## 1 Introduction to MindStudio

1.1 About MindStudio

1.2 Terminology

1.3 Product Modes in Inference Scenario

## 2 Development Environment Setup

2.1 Installation Introduction

2.2 Installing MindStudio on Windows

2.2.1 Windows Supported Features

2.2.2 Installation Workflow

2.2.3 Obtaining Software Packages

2.2.4 Preparing for Installation

2.2.5 Installing MindStudio

2.3 Installing MindStudio on a Linux Physical Machine Using msInstaller

2.3.1 Ubuntu 18.04-x86_64

2.3.2 Ubuntu 18.04-AArch64

2.3.3 CentOS7.6-x86_64

2.3.4 CentOS7.6-AArch64

2.3.5 EulerOS 2.8-AArch64

2.3.6 Kylin OS V10 SP1-AArch64

2.3.7 NeoKylin OS 7.6-AArch64

2.4 Manually Installing MindStudio on a Linux Physical Machine

2.4.1 Installation workflow

2.4.2 Downloading Required Files

2.4.3 Pre-installation Actions

2.4.4 Installing MindStudio

2.4.5 Obtaining Software Packages

2.4.6 Preparing for Installation

2.4.7 Installing MindStudio

2.5 Preparing for Installation

2.5.1 Installation

2.5.2 Preparing for Installation

2.5.3 Creating a MindStudio Container Image Using a Dockerfile

2.5.4 Starting a MindStudio Container

2.5.4.1 Starting a MindStudio Container on an Ascend AI Device

2.5.4.2 Starting the MindStudio Container on a non-Ascend AI Device

2.5.5 Starting the MindStudio Container on a non-Ascend AI Device
3 Supported Operators and Models ................................................................. 84

4 Model Conversion .......................................................................................... 87

5 Model Training .............................................................................................. 114

6 Script Conversion .......................................................................................... 142

7 Application Development ............................................................................ 159
8 Custom Operator Development .............................................................. 222

8.1 Before You Start ........................................................................ 222
8.2 Development Workflow ................................................................ 223
8.2.1 TensorFlow TBE Operator Development Workflow ............ 223
8.2.2 PyTorch TBE Operator Development Workflow .................... 225
8.2.3 MindSpore TBE Operator Development Workflow ............ 227
8.2.4 TensorFlow AI CPU Operator Development Workflow ......... 228
8.3 Operator Analysis ........................................................................ 229
8.4 Project Creation ......................................................................... 232
8.4.1 Creating an Operator Project .................................................. 232
8.4.2 Setting the Python Library .................................................... 242
8.5 TBE Operator Development (TensorFlow) .................................. 246
8.5.1 Operator Code Implementation ............................................. 246
8.5.2 Operator Prototype Definition ............................................... 251
8.5.3 Operator Information Definition ......................................... 253
8.5.4 Operator Plugin Implementation ........................................... 257
8.5.5 UT ...................................................................................... 258
8.5.5.1 Overview ........................................................................ 258
8.5.5.2 APIs ............................................................................... 259
8.5.5.2.1 OpUT ......................................................................... 259
8.5.5.2.2 BroadcastOpUT .......................................................... 264
8.5.5.2.3 ElementwiseOpUT .................................................... 266
8.5.5.2.4 ReduceOpUT .............................................................. 269
8.5.5.3 Creating and Running UT Cases ........................................ 272
8.5.6 Operator Project Building ...................................................... 279
8.5.7 Local Operator Deployment .................................................... 281
8.5.8 Remote Operator Deployment ................................................ 283
8.5.9 ST ....................................................................................... 284
8.6 TBE Operator Development (PyTorch)............................................................................................................................... 302
8.6.1 Operator Code Implementation...................................................................................................................................... 302
8.6.2 Operator Prototype Definition........................................................................................................................................ 307
8.6.3 Operator Information Definition.................................................................................................................................. 309
8.6.4 UT................................................................................................................................................................................... 313
8.6.4.1 Overview................................................................................................................................................................. 313
8.6.4.2 APIs........................................................................................................................................................................ 313
8.6.4.2.1 OpUT................................................................................................................................................................. 313
8.6.4.2.2 BroadcastOpUT............................................................................................................................................ 318
8.6.4.2.3 ElementwiseOpUT........................................................................................................................................ 321
8.6.4.2.4 ReduceOpUT................................................................................................................................................ 323
8.6.4.3 Creating and Running UT Cases.......................................................................................................................... 326
8.6.5 Operator Project Building.............................................................................................................................................. 333
8.6.6 Local Operator Deployment........................................................................................................................................... 336
8.6.7 Remote Operator Deployment...................................................................................................................................... 337
8.6.8 Operator Adaptation in PyTorch................................................................................................................................ 339
8.6.9 ST.................................................................................................................................................................................. 339
8.7 TBE Operator Development (MindSpore).......................................................................................................................... 357
8.7.1 Operator Implementation and Information Registration File......................................................................................... 357
8.7.2 Operator Prototype........................................................................................................................................................ 357
8.7.3 UT................................................................................................................................................................................... 357
8.7.3.1 Overview................................................................................................................................................................. 357
8.7.3.2 APIs........................................................................................................................................................................ 358
8.7.3.2.1 OpUT................................................................................................................................................................. 358
8.7.3.2.2 BroadcastOpUT............................................................................................................................................ 362
8.7.3.2.3 ElementwiseOpUT........................................................................................................................................ 365
8.7.3.2.4 ReduceOpUT................................................................................................................................................ 368
8.7.3.3 Creating and Running UT Cases.......................................................................................................................... 370
8.7.4 ST.................................................................................................................................................................................. 374
8.8 AI CPU Operator Development (TensorFlow).................................................................................................................. 383
8.8.1 Operator Code Implementation...................................................................................................................................... 383
8.8.2 Operator Prototype Definition........................................................................................................................................ 385
8.8.3 Operator Information Definition.................................................................................................................................. 387
8.8.4 Operator Plugin Implementation...................................................................................................................................... 389
8.8.5 Operator Project Building.............................................................................................................................................. 389
8.8.6 Local Operator Deployment........................................................................................................................................... 392
8.8.7 Remote Operator Deployment...................................................................................................................................... 393
8.8.8 ST.................................................................................................................................................................................. 395
8.9 Other Functions and Operations........................................................................................................................................... 412
8.9.1 Operator Project Management........................................................................................................................................ 412
8.9.1.1 Importing an Operator Project........................................................................................................................... 413
8.9.1.2 Editing Project Properties........................................................................................................................................ 414
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.6.2.5 AI Core Metrics View</td>
<td>485</td>
</tr>
<tr>
<td>10.6.3 Analysis Summary</td>
<td>486</td>
</tr>
<tr>
<td><strong>11 Al Core Error Analyzer</strong></td>
<td>489</td>
</tr>
<tr>
<td>11.1 Overview</td>
<td>489</td>
</tr>
<tr>
<td>11.2 Troubleshooting Workflow</td>
<td>490</td>
</tr>
<tr>
<td>11.3 Preparing Data</td>
<td>491</td>
</tr>
<tr>
<td>11.4 Analyzing AI Core Errors</td>
<td>494</td>
</tr>
<tr>
<td>11.5 Locating AI Core Errors</td>
<td>496</td>
</tr>
<tr>
<td><strong>12 Operation Guidelines</strong></td>
<td>498</td>
</tr>
<tr>
<td>12.1 SSH Connection Management</td>
<td>498</td>
</tr>
<tr>
<td>12.1.1 Restrictions</td>
<td>498</td>
</tr>
<tr>
<td>12.1.2 Overview</td>
<td>499</td>
</tr>
<tr>
<td>12.1.3 Adding a New SSH Connection</td>
<td>501</td>
</tr>
<tr>
<td>12.1.4 Deleting an SSH Connection</td>
<td>501</td>
</tr>
<tr>
<td>12.1.5 Modifying an SSH Connection</td>
<td>501</td>
</tr>
<tr>
<td>12.1.6 Copying an SSH Connection</td>
<td>501</td>
</tr>
<tr>
<td>12.1.7 Encrypting the SSH Password</td>
<td>501</td>
</tr>
<tr>
<td>12.1.8 Changing the SSH Password Saving Mode</td>
<td>502</td>
</tr>
<tr>
<td>12.2 Project Management</td>
<td>503</td>
</tr>
<tr>
<td>12.2.1 Opening/Closing/Removing a Project</td>
<td>503</td>
</tr>
<tr>
<td>12.2.2 Creating/Deleting a File</td>
<td>506</td>
</tr>
<tr>
<td>12.3 Run Configuration</td>
<td>507</td>
</tr>
<tr>
<td>12.3.1 Editing a Run Configuration</td>
<td>507</td>
</tr>
<tr>
<td>12.3.2 Pausing and Resuming</td>
<td>508</td>
</tr>
<tr>
<td>12.4 Device Manager</td>
<td>508</td>
</tr>
<tr>
<td>12.5 ADK Manager</td>
<td>509</td>
</tr>
<tr>
<td>12.5.1 Overview</td>
<td>509</td>
</tr>
<tr>
<td>12.5.2 Updating ADK</td>
<td>511</td>
</tr>
<tr>
<td>12.5.3 Switching/Activating ADK</td>
<td>511</td>
</tr>
<tr>
<td>12.5.4 Configuring the Cross-Compilation Environment</td>
<td>513</td>
</tr>
<tr>
<td>12.6 Log Management</td>
<td>514</td>
</tr>
<tr>
<td>12.6.1 Before You Start</td>
<td>514</td>
</tr>
<tr>
<td>12.6.2 Log Management</td>
<td>515</td>
</tr>
<tr>
<td>12.6.3 Log Level Setting</td>
<td>517</td>
</tr>
<tr>
<td>12.7 Shortcuts</td>
<td>518</td>
</tr>
<tr>
<td>12.8 Common Native Functions</td>
<td>519</td>
</tr>
<tr>
<td><strong>13 MindStudio Management (Linux)</strong></td>
<td>521</td>
</tr>
<tr>
<td>13.1 Querying MindStudio Version</td>
<td>521</td>
</tr>
<tr>
<td>13.2 Launching MindStudio</td>
<td>522</td>
</tr>
<tr>
<td>13.3 Exiting MindStudio</td>
<td>522</td>
</tr>
<tr>
<td>13.4 Updating MindStudio</td>
<td>522</td>
</tr>
</tbody>
</table>
13.5 Uninstalling MindStudio................................................................................................................................. 524
13.6 FAQs....................................................................................................................................................................... 525
13.6.1 What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When
"pip3 install" Is Run?.................................................................................................................................................. 525

14 MindStudio Management (Windows)................................................................................................................... 527

15 Appendixes................................................................................................................................................................ 530
15.1 MindStudio Package Overview............................................................................................................................ 530
15.2 Installing Dependencies........................................................................................................................................... 531
15.2.1 Ubuntu 18.04-x86_64......................................................................................................................................... 531
15.2.2 Ubuntu 18.04-aarch64..................................................................................................................................... 536
15.2.3 EulerOS 2.8-aarch64....................................................................................................................................... 542
15.2.4 CentOS7.6-x86_64.......................................................................................................................................... 547
15.2.5 CentOS7.6-AArch64......................................................................................................................................... 552
15.2.6 Kylin OS V10 SP1-aarch64............................................................................................................................ 558
15.2.7 Installing Dependencies (NeoKylin OS 7.6-AArch64)....................................................................................... 563
15.2.8 Installing Dependencies (Windows 10).............................................................................................................. 569
15.3 Operator Specifications........................................................................................................................................... 577
15.3.1 Caffe Operator Specifications........................................................................................................................... 577
15.3.2 TensorFlow Operator Specifications.................................................................................................................. 578
15.3.3 ONNX Operator Specifications........................................................................................................................ 578
15.4 Feature Configuration.......................................................................................................................................... 578
15.4.1 Custom Caffe Network Modification................................................................................................................ 578
15.4.1.1 Overview....................................................................................................................................................... 578
15.4.1.2 Custom Operator List.................................................................................................................................. 579
15.4.1.3 Custom Operator Description...................................................................................................................... 581
15.4.1.4 Samples....................................................................................................................................................... 592
15.4.1.4.1 Modifying Faster R-CNN Prototxt............................................................................................................... 592
15.4.1.4.2 Modifying YOLOv3 Prototxt........................................................................................................................ 594
15.4.1.4.3 Modifying YOLOv2 Prototxt........................................................................................................................ 596
15.4.1.4.4 Modifying SSD Prototxt............................................................................................................................ 598
15.4.1.4.5 Modifying BatchedMatMul Prototxt.......................................................................................................... 600
15.4.1.4.6 Modifying SENet Prototxt......................................................................................................................... 600
15.5 How Do I Determine the Video Stream Format Standard When I Perform CSC on a Model Using
AIPP?......................................................................................................................................................................... 601
15.6 Configuring a System Network Proxy................................................................................................................. 602
15.7 Security Hardening Suggestions.......................................................................................................................... 603
15.8 FAQs........................................................................................................................................................................ 603
15.8.1 What Do I Do If the Copied Content Cannot Be Pasted to the Editor Window Occasionally When
MindStudio Is Opened Remotely on Windows?.......................................................................................................... 603
15.8.2 What Do I Do If the Arrow Keys Occasionally Change to Numbers When MobaXterm Starts
MindStudio to Edit Code?............................................................................................................................................ 604
15.8.3 How Do I Allow Unsigned Access Requests?.................................................................................................. 604
15.8.4 What Do I Do If the GUI Cannot Be Displayed When MindStudio Is Started?......................................... 605
1 Introduction to MindStudio

1.1 About MindStudio

MindStudio is an IntelliJ-based integrated development environment (IDE). It enables application development and debugging and model conversion. It also supports network porting, optimization, and profiling, facilitating application development.

**NOTE**
Read *MindStudio User Guide* in conjunction with the associated development related auxiliary manuals of the product in use.

- For details about the Atlas 200 DK, see *Atlas 200 DK Developer Kit*.
- For details about other products, see the *Ascend CANN documentation*.

### Highlights

- MindStudio provides multiple deployment modes and supports multiple mainstream OSs and container-based deployment, facilitating installation and deployment.
- MindStudio provides a complete operator development process, including the unit test (UT), system test (ST), and TIK operator debugging. It allows you to develop TBE and AI CPU custom operators based on mainstream frameworks such as TensorFlow, PyTorch, and MindSpore.
- For network model development, MindStudio supports model training based on TensorFlow, PyTorch, Caffe, and MindSpore, as well as model conversion of multiple mainstream frameworks. MindStudio integrates the MindInsight, script conversion tool, Ascend Tensor Compiler (ATC), Ascend Model Compression Toolkit (AMCT), and Model Accuracy Analyzer, boosting the efficiency of network model porting, analysis, and optimization.
- For application development, MindStudio integrates tools such as Profiling, compiler, application development interconnected with MindX SDK, and...
visualized pipeline orchestration, providing developers with a graphical integrated development environment (IDE). MindStudio can be used to perform full-pipeline development, covering as project management, compilation, debugging, and performance analysis, which greatly improves the development efficiency.

Architecture

Figure 1-1 shows the MindStudio architecture. The toolchain provides a full coverage including model conversion, model training, custom operator development, application development, project management, compilation, process orchestration, accuracy analysis, log management, performance profiling, and device management.

Features

MindStudio offers the following features:

- Manages projects including project creation, opening, closing, and deletion, as well as project directory creation and project property setting. For details, see 12.2 Project Management.
- Manages SSH connections including adding, modifying, deleting, and copying SSH connections as well as encrypting SSH passwords, and selecting the SSH password saving mode. For details, see 12.1 SSH Connection Management.
- Develops applications using Ascend Computing Language (AscendCL) and MindX SDK, which shortens the learning curve and makes process development smarter with the post-programming services ranging from compilation and execution to result display. For details, see 7 Application Development.
- Develops custom operators using Tensor Boost Engine (TBE) and AI CPUs, which makes operator porting across platforms easier and interconnection to
the Ascend AI Processor faster. For details, see **8 Custom Operator Development**.

- Converts models trained in third-party frameworks into visualized offline models with model interfaces automatically generated in one-click mode, facilitating programming. For details, see **4 Model Conversion**.

- Collects and analyzes system-wide logs of the Ascend AI Processor, improving the efficiency of locating algorithm faults in runtime. The log analysis system enables visualized, unified analysis of cross-platform logs and runtime diagnosis, enhancing its usability. For details, see **12.6 Log Management**.

- Profiles performance in graphical user interface (GUI) mode for the multi-node, multi-module heterogeneous host-device system. The performance and power consumption of the Ascend AI Processor can be synchronously analyzed, facilitating algorithm optimization. For details, see **10 Profiling**.

- Manages devices connected to the host. For details, see **12.4 Device Manager**.

- Analyzes model accuracy by comparing the execution results of Huawei's operators and those of standard Caffe, TensorFlow, and ONNX operators to locate the faults. For details, see **9 Model Accuracy Analyzer**.

- Installs and manages the development kit Ascend-CANN-Toolkit, streamlining AI algorithm development with the Ascend AI Processor. Developers can install the development kit on MindStudio and use MindStudio for quick development or use an independent development kit for development. The Ascend-CANN-Toolkit contains the header files and library files on which the Ascend AI Processor depends, compilation toolchain, and optimization tools. For details, see **12.5 ADK Manager**.

### 1.2 Terminology

#### Host and Device

In a heterogeneous computing architecture, the the Ascend AI Processor and CPU collaborate over the PCIe bus.

- The host is an x86 server, an ARM server, or a Windows PC that is connected to the hardware device powered by the Ascend AI Processor. The host leverages the neural-network (NN) computing capability provided by the Ascend AI Processor to implement services. See the development platform layer in **Figure 1-1**.

- A device is a hardware device powered by the Ascend AI Processor. It connects to a server over the PCIe interface and provides the NN compute capability for the server. See the SoC layer in **Figure 1-1**.

For the Ascend RC form, since the hardware device powered by the the Ascend AI Processor is co-deployed with the interconnected ARM server, the host and device sides are not distinguished and are referred to as the host.

#### ATC

Ascend Tensor Compiler (ATC) converts models trained on third-party frameworks such as Caffe and TensorFlow into offline models compatible with the Ascend AI
Processor. The offline models can be loaded to applications for inference through Graph Engine (GE).

*Figure 1-2* shows the ATC workflow.

*Figure 1-2 ATC workflow*

GE provides a collection of secure, easy-to-use graph construction APIs for graph/operator Intermediate Representation (IR). These APIs can be called to build a model, and set graphs in a model, operators in a graph, and attributes of a model and operators.

**DVPP**

Digital Vision Pre-Processing (DVPP) implements image preprocessing ranging from encoding and decoding to format conversion. DVPP converts the video or image data input from the system memory or Internet into formats supported by the Ascend AI Processor before neural network is computed by the Ascend AI Processor.

**AIPP**

AI Pre-Processing (AIPP) is introduced for hardware-based image preprocessing including color space conversion (CSC), image normalization (by subtracting the mean value or multiplying a factor), image cropping (by specifying the crop start and cropping the image to the size required by the neural network), and much more. Images output by DVPP are aligned images in YUV420SP format. DVPP does not output RGB images. Therefore, AIPP can be used to convert the aligned YUV420SP images and then crop them to the size required by the model.

**YUV420SP**

It is a lossy image color encoding format, which can be YUV420SP_UV or YUV420SP_VU.
1.3 Product Modes in Inference Scenario

The following uses the Ascend AI Processor as an example. If the PCIe works in master mode and supports peripherals, this is referred to as RC mode (or Ascend RC in the following). If the PCIe works in slave mode, this is referred to as EP mode (or Ascend EP in the following).

- The working modes of the Ascend AI Processor are as follows:
  - The Ascend 310 AI Processor supports both the EP and RC modes.
  - The Ascend 910 AI Processor supports only the EP mode.
- The following products support RC mode: Atlas 200 AI accelerator module and Atlas 200 DK developer kit.
  The CPUs of such products run the AI service software specified by the running user directly and connect to peripherals such as network cameras, I²C sensors, and SPI monitors as slave devices.
- The following products support EP mode:
  - Ascend 910 AI Processor: Atlas 800 training server and Atlas 300T training card
  In EP mode, the host acts as the master, the device acts as the slave, and customer AI service programs run on the host. The product functions as a device system and connects to the host system through the PCIe channel, while the host interacts with the device system through the PCIe channel and loads AI tasks to the Ascend AI Processor on the device.

Figure 1-3 shows the products and architecture of the two modes.

- The host is an x86 server, an ARM server, or a Windows PC connected to hardware equipped with an Ascend AI Processor. The host leverages the neural network (NN) compute capability provided by the Ascend AI Processor to implement services.
- A device is a hardware backend equipped with an Ascend AI Processor. It is connected to the server over the PCIe interface and provides the NN compute capability required by the server.
Figure 1-3 RC and EP modes

RC mode
- Atlas 200 AI accelerator module
- Atlas 200 DK developer kit

EP mode
- Atlas 200 AI accelerator module
- Atlas 300I inference card
- Atlas 500 Pro AI edge station
- Atlas 500 Pro AI edge server
- Atlas 800 inference server
- Atlas 300i Pro inference card
- Atlas 800 training server
- Atlas 300T training card
2 Development Environment Setup

2.1 Installation Introduction

Software Packages

- MindStudio is located in the development environment where you can manage projects, write and build code, and convert models. The built applications are executed on the operating environment powered by the Ascend AI Processor. To run a developed project on the Ascend AI Processor, connect MindStudio to the operating environment. Project execution, log diagnosis, and performance profiling require the collaboration between the host and the background service modules of the device powered by the Ascend AI Processor.

- Ascend-CANN-Toolkit: a developer toolkit for developers to efficiently develop models, algorithms, and applications based on the Ascend AI Processor. The developer toolkit can be installed only on the Linux server. After the developer toolkit is installed, developers can use the MindStudio development tool to perform quick development.

Application Scenarios

- (Scenario 1) Install the Ascend-CANN-Toolkit on Ascend AI devices or non-Ascend AI devices. The development kit can be used as the development environment and can be used only for development activities such as code development and compilation (for example, ATC model conversion and pure code development of operators and inference applications). If you want to run applications or perform training, you need to connect to the Ascend AI device where the environment is deployed.
(Scenario 2) Install the Ascend-CANN-Toolkit, npu-firmware installation package, npu-driver installation package, and AI framework on Ascend AI devices. The packages can be used as the development environment and operating environment to run applications or perform training. For details about how to deploy Ascend AI devices, see *CANN Software Installation Guide (ascend-deployer)*.

### Installing MindStudio on Windows

MindStudio can be directly installed on Windows. Before the installation, you need to install and deploy Ascend-CANN-Toolkit on Linux, and then install MindStudio on Windows. After the installation is complete, configure the remote connection mode to establish the connection between the Windows environment where MindStudio is located and the Linux environment where Ascend-CANN-Toolkit is located to implement full-pipeline development. For details, see 2.2 Installing MindStudio on Windows.

![Figure 2-1 Architecture of the installation on Windows](image)

### Installing MindStudio on Linux

MindStudio can be installed on a Linux server. You can use the GNOME Terminal provided by the native desktop to install it. If you are using a Windows server, you can log in to a Linux server in SSH mode to install MindStudio. Since MindStudio is a GUI-based IDE, to install MindStudio using the later method, you need to prepare an SSH terminal (for example, MobaXterm v20.2 or later) integrated with the X server.

- You can use msiInstaller to automatically install the OS dependencies, MindStudio, and Ascend-CANN-Toolkit. For details, see *CANN Software Installation Guide (msInstaller)*.
- For details about how to manually deploy the development environment, see 2.4 Manually Installing MindStudio on a Linux Physical Machine.
- To deploy the development environment in container mode, pull image files through the network to build a MindStudio container. For details, see 2.5 Installing MindStudio on Linux (Container-based).
2.2 Installing MindStudio on Windows

2.2.1 Windows Supported Features

Currently, MindStudio is used on the Windows OS. Table 2-1 lists the supported features.

Table 2-1 Feature list

<table>
<thead>
<tr>
<th>Feature</th>
<th>Sub-Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation and deployment</td>
<td>Upgrading the IDE online</td>
</tr>
<tr>
<td></td>
<td>Decompressing and installing the MindStudio package</td>
</tr>
<tr>
<td></td>
<td>Installing the MindStudio package</td>
</tr>
<tr>
<td>Custom operator development</td>
<td>Creating an operator project</td>
</tr>
<tr>
<td></td>
<td>UT and ST on operator development</td>
</tr>
<tr>
<td></td>
<td>TMModel performance simulation for operator development</td>
</tr>
<tr>
<td></td>
<td>Building and deploying an operator remotely</td>
</tr>
<tr>
<td>Application project</td>
<td>Building and running an ACL application project locally or remotely</td>
</tr>
<tr>
<td></td>
<td>Generating dump files locally or remotely</td>
</tr>
<tr>
<td>Model conversion</td>
<td>Model conversion</td>
</tr>
<tr>
<td></td>
<td>Original model visualization</td>
</tr>
</tbody>
</table>
### 2.2.2 Installation Workflow

**Figure 2-3** MindStudio installation workflow

1. Analyze local environment requirements.
2. Configure the remote environment.
3. Install dependencies.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Sub-Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic AIPP, dynamic ImageSize, and dynamic Batch</td>
<td></td>
</tr>
<tr>
<td>Static AIPP, which supports resizing and padding</td>
<td></td>
</tr>
<tr>
<td>Customizing the output node flag</td>
<td></td>
</tr>
<tr>
<td>Model evaluation report (inference scenario)</td>
<td></td>
</tr>
<tr>
<td>Loading historical configuration files</td>
<td></td>
</tr>
<tr>
<td>Training project</td>
<td>Model training</td>
</tr>
<tr>
<td>Profiling</td>
<td>Local or remote profiling and data display</td>
</tr>
<tr>
<td></td>
<td>Merge Reports</td>
</tr>
<tr>
<td>Instructions</td>
<td>Installing the ADK from the local PC</td>
</tr>
<tr>
<td></td>
<td>Version management of the installed ADK</td>
</tr>
<tr>
<td></td>
<td>ADK version switching</td>
</tr>
<tr>
<td></td>
<td>Device manager</td>
</tr>
<tr>
<td></td>
<td>SSH configuration management</td>
</tr>
</tbody>
</table>
2.2.3 Obtaining Software Packages

Table 2-2 Package description

<table>
<thead>
<tr>
<th>Software Package</th>
<th>Description</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>MindStudio_{version}_win.zip</td>
<td>Installation-free MindStudio package, which contains a graphical IDE.</td>
<td>Obtaining Software Packages</td>
</tr>
<tr>
<td>MindStudio_{version}.exe</td>
<td>MindStudio installation package, which contains a graphical IDE.</td>
<td>Obtaining Software Packages</td>
</tr>
</tbody>
</table>

Validating Software Package

To prevent a software package from being maliciously tampered with during transmission or storage, you need to download the corresponding digital signature file for integrity validation when downloading the software package.

After the software package is downloaded, verify its PGP digital signature by referring to the OpenPGP Signature Verification Guide. If the validation fails, do not use the software package. Contact Huawei technical support.

Before using a software package for installation or upgrade, you also need to verify its digital signature by referring to the OpenPGP Signature Verification Guide to ensure that the software package has not been tampered with.

2.2.4 Preparing for Installation

Local Environment Requirements

Windows 10

Local Installation Dependencies

- JDK 11
- Python3.7.5
- MinGW

For details, see 15.2.8 Installing Dependencies (Windows 10).

Configuring the Remote Environment

1. Configure the MindStudio environment based on the version of the remote Linux server. For details, see 2.4.3 Pre-installation Actions and 2.4.5 Configuring the Compilation Environment.
NOTE

You do not need to install JDK 11 when installing remote environment dependencies. Install JDK 11 on the local Windows host by referring to 15.2.8 Installing Dependencies (Windows 10).

2. Download the Ascend-CANN-Toolkit development kit from Ascend Community and run the following commands as the user created in 1 to install the development kit:

```bash
chmod +x *.run
./*.run --install
```

In the preceding commands, `*.run` indicates `Ascend-cann-toolkit_{version}_linux-{arch}.run`. Replace it with the actual development kit.

2.2.5 Installing MindStudio

Prerequisites

Prepare for MindStudio installation. See 2.2.3 Obtaining Software Packages and 2.2.4 Preparing for Installation.

Procedure

**Step 1** Double-click `MindStudio_{version}.exe` to install MindStudio.

If you use `MindStudio_{version}_win.zip` to install MindStudio, skip Steps **Step 1** to **Step 7**. Double-click `MindStudio64.exe` in the `bin` directory generated after the decompression, and start from Step **Step 8** to configure MindStudio.

**Step 2** On the MindStudio Setup page, click Next, as shown in Figure 2-4.

![MindStudio Setup](image)

**Figure 2-4 MindStudio Setup**

Welcome to MindStudio Setup

Setup will guide you through the installation of MindStudio.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.
Step 3  Select the MindStudio installation path and click Next. See Figure 2-5.

Figure 2-5 MindStudio installation path

Step 4  Select installation options as required and click Next. See Figure 2-6.

Figure 2-6 Selecting installation options

- **Create Desktop Shortcut**: creates a desktop shortcut. You can select 32-bit launcher or 64-bit launcher based on the system configuration.
- **Update PATH variable (restart needed)**: adds the path of the MindStudio boot file to the environment variable PATH so that MindStudio can be started directly from the system command line.

- **Update context menu**: Select Add "Open Folder as Project" and right-click the folder to open it as a MindStudio project.

- **Create Associations**: By default, this option is not selected.

**Step 5** Select or create the start menu folder in the MindStudio installation path, and click **Install**. See **Figure 2-7**.

![Choose Start Menu Folder](image)

**Figure 2-7** Start Menu Folder

**Step 6** Start to install MindStudio. After the installation is complete, click **Next**. See **Figure 2-8**.

![Choose Start Menu Folder](image)
Step 7 If you want to run MindStudio directly after the installation, select Run MindStudio and click Finish. See Figure 2-9.

Step 8 Open the Import MindStudio Settings dialog box, as shown in Figure 2-10.
Figure 2-10 Environment settings

- **Config or installation folder**: imports MindStudio settings from an existing directory.
  The directory is `C:\Users\Individual User\AppData\Roaming\MindStudioMS-x.x`. The MindStudio settings include the personalized settings of the project UI such as the background color.
- **Do not import settings** (default): creates a configuration file.

Select an option and click **OK**.

**NOTE**

- If an earlier MindStudio installation is detected, the **Previous version** parameter is displayed in the dialog box shown in Figure 2-10 when you are installing a later version. You can choose whether to import the settings from the earlier version as required.
- If MindStudio of the same version is detected and the `C:\Users\Individual User\AppData\Roaming\Huawei\MindStudioMS-x.x` and `C:\Users\Individual User\.mindstudio` directories are not deleted when you uninstall the previous version, the re-installation proceeds to **Step 14** from Step **Step 8**.

**Step 9** If the dialog box shown in Figure 2-11 is displayed:

**NOTE**

This dialog box is also displayed when you use MindStudio. You can determine whether to click **Accept** based on the site requirements. If you choose **File > Settings... > Tools > Server Certificates** on the menu bar and select **Accept non-trusted certificates automatically**, the certificates will be automatically accepted and this dialog box will not be displayed when you use MindStudio.
Figure 2-11 Untrusted server certificate

- Click **Accept** to allow MindStudio to connect to the upgrade server.
- Click **Reject** to reject MindStudio to connect to the upgrade server. This dialog box is displayed when you restart MindStudio next time.

**Step 10** On the page shown in Figure 2-12, click to connect the Windows environment where MindStudio is located to the remote development environment where Ascend-CANN-Toolkit is located.
Step 11 On the page for configuring the SSH connection, click to configure the SSH connection parameters. Click OK, as shown in Figure 2-13.

Table 2-3 Parameters and icons

<table>
<thead>
<tr>
<th>Parameter/Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Destination IP address of a connection.</td>
</tr>
<tr>
<td>Parameter/Icon</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Port</td>
<td>Port number of the destination IP address of the connection.</td>
</tr>
<tr>
<td>User name</td>
<td>Username for logging in to the target host.</td>
</tr>
<tr>
<td>Authentication type</td>
<td>Authentication method. Two methods are available:</td>
</tr>
<tr>
<td></td>
<td>● <strong>PASSWORD</strong>: Enter a password for authentication.</td>
</tr>
<tr>
<td></td>
<td>● <strong>KEY_PAIR</strong>: Use an SSH key for authentication.</td>
</tr>
<tr>
<td></td>
<td>Prepare the key in advance by yourself.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong></td>
</tr>
<tr>
<td></td>
<td>In the EulerOS 2.8 AArch architecture or Kylin OS V10 SP1 AArch architecture, when <strong>KEY_PAIR</strong> is used for identity authentication, you need to run the <code>ssh-keygen -m PEM</code> or <code>ssh-keygen -t rsa -m PEM</code> command to apply for an SSH key.</td>
</tr>
<tr>
<td>Password</td>
<td>Available when <strong>Authentication type</strong> is set to <strong>PASSWORD</strong>. Enter the correct password. You can select <strong>Save password</strong> to save the password.</td>
</tr>
<tr>
<td>Private key file</td>
<td>Available when <strong>Authentication type</strong> is set to <strong>KEY_PAIR</strong>. Select a local private key file.</td>
</tr>
<tr>
<td>Passphrase</td>
<td>Available when <strong>Authentication type</strong> is set to <strong>KEY_PAIR</strong>. Enter the correct passphrase. You can select <strong>Save passphrase</strong> to save the passphrase.</td>
</tr>
<tr>
<td>Test Connection</td>
<td>Checks whether the connection is successful.</td>
</tr>
<tr>
<td>SSH tunnel</td>
<td>Mapping between the local port and the remote port.</td>
</tr>
<tr>
<td></td>
<td>The format is <code>&lt;local port&gt;:&lt;remote port&gt;</code>.</td>
</tr>
<tr>
<td>Remote work path</td>
<td>Remote work path.</td>
</tr>
<tr>
<td>+</td>
<td>Adds a new SSH connection.</td>
</tr>
<tr>
<td>-</td>
<td>Deletes an SSH connection. Select an SSH connection to be deleted and click this icon.</td>
</tr>
<tr>
<td></td>
<td>Save as icon. Select an SSH connection to be saved and click this icon.</td>
</tr>
<tr>
<td></td>
<td>Modifies the alias of an SSH connection. Select the SSH connection to be modified and click this icon.</td>
</tr>
<tr>
<td>OK</td>
<td>Saves the SSH configuration. Click this icon to save and apply the current SSH connection configuration, and close the configuration window.</td>
</tr>
<tr>
<td>Cancel</td>
<td>Cancels the current configuration. Click this icon to cancel the current configuration and close the configuration window.</td>
</tr>
<tr>
<td>Apply</td>
<td>Applies the SSH configuration. Click this icon to save and apply the current configuration.</td>
</tr>
</tbody>
</table>
Step 12 Enter the Ascend-CANN-Toolkit installation path in the remote environment and click Finish. See Figure 2-14.

Figure 2-14 Ascend-CANN-Toolkit installation path

Step 13 The User Experience Improvement Program page is displayed, as shown in Figure 2-15. You can set Name and Organization as required and click the User Experience Improvement Program link to read the details. Selecting the check box ☑️ indicates that you have read the content.
Figure 2-15 User experience improvement program

- Click **Agree**. MindStudio will collect user information for the experience improvement program.
- Click **Disagree**. The page will be closed, and MindStudio will not collect user information for the experience improvement program.

**Step 14** If no error is reported and the welcome window is displayed, MindStudio is successfully installed. See **Figure 2-16**.
### Figure 2-16 Welcome to MindStudio

- **Create New Project**: creates a project.
- **Open or Import**: opens or imports a project.
- **Welcome System Profiling**: opens the **System Profiling** window.

**Step 15** Start MindStudio. Choose **File > Settings... > Plugins** to search for and install the following plugin tools:
- Grep Console
- Pylint
- Python Community Edition

---End

### 2.3 Installing MindStudio on a Linux Physical Machine Using msInstaller

You can use msInstaller to automatically install the OS dependencies, MindStudio, and Ascend-CANN-Toolkit. For details, see *CANN Software Installation Guide (msInstaller)*.

**NOTE**

- For details about how to manually install MindStudio, see **2.4 Manually Installing MindStudio on a Linux Physical Machine**.
- For details about how to set up the Atlas 200 DK environment, see "Environment Deployment" in *Atlas 200 DK Developer Kit*.
2.4 Manually Installing MindStudio on a Linux Physical Machine

2.4.1 Installation workflow

![Figure 2-17 MindStudio installation workflow]

- Start
- Obtain software packages.
- Prepare for installation.
- Install MindStudio.
- End

1. Analyze environment requirements.
2. Prepare the MindStudio installation user.
3. (Optional) Configure the permission of the MindStudio installation user.
4. Install dependencies.

2.4.2 Downloading Required Files

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>MindStudio_{version}_linux.tar.gz</td>
<td>MindStudio package, a GUI-based IDE</td>
<td>Software package</td>
</tr>
<tr>
<td>File</td>
<td>Description</td>
<td>How to Obtain</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>---------------</td>
</tr>
</tbody>
</table>
| Ascend-cann-toolkit_{version}_linux-{arch}.run | Ascend-CANN-Toolkit, including auxiliary toolkit and APIs.  
● If an installation is available in your environment, do not install it repeatedly.  
● If no installation is available in your environment, download its runfile and install it in the MindStudio installation process. | Software package |

Validating Software Package

To prevent a software package from being maliciously tampered with during transmission or storage, you need to download the corresponding digital signature file for integrity validation when downloading the software package.

After the software package is downloaded, verify its PGP digital signature by referring to the OpenPGP Signature Verification Guide. If the validation fails, do not use the software package. Contact Huawei technical support.

Before using a software package for installation or upgrade, you also need to verify its digital signature by referring to the OpenPGP Signature Verification Guide to ensure that the software package has not been tampered with.

2.4.3 Pre-installation Actions

2.4.3.1 Ubuntu 18.04-x86_64

Environment Requirements

The minimum hardware and OS requirements for installing MindStudio are as follows:
Table 2-5 Ubuntu OS mapping information

<table>
<thead>
<tr>
<th>Item</th>
<th>Restriction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>● Recommended memory size: 8 GB (at least 4 GB)</td>
<td>● If the memory size of the Linux host is 4 GB, the size of the model file should be less than or equal to 350 MB for model conversion on MindStudio. Otherwise, the OS may be unstable.</td>
</tr>
<tr>
<td></td>
<td>● Minimum drive space: 6 GB</td>
<td>● If the configurations of the Linux host are upgraded, for example, the memory is expanded to 8 GB, the supported operation specifications should be increased proportionally. If the memory is expanded from 4 GB to 8 GB, keep the model file size within 700 MB.</td>
</tr>
<tr>
<td>OS</td>
<td>Ubuntu 18.04 (x86_64)</td>
<td>● Available at <a href="http://releases.ubuntu.com/releases/">http://releases.ubuntu.com/releases/</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>● During model conversion, if a custom TE plugin is loaded, the UI may be suspended. In this case, run the <code>uname -r</code> or <code>cat /proc/version</code> command to check whether the Linux kernel version is earlier than 4.18. If yes, install the Linux kernel patch available at <a href="https://bugzilla.kernel.org/attachment.cgi?id=277305">https://bugzilla.kernel.org/attachment.cgi?id=277305</a>.</td>
</tr>
<tr>
<td>System language</td>
<td>en_US.UTF-8</td>
<td>● Only English version is supported.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● Run the <code>locale</code> command as any user to query the encoding format in any directory. If the system displays <code>LANG=en_US.UTF-8</code>, the encoding format is correct. Otherwise, run the <code>vim /etc/default/locale</code> command as the <code>root</code> user to modify LANG=en_US.UTF-8 and reboot the system (using the <code>reboot</code> command) for the settings to take effect.</td>
</tr>
</tbody>
</table>

(Optional) Preparing the MindStudio Installation User

- If an Ascend-CANN-Toolkit installation is available, install MindStudio as the Ascend-CANN-Toolkit installation user.
- If there is no Ascend-CANN-Toolkit installation, prepare an installation user as follows:

  You can install MindStudio as any user (`root` or non-root).

  - To install MindStudio as the `root` user, skip this section.
  - To install MindStudio as an existing non-root user, ensure that the user has the read, write, and execute permissions on the `$HOME` directory.
To install MindStudio as a new non-root user, perform the following steps as the **root** user. This document uses this scenario as an example to describe how to install MindStudio.

a. Create a MindStudio installation user and set the $HOME directory of the user:

```bash
useradd -g HwHiAiUser -d /home/username -m username -s /bin/bash
```

**NOTE**

The user must be at the same group as the Driver running user. Otherwise, add the user to the Driver running user group. Assume the user group is HwHiAiUser.

You can run the `useradd -g HwHiAiUser -d /home/username -m username -s /bin/bash` command to create a MindStudio installation user and add it to the group.

b. Run the following command to set the user password:

```bash
passwd username
```

`username` indicates the name of the MindStudio installation user. The umask value of the user cannot be greater than 0027.

- You can view the `umask` value by running the `umask` command.
- You can change the `umask` value by running the `umask NewValue` command.

If the value of `umask` is changed using the preceding method, `NewValue` takes effect only in the current window. To set a permanent value of `umask`, modify the `~/.bashrc` file as follows:

1) Run the following command in any directory to open the `.bashrc` file:

   ```bash
   vi ~/.bashrc
   ```

   Append `umask NewValue` to the file.

2) Run the `:wq!` command to save the file and exit.

3) Run the `source ~/.bashrc` command for the modification to take effect immediately.

### (Optional) Configuring the Permission of the MindStudio Installation User

Skip the following part if you install the component as the root user.

Before installing MindStudio, you need to download related dependent software. The following commands may be used during the installation. You need to perform the following operations as the root user to escalate privileges for common users to ensure they can use the commands.

#### Step 1
Open the `/etc/sudoers` file:

```bash
chmod u+w /etc/sudoers
vi /etc/sudoers
```

#### Step 2
Add the following lines to the file:

```bash
username ALL=(ALL:ALL) NOPASSWD:SETENV:/usr/bin/apt-get, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/cp, /usr/bin/make install, /usr/bin/pip3, /usr/bin/pip3.7, /usr/bin/pip3.7.5, /usr/bin/python3.7.5, /bin/ln, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/local/python3.7.5/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/local/python3.7.5/bin/pip3.7.5, /usr/local/python3.7.5/bin/pip3.7.5, /usr/bin/unzip, /usr/sbin/update-alternatives, /usr/bin/add-apt-repository, /usr/local/python3.7.5/bin/pip3
```

Replace `username` with the actual installation user.
If the `/etc/sudoers` file does not contain `#includedir /etc/sudoers.d`, append the line to the file.

After the MindStudio installation is complete, retaining the preceding privilege escalation configuration may cause risks. Therefore, you need to cancel the sudo permission immediately after the installation is complete. To cancel the permission, delete all the trustlisted commands of the corresponding common user.

If a message is displayed indicating that the permission is incorrect during the uninstallation, you also need to configure the sudo permission. After the uninstallation is complete, cancel the sudo permission.

**Step 3** Run the `:wq!` command to save the file.

**Step 4** Remove the write permission on the `/etc/sudoers` file:
```
chmod u-w /etc/sudoers
```

---End

**Checking Sources**

MindStudio installation requires the download of related dependencies. Ensure that the server where MindStudio is installed can be connected to the network.

Run the following command as the root user to check whether the source is available:
```
apt-get update
```

**NOTE**

If an error is reported during the command execution, check whether the network connection is normal or replace the source in the `/etc/apt/sources.list` file with a valid one.

**Installing Dependencies**

For details, see 15.2.1 Ubuntu 18.04-x86_64.

2.4.3.2 Ubuntu 18.04-AArch64

**Environment Requirements**

The minimum hardware and OS requirements for installing MindStudio are as follows:
Table 2-6 Ubuntu OS mapping information

<table>
<thead>
<tr>
<th>Item</th>
<th>Restriction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardw</td>
<td>● Recommended memory size: 8 GB (at least 4 GB)</td>
<td>● If the memory size of the Linux host is 4 GB, the size of the model file should be less than or equal to 350 MB for model conversion on MindStudio. Otherwise, the OS may be unstable.</td>
</tr>
<tr>
<td></td>
<td>● Minimum drive space: 6 GB</td>
<td>● If the configurations of the Linux host are upgraded, for example, the memory is expanded to 8 GB, the supported operation specifications should be increased proportionally. If the memory is expanded from 4 GB to 8 GB, keep the model file size within 700 MB.</td>
</tr>
</tbody>
</table>

| OS          | Ubuntu 18.04 (AArch64)                                                      | ● During model conversion, if a custom TE plugin is loaded, the UI may be suspended. In this case, run the `uname -r` or `cat /proc/version` command to check whether the Linux kernel version is earlier than 4.18. If yes, install the Linux kernel patch available at [https://bugzilla.kernel.org/attachment.cgi?id=277305](https://bugzilla.kernel.org/attachment.cgi?id=277305). |
|             |                                                                            | ● For details about how to upgrade the Linux kernel, visit [https://bbs.huaweicloud.com/forum/thread-22441-1-1.html](https://bbs.huaweicloud.com/forum/thread-22441-1-1.html). |

<table>
<thead>
<tr>
<th>System language</th>
<th>en_US.UTF-8</th>
<th>● Only English version is supported.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>● Run the <code>locale</code> command as any user to query the encoding format in any directory. If the system displays LANG=en_US.UTF-8, the encoding format is correct. Otherwise, run the <code>vim /etc/default/locale</code> command as the <code>root</code> user to modify LANG=en_US.UTF-8 and reboot the system (using the <code>reboot</code> command) for the settings to take effect.</td>
</tr>
</tbody>
</table>

(Optional) Preparing the MindStudio Installation User

- If an Ascend-CANN-Toolkit installation is available, install MindStudio as the Ascend-CANN-Toolkit installation user.
- If there is no Ascend-CANN-Toolkit installation, prepare an installation user as follows:

  You can install MindStudio as any user (root or non-root).
  - To install MindStudio as the root user, skip this section.
  - To install MindStudio as an existing non-root user, ensure that the user has the read, write, and execute permissions on the $HOME directory.
To install MindStudio as a new non-root user, perform the following steps as the root user. This document uses this scenario as an example to describe how to install MindStudio.

a. Create a MindStudio installation user and set the $HOME directory of the user:

```
useradd -g HwHiAiUser -d /home/username -m username -s /bin/bash
```

**NOTE**

The user must be at the same group as the Driver running user. Otherwise, add the user to the Driver running user group. Assume the user group is HwHiAiUser. You can run the `useradd -g HwHiAiUser -d /home/username -m username -s /bin/bash` command to create a MindStudio installation user and add it to the group.

b. Run the following command to set the user password:

```
passwd username
```

`username` indicates the name of the MindStudio installation user. The umask value of the user cannot be greater than 0027.

- You can view the umask value by running the `umask` command.
- You can change the umask value by running the `umask NewValue` command.

If the value of umask is changed using the preceding method, `NewValue` takes effect only in the current window. To set a permanent value of umask, modify the `~/.bashrc` file as follows:

1) Run the following command in any directory to open the `.bashrc` file:

```
vi ~/.bashrc
```

Append `umask NewValue` to the file.

2) Run the `:wq!` command to save the file and exit.

3) Run the `source ~/.bashrc` command for the modification to take effect immediately.

(Optional) Configuring the Permission of the MindStudio Installation User

Skip the following part if you install the component as the root user.

Before installing MindStudio, you need to download related dependent software. The following commands may be used during the installation. You need to perform the following operations as the root user to escalate privileges for common users to ensure they can use the commands.

**Step 1** Open the `/etc/sudoers` file:

```
chmod u+w /etc/sudoers
vi /etc/sudoers
```

**Step 2** Add the following lines to the file:

```
username ALL=(ALL:ALL) NOPASSWD:SETENV:/usr/bin/apt-get, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/cp, /usr/bin/make install, /usr/bin/pip3.7, /usr/bin/pip3.7.5, /usr/bin/python3.7.5, /bin/ln, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/local/python3.7.5/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/python3.7.5, /usr/local/python3.7.5/bin python3.7.5, /usr/local/python3.7.5/bin/unzip, /usr/sbin/update-alternatives, /usr/bin/add-apt-repository, /usr/local/python3.7.5/bin/pip3
```

Replace `username` with the actual installation user.
NOTE

- If the `/etc/sudoers` file does not contain `#includedir /etc/sudoers.d`, append the line to the file.
- After the MindStudio installation is complete, retaining the preceding privilege escalation configuration may cause risks. Therefore, you need to cancel the sudo permission immediately after the installation is complete. To cancel the permission, delete all the trustlisted commands of the corresponding common user.
- If a message is displayed indicating that the permission is incorrect during the uninstallation, you also need to configure the sudo permission. After the uninstallation is complete, cancel the sudo permission.

Step 3  Run the `:wq!` command to save the file.

Step 4  Remove the write permission on the `/etc/sudoers` file:

```
chmod u-w /etc/sudoers
```

---End

Checking Sources

MindStudio installation requires the download of related dependencies. Ensure that the server where MindStudio is installed can be connected to the network.

Run the following command as the root user to check whether the source is available:

```
apt-get update
```

NOTE

If an error is reported during the command execution, check whether the network connection is normal or replace the source in the `/etc/apt/sources.list` file with a valid one.

Installing Dependencies

For details, see 15.2.2 Ubuntu 18.04-aarch64.

2.4.3.3 EulerOS 2.8-AArch64

Environment Requirements

The minimum hardware and OS requirements for installing MindStudio are as follows:
### Table 2-7 OS mapping information

<table>
<thead>
<tr>
<th>Item</th>
<th>Restriction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>● Recommended memory size: 8 GB (at least 4 GB)</td>
<td>● If the memory size of the Linux host is 4 GB, the size of the model file should be less than or equal to 350 MB for model conversion on MindStudio. Otherwise, the OS may be unstable.</td>
</tr>
<tr>
<td></td>
<td>● Minimum disk space: 6 GB</td>
<td>● If the configurations of the Linux host are upgraded, for example, the memory is expanded to 8 GB, the supported operation specifications should be increased proportionally. If the memory is expanded from 4 GB to 8 GB, keep the model file size within 700 MB.</td>
</tr>
<tr>
<td>OS</td>
<td>EulerOS 2.8 (AArch64)</td>
<td>--</td>
</tr>
<tr>
<td>System</td>
<td>en_US.UTF-8</td>
<td>● Only English version is supported.</td>
</tr>
<tr>
<td>language</td>
<td></td>
<td>● Run the <code>locale</code> command as any user to query the encoding format in any directory. If the system displays LANG=en_US.UTF-8, the encoding format is correct. Otherwise, run the <code>vim /etc/default/locale</code> command as the root user to modify LANG=en_US.UTF-8 and reboot the system (using the <code>reboot</code> command) for the settings to take effect.</td>
</tr>
</tbody>
</table>

(Optimal) Preparing the MindStudio Installation User

- If an Ascend-CANN-Toolkit installation is available, install MindStudio as the Ascend-CANN-Toolkit installation user.
- If there is no Ascend-CANN-Toolkit installation, prepare an installation user as follows:

You can install MindStudio as any user (root or non-root).

- To install MindStudio as the root user, skip this section.
- To install MindStudio as an existing non-root user, ensure that the user has the read, write, and execute permissions on the $HOME directory.
- To install MindStudio as a new non-root user, perform the following steps as the root user. This document uses this scenario as an example to describe how to install MindStudio.

a. Create a MindStudio installation user and set the $HOME directory of the user:

```bash
useradd -g HwHiAiUser -d /home/username -m username -s /bin/bash
```
The user must be at the same group as the Driver running user. Otherwise, add the user to the Driver running user group. Assume the user group is `HwHiAiUser`.

You can run the `useradd -g HwHiAiUser -d /home/username -m username -s /bin/bash` command to create a MindStudio installation user and add it to the group.

b. Run the following command to set the user password:

```
passwd username
```

`username` indicates the name of the MindStudio installation user. The `umask` value of the user cannot be greater than 0027.

- You can view the `umask` value by running the `umask` command.
- You can change the `umask` value by running the `umask NewValue` command.

If the value of `umask` is changed using the preceding method, `NewValue` takes effect only in the current window. To set a permanent value of `umask`, modify the `~/.bashrc` file as follows:

1) Run the following command in any directory to open the `.bashrc` file:

```
vi ~/.bashrc
```

Append `umask NewValue` to the file.

2) Run the `:wq!` command to save the file and exit.

3) Run the `source ~/.bashrc` command for the modification to take effect immediately.

(Required) Configuring the Permission of the MindStudio Installation User

Skip the following part if you install the component as the `root` user.

Before installing MindStudio, you need to download related dependent software. The following commands may be used during the installation. You need to perform the following operations as the `root` user to escalate privileges for common users to ensure they can use the commands.

1. Open the `/etc/sudoers` file:

```
chmod u+w /etc/sudoers
vi /etc/sudoers
```

2. Under `root ALL=(ALL:ALL) ALL` in the file, add the following content:

```
username ALL=(ALL:ALL) NOPASSWD:SETENV:/usr/bin/yum, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/cp, /usr/bin/make install, /usr/bin/pip3, /usr/bin/pip3.7, /usr/bin/python3.7, /bin/ln, /bin/ln -s /usr/local/python3.7.5/bin/python3.7 /usr/local/python3.7.5/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip /usr/local/python3.7.5/bin/pip3 /usr/local/python3.7.5/bin/pip3.7 /usr/local/python3.7.5/bin/pip3.7.5, /usr/bin/unzip, /usr/sbin/update-alternatives, /usr/bin/add-apt-repository, /usr/local/python3.7.5/bin/pip3
```

Replace `username` with the actual installation user.
If the `/etc/sudoers` file does not contain `#includedir /etc/sudoers.d`, append the line to the file.

After the MindStudio installation is complete, retaining the preceding privilege escalation configuration may cause risks. Therefore, you need to cancel the sudo permission immediately after the installation is complete. To cancel the permission, delete all the trustlisted commands of the corresponding common user.

If a message is displayed indicating that the permission is incorrect during the uninstallation, you also need to configure the sudo permission. After the uninstallation is complete, cancel the sudo permission.

3. Run the `:wq!` command to save the file.
4. Remove the write permission on the `/etc/sudoers` file:
   ```bash
   chmod u-w /etc/sudoers
   ```

## Checking Sources

MindStudio installation requires the download of related dependencies. Ensure that the server where MindStudio is installed can be connected to the network.

Run the following command as the `root` user to check whether the source is available:

```bash
yum repolist
```

### NOTE

If an error is reported during the command execution, check the Internet access or replace the sources in the `/etc/yum.repos.d/xxxx.repo` file with valid ones.

## Installing Dependencies

For details, see [15.2.3 EulerOS 2.8-aarch64](#).

### 2.4.3.4 CentOS7.6-x86_64

## Environment Requirements

The minimum hardware and OS requirements for installing MindStudio are as follows:
Table 2-8 OS mapping information

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
</table>
| Hardware      | • Recommended memory size: 8 GB (at least 4 GB)  
                • Minimum drive space: 6 GB                                                              | • If the memory size of the Linux host is 4 GB, the size of the model file should be less than or equal to 350 MB for model conversion on MindStudio. Otherwise, the OS may be unstable.  
                                    • If the configurations of the Linux host are upgraded, for example, the memory is expanded to 8 GB, the supported operation specifications should be increased proportionally.  
                                  If the memory is expanded from 4 GB to 8 GB, keep the model file size within 700 MB. |
| OS            | CentOS 7.6 (x86_64)    | --                                                                                                                                          |
| System language | en_US.UTF-8           | • Only English version is supported.  
                                    • Run the `locale` command as any user to query the encoding format in any directory. If the system displays LANG=en_US.UTF-8, the encoding format is correct. Otherwise, run the `vim /etc/default/locale` command as the root user to modify LANG=en_US.UTF-8 and reboot the system (using the `reboot` command) for the settings to take effect. |

(Optional) Preparing the MindStudio Installation User

- If an Ascend-CANN-Toolkit installation is available, install MindStudio as the Ascend-CANN-Toolkit installation user.
- If there is no Ascend-CANN-Toolkit installation, prepare an installation user as follows:

You can install MindStudio as any user (**root** or non-root).

- To install MindStudio as the **root** user, skip this section.
- To install MindStudio as an existing non-root user, ensure that the user has the read, write, and execute permissions on the `$HOME` directory.
- To install MindStudio as a new non-root user, perform the following steps as the **root** user. This document uses this scenario as an example to describe how to install MindStudio.
  a. Create a MindStudio installation user and set the `$HOME` directory of the user:

```
useradd -g HwHiAiUser -d /home/username -m username -s /bin/bash
```
The user must be at the same group as the Driver running user. Otherwise, add the user to the Driver running user group. Assume the user group is HwHiAiUser. You can run the useradd -g HwHiAiUser -d /home/username -m username -s /bin/bash command to create a MindStudio installation user and add it to the group.

b. Run the following command to set the user password:

```bash
passwd username
```

username indicates the name of the MindStudio installation user. The umask value of the user cannot be greater than 0027.

- You can view the umask value by running the umask command.
- You can change the umask value by running the umask NewValue command.

If the value of umask is changed using the preceding method, NewValue takes effect only in the current window. To set a permanent value of umask, modify the ~/.bashrc file as follows:

1) Run the following command in any directory to open the .bashrc file:

```bash
vi ~/.bashrc
```

Append umask NewValue to the file.

2) Run the :wq! command to save the file and exit.

3) Run the source ~/.bashrc command for the modification to take effect immediately.

(Optional) Configuring the Permission of the MindStudio Installation User

Skip the following part if you install the component as the root user.

Before installing MindStudio, you need to download related dependent software. The following commands may be used during the installation. You need to perform the following operations as the root user to escalate privileges for common users to ensure they can use the commands.

1. Open the /etc/sudoers file:

```bash
chmod u+w /etc/sudoers
vi /etc/sudoers
```

2. Under root ALL=(ALL:ALL) ALL in the file, add the following content:

```bash
username ALL=(ALL:ALL) NOPASSWD:SETENV:/usr/bin/yum, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/cp, /usr/bin/make install, /usr/bin/pip3, /usr/bin/pip3.7, /usr/bin/python3.7, /bin/ln, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/local/python3.7.5/bin/python3.7.5, /usr/local/python3.7.5/bin/pip3, /usr/local/python3.7.5/bin/pip3.7.5, /usr/bin/unzip, /usr/sbin/update-alternatives, /usr/bin/add-apt-repository, /usr/local/python3.7.5/bin/pip3
```

Replace username with the actual installation user.
NOTE

- If the `/etc/sudoers` file does not contain `#includedir /etc/sudoers.d`, append the line to the file.
- After the MindStudio installation is complete, retaining the preceding privilege escalation configuration may cause risks. Therefore, you need to cancel the sudo permission immediately after the installation is complete. To cancel the permission, delete all the trustlisted commands of the corresponding common user.
- If a message is displayed indicating that the permission is incorrect during the uninstallation, you also need to configure the sudo permission. After the uninstallation is complete, cancel the sudo permission.

3. Run the `:wq!` command to save the file.
4. Remove the write permission on the `/etc/sudoers` file:
   ```bash
   chmod u-w /etc/sudoers
   ```

Checking Sources

MindStudio installation requires the download of related dependencies. Ensure that the server where MindStudio is installed can be connected to the network.

Run the following command as the `root` user to check whether the source is available:

```bash
yum repolist
```

NOTE

If an error is reported during the command execution, check the Internet access or replace the sources in the `/etc/yum.repos.d/xxxx.repo` file with valid ones.

Installing Dependencies

For details, see 15.2.4 CentOS7.6-x86_64.

2.4.3.5 CentOS7.6-AArch64

Environment Requirements

The minimum hardware and OS requirements for installing MindStudio are as follows:
Table 2-9 OS mapping information

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Recommended memory size: 8 GB (at least 4 GB)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Minimum drive space: 6 GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● If the memory size of the Linux host is 4 GB, the size of the model file should be less than or equal to 350 MB for model conversion on MindStudio. Otherwise, the OS may be unstable.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● If the configurations of the Linux host are upgraded, for example, the memory is expanded to 8 GB, the supported operation specifications should be increased proportionally. If the memory is expanded from 4 GB to 8 GB, keep the model file size within 700 MB.</td>
<td></td>
</tr>
<tr>
<td>OS</td>
<td>CentOS 7.6 (AArch_64)</td>
<td>--</td>
</tr>
<tr>
<td>System language</td>
<td>en_US.UTF-8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Only English version is supported.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Run the <code>locale</code> command as any user to query the encoding format in any directory. If the system displays LANG=en_US.UTF-8, the encoding format is correct. Otherwise, run the <code>vim /etc/default/locale</code> command as the root user to modify LANG=en_US.UTF-8 and reboot the system (using the <code>reboot</code> command) for the settings to take effect.</td>
<td></td>
</tr>
</tbody>
</table>

(Optional) Preparing the MindStudio Installation User

- If an Ascend-CANN-Toolkit installation is available, install MindStudio as the Ascend-CANN-Toolkit installation user.
- If there is no Ascend-CANN-Toolkit installation, prepare an installation user as follows:

You can install MindStudio as any user (root or non-root).

- To install MindStudio as the root user, skip this section.
- To install MindStudio as an existing non-root user, ensure that the user has the read, write, and execute permissions on the $HOME directory.
- To install MindStudio as a new non-root user, perform the following steps as the root user. This document uses this scenario as an example to describe how to install MindStudio.

a. Create a MindStudio installation user and set the $HOME directory of the user:

```
useradd -g HwHiAiUser -d /home/username -m username -s /bin/bash
```
The user must be at the same group as the Driver running user. Otherwise, add the user to the Driver running user group. Assume the user group is `HwHiAiUser`. You can run the `useradd -g HwHiAiUser -d /home/username -m username -s /bin/bash` command to create a MindStudio installation user and add it to the group.

b. Run the following command to set the user password:

```bash
cd /home/username
passwd username
```

*username* indicates the name of the MindStudio installation user. The *umask* value of the user cannot be greater than `0027`.

- You can view the *umask* value by running the `umask` command.
- You can change the *umask* value by running the `umask NewValue` command.

   If the value of *umask* is changed using the preceding method, *NewValue* takes effect only in the current window. To set a permanent value of *umask*, modify the `~/.bashrc` file as follows:

1) Run the following command in any directory to open the `.bashrc` file:

```bash
vi ~/.bashrc
```

Append `umask NewValue` to the file.

2) Run the `:wq!` command to save the file and exit.

3) Run the `source ~/.bashrc` command for the modification to take effect immediately.

**(Optional) Configuring the Permission of the MindStudio Installation User**

Skip the following part if you install the component as the *root* user.

Before installing MindStudio, you need to download related dependent software. The following commands may be used during the installation. You need to perform the following operations as the *root* user to escalate privileges for common users to ensure they can use the commands.

1. Open the `/etc/sudoers` file:

   ```bash
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```

2. Under *root ALL=(ALL:ALL) ALL* in the file, add the following content:

   ```bash
   username ALL=(ALL:ALL) NOPASSWD:SETENV:/usr/bin/yum, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/cp, /usr/bin/make install, /usr/bin/pip3, /usr/bin/pip3.7, /usr/bin/python3.7, /usr/bin/python3, /bin/ln, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/local/python3.7.5/bin/python3.7, /usr/bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/local/python3.7.5/bin/pip3.7, /usr/bin/unzip, /usr/bin/add-apt-repository, /usr/local/python3.7.5/bin/pip3
   ```

   Replace *username* with the actual installation user.
3. Run the `:wq!` command to save the file.
4. Remove the write permission on the `/etc/sudoers` file:
   
   ```sh
   chmod u-w /etc/sudoers
   ```

### Checking Sources

MindStudio installation requires the download of related dependencies. Ensure that the server where MindStudio is installed can be connected to the network.

Run the following command as the root user to check whether the source is available:

```sh
yum repolist
```

- **NOTE**

  If an error is reported during the command execution, check the Internet access or replace the sources in the `/etc/yum.repos.d/xxxx.repo` file with valid ones.

### Installing Dependencies

For details, see 15.2.5 CentOS7.6-AArch64.

#### 2.4.3.6 Kylin OS V10 SP1-AArch64

### Environment Requirements

The minimum hardware and OS requirements for installing MindStudio are as follows:
Table 2-10 OS mapping information

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td></td>
<td>• Recommended memory size: 8 GB (at least 4 GB)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Minimum disk space: 6 GB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If the memory size of the Linux host is 4 GB, the size of the model file should be less than or equal to 350 MB for model conversion on MindStudio. Otherwise, the OS may be unstable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If the configurations of the Linux host are upgraded, for example, the memory is expanded to 8 GB, the supported operation specifications should be increased proportionally.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the memory is expanded from 4 GB to 8 GB, keep the model file size within 700 MB.</td>
</tr>
<tr>
<td>OS</td>
<td>Kylin V10</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>(AArch_64)</td>
<td></td>
</tr>
<tr>
<td>System language</td>
<td>en_US.UTF-8</td>
<td>• Only English version is supported.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Run the <code>locale</code> command as any user to query the encoding format in any directory. If the system displays LANG=en_US.UTF-8, the encoding format is correct. Otherwise, run the <code>vim /etc/default/locale</code> command as the root user to modify LANG=en_US.UTF-8 and reboot the system (using the <code>reboot</code> command) for the settings to take effect.</td>
</tr>
</tbody>
</table>

(Optional) Preparing the MindStudio Installation User

- If an Ascend-CANN-Toolkit installation is available, install MindStudio as the Ascend-CANN-Toolkit installation user.
- If there is no Ascend-CANN-Toolkit installation, prepare an installation user as follows:

You can install MindStudio as any user (`root` or non-root).

- To install MindStudio as the `root` user, skip this section.
- To install MindStudio as an existing non-root user, ensure that the user has the read, write, and execute permissions on the `$HOME` directory.
- To install MindStudio as a new non-root user, perform the following steps as the `root` user. This document uses this scenario as an example to describe how to install MindStudio.

  a. Create a MindStudio installation user and set the `$HOME` directory of the user:

```bash
useradd -g HwHiAiUser -d /home/username -m username -s /bin/bash
```
The user must be at the same group as the Driver running user. Otherwise, add the user to the Driver running user group. Assume the user group is HwHiAiUser. You can run the `useradd -g HwHiAiUser -d /home/username -m username -s /bin/bash` command to create a MindStudio installation user and add it to the group.

b. Run the following command to set the user password:

```
passwd username
```

*username* indicates the name of the MindStudio installation user. The *umask* value of the user cannot be greater than **0027**.

- You can view the *umask* value by running the `umask` command.
- You can change the *umask* value by running the `umask NewValue` command.

If the value of *umask* is changed using the preceding method, *NewValue* takes effect only in the current window. To set a permanent value of *umask*, modify the `~/.bashrc` file as follows:

1. Run the following command in any directory to open the `.bashrc` file:

   ```
   vi ~/.bashrc
   ```

2. Append `umask NewValue` to the file.

3. Run the `:wq!` command to save the file and exit.

4. Run the `source ~/.bashrc` command for the modification to take effect immediately.

(Optional) Configuring the Permission of the MindStudio Installation User

Skip the following part if you install the component as the root user.

Before installing MindStudio, you need to download related dependent software. The following commands may be used during the installation. You need to perform the following operations as the root user to escalate privileges for common users to ensure they can use the commands.

1. Open the `/etc/sudoers` file:

   ```
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```

2. Under `root ALL=(ALL:ALL) ALL` in the file, add the following content:

   ```
   username ALL=(ALL:ALL) NOPASSWD:SETENV:/usr/bin/yum, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/cp, /usr/bin/make install, /usr/bin/pip3, /usr/bin/pip3.7, /usr/bin/python3.7.5, /usr/bin/pip3.7, /usr/local/python3.7.5/bin/python3.7.5 /usr/local/python3.7.5/bin/python3.7.5 /usr/bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/local/python3.7.5/bin/pip3 /usr/local/python3.7.5/bin/pip3.7 /usr/local/python3.7.5/bin/pip3.7.5 /usr/local/python3.7.5/bin/python3.7.5 /usr/local/python3.7.5/bin/unzip, /usr/sbin/update-alternatives, /usr/bin/add-apt-repository, /usr/local/python3.7.5/bin/pip3
   ```

Replace *username* with the actual installation user.
● If the `/etc/sudoers` file does not contain `#includedir /etc/sudoers.d`, append the line to the file.

● After the MindStudio installation is complete, retaining the preceding privilege escalation configuration may cause risks. Therefore, you need to cancel the sudo permission immediately after the installation is complete. To cancel the permission, delete all the trustlisted commands of the corresponding common user.

● If a message is displayed indicating that the permission is incorrect during the uninstallation, you also need to configure the sudo permission. After the uninstallation is complete, cancel the sudo permission.

3. Run the `:wq!` command to save the file.

4. Remove the write permission on the `/etc/sudoers` file:
   ```
   chmod u-w /etc/sudoers
   ```

### Checking Sources

MindStudio installation requires the download of related dependencies. Ensure that the server where MindStudio is installed can be connected to the network.

Run the following command as the root user to check whether the source is available:
```
yum repolist
```

**NOTE**

If an error is reported during the command execution, check the Internet access or replace the sources in the `/etc/yum.repos.d/xxxx.repo` file with valid ones.

### Installing Dependencies

For details, see [15.2.6 Kylin OS V10 SP1-aarch64](#).

### 2.4.3.7 NeoKylin OS 7.6-AArch64

### Environment Requirements

The minimum hardware and OS requirements for installing MindStudio are as follows:
Table 2-11 OS mapping information

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
</table>
| Hardware   |                          | • Recommended memory size: 8 GB (at least 4 GB)  
                           | • Minimum disk space: 6 GB  
                           | • If the memory size of the Linux host is 4 GB, the size of the model file should be less than or equal to 350 MB for model conversion on MindStudio. Otherwise, the OS may be unstable.  
                           | • If the configurations of the Linux host are upgraded, for example, the memory is expanded to 8 GB, the supported operation specifications should be increased proportionally.  
                           | • If the memory is expanded from 4 GB to 8 GB, keep the model file size within 700 MB.                                                   |
| OS         | NeoKylin 7.6 (AArch_64)  | --                                                                                                                                            |
| System     | en_US.UTF-8              | • Only English version is supported.  
                           | language    |                          | • Run the locale command as any user to query the encoding format in any directory. If the system displays LANG=en_US.UTF-8, the encoding format is correct. Otherwise, run the vim /etc/default/locale command as the root user to modify LANG=en_US.UTF-8 and reboot the system (using the reboot command) for the settings to take effect. |

(Optional) Preparing the MindStudio Installation User

- If an Ascend-CANN-Toolkit installation is available, install MindStudio as the Ascend-CANN-Toolkit installation user.
- If there is no Ascend-CANN-Toolkit installation, prepare an installation user as follows:

You can install MindStudio as any user (root or non-root).

- To install MindStudio as the root user, skip this section.
- To install MindStudio as an existing non-root user, ensure that the user has the read, write, and execute permissions on the $HOME directory.
- To install MindStudio as a new non-root user, perform the following steps as the root user. This document uses this scenario as an example to describe how to install MindStudio.

  a. Create a MindStudio installation user and set the $HOME directory of the user:

```bash
useradd -g HwHiAiUser -d /home/username -m username -s /bin/bash
```
NOTE

The user must be at the same group as the Driver running user. Otherwise, add the user to the Driver running user group. Assume the user group is HwHiAiUser.

You can run the `useradd -g HwHiAiUser -d /home/username -m username -s /bin/bash` command to create a MindStudio installation user and add it to the group.

b. Run the following command to set the user password:

```
passwd username
```

`username` indicates the name of the MindStudio installation user. The `umask` value of the user cannot be greater than 0027.

- You can view the `umask` value by running the `umask` command.
- You can change the `umask` value by running the `umask NewValue` command.

If the value of `umask` is changed using the preceding method, `NewValue` takes effect only in the current window. To set a permanent value of `umask`, modify the `~/.bashrc` file as follows:

1. Run the following command in any directory to open the `.bashrc` file:

```
vi ~/.bashrc
```

Append `umask NewValue` to the file.

2. Run the `:wq!` command to save the file and exit.

3. Run the `source ~/.bashrc` command for the modification to take effect immediately.

(Optional) Configuring the Permission of the MindStudio Installation User

Skip the following part if you install the component as the root user.

Before installing MindStudio, you need to download related dependent software. The following commands may be used during the installation. You need to perform the following operations as the root user to escalate privileges for common users to ensure they can use the commands.

1. Open the `/etc/sudoers` file:

```
chmod u+w /etc/sudoers
vi /etc/sudoers
```

2. Under `root ALL=(ALL:ALL) ALL` in the file, add the following content:

```
username ALL=(ALL:ALL) NOPASSWD:SETENV:/usr/bin/yum, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/cp, /usr/bin/make install, /usr/bin/pip3, /usr/bin/pip3.7, /usr/bin/python3.7, /usr/bin/python3.7.5, /bin/ln, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/local/python3.7.5/bin/python3.7, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/local/python3.7.5/bin/pip3.7.5, /usr/bin/unzip, /usr/sbin/update-alternatives, /usr/bin/add-apt-repository, /usr/local/python3.7.5/bin/pip3
```

Replace `username` with the actual installation user.
3. Run the `:wq!` command to save the file.
4. Remove the write permission on the `/etc/sudoers` file:
   
   ```bash
   chmod u-w /etc/sudoers
   ```

### Checking Sources

MindStudio installation requires the download of related dependencies. Ensure that the server where MindStudio is installed can be connected to the network.

Run the following command as the root user to check whether the source is available:

```bash
yum repolist
```

**NOTE**

If an error is reported during the command execution, check the Internet access or replace the sources in the `/etc/yum.repos.d/xxxx.repo` file with valid ones.

### Installing Dependencies

For details, see [15.2.7 Installing Dependencies (NeoKylin OS 7.6-AArch64)](#).

### 2.4.4 Installing MindStudio

**NOTICE**

- Only one MindStudio installation is allowed on a single server. Before installing MindStudio, uninstall any previous MindStudio installation by referring to [13.5 Uninstalling MindStudio](#).
- Log in to the OS as the MindStudio installation user. It is not supported to switch to the MindStudio installation user after login as a different user.

### Prerequisites

[2.4.2 Downloading Required Files](#) and [2.4.3 Pre-installation Actions](#) have been completed.

### Procedure

**Step 1** Upload the MindStudio package as the MindStudio installation user.
• If there is an Ascend-CANN-Toolkit installation, upload the MindStudio package to the MindStudio installation server.
• If there is no Ascend-CANN-Toolkit installation, upload the MindStudio package and Ascend-CANN-Toolkit runfile to the MindStudio installation server.

**Step 2** Extract the MindStudio tar.gz package.

Run the following command as the MindStudio installation user to extract the MindStudio_{version}_linux.tar.gz package:

```
tar -zxvf MindStudio_{version}_linux.tar.gz
```

For details about the content extracted from the package, see Table 15-1.

**Step 3** As the MindStudio installation user, run the following command in the MindStudio/bin directory generated after package extraction:

```
cd MindStudio/bin
./MindStudio.sh
```

**NOTE**

If the error message “Failed to initialize graphics environment” or “Unable to detect graphics environment” is displayed when MindStudio is launched, rectify the fault by following the instructions in 15.8.4 What Do I Do If the GUI Cannot Be Displayed When MindStudio Is Started?

When MindStudio is launched for the first time, it will prompt you to install the necessary third-party dependencies. You are advised not to launch MindStudio in the background. The installation script automatically checks whether the path is valid.

1. If the path is valid, the dependencies are checked. If all the dependencies are installed, go to Step 4. Otherwise, install any missing dependency as prompted. After all dependencies are installed, run the installation script again.

2. If messages similar to the following are displayed:

   There is no python3.7.5 in path /usr/local/python3.7.5. Please enter your choice:
   1. choose local python3.7.5
   2. install python3.7.5

   a. Enter 1. The system prompts "Enter local python path:". Enter the local path where Python 3.7.5 has been installed. The script checks whether the path is valid.
      i. If the path is valid, the dependencies are checked. If all the dependencies are installed, go to Step 4. Otherwise, install any missing dependency as prompted.
      ii. If the path check fails, return to the message prompt. Check whether the entered path is correct or enter 2 to go to **b**.

   b. Enter 2 and enter a new installation path to install Python 3.7.5 as prompted. You can also retain the default installation path /usr/local/python3.7.5.

      Enter python install path. Default value is /usr/local/python3.7.5:
      /home/hisisoc/software/python3.7.5

      Install Python 3.7.5 and related dependencies as prompted. After the installation is complete, run the installation script again.
Step 4 Open the Import MindStudio Settings dialog box, as shown in Figure 2-18.

Figure 2-18 Environment settings

- **Config or installation folder**: imports MindStudio settings from an existing directory. The directory is $HOME/.config/Huawei/MindStudioMS-{version}. The MindStudio settings include the personalized settings of the project UI such as the background color.
- **Do not import settings** (default): creates a configuration file.

Select an option and click OK.

**NOTE**
- If an earlier MindStudio installation is detected, the Previous version parameter is displayed in the dialog box shown in Figure 2-18 when you are installing a later version. You can choose whether to import the settings from the earlier version as required.
- If this version of MindStudio has been installed on the server earlier, and the SHOME/.cache/Huawei/MindStudioMS-{version} and SHOME/.config/Huawei/MindStudioMS-{version} directories have not been deleted during the uninstallation, you will be directed to Step 10 from Step 3 when you try to install this version again.

Step 5 If the dialog box shown in Figure 2-19 is displayed:

**NOTE**
This dialog box is also displayed when you use MindStudio. You can determine whether to click Accept based on the site requirements. If you choose File > Settings... > Tools > Server Certificates on the menu bar and select Accept non-trusted certificates automatically, the certificates will be automatically accepted and this dialog box will not be displayed when you use MindStudio.
Click **Accept** to allow MindStudio to connect to the upgrade server.

Click **Reject** to reject MindStudio to connect to the upgrade server. This dialog box is displayed when you restart MindStudio next time.

**Step 6** The window shown in **Figure 2-20** is displayed.

**NOTE**

- If the page is not completely displayed, rectify the fault by following the instructions in Chapter 15.8.7 on how to solve the problem that Chinese characters are displayed as garbled characters and the GUI is incompletely displayed or is not well-organized.
- When selecting the installation path for Ascend-CANN-Toolkit, do not select the soft link path that is automatically created by the system when the development kit is installed.
If there is an Ascend-CANN-Toolkit installation: Click next to Ascend Toolkit Path. In the dialog box that is displayed, select the Ascend-CANN-Toolkit installation path (including the version number). Click OK. The Ascend-CANN-Toolkit version information is displayed in the Ascend Toolkit Version text box, which is not configurable. See Figure 2-21. Click Finish, moving on to Step 9.

Figure 2-21 Selecting the Ascend-CANN-Toolkit installation path
- If there is no Ascend-CANN-Toolkit installation: Click next to Ascend Toolkit Path. In the dialog box that is displayed, select the Ascend-CANN-Toolkit runfile uploaded in 2.4.3 Pre-installation Actions and select its path, as shown in Figure 2-22. Click Next, moving on to Step 7.

**Figure 2-22** Selecting the Ascend-CANN-Toolkit path

![Image of Toolkit Version Setting](image)

**Step 7** Verify the current settings.

**Figure 2-23** Current settings

![Image of Verify Settings](image)
If you have installed an Ascend-CANN-Toolkit of the same version in the specified Install Location, a message is displayed, indicating that a previous installation is found.

**Step 8**  Click Next, as shown in Figure 2-24.

![Figure 2-24 Installing Ascend Toolkit](image)

Click Show Details to check the Ascend-CANN-Toolkit installation details or click Hide Details to hide the details.

If the following message is displayed, the installation is successful. In this case, the Finish button becomes available and other buttons are unavailable.

Installation succeeded. Please click "Finish".

Click Finish. A dialog box is displayed, prompting you to restart MindStudio, as shown in Figure 2-25.

![Figure 2-25 Restarting MindStudio](image)

Click OK to restart MindStudio.

**Step 9**  The User Experience Improvement Program page is displayed. See Figure 2-26. You can set Name and Organization as required and click the User Experience Improvement Program link to read the details. Selecting the check box indicates that you have read the content.
Figure 2-26 User experience improvement program

- Click **Agree**. MindStudio will collect user information for the experience improvement program.
- Click **Disagree**. The page will be closed, and MindStudio will not collect user information for the experience improvement program.

**Step 10** If no error is reported and the welcome window is displayed, MindStudio is successfully installed.

Find the following new directories in $HOME on the MindStudio installation server:

- **$HOME/.mindstudio**: stores the MindStudio property file `mindstudio.properties` (recording the Toolkit installation path) and records of operations performed by users on the privacy statement window.
- **$HOME/.cache/Huawei/MindStudioMS-{version}**: stores runtime logs (`log/idea.log`) generated after MindStudio is installed and launched.
- **$HOME/.config/Huawei/MindStudioMS-{version}**: stores user configuration, including personalized settings (such as the background color) of the project UI.

The parameters in the welcome window are described as follows:

- **Create new project**: creates a project and save it in $HOME/AscendProjects.
- **Open or Import**: opens or imports a project.
- **Welcome System Profiling**: opens the System Profiling window.
Step 11  Launch MindStudio and choose File > Settings > Plugins from the main menu to install the following plugins:

- Grep Console
- Pylint
- Python Community Edition

---End

2.4.5 Configuring the Compilation Environment

After MindStudio is installed and the project window is displayed, you need to configure the compilation environment before getting started to develop operators or projects.

Prerequisites

- The version of the Ascend-CANN-Toolkit on the MindStudio installation server must be the same as that of the CANN software package in the operating environment.
- The Ascend-CANN-Toolkit must be installed on the MindStudio installation server, with the corresponding CANN software package in the operating environment.

Installing the Compiler

Install different compilers based on the product form, and the OS and architecture of the operating environment.

⚠️ NOTE

After the compilation environment is configured, restart MindStudio for the configuration to take effect.

- Form Ascend EP
## Table 2-12: Compiler commands

<table>
<thead>
<tr>
<th>MindStudio Installation Server OS and Architecture</th>
<th>OS and Architecture of the Operating Environment</th>
<th>Compiler</th>
</tr>
</thead>
</table>
| Ubuntu 18.04+x86                                 | CentOS 7.6 (x86)                                | Run the `g++ --version` command on the MindStudio installation server as the MindStudio installation user to check whether the compiler has been installed. If not, run the following installation command to install the compiler:  
  ```
sudo apt-get install -y g++
  ```  
  If there are multiple cross compilers on the MindStudio installation server, create a soft link between the used cross compiler and the required cross compiler version. The command is as follows:  
  ```
sudo ln -s Cross compiler of another version Required version
  ```  |
| CentOS 7.6                                       | EulerOS 2.8 +aarch64                            | Run the `aarch64-linux-gnu-g++ --version` command on the MindStudio installation server as the MindStudio installation user to check whether the compiler has been installed. If not, run the following command to install the compiler:  
  ```
sudo apt-get install -y g++-aarch64-linux-gnu
  ```  
  If there are multiple cross compilers on the MindStudio installation server, create a soft link between the used cross compiler and the required cross compiler version. The command is as follows:  
  ```
sudo ln -s Cross compiler of another version Required version
  ```  |
| Centos 7.6                                       | Ubuntu 18.04+x86                                | Run the `gcc --version` command on the MindStudio installation server as the MindStudio installation user to check whether the compiler has been installed. If not, install the compiler as follows:  
  1. Install `gcc-7.3.0`. For details, see [Installing GCC 7.3.0](#).  
  2. Configure environment variables.  
     ```
     export LD_LIBRARY_PATH=${install_path}/lib64:$LD_LIBRARY_PATH
     export PATH=${install_path}/bin:$PATH
     ```

<table>
<thead>
<tr>
<th>MindStudio Installation Server OS and Architecture</th>
<th>Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubuntu18.04/CentOS7.6/EulerOS2.8+aarch64</td>
<td>Run the <strong>aarch64-linux-gnu-gcc --version</strong> command on the MindStudio installation server as the MindStudio installation user to check whether the compiler has been installed. If not, install the compiler as follows:</td>
</tr>
<tr>
<td></td>
<td>1. Obtain the installation package.</td>
</tr>
<tr>
<td></td>
<td>2. Decompress the installation package.</td>
</tr>
<tr>
<td></td>
<td><code>tar -xvJf gcc-linaro-7.5.0-2019.12-x86_64_aarch64-linux-gnu.tar.xz</code></td>
</tr>
<tr>
<td></td>
<td>3. Set the environment variable.</td>
</tr>
<tr>
<td></td>
<td><code>vi ~/.bashrc</code></td>
</tr>
<tr>
<td></td>
<td><code>export PATH=$HOME/gcc-linaro-7.5.0-2019.12-x86_64_aarch64-linux-gnu/bin:$PATH</code></td>
</tr>
<tr>
<td></td>
<td>4. Press <strong>Esc</strong> and run the :wq! command to save the configurations and exit.</td>
</tr>
<tr>
<td></td>
<td>5. Run the <strong>source ~/.bashrc</strong> command for the modifications to take effect.</td>
</tr>
<tr>
<td></td>
<td>If there are multiple cross compilers on the MindStudio installation server, create a soft link between the used cross compiler and the required cross compiler version. The command is as follows:</td>
</tr>
<tr>
<td></td>
<td><code>sudo ln -s Cross compiler of another version Required version</code></td>
</tr>
</tbody>
</table>

- Form Ascend RC
### Table 2-13 Compiler commands

<table>
<thead>
<tr>
<th>MindStudio Installation Server OS and Architecture</th>
<th>Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubuntu 18.04+aarch64</td>
<td>Run the <code>aarch64-linux-gnu-gcc --version</code> command on the MindStudio installation server as the MindStudio installation user to check whether the compiler has been installed. If not, run the following command to install the compiler: <code>sudo apt-get install g++-aarch64-linux-gnu</code></td>
</tr>
</tbody>
</table>
| CentOS 7.6+aarch64 | Run the `aarch64-linux-gnu-g++ --version` command on the MindStudio installation server as the MindStudio installation user to check whether the compiler has been installed. If not, install the compiler as follows:  
1. Obtain the installation package.  
2. Decompress the installation package.  
   `tar -zxvf gcc-linaro-7.5.0-2019.12-x86_64_aarch64-linux-gnu.tar.xz`  
3. Set the environment variables.  
   `sudo vi /etc/profile`  
   `export PATH=/root/gcc-linaro-7.5.0-2019.12-x86_64_aarch64-linux-gnu/bin:$PATH`  
4. Press Esc and run the `:wq!` command to save the configurations and exit.  
5. Run the `source /etc/profile` command.  
   If there are multiple cross compilers on the MindStudio installation server, create a soft link between the used cross compiler and the required cross compiler version. The command is as follows: `sudo ln -s Cross compiler of another version Required version` |

## 2.5 Installing MindStudio on Linux (Container-based)
2.5.1 Installation Workflow

![MindStudio Installation workflow](image)

Figure 2-27 MindStudio Installation workflow

2.5.2 Preparing for Installation

- Install Docker on the host. You can run the `docker --version` command to check whether Docker has been installed.
- Add the installation user (for example, `HwHiAiUser`) to the `docker` group. You can run the `usermod -a -G docker HwHiAiUser` command to add the `HwHiAiUser` user to the `docker` group. In some environments, you need to run the `chmod 666 /var/run/docker.sock` command to ensure that the installation user has the permission to operate Docker.
- Connect the host to the Internet by referring to 15.6 Configuring a System Network Proxy for downloading the required image.
- Obtain the files listed in Table 2-14 and copy them to any directory on the host.

Table 2-14 Required files

<table>
<thead>
<tr>
<th>Scenario</th>
<th>File</th>
<th>Description</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>MindStudio inference container</td>
<td>MindStudio_{version}_linux.tar.gz</td>
<td>MindStudio package</td>
<td>Link</td>
</tr>
<tr>
<td>based on CentOS 7.8 AArch64</td>
<td>Ascend-cann-toolkit_{version}_linux-{arch}.run</td>
<td>Ascend-CANN-Toolkit runfile</td>
<td>Link</td>
</tr>
</tbody>
</table>
### Scenario File Description How to Obtain

<table>
<thead>
<tr>
<th>Scenario</th>
<th>File</th>
<th>Description</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>matplotlib-3.3.2.tar.gz</td>
<td>Matplotlib, Python's drawing library. Use Matplotlib with NumPy to provide an open-source alternative to MATLAB.</td>
<td>Link</td>
</tr>
<tr>
<td></td>
<td>freetype-2.6.1.tar.gz</td>
<td>FreeType, an open-source and portable font engine that provides a unified interface for accessing multiple font files.</td>
<td>Link</td>
</tr>
<tr>
<td></td>
<td>ascend_install.info</td>
<td>Installation log file</td>
<td>Required only in the scenario where an AI CPU operator is used for inference and commissioning. Default path: <code>/etc/ascend_install.info</code> on the host</td>
</tr>
<tr>
<td></td>
<td>version.info</td>
<td>Driver version file</td>
<td>Required in the scenario where an AI CPU operator is used for inference and commissioning. Default path: <code>/usr/local/Ascend/driver/version.info</code> on the host</td>
</tr>
<tr>
<td>MindStudio inference container based on Euler 2.8 AArch64</td>
<td>MindStudio_{version}_linux.tar.gz</td>
<td>MindStudio package</td>
<td>Link</td>
</tr>
<tr>
<td></td>
<td>Ascend-cann-toolkit_{version}_linux-{arch}.run</td>
<td>Ascend-CANN-Toolkit runfile</td>
<td>Link</td>
</tr>
<tr>
<td>Scenario</td>
<td>File</td>
<td>Description</td>
<td>How to Obtain</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>-------------</td>
<td>---------------</td>
</tr>
<tr>
<td>matplotlib-3.3.3.tar.gz</td>
<td>Matplotlib, Python's drawing library. Use Matplotlib with NumPy to provide an open-source alternative to MATLAB.</td>
<td>Link</td>
<td></td>
</tr>
<tr>
<td>ascend_install.info</td>
<td>Installation log file</td>
<td>Required only in the scenario where an AI CPU operator is used for inference and commissioning. Default path: <code>/etc/ascend_install.info</code> on the host</td>
<td></td>
</tr>
<tr>
<td>version.info</td>
<td>Driver version file</td>
<td>Required in the scenario where an AI CPU operator is used for inference and commissioning. Default path: <code>/usr/local/Ascend/driver/version.info</code> on the host</td>
<td></td>
</tr>
<tr>
<td>MindStudio inference container based on Ubuntu18.04 AArch64</td>
<td>MindStudio_{version}_linux.tar.gz</td>
<td>MindStudio package</td>
<td>Link</td>
</tr>
<tr>
<td>Ascend-cann-toolkit_{version}_linux-{arch}.run</td>
<td>Ascend-CANN-Toolkit runfile</td>
<td>Link</td>
<td></td>
</tr>
<tr>
<td>matplotlib-3.3.3.tar.gz</td>
<td>Matplotlib, Python's drawing library. Use Matplotlib with NumPy to provide an open-source alternative to MATLAB.</td>
<td>Link</td>
<td></td>
</tr>
<tr>
<td>Scenario</td>
<td>File</td>
<td>Description</td>
<td>How to Obtain</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------</td>
<td>-------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>ascend_install.info</td>
<td>Installation log file</td>
<td>Required in the scenario where an AI CPU operator is used for inference and commissioning. Default path: <code>/etc/ascend_install.info</code> on the host</td>
</tr>
<tr>
<td></td>
<td>version.info</td>
<td>Driver version file</td>
<td>Required in the scenario where an AI CPU operator is used for inference and commissioning. Default path: <code>/usr/local/Ascend/driver/version.info</code> on the host</td>
</tr>
<tr>
<td></td>
<td>MindStudio_{version}_linux.tar.gz</td>
<td>MindStudio package</td>
<td>Link</td>
</tr>
<tr>
<td></td>
<td>Ascend-cann-toolkit_{version}_linux-{arch}.run</td>
<td>Ascend-CANN-Toolkit runfile</td>
<td>Link</td>
</tr>
<tr>
<td>Scenario</td>
<td>File</td>
<td>Description</td>
<td>How to Obtain</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>-------------</td>
<td>---------------</td>
</tr>
<tr>
<td></td>
<td>ascend_install.info</td>
<td>Installation log file</td>
<td>Required only in the scenario where an AI CPU operator is used for inference and commissioning. Default path: /etc/ascend_install.info on the host</td>
</tr>
<tr>
<td></td>
<td>version.info</td>
<td>Driver version file</td>
<td>Required in the scenario where an AI CPU operator is used for inference and commissioning. Default path: /usr/local/Ascend/driver/version.info on the host</td>
</tr>
<tr>
<td>MindStudio inference container based on Ubuntu18.04 x86_64</td>
<td>MindStudio_{version}_linux.tar.gz</td>
<td>MindStudio package</td>
<td>Link</td>
</tr>
<tr>
<td></td>
<td>Ascend-cann-toolkit_{version}_linux-{arch}.run</td>
<td>Ascend-CANN-Toolkit runfile</td>
<td>Link</td>
</tr>
<tr>
<td>Scenario</td>
<td>File</td>
<td>Description</td>
<td>How to Obtain</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>ascend_install.info</td>
<td>Installation log file</td>
<td>Required only in the scenario where an AI CPU operator is used for inference and commissioning. Default path: <code>/etc/ascend_install.info</code>.</td>
</tr>
<tr>
<td></td>
<td>version.info</td>
<td>Driver version file</td>
<td>Required in the scenario where an AI CPU operator is used for inference and commissioning. Default path: <code>/usr/local/Ascend/driver/version.info</code>.</td>
</tr>
<tr>
<td>MindStudio training container based on CentOS 7.8 AArch64</td>
<td>MindStudio_{version}_linux.tar.gz</td>
<td>MindStudio package</td>
<td><a href="#">Link</a></td>
</tr>
<tr>
<td></td>
<td>Ascend-cann-toolkit_{version}_linux-{arch}.run</td>
<td>Ascend-CANN-Toolkit runfile</td>
<td><a href="#">Link</a></td>
</tr>
<tr>
<td></td>
<td>matplotlib-3.3.2.tar.gz</td>
<td>Matplotlib, Python's drawing library. Use Matplotlib with NumPy to provide an open-source alternative to MATLAB.</td>
<td><a href="#">Link</a></td>
</tr>
<tr>
<td></td>
<td>freetype-2.6.1.tar.gz</td>
<td>FreeType, an open-source and portable font engine that provides a unified interface for accessing multiple font files.</td>
<td><a href="#">Link</a></td>
</tr>
<tr>
<td>Scenario</td>
<td>File</td>
<td>Description</td>
<td>How to Obtain</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>MindInsight file</td>
<td>mindinsight-{version}-cp37-cp37m-lin...</td>
<td>MindInsight wheel file</td>
<td>Link</td>
</tr>
<tr>
<td>GNU Multiple Precision Arithmetic Library</td>
<td>gmp-6.1.2.tar.xz</td>
<td>GMP is a free library for arbitrary-precision arithmetic, operating on signed integers, rational numbers, and floating-point numbers. There is no practical limit to the precision except the ones implied by the available memory in the machine GMP runs on. GMP has a rich set of functions, and the functions have a regular interface.</td>
<td>Link</td>
</tr>
<tr>
<td>Installation log file</td>
<td>ascend_install.info</td>
<td>Installation log file</td>
<td></td>
</tr>
<tr>
<td>Driver version file</td>
<td>version.info</td>
<td>Driver version file</td>
<td></td>
</tr>
<tr>
<td>MindStudio package</td>
<td>MindStudio_{version}_linux.tar.gz</td>
<td>MindStudio package</td>
<td>Link</td>
</tr>
<tr>
<td>Ascend-CANN-Toolkit runfile</td>
<td>Ascend-cann-toolkit_{version}_linux-{arch}.run</td>
<td>Ascend-CANN-Toolkit runfile</td>
<td>Link</td>
</tr>
<tr>
<td>MindSpore wheel file</td>
<td>mindspore_ascend-{version}-cp37-cp37m-lin...</td>
<td>MindSpore wheel file</td>
<td>Link</td>
</tr>
<tr>
<td>MindInsight wheel file</td>
<td>mindinsight-{version}-cp37-cp37m-lin...</td>
<td>MindInsight wheel file</td>
<td>Link</td>
</tr>
<tr>
<td>Scenario</td>
<td>File</td>
<td>Description</td>
<td>How to Obtain</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Scenario</td>
<td>matplotlib-3.3.3.tar.gz</td>
<td>Matplotlib, Python's drawing library. Use Matplotlib with NumPy to provide an open-source alternative to MATLAB.</td>
<td>Link</td>
</tr>
<tr>
<td>GMP</td>
<td>gmp-6.1.2.tar.xz</td>
<td>GMP is a free library for arbitrary-precision arithmetic, operating on signed integers, rational numbers, and floating-point numbers. There is no practical limit to the precision except the ones implied by the available memory in the machine GMP runs on. GMP has a rich set of functions, and the functions have a regular interface.</td>
<td>Link</td>
</tr>
<tr>
<td>MindStudio training container based on Ubuntu 18.04 AArch64</td>
<td>ascend_install.info</td>
<td>Installation log file</td>
<td></td>
</tr>
<tr>
<td>MindStudio package</td>
<td>version.info</td>
<td>Driver version file</td>
<td>Default path: /etc/ascend_install.info on the host</td>
</tr>
<tr>
<td>MindStudio package</td>
<td>MindStudio_{version}_linux.tar.gz</td>
<td>MindStudio package</td>
<td>Link</td>
</tr>
<tr>
<td>Ascend-CANN-Toolkit runfile</td>
<td>Ascend-cann-toolkit_{version}_linux-{arch}.run</td>
<td>Ascend-CANN-Toolkit runfile</td>
<td>Link</td>
</tr>
<tr>
<td>MindSpore wheel file</td>
<td>mindspore_ascend-{version}-cp37-cp37m-linux-{arch}.whl</td>
<td>MindSpore wheel file</td>
<td>Link</td>
</tr>
<tr>
<td>MindInsight wheel file</td>
<td>mindinsight-{version}-cp37-cp37m-linux-{arch}.whl</td>
<td>MindInsight wheel file</td>
<td>Link</td>
</tr>
<tr>
<td>Scenario</td>
<td>File</td>
<td>Description</td>
<td>How to Obtain</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>-------------</td>
<td>---------------</td>
</tr>
<tr>
<td></td>
<td>matplotlib-3.3.3.tar.gz</td>
<td>Matplotlib, Python’s drawing library. Use Matplotlib with NumPy to provide an open-source alternative to MATLAB.</td>
<td>Link</td>
</tr>
<tr>
<td></td>
<td>gmp-6.1.2.tar.xz</td>
<td>GMP is a free library for arbitrary-precision arithmetic, operating on signed integers, rational numbers, and floating-point numbers. There is no practical limit to the precision except the ones implied by the available memory in the machine GMP runs on. GMP has a rich set of functions, and the functions have a regular interface.</td>
<td>Link</td>
</tr>
<tr>
<td></td>
<td>ascend_install.info</td>
<td>Installation log file</td>
<td>Default path: <code>/etc/ascend_install.info</code> on the host</td>
</tr>
<tr>
<td></td>
<td>version.info</td>
<td>Driver version file</td>
<td>Default path: <code>/usr/local/Ascend/driver/version.info</code> on the host</td>
</tr>
<tr>
<td></td>
<td>MindStudio_{version}_linux.tar.gz</td>
<td>MindStudio package</td>
<td>Link</td>
</tr>
<tr>
<td></td>
<td>Ascend-cann-toolkit_{version}_linux-{arch}.run</td>
<td>Ascend-CANN-Toolkit runfile</td>
<td>Link</td>
</tr>
<tr>
<td></td>
<td>mindinsight-{version}-cp37-cp37m-linu...</td>
<td>MindInsight wheel file</td>
<td>Link</td>
</tr>
</tbody>
</table>
### Scenario

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>gmp-6.1.2.tar.xz</td>
<td>GMP is a free library for arbitrary-precision arithmetic, operating on signed integers, rational numbers, and floating-point numbers. There is no practical limit to the precision except the ones implied by the available memory in the machine GMP runs on. GMP has a rich set of functions, and the functions have a regular interface.</td>
<td>Link</td>
</tr>
<tr>
<td>ascend_install.info</td>
<td>Installation log file</td>
<td>Default path: <code>/etc/ascend_install.info</code> on the host</td>
</tr>
<tr>
<td>version.info</td>
<td>Driver version file</td>
<td>Default path: <code>usr/local/Ascend/driver/version.info</code> on the host</td>
</tr>
<tr>
<td>MindStudio_{version}_linux.tar.gz</td>
<td>MindStudio package</td>
<td>Link</td>
</tr>
<tr>
<td>Ascend-cann-toolkit_{version}_linux-{arch}.run</td>
<td>Ascend-CANN-Toolkit runfile</td>
<td>Link</td>
</tr>
<tr>
<td>mindspore_ascend-{version}-cp37-cp37m-linux-{arch}.whl</td>
<td>MindSpore wheel file</td>
<td>Link</td>
</tr>
<tr>
<td>mindinsight-{version}-cp37-cp37m-linux-{arch}.whl</td>
<td>MindInsight wheel file</td>
<td>Link</td>
</tr>
</tbody>
</table>
GMP is a free library for arbitrary-precision arithmetic, operating on signed integers, rational numbers, and floating-point numbers. There is no practical limit to the precision except the ones implied by the available memory in the machine GMP runs on. GMP has a rich set of functions, and the functions have a regular interface.

### Host Requirements

**Table 2-15** Host requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Requirement</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>• Recommended memory size: 8 GB (at least 4 GB)</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>• Minimum drive space: 30 GB</td>
<td></td>
</tr>
</tbody>
</table>

**Table 2-16** describes the mapping between the host OS and the container OS.
### Table 2-16 Host and container OS mapping

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Host OS</th>
<th>Container OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AArch64</td>
<td>--</td>
<td>CentOS 7.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Euler 2.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ubuntu 18.04</td>
</tr>
<tr>
<td>x86_64</td>
<td>--</td>
<td>CentOS 7.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ubuntu 18.04</td>
</tr>
</tbody>
</table>

#### 2.5.3 Creating a MindStudio Container Image Using a Dockerfile

You can build container images by using the provided Dockerfile samples with only slight tweaking.

**Obtaining the Dockerfile Samples**

**Step 1** Decompress the MindStudio tar.gz package.

Run the following command as the MindStudio installation user to decompress the `MindStudio_{version}_linux.tar.gz` package:

```
tar -zxvf MindStudio_{version}_linux.tar.gz
```

**Step 2** Go to the directory generated after the decompression and select the required Dockerfile.

```
cd ./MindStudio
```

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>./docker/infer/aarch64-linux/centos/Dockerfile</td>
<td>Dockerfile sample for CentOS 7.8 (ARM), which is applicable to inference scenario</td>
</tr>
<tr>
<td>./docker/infer/aarch64-linux/euler/Dockerfile</td>
<td>Dockerfile sample for EulerOS 2.8 (ARM), which is applicable to inference scenario</td>
</tr>
<tr>
<td>./docker/infer/aarch64-linux/ubuntu/Dockerfile</td>
<td>Dockerfile sample for Ubuntu 18.04 (ARM), which is applicable to inference scenarios</td>
</tr>
<tr>
<td>./docker/infer/x86_64-linux/centos/Dockerfile</td>
<td>Dockerfile sample for CentOS 7.6 (x86_64), which is applicable to inference scenarios</td>
</tr>
<tr>
<td>./docker/infer/x86_64-linux/ubuntu/Dockerfile</td>
<td>Dockerfile sample for Ubuntu 18.04 (x86_64), which is applicable to inference scenarios</td>
</tr>
<tr>
<td>./docker/train/aarch64-linux/centos/Dockerfile</td>
<td>Dockerfile sample for CentOS 7.8 (ARM), which is applicable to training scenario</td>
</tr>
<tr>
<td>./docker/train/aarch64-linux/euler/Dockerfile</td>
<td>Dockerfile sample for EulerOS 2.8 (ARM), which is applicable to training scenario</td>
</tr>
</tbody>
</table>
Building an Image Using a Dockerfile

**Step 1** Modify the Dockerfile based on the actual situation by referring to **Obtaining the Dockerfile Samples**. Table 2-17 lists the items that must be modified. Other items should be modified as required.

### Table 2-17 Dockerfile tweaks

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Option</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>For building a CentOS 7.8 AArch64-based MindStudio container image for inference use</td>
<td>CANN_TOOLKIT_NAME</td>
<td>Ascend-CANN-Toolkit runfile name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td></td>
<td>MINDSTUDIO_NAME</td>
<td>MindStudio package name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td>For building a EulerOS 2.8 AArch64-based MindStudio container image for inference use</td>
<td>CANN_TOOLKIT_NAME</td>
<td>Ascend-CANN-Toolkit runfile name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td></td>
<td>MINDSTUDIO_NAME</td>
<td>MindStudio package name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td>For building a Ubuntu18.04 AArch64-based MindStudio container image for inference use</td>
<td>CANN_TOOLKIT_NAME</td>
<td>Ascend-CANN-Toolkit runfile name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td></td>
<td>MINDSTUDIO_NAME</td>
<td>MindStudio package name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td>For building a CentOS 7.6 x86_64-based MindStudio container</td>
<td>CANN_TOOLKIT_NAME</td>
<td>Ascend-CANN-Toolkit runfile name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td>Scenario</td>
<td>Option</td>
<td>Description</td>
<td>Remarks</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>image for inference use</td>
<td>MINDSTUDIO_NAME</td>
<td>MindStudio package name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td>For building a Ubuntu18.04 x86_64-based MindStudio container image for inference use</td>
<td>CANN_TOOLKIT_NAME</td>
<td>Ascend-CANN-Toolkit runfile name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td></td>
<td>MINDSTUDIO_NAME</td>
<td>MindStudio package name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td>For building a CentOS 7.8 AArch64-based MindStudio container image for training use</td>
<td>CANN_TOOLKIT_NAME</td>
<td>Ascend-CANN-Toolkit runfile name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td></td>
<td>MINDSTUDIO_NAME</td>
<td>MindStudio package name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td></td>
<td>MINDINSIGHT_NAME</td>
<td>MindInsight wheel file name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td>For building a EulerOS 2.8 AArch64-based MindStudio container image for training use</td>
<td>CANN_TOOLKIT_NAME</td>
<td>Ascend-CANN-Toolkit runfile name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td></td>
<td>MINDSTUDIO_NAME</td>
<td>MindStudio package name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td></td>
<td>MINDINSIGHT_NAME</td>
<td>MindInsight wheel file name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td>For building a Ubuntu18.04 AArch64-based MindStudio container image for training use</td>
<td>CANN_TOOLKIT_NAME</td>
<td>Ascend-CANN-Toolkit runfile name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td></td>
<td>MINDSTUDIO_NAME</td>
<td>MindStudio package name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td></td>
<td>MINDINSIGHT_NAME</td>
<td>MindInsight wheel file name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td>Scenario</td>
<td>Option</td>
<td>Description</td>
<td>Remarks</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>-------------------------</td>
<td>---------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>For building a CentOS 7.6 x86_64-based MindStudio container image for training use</td>
<td>MINDINSIGHT_NAME</td>
<td>MindInsight wheel file name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td>For building a CentOS 7.6 x86_64-based MindStudio container image for training use</td>
<td>CANN_TOOLKIT_NAME</td>
<td>Ascend-CANN-Toolkit runfile name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td>For building a CentOS 7.6 x86_64-based MindStudio container image for training use</td>
<td>MINDSTUDIO_NAME</td>
<td>MindStudio package name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td>For building a CentOS 7.6 x86_64-based MindStudio container image for training use</td>
<td>MINDINSIGHT_NAME</td>
<td>MindInsight wheel file name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td>For building a Ubuntu18.04 x86_64-based MindStudio container image for training use</td>
<td>CANN_TOOLKIT_NAME</td>
<td>Ascend-CANN-Toolkit runfile name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td>For building a Ubuntu18.04 x86_64-based MindStudio container image for training use</td>
<td>MINDSTUDIO_NAME</td>
<td>MindStudio package name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td>For building a Ubuntu18.04 x86_64-based MindStudio container image for training use</td>
<td>MINDSPORE_NAME</td>
<td>MindSpore wheel file name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
<tr>
<td>For building a Ubuntu18.04 x86_64-based MindStudio container image for training use</td>
<td>MINDINSIGHT_NAME</td>
<td>MindInsight wheel file name</td>
<td>Set to the actual file name as described in Table 2-14.</td>
</tr>
</tbody>
</table>

**Step 2**  Copy the dependencies listed in Table 2-18 to the directory where the Dockerfile is stored.

**Table 2-18** Dockerfile dependencies

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td>For building a CentOS 7.8 AArch64-based MindStudio container image for inference use</td>
<td>1. MindStudio_{version}_linux.tar.gz</td>
</tr>
<tr>
<td>For building a CentOS 7.8 AArch64-based MindStudio container image for inference use</td>
<td>2. Ascend-cann-toolkit_{version}_linux-{arch}.run</td>
</tr>
<tr>
<td>For building a CentOS 7.8 AArch64-based MindStudio container image for inference use</td>
<td>3. matplotlib-3.3.2.tar.gz</td>
</tr>
<tr>
<td>For building a CentOS 7.8 AArch64-based MindStudio container image for inference use</td>
<td>4. freetype-2.6.1.tar.gz</td>
</tr>
<tr>
<td>For building a CentOS 7.8 AArch64-based MindStudio container image for inference use</td>
<td>5. ascend_install.info (required only for AI CPU operator inference and commissioning)</td>
</tr>
<tr>
<td>For building a CentOS 7.8 AArch64-based MindStudio container image for inference use</td>
<td>6. version.info (required only for AI CPU operator inference and commissioning)</td>
</tr>
<tr>
<td>Use Case</td>
<td>Dependency</td>
</tr>
<tr>
<td>---------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| For building a EulerOS 2.8 AArch64-based MindStudio container image for inference use | 1. MindStudio_{version}_linux.tar.gz  
2. Ascend-cann-toolkit_{version}_linux-{arch}.run  
3. matplotlib-3.3.3.tar.gz  
4. ascend_install.info (required only for AI CPU operator inference and commissioning)  
5. version.info (required only for AI CPU operator inference and commissioning) |
| For building a Ubuntu18.04 AArch64-based MindStudio container image for inference use | 1. MindStudio_{version}_linux.tar.gz  
2. Ascend-cann-toolkit_{version}_linux-{arch}.run  
3. matplotlib-3.3.3.tar.gz  
4. ascend_install.info (required only for AI CPU operator inference and commissioning)  
5. version.info (required only for AI CPU operator inference and commissioning) |
| For building a CentOS 7.6 x86_64-based MindStudio container image for inference use | 1. MindStudio_{version}_linux.tar.gz  
2. Ascend-cann-toolkit_{version}_linux-{arch}.run  
3. ascend_install.info (required only for AI CPU operator inference and commissioning)  
4. version.info (required only for AI CPU operator inference and commissioning) |
| For building a Ubuntu18.04 x86_64-based MindStudio container image for inference use | 1. MindStudio_{version}_linux.tar.gz  
2. Ascend-cann-toolkit_{version}_linux-{arch}.run  
3. ascend_install.info (required only for AI CPU operator inference and commissioning)  
4. version.info (required only for AI CPU operator inference and commissioning) |
| For building a CentOS 7.8 AArch64-based MindStudio container image for training use | 1. MindStudio_{version}_linux.tar.gz  
2. Ascend-cann-toolkit_{version}_linux-{arch}.run  
3. mindinsight-{version}-cp37-cp37m-linux_{arch}.whl  
4. matplotlib-3.3.2.tar.gz  
5. freetype-2.6.1.tar.gz  
6. gmp-6.1.2.tar.xz  
7. ascend_install.info  
8. version.info |
<table>
<thead>
<tr>
<th>Use Case</th>
<th>Dependency</th>
</tr>
</thead>
</table>
| For building a EulerOS 2.8 AArch64-based MindStudio container image for training use | 1. MindStudio_{version}_linux.tar.gz  
2. Ascend-cann-toolkit_{version}_linux-{arch}.run  
3. mindspore_ascend-{version}-cp37-cp37m-linux_{arch}.whl  
4. mindinsight-{version}-cp37-cp37m-linux_{arch}.whl  
5. matplotlib-3.3.3.tar.gz  
6. gmp-6.1.2.tar.xz  
7. ascend_install.info  
8. version.info |
| For building a Ubuntu18.04 AArch64-based MindStudio container image for training use | 1. MindStudio_{version}_linux.tar.gz  
2. Ascend-cann-toolkit_{version}_linux-{arch}.run  
3. mindspore_ascend-{version}-cp37-cp37m-linux_{arch}.whl  
4. mindinsight-{version}-cp37-cp37m-linux_{arch}.whl  
5. matplotlib-3.3.3.tar.gz  
6. gmp-6.1.2.tar.xz  
7. ascend_install.info  
8. version.info |
| For building a CentOS 7.6 x86_64-based MindStudio container image for training use | 1. MindStudio_{version}_linux.tar.gz  
2. Ascend-cann-toolkit_{version}_linux-{arch}.run  
3. mindspore_ascend-{version}-cp37-cp37m-linux_{arch}.whl  
4. mindinsight-{version}-cp37-cp37m-linux_{arch}.whl  
5. gmp-6.1.2.tar.xz  
6. ascend_install.info  
7. version.info |
| For building a Ubuntu18.04 x86_64-based MindStudio container image for training use | 1. MindStudio_{version}_linux.tar.gz  
2. Ascend-cann-toolkit_{version}_linux-{arch}.run  
3. mindspore_ascend-{version}-cp37-cp37m-linux_{arch}.whl  
4. mindinsight-{version}-cp37-cp37m-linux_{arch}.whl  
5. gmp-6.1.2.tar.xz  
6. ascend_install.info  
7. version.info |

**Step 3** (Optional) In the scenario of AI CPU operator inference and commissioning, add the following two lines after the `COPY SMINDSTUDIO_NAME /root` line in the corresponding Dockerfile:

```
COPY ascend_install.info /etc/
COPY version.info /usr/local/Ascend/driver/
```

**Step 4** Run the `build` command in the directory where the Dockerfile is stored.

```
docker build -t image_name .
```

The ending period indicates the context path of the current execution. Do not omit it. Fill in `image_name` in the `image_name:image_tag` format.
If "Successfully built" is displayed, the image is successfully built.

**Step 5** Run the following command to view the image details:

```bash
docker images
```

**Example:**

<table>
<thead>
<tr>
<th>REPOSITORY</th>
<th>TAG</th>
<th>IMAGE ID</th>
<th>CREATED</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>mindstudio</td>
<td>v1</td>
<td>d82746acd7f0</td>
<td>27 minutes ago</td>
<td>10.2G</td>
</tr>
</tbody>
</table>

**NOTE**

During image building, an intermediate image named `none` is generated. You are advised to delete it to save space.

---

### 2.5.4 Starting a MindStudio Container

#### 2.5.4.1 Starting a MindStudio Container on an Ascend AI Device

**NOTICE**

To start a MindStudio container on an Ascend AI device, ensure that the firmware and driver have been installed on the host.

**Step 1** Log in to the host as the installation user and import necessary environment variables.

The following is an example.

```bash
#NPUs on the host, such as /dev/davinci0, /dev/davinci1, and /dev/davinci2
export device=/dev/davinci0
#The SSH port (22) in the container is mapped to the host port (20000). The host port must be available.
export port=20000
# Set the DISPLAY environment variable.
export DISPLAY=host IP address that displays the GUI:0
# Set the name and tag of the built MindStudio image.
export image_name=mindstudio:v1
# Set the name of the running container.
export container_name=mindstudio
```

**Step 2** Run the following command on the host to run a container based on the built MindStudio image. (The following environment variables have been set in Step 1.)

```bash
docker run -itd \
  --device=${device} \
  --device=/dev/davinci_manager \
  --device=/dev/devmm_svm \
  --device=/dev/hisi_hdc \
  --cap-add=SYS_PTRACE --cap-add=SYS_ADMIN \
  --cap-add=IPC_LOCK \
  -v /sys/kernel/debug:/sys/kernel/debug \
  -v /usr/local/Ascend/driver:/usr/local/Ascend/driver \
  -v $HOME/.Xauthority:/root/.Xauthority \
  -e DISPLAY=${DISPLAY} \
  -p ${port}:22 \
  --name ${container_name} \
  ${image_name}
```
### Table 2-19 Command-line options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-itd</td>
<td>-i: runs the container in interactive mode.</td>
</tr>
<tr>
<td></td>
<td>-t: reallocates a pseudo-input terminal to the container.</td>
</tr>
<tr>
<td></td>
<td>-d: runs the container in the background and returns the container ID.</td>
</tr>
<tr>
<td>--device=XXX</td>
<td>Maps the devices (NPU and management devices) on the host to the container.</td>
</tr>
<tr>
<td>--cap-add=SYS_PTRACE --cap-add=SYS_ADMIN</td>
<td>Sets the permission to start the System Profiling tool in the container.</td>
</tr>
<tr>
<td>--cap-add=IPC_LOCK</td>
<td>(Optional) This parameter needs to be configured when the SSH password storage mode is set to In native Keychain.</td>
</tr>
<tr>
<td>-v host_dir:container_dir</td>
<td>Mount a directory on the host machine to the container.</td>
</tr>
<tr>
<td>-e DISPLAY=$DISPLAY</td>
<td>Sets the DISPLAY environment variable in the container to the DISPLAY environment variable on the host machine.</td>
</tr>
<tr>
<td>-p ${port}:22</td>
<td>Maps SSH port 22 of the container to a port on the host. The port on the host must be available. You can run the `netstat -anp</td>
</tr>
<tr>
<td>--name ${container_name}</td>
<td>Sets the container name.</td>
</tr>
<tr>
<td>${image_name}</td>
<td>Sets the image that runs the container, in <code>image_name:image_tag</code> format.</td>
</tr>
</tbody>
</table>

If the container ID is displayed after the command is executed, the MindStudio container has been started.

**Step 3** Access the container.

**Method 1**: Run the `docker exec` command to access the container.

```bash
docker exec -it ${container_name} /bin/bash
```

**Method 2**: Access the container through SSH.

```
# For the first use, run the `docker exec` command to access the container and run the `passwd` command to set the password of the `root` user.
ssh root@{hostIP} -p ${port}
```

**Step 4** (Optional) Start System Profiling in the container.

```
cd /usr/local/Ascend/ascend-toolkit/latest/toolkit/tools/system-profiling/script/
./start.sh
```
Step 5  Launch MindStudio in the container.

```
cd /root/MindStudio/bin
./MindStudio.sh
```

When MindStudio is launched for the first time, it will prompt you to install the necessary third-party dependencies. You are advised not to launch MindStudio in the background. The installation script automatically checks whether the path is valid.

1. If the path is valid, the dependencies are checked. If all the dependencies are installed, go to Step 6. Otherwise, install any missing dependency as prompted. After all dependencies are installed, run the installation script again.

2. If messages similar to the following are displayed:

   ```
   There is no python3.7.5 in path /usr/local/python3.7.5. Please enter your choice:
   1. choose local python3.7.5
   2. install python3.7.5
   ```

   a. Enter 1. The system prompts "Enter local python path:". Enter the local path where Python 3.7.5 has been installed. The script checks whether the path is valid.

   i. If the path is valid, the dependencies are checked. If all the dependencies are installed, go to Step 6. Otherwise, install any missing dependency as prompted.

   ii. If the path check fails, return to the message prompt. Check whether the entered path is correct or enter 2 to go to b.

   b. Enter 2 and enter a new installation path to install Python 3.7.5 as prompted. You can also retain the default installation path `/usr/local/python3.7.5`.

      ```
      Enter python install path. Default value is /usr/local/python3.7.5:
      /home/hisisoc/software/python3.7.5
      ```

      Install Python 3.7.5 and related dependencies as prompted. After the installation is complete, run the installation script again.

Step 6  Open the Import MindStudio Settings dialog box, as shown in Figure 2-28.

![Figure 2-28 Environment settings](image)

- **Config or installation folder**: imports MindStudio settings from an existing directory.
  The directory is $HOME/.config/Huawei/MindStudioMS-x.x. The MindStudio settings include the personalized settings of the project UI such as the background color.

- **Do not import settings** (default): creates a configuration file.

Select an option and click OK.
**Step 7** The dialog box shown in **Figure 2-29** is displayed.

**Figure 2-29** Untrusted server certificate

- Click **Accept** to allow MindStudio to connect to the upgrade server.
- Click **Reject** to reject MindStudio to connect to the upgrade server. This dialog box is displayed when you restart MindStudio next time.

**Step 8** The window shown in **Figure 2-30** is displayed.

- If the page is not completely displayed, rectify the fault by following the instructions in **15.8.7 How Do I Solve the Problem that Chinese Characters Are Displayed as Garbled Characters and the GUI Is Incompletely Displayed or Is Not Well-Organized?**.
- When selecting the installation path for Ascend-CANN-Toolkit, do not select the soft link path that is automatically created by the system when the development kit is installed.
If there is an Ascend-CANN-Toolkit installation: Click next to Ascend Toolkit Path. In the dialog box that is displayed, select the Ascend-CANN-Toolkit installation path (including the version number). Click OK. The Ascend-CANN-Toolkit version information is displayed in the Ascend Toolkit Version text box, which is not configurable, as shown in Figure 2-31. Click Finish, moving on to Step 11.

Figure 2-30 Selecting Ascend-CANN-Toolkit path

Figure 2-31 Selecting the Ascend-CANN-Toolkit installation path
If there is no Ascend-CANN-Toolkit installation: Click next to **Ascend Toolkit Path**. In the dialog box that is displayed, select the Ascend-CANN-Toolkit runfile uploaded in **Table 2-14** and select its path, as shown in **Figure 2-32**. Click **Next**, moving on to **Step 9**.

**Figure 2-32** Selecting the Ascend-CANN-Toolkit path

---

**Step 9** Verify the current settings.

**Figure 2-33** Current settings
### NOTE

If you have installed an Ascend-CANN-Toolkit of the same version in the specified Install Location, a message is displayed, indicating that a previous installation is found.

**Step 10** Click **Next**, as shown in **Figure 2-34**.

**Figure 2-34** Installation progress

Click **Show Details** to check the Ascend-CANN-Toolkit installation details or click **Hide Details** to hide the details.

If the following message is displayed, the installation is successful. In this case, the **Finish** button becomes available and other buttons are unavailable.

Installation succeeded. Please click "Finish".

Click **Finish**. A dialog box is displayed, prompting you to restart MindStudio, as shown in **Figure 2-35**.

**Figure 2-35** Restarting MindStudio

Click **OK** to restart MindStudio.

**Step 11** The **User Experience Improvement Program** page is displayed. See **Figure 2-26**. **Name** and **Organization** are optional. Click the **User Experience Improvement Program** link to read the details. Selecting the check box indicates that you have read the content.
Figure 2-36 User experience improvement program

- Click **Agree**. MindStudio will collect user information for the experience improvement program.
- Click **Disagree** to close the window.

**Step 12** If no error is reported and the welcome window is displayed, MindStudio is successfully installed.

Find the following new directories in $HOME$ on the MindStudio installation server:

- $HOME/.mindstudio$: stores the MindStudio property file `mindstudio.properties` (recording the Toolkit installation path) and records of operations performed by users on the privacy statement window.
- $HOME/.cache/Huawei/MindStudioMS-{version}$: stores runtime logs (`log/idea.log`) generated after MindStudio is installed and launched.
- $HOME/.config/Huawei/MindStudioMS-{version}$: stores user configuration, including personalized settings (such as the background color) of the project UI.

The parameters in the welcome window are described as follows:

- **Create new project**: creates a project and save it in $HOME/AscendProjects$.
- **Open or Import**: opens or imports a project.
- **Welcome System Profiling**: opens the **System Profiling** window.

**Step 13** Launch MindStudio and choose **File > Settings > Plugins** from the main menu to install the following plugins:
2.5.4.2 Starting the MindStudio Container on a non-Ascend AI Device

NOTICE

The procedure for starting a MindStudio container on a non-Ascend AI device is the same as that for starting a MindStudio container on an Ascend AI device, except that the commands for importing environment variables in step 1 and running the container in step 2 are different.

**Step 1** Import environment variables to the host.

The following is an example:

```bash
# The SSH port (22) in the container is mapped to the host port (20000). The host port must be available.
export port=20000
# Set the DISPLAY environment variable.
export DISPLAY=host IP address that displays the GUI:0
# Set the name and tag of the built MindStudio image.
export image_name=mindstudio:v1
# Set the name of the running container.
export container_name=mindstudio
```

**Step 2** Run the following command on the host to run a container based on the built MindStudio image: (The following environment variables have been set in **Step 1**.)

```bash
docker run -itd -v $HOME/.Xauthority:/root/.Xauthority -e DISPLAY=$DISPLAY -p ${port}:22 --name ${container_name} --cap-add=IPC_LOCK ${image_name}
```

### Table 2-20 Command-line options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-itd</td>
<td>-i: runs the container in interactive mode.</td>
</tr>
<tr>
<td></td>
<td>-t: reallocates a pseudo-input terminal to the container.</td>
</tr>
<tr>
<td></td>
<td>-d: runs the container in the background and returns the container ID.</td>
</tr>
<tr>
<td>-v host_dir:container_dir</td>
<td>Mount a directory on the host machine to the container.</td>
</tr>
<tr>
<td>-e DISPLAY=$DISPLAY</td>
<td>Sets the DISPLAY environment variable in the container to the DISPLAY environment variable on the host machine.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>-p ${port}:22</td>
<td>Maps SSH port 22 of the container to a port on the host. The port on the host must be available. You can run the `netstat -anp</td>
</tr>
<tr>
<td>--name $</td>
<td>Sets the container name.</td>
</tr>
<tr>
<td>{container_name}</td>
<td></td>
</tr>
<tr>
<td>--cap-add=IPC_LOCK</td>
<td>(Optional) This parameter needs to be configured when the SSH password storage mode is set to In native Keychain.</td>
</tr>
<tr>
<td>${image_name}</td>
<td>Sets the image that runs the container, in <code>image_name:image_tag</code> format.</td>
</tr>
</tbody>
</table>

If the container ID is displayed after the command is executed, the MindStudio container has been started.

**Step 3** Proceed from step 3 described in 2.5.4.1 Starting a MindStudio Container on an Ascend AI Device.

----End

### 2.5.5 Deleting a MindStudio Container

**Step 1** Stop the running MindStudio container.

`docker stop ${container_name}`

**Step 2** Delete the MindStudio container.

`docker rm ${container_name}`

**Step 3** (Optional) Delete the MindStudio image.

`docker rmi ${image_name}`

----End
3 Supported Operators and Models

The Supported Operators and Supported Models dialog boxes help developers understand the operators and models supported by the current ADK version.

3.1 Supported Operators

3.2 Supported Models

3.1 Supported Operators

Query Portals

You can navigate to the Supported Operators dialog box in either of the following ways:

- Choose Ascend > View Supported Operators from the main menu.
- Click on the main menu.

NOTE

If no valid Ascend-cann-toolkit_*.run is installed, the error message "No ADK is found. Please (re)install an ADK in ADK Manager first" is displayed. In this case, check whether Ascend-cann-toolkit_*.run is correctly installed.

Supported Operators Dialog Box

Select View Supported Operators or click the icon . The Supported Operators dialog box is displayed, as shown in Figure 3-1.

- You can query the operator by name in Search by Name, or filter the operator based on the operator type or hardware model. Operator information is displayed as follows:
  - Operator Type
  - Hardware Type: hardware supported by the operator
Figure 3-1 Supported Operators

<table>
<thead>
<tr>
<th>Operator Type</th>
<th>Hardware Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abs</td>
<td>ascend310</td>
</tr>
<tr>
<td>AccumulateN1V2</td>
<td>ascend310</td>
</tr>
<tr>
<td>Acos</td>
<td>ascend310</td>
</tr>
<tr>
<td>Acosh</td>
<td>ascend310</td>
</tr>
<tr>
<td>Add</td>
<td>ascend310</td>
</tr>
<tr>
<td>AddN</td>
<td>ascend310</td>
</tr>
<tr>
<td>AddN2</td>
<td>ascend310</td>
</tr>
<tr>
<td>App</td>
<td>ascend310</td>
</tr>
<tr>
<td>ApproximatelyEqual</td>
<td>add2</td>
</tr>
<tr>
<td>ArgMax</td>
<td>ascend310</td>
</tr>
<tr>
<td>ArgMin</td>
<td>ascend310</td>
</tr>
<tr>
<td>ArgMaxWithValue</td>
<td>ascend310</td>
</tr>
<tr>
<td>ArgMinWithValue</td>
<td>ascend310</td>
</tr>
<tr>
<td>AscendAntiQuart</td>
<td>ascend310</td>
</tr>
<tr>
<td>AscendQuart</td>
<td>ascend310</td>
</tr>
<tr>
<td>Asin</td>
<td>ascend310</td>
</tr>
<tr>
<td>Asinh</td>
<td>ascend310</td>
</tr>
<tr>
<td>Assign</td>
<td>ascend310</td>
</tr>
<tr>
<td>AssignAdd</td>
<td>ascend310</td>
</tr>
<tr>
<td>AssignSub</td>
<td>ascend310</td>
</tr>
</tbody>
</table>

- Select an operator type to see more details on the detail page, as is shown in Figure 3-2.

Figure 3-2 Operator details (example)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>opInfo</td>
<td></td>
</tr>
<tr>
<td>flagPartial</td>
<td>False</td>
</tr>
<tr>
<td>engine</td>
<td>DNN_VM_AICPU</td>
</tr>
<tr>
<td>opsFlag</td>
<td>OPS_FLAG_CLOSE</td>
</tr>
<tr>
<td>flagAsync</td>
<td>False</td>
</tr>
<tr>
<td>formatAgnostic</td>
<td>True</td>
</tr>
<tr>
<td>opKernelLib</td>
<td>TFKernel</td>
</tr>
<tr>
<td>computeCost</td>
<td>100</td>
</tr>
<tr>
<td>subTypeOfInferShape</td>
<td>1</td>
</tr>
<tr>
<td>input0</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>x</td>
</tr>
<tr>
<td>type</td>
<td>DT_INT8, DT_INT16, DT_FLOAT16, DT_FL...</td>
</tr>
<tr>
<td>output0</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>y</td>
</tr>
<tr>
<td>type</td>
<td>DT_INT8, DT_INT16, DT_FLOAT16, DT_FL...</td>
</tr>
</tbody>
</table>

3.2 Supported Models

Query Portals

You can access the model query page in either of the following ways:
Choose Ascend > View Supported Models from the main menu.

Click ☰ on the main menu.

**Supported Models Window**

Obtain supported models at ModelZoo in the Ascend Community.

With Internet access, you can obtain the background information. The pre-trained models are displayed on the Supported Models dialog box. As shown in Figure 3-3, the Model Name, Application Area, and Precision are displayed.

**Figure 3-3 Supported Models**

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Application Area</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATT DFL03 from TensorflowAscend 2.0</td>
<td>Object Detection</td>
<td>PS6.0</td>
</tr>
<tr>
<td>ATT ResNet from Tensorflow - Ascend 5.0</td>
<td>Image Classification</td>
<td>PS6.0</td>
</tr>
<tr>
<td>ATT VGG16 from Tensorflow - Ascend 5.0</td>
<td>Image Classification</td>
<td>PS6.0</td>
</tr>
<tr>
<td>ATT VGG19 from Tensorflow - Ascend 5.0</td>
<td>Image Classification</td>
<td>PS6.0</td>
</tr>
<tr>
<td>ATTsqueeze from Tensorflow - Ascend 2.0</td>
<td>Image Classification</td>
<td>PS6.0</td>
</tr>
<tr>
<td>ATT ResNet C50 from TensorflowAscend 6.0</td>
<td>Image Classification</td>
<td>PS6.0</td>
</tr>
<tr>
<td>ATT ResNet-101 from TensorflowAscend 5.0</td>
<td>Image Classification</td>
<td>PS6.0</td>
</tr>
<tr>
<td>ATT ResNet-152 from TensorflowAscend 5.0</td>
<td>Image Classification</td>
<td>PS6.0</td>
</tr>
<tr>
<td>ATT MobileNet2 from Tensorflow - Ascend 5.0</td>
<td>Image Classification</td>
<td>PS6.0</td>
</tr>
<tr>
<td>ATT MobileNet5 from Tensorflow - Ascend 5.0</td>
<td>Image Classification</td>
<td>PS6.0</td>
</tr>
<tr>
<td>ATT MetaResNet from Tensorflow - Ascend 5.0</td>
<td>Classification</td>
<td>PS6.0</td>
</tr>
<tr>
<td>ATT InceptionResNet from TensorflowAscend 2.0</td>
<td>Image Classification</td>
<td>PS6.0</td>
</tr>
<tr>
<td>ATT EfficientNetL2 from TensorflowAscend 3.0</td>
<td>Image Classification</td>
<td>PS6.0</td>
</tr>
<tr>
<td>ATT DenseNet from TensorflowAscend 5.0</td>
<td>Image Classification</td>
<td>PS6.0</td>
</tr>
<tr>
<td>ATT EfficientNetL3 from TensorflowAscend 5.0</td>
<td>Image Classification</td>
<td>PS6.0</td>
</tr>
<tr>
<td>ATT BERT Large (Chinese) from TensorflowAscend 3.0</td>
<td>Sentiment analysis</td>
<td>PS6.0</td>
</tr>
<tr>
<td>ATT BERT Large from TensorflowAscend 3.0</td>
<td>Image Classification</td>
<td>PS6.0</td>
</tr>
</tbody>
</table>

If the background information fails to be obtained, the ModelZoo dialog box is displayed, showing the URL of the ModelZoo, as shown in Figure 3-4. Users can directly access the URL to query the supported models.

**Figure 3-4 ModelZoo**

Get supported model info failed. Please refer: https://huaweicloud.com/ascend/resources/modelzoo
4 Model Conversion

4.1 Before You Start

Models trained under frameworks such as Caffe and TensorFlow can be converted into offline models compatible with the Ascend AI Processor by using the ATC. During offline model conversion, you can enable operator scheduling tuning, weight data rearrangement, and memory optimization, thereby preprocessing your models without depending on the device. Figure 4-1 shows the ATC architecture.

**Figure 4-1 ATC architecture**

```
Caffe, TensorFlow, MindSpore, ONNX, ...
```

```
ATC

Prepare graph  Optimize graph  Partition graph  Build graph
```

```
Model file
```

```
Model .om file
```
Reference Documents

Table 4-1 Reference documents

<table>
<thead>
<tr>
<th>Product</th>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlas 200 DK</td>
<td>● <em>ATC Instructions</em></td>
</tr>
<tr>
<td></td>
<td>● <em>Application Software</em></td>
</tr>
<tr>
<td></td>
<td><em>Development Guide (C&amp;C++)</em></td>
</tr>
<tr>
<td>Others</td>
<td>● <em>ATC Instructions (Inference)</em></td>
</tr>
<tr>
<td></td>
<td>● <em>Application Software</em></td>
</tr>
<tr>
<td></td>
<td><em>Development Guide (C&amp;C++)</em></td>
</tr>
<tr>
<td></td>
<td><em>(Inference)</em></td>
</tr>
</tbody>
</table>

Restrictions

Before model conversion, pay attention to the following restrictions:

- To convert a network such as Faster R-CNN, YOLOv3, or YOLOv2 into an offline model compatible with the the Ascend AI Processor, you must modify the .prototxt model file by referring to 15.4.1 Custom Caffe Network Modification.
- Model conversion is supported for Caffe, TensorFlow, ONNX, and MindSpore frameworks.
  - If the original framework type is Caffe, ONNX, or MindSpore, the input data type can be FP32, FP16 (implemented by setting the input argument --input_fp16_nodes, which is not supported by the MindSpore framework), or UINT8 (implemented by configuring data preprocessing).
  - If the original framework type is TensorFlow, the input data type can be FP16, FP32, UINT8, INT32, INT64, or BOOL. Note that INT64 is not supported and needs to be changed to INT32.
- For a Caffe model, op name and op type in the model file (.prototxt) and weight file (.caffemodel) must be consistent (case sensitive).
- For a TensorFlow model, only the FrozenGraphDef format is supported.
- Dynamic shape input is not supported, for example, NHWC = [?, ?, ?, 3]. The dimension sizes must be static.
- For a Caffe model, the input data is up to 4-dimensional and operators such as reshape and expanddim do not support 5-dimensional output.
- Except the const operator, the input and output at all layers in a model must meet the condition dim! = 0.
- For the full list of operators and limitations, see 15.3.1 Caffe Operator Specifications and 15.3.2 TensorFlow Operator Specifications.
4.2 Model Conversion

Portals

- You can navigate to the **Model Converter** in either of the following ways:
  - Choose **Ascend > Model Converter** on the menu bar.
  - Choose **View > Appearance > Toolbar** on the menu bar. Click 📝 on the displayed toolbar below the menu bar.
  
  For details, see **Procedure**.

- You can also run the atc command to convert the model as follows:
  a. Click the **Terminal** window bar in the lower part of the MindStudio main window.
  b. Understand the functions of related command-line options by referring to "Command-line Options" and set environment variables by referring to "Getting Started with ATC" in **ATC Tool Instructions** in **Reference Documents**.
  c. Run the **atc** command to start conversion. For details, see section "Getting Started with ATC".

Prerequisites

The model file and weight file of the model have been uploaded to the development environment where the Ascend-CANN-Toolkit resides by the MindStudio installation user.

Procedure

**Step 1** Go to the model conversion page and upload the model file and weight file on the **Model Information** tab page. See **Figure 4-2**. The actual page may vary depending on the version.
**Figure 4-2 Model Information**

Table 4-2 describes the parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model File</td>
<td>(Required) Selects the model file.</td>
</tr>
<tr>
<td></td>
<td>Select a model file in either of the following ways:</td>
</tr>
<tr>
<td></td>
<td>• Click the folder icon on the right, browse for the model file on the server, and upload the file.</td>
</tr>
<tr>
<td></td>
<td>• Enter the path of the model file on the server in the text box, including the file name extension.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong></td>
</tr>
<tr>
<td></td>
<td>In the Windows OS, you can only manually enter the path of the model file on the backend server, including the model file name extension.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Weight File      | Selects the weight file. Required when the original framework is Caffe.  
  - If the model file and weight file are placed in the same directory on the server and their file names are the same, the weight file is automatically selected after the model file is selected.  
  - If the model file and weight file are stored in different directories on the server or in the same directory with different file names:  
    - Upload the weight file manually. Click the folder icon on the right, select the weight file corresponding to the model file from the server path, and upload the weight file.  
    - Enter the path of the .caffemodel weight file on the server in the text box, including the file name extension.  
  **NOTE**  
  If the file path is not automatically entered, in the Windows OS, you can only manually enter the path of the weight file on the backend server, including the weight file name extension. |
| Model Name       | (Required) Sets the model name. After a model file is selected, this parameter is automatically filled. You can change the name as required.  
  Enter a maximum of 64 characters. Only letters (a–z and A–Z), numbers (0–9), underscores (_), and hyphens (-) are allowed.  
  If a model file with the same name already exists in the output path of model conversion, a message is displayed after you click **Next**, asking you to replace the original file or rename the current model. |
| Target SoC Version| Sets the target SoC version. Set this parameter based on the chip form in the board environment. |
| Input Format     | Input data format.  
  - If the original framework is Caffe, **NCHW** and **ND** (any format with \( N \leq 4 \)) are supported. Defaults to **NCHW**.  
  - If the original framework is ONNX, the value is **NCHW**.  
  - If the original framework is MindSpore, the value is **NCHW**.  
  - If the original framework is TensorFlow, **NCHW**, **NHWC**, **ND**, **NCDHW**, and **NDHWC** are supported. Defaults to **NHWC**. |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADK Machine</td>
<td>The system automatically enters the value. This parameter specifies the SSH address for remotely connecting to the ADK environment. The format is <code>&lt;username&gt;@localhost:port number</code>.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Input Nodes</td>
<td>Displays the model input node information.</td>
</tr>
<tr>
<td></td>
<td>If a model file is parsed successfully, the shape and type information of the model input nodes will be displayed below.</td>
</tr>
<tr>
<td></td>
<td>If <strong>Input Nodes</strong> cannot be parsed from the model file selected, you need to manually enter the information. Click + on the right. In the</td>
</tr>
<tr>
<td></td>
<td>displayed dialog box, enter the name, shape, and data type of each input node. Note that the shape allows only commas (,) and digits (-1 and  1–9). Click</td>
</tr>
<tr>
<td></td>
<td>to delete a node.</td>
</tr>
<tr>
<td></td>
<td>If a model has multiple inputs, the shape and type information of each input node will be displayed below <strong>Input Nodes</strong> after the model is parsed successfully.</td>
</tr>
<tr>
<td></td>
<td>● <strong>Shape</strong>: specifies the input shape. The example shown in Figure 4-2 indicates N (number of images processed per batch), C (channel count),</td>
</tr>
<tr>
<td></td>
<td>H (height), and W (width) of the input data, respectively. For example, the channel count of RGB images is 3. If AIPP is enabled, the values of</td>
</tr>
<tr>
<td></td>
<td>H and W are the height and width of the AIPP output.</td>
</tr>
<tr>
<td></td>
<td>– Dynamic batch size. Applies to the scenario where the number of images processed per inference batch is unfixed.</td>
</tr>
<tr>
<td></td>
<td>Set N in the <strong>Shape</strong> text box to -1 and click the rest parameters or <strong>Next</strong>. The <strong>Dynamic Batch</strong> text box is displayed under <strong>Shape</strong>. Enter</td>
</tr>
<tr>
<td></td>
<td>the dynamic batch size choices (separated by commas) in the text box. A maximum of 100 choices are supported. Try to keep the batch size of each</td>
</tr>
<tr>
<td></td>
<td>choice within [1, 2048]. For example, you can enter: 1,2,4,8</td>
</tr>
<tr>
<td></td>
<td>If the <strong>Dynamic Batch</strong> parameter is set during model conversion, you need to add a call to aclmdlSetDynamicBatchSize before the aclmdlExecute call</td>
</tr>
<tr>
<td></td>
<td>to set the actual batch size when running an application project for model inference. For details about how to use the aclmdlSetDynamicBatch-</td>
</tr>
<tr>
<td></td>
<td>Size API, see &quot;AscendCL API Reference &gt; Model Loading and Execution&quot; in Application Software Development Guide (C&amp;C++) (Inference).</td>
</tr>
<tr>
<td></td>
<td>– Dynamic image size. Applies to the scenario where image size per inference batch is unfixed.</td>
</tr>
<tr>
<td></td>
<td>Set H and W in the <strong>Shape</strong> text box to -1 and click the rest parameters or <strong>Next</strong>. The <strong>Dynamic Image Size</strong> text box is displayed under</td>
</tr>
<tr>
<td></td>
<td><strong>Shape</strong>. Enter the dynamic image size choices (separated by commas) in the text box. A maximum of 100 choices are supported. For example, you can enter: 112,112. 224, 224</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>If the Dynamic Image Size parameter is set during model conversion, you need to add a call to aclmdlSetDynamicHWSize before the aclmdlExecute call to set the actual image size when running an application project for model inference.</strong> For details about how to use the aclmdlSetDynamicHWSize API, see &quot;AscendCL API Reference &gt; Model Loading and Execution&quot; in Application Software Development Guide (C&amp;C++) (Inference). If dynamic image size is enabled during model conversion and the data preprocessing function in Data Pre-Processing is required, the Crop and Padding functions are unavailable. Dynamic batch size and dynamic image size are mutually exclusive.</td>
<td></td>
</tr>
<tr>
<td><strong>Type</strong>: specifies the data type of the input node. If the original framework type is Caffe or ONNX, the supported data types are FP32, FP16, and UINT8. If the original framework type is MindSpore, the supported data types are FP32 and UINT8. If the original framework type is TensorFlow, the supported input data types are FP32, FP16, UINT8, Int32, Int64 and Bool. The Data Pre-Processing tab page (for configuring the AIPP function) in Step 2 can be configured only when <strong>Type</strong> is set to UINT8. If a model has multiple inputs, the Data Pre-Processing tab page can be configured only when <strong>Type</strong> is set to UINT8. If the value of <strong>Type</strong> is UINT8 but the H and W information in Shape cannot be obtained, the Data Pre-Processing tab page cannot be configured. <strong>NOTE</strong> If the original framework is MindSpore, <strong>Input Nodes</strong> does not automatically parse the input information in the corresponding model. You need to manually enter the information. If the information is not specified, the ATC tool automatically parses necessary information from the model when the atc command is executed in the command line.</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Output Nodes</td>
<td>Displays the model output node information. Click <strong>Select</strong>. Right-click a node and choose <strong>Select</strong> from the shortcut menu, turning the node blue. Click <strong>OK</strong>. All selected operators are displayed under <strong>Output Nodes</strong>. You can right-click any unwanted operator and choose <strong>Deselect</strong> from the shortcut menu to cancel the selection.</td>
</tr>
</tbody>
</table>
|                 | - **Op Name**: operator name.  
|                 | - **Data Type**: output data type, selected from **FP32**, **UINT8**, and **FP16**.                                                                                                                                                                                                                                                      |
|                 | You can deselect any unwanted operators as required under **Output Nodes**. This section assumes that all operators at the selected layer are retained as the model output.                                                                                                                                                                                                 |
|                 | This function applies to the scenario where you want to check the parameters of a specific layer. Select the layer and deselect unwanted operators as required under **Output Nodes**. After model conversion, in the corresponding .om model file, the output of the operator is used as the output of the model. For details, see **4.3 Visualizing a Model**. |
|                 | **NOTE**  
|                 | - If no layer is selected or no operator is added to **Output Nodes**, the output of the model is the outputs of the operators at the last layer.  
|                 | - If a layer is selected and one or more operators are retained under **Output Nodes**, the output of the retained operators is used as the output of the model.  
|                 | - If a selected operator is fused during model conversion, the operator cannot be specified as the output node.  
|                 | - If the selected model contains an unsupported operator, click **Select**. The failure log is generated in the **Output window** of MindStudio, specifying that the operator is not supported and its shape is not accessible. Such operator is highlighted in red in the displayed **Model Visualizer**. In this case, you cannot obtain the output format and shape of the operator.  
|                 | - If the original framework is MindSpore, this parameter cannot be edited to specify an output node.                                                                                                                                                                                                                                                                 |
| Load Configuration | Loads the configuration file of the last successful conversion. Whether a user model is successfully converted, a **${Model Name}_config.json** configuration file is generated in **$HOME/modelzoo/${Model Name}/device/**. The configuration file records the model conversion configuration, including the model path, model name, input and output configurations, and data preprocessing configuration. You can click **Load Configuration** to load an existing configuration file and the corresponding configuration information will be automatically filled. |
Click 🛠 on the right to view the model graph in a visualized way, as shown in Figure 4-3. (This function is not available for MindSpore models.)

**NOTE**

- You can scroll the mouse wheel up or down to move the view upward or downward.
- You can press and hold Ctrl and scroll the mouse wheel up and down to zoom in or zoom out the view.

**Figure 4-3 Model visualizer**

- **View the operator details.**
  Select a layer. The layer is outlined green and all its operators are displayed on the top right. Click up to expand the operator details, including the name, type, and attributes of each operator. Click down to collapse the operator details. Click next to view the details of a specific metric. The information displayed is the same as that defined in the model file.

- **View the output format and shape of an operator.**
  Figure 4-3 also displays the output format and shape (for example, \(1,3,224,224\) and \(1,64,112,112\)) of each operator on the connection lines. When you move the cursor to a shape, the output format of the operator will be displayed, for example, **NCHW**.

If the selected model contains an unsupported operator, click 🛠 on the right. The failure log is generated in the **Output** window of MindStudio, specifying that the operator is not supported and its shape is not accessible. Such operator is highlighted in red in the displayed **Model Visualizer**. In this
case, you cannot obtain the output format and shape of the operator. To rectify the fault in this scenario, see Exception Handling.

- **Search for an operator.**

  On the right of Figure 4-3, click **Op Search**. The search area is displayed on the lower right, as shown in Figure 4-4.

  On the **Op Search** tab page, all the operators on the network are listed below. Select an operator, adding a green outline to the operator in the graph. The details about the operator are displayed in the upper right.

**Figure 4-4 Searching for an operator**

- **Find operator details.**

  Select a layer and click **on the upper right. All operators at this layer are displayed below. Click  or press **Ctrl+F** to find a specific operator metric in the displayed **Find** dialog box, as shown in Figure 4-5.

    - **Case sensitive**: If selected, the search operation follows the exact letter case of the entered keyword.
    - **Wrap Search**: If selected, the search operation is continued from the start of the list when the end of the list is reached, until all operators are covered.
    - **Backward**: If selected, the search operation is from current position to the start of the list.

  In the Figure 4-5 dialog box, enter the keyword to search for in the text box, select required search options, and click **Find**. The matched information in the list will be highlighted. If no match is found, the message **Value not found** is displayed. Click **Close** to exit the search.
Step 2 Click Next. The Data Pre-Processing tab page is displayed. See Figure 4-6. The tab page varies according to the version.

The data preprocessing capability is backed by the AI Pre-processing (AIPP) module of the Ascend AI Processor. It enables hardware-based image preprocessing including color space conversion (CSC), image normalization (by subtracting the mean value or multiplying a factor), image cropping (by specifying the crop start and cropping the image to the size required by the neural network), and much more.

Only when Type is set to Uint8 for Input Nodes on the Model Information tab page, the data preprocessing function can be configured for an input node on the Data Pre-Processing tab page. For a model with multiple inputs, data preprocessing can be configured when each input node can obtain the width and height from the Shape information, and Type of each input node is Uint8.
Table 4-3 describes the parameters.

**Table 4-3 Parameter description**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Pre-processing Mode</td>
<td>Sets the image preprocessing mode.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Static</strong>: static AIPP.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Dynamic</strong>: dynamic AIPP.</td>
</tr>
</tbody>
</table>

If you set **Image Pre-processing Mode** to **Static**, also configure the following parameters:

<table>
<thead>
<tr>
<th>Input Node: (data)</th>
<th>Sets the AIPP switch by node. This parameter is available only when <strong>Type</strong> of the data node is set to <strong>Uint8</strong> in the <strong>Input Nodes</strong> area, as shown in Figure 4-2. For a TensorFlow model, this parameter is available only when <strong>Type</strong> of the data node is set to <strong>FP32</strong>, <strong>Int32</strong>, <strong>Int64</strong>, or <strong>Bool</strong> in the <strong>Input Nodes</strong> area, as shown in Figure 4-2.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Node: (im_info)</td>
<td>Switches AIPP on for the second input node. This parameter is available only when the model has two inputs. This parameter is available only when <strong>Type</strong> of the im_info node is set to <strong>Uint8</strong> in the <strong>Input Nodes</strong> area (as shown in Figure 4-2) and the W and H shape information are well parsed. Note that <strong>im_info</strong> in <strong>Input Node: (im_info)</strong> varies depending on the parsed model.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Input Image Format</td>
<td>Sets the input image format.</td>
</tr>
<tr>
<td></td>
<td>If <strong>Target SoC Version</strong> in <strong>Step 1</strong> is set to <strong>Ascend310</strong>, <strong>Ascend610</strong>, <strong>Ascend615</strong>, <strong>Ascend710</strong>, or <strong>Ascend710Pro</strong>, the supported formats include: YUV420 sp, YVU420 sp, RGB package, BGR package, YUV400</td>
</tr>
<tr>
<td></td>
<td>If <strong>Target SoC Version</strong> in <strong>Step 1</strong> is set to <strong>Hi3796CV300CS</strong>, <strong>Hi3796CV300ES</strong>, or <strong>SD3403</strong>, the supported formats include: YUV420 sp, YVU420 sp, YVU422 sp, YVU422 sp, YUV400, RGB package, BGR package, ARGB package, RGBA package, ABGR package, BGRA package</td>
</tr>
<tr>
<td></td>
<td>- If the setting is <strong>YUV420 sp</strong>, <strong>YVU420 sp</strong>, <strong>YUV422 sp</strong>, or <strong>YVU422 sp</strong>, the following data types are available on the right: BT.601 (Video Range), BT.601 (Full Range), BT.709 (Video Range), BT.709 (Full Range)</td>
</tr>
<tr>
<td></td>
<td>Different data types correspond to different CSC configurations. (The CSC factors are stored in the <code>insert_op.cfg</code> configuration file after model conversion.)</td>
</tr>
<tr>
<td></td>
<td>- BT.601 is the standard for standard-definition television (SDTV).</td>
</tr>
<tr>
<td></td>
<td>- BT.709 is the standard for high-definition television (HDTV).</td>
</tr>
<tr>
<td></td>
<td>The two standards are classified into narrow range (<strong>Video Range</strong>) and wide range (<strong>Full Range</strong>) according to their representation range.</td>
</tr>
<tr>
<td></td>
<td>The representation range of the narrow range is <strong>[Y [16...235]]</strong>, and that of the wide range is <strong>[Y/Cb/Cr [0...255]]</strong>, <strong>[R/G/B [0...255]]</strong>.</td>
</tr>
<tr>
<td></td>
<td>For details about how to determine the standard of the input data, see <strong>15.5 How Do I Determine the Video Stream Format Standard When I Perform CSC on a Model Using AIPP?</strong></td>
</tr>
<tr>
<td></td>
<td>- If the setting is <strong>YUV400</strong>, CSC is not supported.</td>
</tr>
<tr>
<td></td>
<td>- If the setting is <strong>RGB package, BGR package, ARGB package, RGBA package, ABGR package</strong> or <strong>BGRA package</strong>, no input data type option is displayed on the right. After the model conversion is complete, the values of the following parameters in the data preprocessing configuration file <code>insert_op.cfg</code> vary depending on the setting of <strong>Model Image Format</strong>:</td>
</tr>
<tr>
<td></td>
<td>- If the setting is <strong>BGR package</strong>, the output is in RGB format. If the setting is <strong>RGB package</strong>, the output is in BGR format.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| # Whether to enable R/B or U/V channel before CSC | **rbuv_swap_switch**: true  
  - If the setting is **BGR package**, the output is in BGR format. If the setting is **RGB package**, the output is in RGB format.  
  **rbuv_swap_switch**: false  
  - If the setting is **ARGB package**, the output is in RGBA format. If the setting is **ABGR package**, the output is in BGRA format.  
  **rbuv_swap_switch**: false  
  - If the setting is **RGBA package**, the output is in RGBA format. If the setting is **BGRA package**, the output is in BGRA format.  |

| Input Image Resolution                | Sets the input image size.  
  If **Input Image Format** is set to **YUV420 sp**, the width and height must be even numbers. |

| Model Image Format                    | Sets the model image format.  
  The model image format varies with the input image format.  
  - If **Input Image Format** is set to **YUV420 sp**, **YUV420 sp**, **YUV422 sp**, **YUV422 sp**, or **RGB package**, **Model Image Format** can be set to **RGB** or **BGR**.  
  - If **Input Image Format** is set to **BGR package**, **Model Image Format** can be set to **RGB**, **BGR**, or **GRAY**.  
  - If the input image format is **YUV400**, **Model Image Format** can only be set to **GRAY**.  
  - If **Input Image Format** is set to **ARGB package**, **RGBA package**, **ABGR package**, or **BGRA package**, **Model Image Format** can be set to **RGBA** or **BGRA**.  
  The toggle is also the CSC switch (default to on). Turn the toggle switch on when the Input Image Format is inconsistent with the Model Image Format. |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crop</td>
<td>Sets the cropping switch (default off). If the toggle is switched on, the following two parameters are displayed:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Cropping Start</strong>: start of image cropping. The value range of <strong>Cropping Start [H][W]</strong> is narrower than that of <strong>Input Image Resolution [H][W]</strong>.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Cropping Area</strong>: size of the cropped image. The default width and height are consistent with those in the <strong>Shape</strong> text box in the <strong>Input Nodes</strong> area, as shown in <strong>Figure 4-2</strong>. The specified width and height cannot be greater than those of <strong>Input Image Resolution</strong>.</td>
</tr>
<tr>
<td></td>
<td>Note the following restrictions on image cropping:</td>
</tr>
<tr>
<td></td>
<td>• If <strong>Input Image Format</strong> is set to <strong>YUV420 sp</strong> or <strong>YVU420 sp</strong>, the values of <strong>Cropping Start [H][W]</strong> must be even numbers.</td>
</tr>
<tr>
<td></td>
<td>• If <strong>Input Image Format</strong> is set to <strong>YUV422 sp</strong> or <strong>YVU422 sp</strong>, the values of <strong>Cropping Start [W]</strong> must be even numbers.</td>
</tr>
<tr>
<td></td>
<td>• If <strong>Input Image Format</strong> is set to other values, there is no restriction on <strong>Cropping Start [H][W]</strong>.</td>
</tr>
<tr>
<td></td>
<td>• If image cropping is switched on: <strong>Input Image Resolution &gt;= Cropping Area + Cropping Start</strong></td>
</tr>
<tr>
<td></td>
<td>If image cropping is switched on with padding switched off, <strong>Cropping Area [H][W]</strong> can be set to 0 or left empty. In this case, the <strong>Cropping Area [H][W]</strong> values are obtained from the H and W shape information in the <strong>Input Nodes</strong> area (the height and width of the model input), as shown in <strong>Figure 4-2</strong>.</td>
</tr>
<tr>
<td>Padding</td>
<td>Sets the image padding switch (default off). The value range of <strong>Padding Area [L][R][B][T]</strong> is [0, 32]. Make sure the output height and width (after AIPP with padding is performed) are consistent with those required by the model.</td>
</tr>
<tr>
<td>Normalization</td>
<td>Sets the normalization switch. After this toggle is switched on, the <strong>Mean</strong>, <strong>Min</strong>, and <strong>Variance</strong> lines are displayed. <strong>Conversion Type</strong> defines the calculation rule.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td>Sets the mean value of each channel. This line is available only when <strong>Normalization</strong> is switched on.</td>
</tr>
<tr>
<td></td>
<td>- If <strong>Input Image Format</strong> is YUV420 sp, YVU420 sp, YUV422 sp, YVU422 sp, YUV400, RGB package, or BGR package, this parameter is displayed as <strong>Mean: [R][G][B]</strong>. The default mean values of the channels are 104, 117, and 123, respectively.</td>
</tr>
<tr>
<td></td>
<td>- If <strong>Input Image Format</strong> is ARGB package, RGBA package, ABGR package, or BGRA package, this parameter is displayed as <strong>Mean: [R][G][B][A]</strong>. The default mean values of the channels are 104, 117, 123, and 0, respectively.</td>
</tr>
<tr>
<td><strong>Min</strong></td>
<td>Sets the minimum value of each channel. This line is available only when <strong>Normalization</strong> is switched on.</td>
</tr>
<tr>
<td></td>
<td>- If <strong>Input Image Format</strong> is YUV420 sp, YVU420 sp, YUV422 sp, YVU422 sp, YUV400, RGB package, or BGR package, this parameter is displayed as <strong>Min: [R][G][B]</strong>. The default value is 0 for each channel.</td>
</tr>
<tr>
<td></td>
<td>- If <strong>Input Image Format</strong> is ARGB package, RGBA package, ABGR package, or BGRA package, this parameter is displayed as <strong>Min: [R][G][B][A]</strong>. The default value is 0 for each channel.</td>
</tr>
<tr>
<td><strong>Variance</strong></td>
<td>Sets the variance value of each channel. This line is available only when <strong>Normalization</strong> is switched on.</td>
</tr>
<tr>
<td></td>
<td>- If <strong>Input Image Format</strong> is YUV420 sp, YVU420 sp, YUV422 sp, YVU422 sp, YUV400, RGB package, or BGR package, this parameter is displayed as <strong>Variance: [R][G][B]</strong>. The default value is 1.0 for each channel.</td>
</tr>
<tr>
<td></td>
<td>- If <strong>Input Image Format</strong> is ARGB package, RGBA package, ABGR package, or BGRA package, this parameter is displayed as <strong>Variance: [R][G][B][A]</strong>. The default value is 1.0 for each channel.</td>
</tr>
</tbody>
</table>

If you set **Image Pre-processing Mode** to **Dynamic**, also configure the following parameters:

| Input Node: (data) | Sets the AIPP switch by node. This parameter is available only when **Type** of the data node is set to Uint8 in the **Input Nodes** area, as shown in **Figure 4-2**. |
| Input Node: (im_info) | Switches dynamic AIPP on for the second input node. This parameter is available only when the model has two inputs. This parameter is available only when **Type** of the im_info node is set to Uint8 in the **Input Nodes** area (as shown in **Figure 4-2**) and the W and H shape information are well parsed. Note that **im_info** in **Input Node: (im_info)** varies depending on the parsed model. |
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Image Size (Byte)</td>
<td>Sets the maximum size of the input image. Required in the dynamic AIPP scenario. (In the dynamic batch size scenario, $N$ is set to the maximum batch size.)</td>
</tr>
</tbody>
</table>

- If **Input Image Format** is set to **YUV400_U8**: Max Image Size $\geq N \times \text{Input Image Resolution [W]} \times \text{Input Image Resolution [H]} \times 1$
- If **Input Image Format** is set to **YUV420SP_U8**: Max Image Size $\geq N \times \text{Input Image Resolution [W]} \times \text{Input Image Resolution [H]} \times 1.5$
- If **Input Image Format** is set to **XRGB8888_U8**: Max Image Size $\geq N \times \text{Input Image Resolution [W]} \times \text{Input Image Resolution [H]} \times 4$.
- If **Input Image Format** is set to **RGB888_U8**: Max Image Size $\geq N \times \text{Input Image Resolution [W]} \times \text{Input Image Resolution [H]} \times 3$

---

**NOTICE**

- Note the following requirements on the input image format for model inference:
  
  If AIPP is switched on in model conversion, model inference requires the NHWC inputs. In this case, the input node configured with AIPP will be changed accordingly, which may be different from that specified by **Input Format** on the **Model Information** tab page in **Step 1**.

---

**Step 3** Click **Next**. The **Advanced Options Preview** tab page is displayed. See **Figure 4-7**. The tab page varies according to the version.
The parameters are described as follows.

Table 4-4 Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Operator Fusion** | Sets the fusion switch.  
  ● Switch the toggle on to disable the fusion function.  
  ● Switch the toggle off to enable the fusion function. The default setting is off.  
  **NOTE**  
  When the fusion function is disabled, only the fusion patterns specified in the configuration file are disabled. For details about the available fusion patterns that can be disabled, see **Graph Fusion and UB Fusion Patterns**. Other fusion patterns cannot be disabled due to the system mechanism.  

To convert a model quantized using AMCT into an offline model (.om) and analyze the model accuracy, switch the toggle on. When the toggle is switched on, the **fusion_switch.cfg** configuration file is generated in the same directory as the .om model file, which records the features that are disabled. |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto Tune Mode</td>
<td>Sets the operator auto tuning mode. When using this function, ensure that MindStudio and operating environment are deployed on the same server. Determines whether to tune the TBE operator to maximize the the Ascend AI Processor performance.</td>
</tr>
<tr>
<td></td>
<td><strong>Genetic Algorithm</strong> (GA): tunes Cube operators.</td>
</tr>
<tr>
<td></td>
<td><strong>Reinforcement Learning</strong> (RL): tunes Vector operators.</td>
</tr>
<tr>
<td></td>
<td><strong>Genetic Algorithm and Reinforcement Learning</strong>: tunes Cube and Vector operators.</td>
</tr>
<tr>
<td></td>
<td>Note the following:</td>
</tr>
<tr>
<td></td>
<td><strong>GA tuning:</strong></td>
</tr>
<tr>
<td></td>
<td>– The <code>{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/opf/data/tiling/${soc_version}/built-in/</code> directory stores the built-in repository and cost model. If a model shape does not hit the built-in repository, tuning is initiated.</td>
</tr>
<tr>
<td></td>
<td>– The <code>{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/atc/data/tiling/${soc_version}/custom/</code> directory stores the repository generated after tuning. If a repository exists in this repository, the repository will be updated. If no repository exists in the directory, a new repository is created.</td>
</tr>
<tr>
<td></td>
<td>Tiling policies are read from the custom repository during model build only when <strong>Auto Tune Mode=GA</strong> or <strong>Auto Tune Mode=&quot;RL,GA&quot;</strong> is included in the model conversion command. Otherwise, Auto Schedule is used during model build.</td>
</tr>
<tr>
<td></td>
<td><strong>RL tuning:</strong></td>
</tr>
<tr>
<td></td>
<td>– The built-in repository is generated to the <code>{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/opf/data/rl/${soc_version}/built-in/</code> directory, which contains the tiling policies of frequently used shapes.</td>
</tr>
<tr>
<td></td>
<td>– After tuning, a new repository (which outperforms the built-in repository or when no built-in repository is available) is generated to the <code>{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/atc/data/rl/${soc_version}/custom/</code> directory.</td>
</tr>
</tbody>
</table>
### Parameter Description

**NOTE**
- If a repository exists in the **custom** directory and the operator logic is changed (for example, the support for ND input is added to the GEMM operator), you need to set the following environment variable in the MindStudio installation server and perform tuning again.
  ```
  export REPEAT_TUNE=True
  ```
- When Auto Tune is enabled, memory allocation is needed for board evaluation. The Auto Tune tool may use more memory than the ATC tool. The required memory is related to the device count used at the same time and can be estimated as follows: \( 2 \times \text{Device count} \times \text{Input data size} \). If the memory size exceeds the size of the ATC working memory, tuning fails.
- During GA auto tuning, the device resource is exclusively occupied. Therefore, other operations that require the device resource are not allowed. If the tuning fails, stop other processes and perform the tuning again. If there are build services running on the host, the tuning duration is affected.

**Additional Arguments**
Adds additional ATC command-line arguments. Enters space-delimited arguments that are not available on the GUI but supported by the ATC tool. A maximum of 2048 characters are allowed.

For details about the available arguments such as `--log=info`, see "Command-line Options" in ATC Instructions.

If the GUI parameters and corresponding command-line arguments in the ATC tool are configured for the same function, the command-line arguments will take effect with a higher priority.

**Command Preview**
Previews the ATC arguments configured for model conversion. This setting cannot be modified.

After required parameters are configured on all tab pages, you can preview the UI configuration in the CLI. Assume that Auto Tune Mode is set to Genetic Algorithm.

In this case, `--auto_tune_mode="GA"` will be displayed in Command Preview.

If you set an existing parameter in Additional Arguments, for example, `--auto_tune_mode="GA,RL"`, the corresponding command-line argument is added to Command Preview. During model conversion, the recently added argument in Command Preview will overwrite the existing one.

**Step 4** Click **Finish** to start model conversion.

The model conversion log records are printed to the **Output** window in the lower part of MindStudio. If the message "Model converted successfully" is displayed, the model conversion is complete. The **Output** window also displays the model conversion commands, environment variables, model conversion result, model output path, and model conversion log path.
Step 5  After the model conversion is complete, find the generated .om model file and $modelname_config.json configuration file in the $HOME/modelzoo/resnet50/device directory in the operating environment.

- If Data Preprocessing is switched on, the data preprocessing configuration file insert_op.cfg is generated in the same directory as the .om file.
- If Operator Fusion is switched off, the fusion_switch.cfg configuration file is generated in the same directory as the .om model file to record the functions that are disabled.

Find the model conversion log file ModelConvert.txt in $HOME/modelzoo/resnet50.

```
hisisoc@dgphicprd32833:~/modelzoo/resnet50/device$ ll
total 150636
drwxr-x--- 2 hisisoc hisisoc      4096 Mar 10 16:46 ./
drwxr-x--- 3 hisisoc hisisoc      4096 Mar 10 16:45 ../
-rw-r----- 1 hisisoc hisisoc        98 Mar 10 15:55 fusion_switch.cfg --Fusion switch configuration file
-rw-r----- 1 hisisoc hisisoc       453 Mar 10 16:45 insert_op.cfg --Data preprocessing configuration file
-rw------- 1 hisisoc hisisoc      2095 Mar 10 18:03 resnet50_config.json --Model conversion configuration file. You can load this file to use the same configuration in future model conversion.
-rw------- 1 hisisoc hisisoc      51581408 Mar 10 16:46 resnet50.om --Model file running on the board
```

----End

Exception Handling

- **Symptom**

  If the selected model file contains operators unsupported by the Ascend AI Processor, the network analysis report is displayed during model conversion, as shown in Figure 4-8.

**Figure 4-8 Network analysis report**

In the Summary area on the left:

- **All Operator**: indicates the number of operators in the model to be converted. The unsupported operators are also counted.

  Click **All Operator**. In the Result Details area on the right, details about all operators of the model are displayed, including the operator type, operator name, and parsing result. If the operator fails to be parsed, the cause of the parsing failure is displayed in Description.

- **Unsupported Operator**: indicates the number of operators that are not supported by model conversion. Below will display the classification and specific reasons.

  Click **Unsupported Operator**. All unsupported operators are displayed on the right.

- **Solution**

  a. In the network analysis report window shown in Figure 4-8, choose Result Details on the right. If failed is shown in the Result column for an operator, the operator information is selected. Click one of solutions.
under **Operation**, for example, **Creator Operator**, to create a custom operator project.

If an operator project has been opened, a dialog box is displayed, as shown in **Figure 4-9**. You can add an operator to the current project or create a new project. If no operator project exists, a dialog box for creating an operator project is displayed.

For details about how to create a custom operator project, see **8.4 Project Creation**.

**Figure 4-9** Message displayed upon the creation of an operator project

![Figure 4-9](image)

b. Create the custom operator project by taking the given procedure.

**Operator Type** custom operator in **New Project > Ascend Operator** is automatically filled based on the operator type selected in the network analysis report. After the project is created, it is stored in **$HOME/AscendProjects** by default.

The directory structure and main files of the operator project are as follows:

```
├── .idea
├── build                                  //Intermediate files generated after build
└── cmake                                  //Directory of public files related to build
    └── framework                           //Directory of the operator plugin implementation files
        └── tf_plugin                         //Directory of the operator plugin files and build rule files of
            └── tensorflow_add_plugin.cpp
```

c. Develop the custom operator. For details, see **8 Custom Operator Development**.

After the custom operator is developed, try to convert the model again.

### 4.3 Visualizing a Model

The **Model Visualizer** of MindStudio allows you to view the network topology and operators of a converted offline model (.om) in a visualized way.

**Step 1** Choose **Ascend > Model Visualizer**, or click ![Model Visualizer Icon]. In the dialog box shown in **Figure 4-10**, select a model file to be visualized, for example, **resnet50**.
Figure 4-10 Selecting a model to be visualized

Step 2  Choose resnet50 > device, select the resnet50.om model file that has been converted in 4.2 Model Conversion, and click OK. The Model Visualizer dialog box is displayed, as shown in Figure 4-11.

Figure 4-11 Model Visualizer

- View the operator details.
  All operators in the model are displayed, as shown in Figure 4-11. Select an operator. The operator is outlined green and all its details are displayed in the top right area, including the operator name and the inputs and outputs. Click
to expand the details. Click ▶ to view the details of a specific metric. Click ✅ to collapse the details.

The general view of the model is displayed in the lower right corner, including the context (outlined in blue) of the selected operator on the network. The detailed view and the general view move and zoom synchronously.

**NOTE**

- You can scroll the mouse wheel up or down to move the view upward or downward.
- You can press and hold Ctrl and scroll the mouse wheel up and down to zoom in or zoom out the view.

- **View the output format and shape of an operator.**

  Figure 4-11 also displays the output format and shape (for example, 1,224,224,4) of each operator on the connection lines. You can move the cursor to a shape, the output format of the operator is shown, for example, NCHW and NC1HWC0.

- **Search for an operator.**

  On the right of Figure 4-11, click Op Search. The search area is displayed on the lower right, as shown in Figure 4-12.

**Figure 4-12 Searching for an operator**

All operators on the network are listed in the search area. Enter an operator name in the search text box. The matched operators are displayed. Select an operator. The operator is outlined green and its details are displayed in the top right area, including the operator name and the inputs and outputs.

- **Find operator details.**

  Select a layer and click ▲ on the upper right. All operators at this layer are displayed below. Click ☐ or press Ctrl+F to find a specific operator metric in the displayed Find dialog box, as shown in Figure 4-13.

  - **Case sensitive**: If selected, the search operation follows the exact letter case of the entered keyword.
  - **Wrap Search**: If selected, the search operation is continued from the start of the list when the end of the list is reached, until all operators are covered.
  - **Backward**: If selected, the search operation is from current position to the start of the list.
In the Figure 4-13 dialog box, enter the keyword to search for in the text box, select required search options, and click Find. The matched information in the list will be highlighted. If no match is found, the message Value not found is displayed. Click Close to exit the search.

Figure 4-13 Finding operator details

- **View the model output nodes.**
  During model conversion, the operators `res2a_branch2a`, `bn2a_branch2a`, `scale2a_branch2a`, and `res2a_branch2a_relu` are selected as the output nodes of the model, as shown in Figure 4-14. You can click the TransData operator to view the input details.

Figure 4-14 Differences before and after operator selection

- **View the data types of the model output nodes.**
  If the output type of a node is set by specifying the Output Nodes in the Model Information tab page in 4.2 Model Conversion, you can click the NetOutput node to view the data types of the input nodes after model conversion is complete.
  
  If only Output Type in the Model Information tab page is set during model conversion and no node is selected as output, click the NetOutput node to view the input data type after model conversion is complete.
  
  If `dtype` is DT_FLOAT, the output data type is FP32. If `dtype` is DT_FLOAT16, the output data type is FP16. If `dtype` is DT_UINT8, the output data type is UINT8.
Figure 4-15 Viewing the data types of the model output node
5 Model Training

5.1 Overview

MindStudio runs the training framework, submits the script, dataset, and parameters executed by the framework to GE, and instructs GE to perform network analysis and output analysis results through APIs. MindStudio displays the network analysis results on the GUI. You can trigger the custom operator process to operators that are not supported on the network. In the custom operator process, the operator information library, operator prototype library, and operator plugins can be automatically generated. The MindStudio-based training process is shown in Figure 5-1.

NOTE

NeoKylin OS 7.6-AArch64 provides only the entry for creating and configuring a training project, but does not support model training.
Before installing the TensorFlow framework, ensure that the framework plugin package `Ascend-cann-tfplugin_xxx.run` has been installed in the operating environment. For details about the installation method, see “Installing the Framework Plugin Package” in the CANN Software Installation Guide.

5.2 Procedure

Creating a Training Project

Step 1
Navigate to the New Project dialog box, as shown in Figure 5-2.

- If this is your first login to MindStudio, click Create new project on the MindStudio welcome window to go to the New Project dialog box.
- If this is not your first login to MindStudio, choose File > New > Project... from the menu bar to go to the New Project dialog box.
Step 2  Create a training project.

1. Choose **Ascend Training** on the navigation bar on the left, as shown in Figure 5-2. On the right, configure the training project information, as shown in Table 5-1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Project name (user-defined). Enter a maximum of 64 characters, starting with a letter. Only letters, digits, hyphens (-), and underscores (_) are allowed.</td>
<td>MyTraining</td>
</tr>
<tr>
<td>Description</td>
<td>Project description (user-defined).</td>
<td>Optional.</td>
</tr>
<tr>
<td>ADK Version</td>
<td>Version number of the ADK.</td>
<td>Select the current ADK version.</td>
</tr>
<tr>
<td>Project Location</td>
<td>Project path.</td>
<td>Use the default value.</td>
</tr>
</tbody>
</table>

2. Click **Next**.

3. **Figure 5-3** shows the page for selecting a training project. Select a framework project under **Templates** or select a built-in sample under **Samples**. In the following figure, a framework project under **Templates** is selected. Click **Finish** to complete the training project creation.
Note

If the TensorFlow project or a sample is used, ensure that the framework plugin package `Ascend-cann-tfplugin_xxx.run` has been installed in the local or remote operating environment for model training. For details about the installation method, see "Installing the Framework Plugin Package" in the CANN Software Installation Guide.

4. If there is already an active project in the window, the message shown in Figure 5-4 is displayed.
   - Clicking This Window opens the created project in the current window.
   - Clicking New Window opens the created project in a new window.

Figure 5-4 Prompt for opening a new project

| NOTICE |

The default path of the new project is under `AscendProjects` in the home directory of the installation user.

Step 3 View the directory structure and main files of the training project.

```
├── .idea
├── data                                  // Dataset directory
├── .project                                 // Project information file, including the project type, project description, target device type, and ADK version
└── train.py                                // Training script file, which is an empty file. You can create a training
```
Before executing a training project, you need to manually upload the dataset to the training machine. The dataset directory is user-defined.

---End

**Importing a Training Project**

If a training project exists, you do not need to create one. You can directly import the training project through MindStudio. The procedure is as follows:

**Step 1** Use MindStudio to import the training project.
- If you log in to MindStudio for the first time, click **Open or Import** on the MindStudio welcome page, select the project to be imported, and click **OK**.
- If it is not your first login to MindStudio, choose **File > New > Open...** from the top menu bar, select the project to be imported, and click **OK**.

**Step 2** If there is already an active project in the window, the message shown in **Figure 5-5** will be displayed.
- Click **This Window** to open the created project in the current window.
- Clicking **New Window** opens the created project in a new window.

**Figure 5-5** Prompt for opening a new project

**Step 3** If MindStudio detects that the .project project configuration file does not exist, the window shown in **Figure 5-6** is displayed, helping you create the .project file.

**Figure 5-6** Importing project configurations

The parameters in the window are described as follows. You can select the parameters based on the actual scenario.

- **Type**: The options are Ascend App (application project) and Ascend Training (training project). The default value is Ascend App. Select Ascend Training for the training project.
● **Framework**: Select Caffe, Pytorch, MindSpore, or TensorFlow as required.
● **Project Desc**: project description. The value is **Training Project** for a training project.

**Step 4** Click **Create** to create the .project file.

If you click **Cancel**, the project configuration file will not be properly created and MindStudio cannot identify the project type. In this case, the dialog box shown in **Figure 5-7** is displayed. After you click **OK** in the dialog box, the imported project file can still be opened. When you import the project file next time, the dialog box shown in **Figure 5-6** is displayed to help you create the project.

**Figure 5-7** Unsupported project type

---End

**Setting Run Configuration**

**Step 1** Set running parameters.

1. Choose **Run > Edit Configurations...** or click the menu shown in **Figure 5-8**. The **Run Configurations** dialog box is displayed.

**Figure 5-8** Shortcut menu of the Edit Configurations dialog box

2. In the **Run Configurations** dialog box, click + and select **Ascend Training**. The **Run/Debug Configurations** dialog box is displayed.
   - If **Run Mode** is set to **Remote Run**, the configuration window shown in **Figure 5-9** is displayed.
If Run Mode is set to Local Run, the configuration window shown in Figure 5-10 is displayed.

Set run configurations of the training project on the right, as described in Table 5-2.
Table 5-2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Project name (user-defined). Enter a maximum of 64 characters, starting with a letter. Only letters, digits, hyphens (-), and underscores (_) are allowed.</td>
<td>MyTraining</td>
</tr>
<tr>
<td>Executable</td>
<td>Entry file of the training project.</td>
<td>/home/xxx/AscendProjects/MyTraining5/data/download_cifar10.sh</td>
</tr>
<tr>
<td>Run Mode</td>
<td>Run mode, either Remote Run (default) or Local Run.</td>
<td>Remote Run</td>
</tr>
<tr>
<td>SSH Connection</td>
<td>IP address of the remote training server, which is configured by choosing Ascend &gt; SSH Configurations from the main menu.</td>
<td>Optional</td>
</tr>
<tr>
<td>Command Arguments</td>
<td>Command-line arguments for training.</td>
<td>Optional</td>
</tr>
<tr>
<td>Environment Variables</td>
<td>Environment variables of the training project.</td>
<td>Optional</td>
</tr>
</tbody>
</table>

3. Click OK, and training project information is created.

**Step 2** Upload the dataset, create a training script file, and create the execution entry file.

1. Upload the dataset to the remote training server. The dataset import path in the training script must be the same as the path of the uploaded dataset.
2. Create an executable file and a training script.
3. Create the execution entry file `train.py`. The following is a simple example:
   ```python
   if __name__ == '__main__':
       val = os.system('./npu.sh npu.py')
   
   npu.sh is the executable file of the training script created by the user, and npu.py is the training code. For details about how to write the executable of the npu.sh script, see "Script Execution" in Manual Porting and Training.
4. After the preceding steps, the training project directory is shown in the following as an example.
   ```bash
   ├── .idea
   │     └── data                                  // Dataset directory
   │          └── .project                          // Project information file, including the project type, project
description, target device type, and ADK version
   │          └── train.py                         // Training script file, which is an empty file. You can create a
training script here.
   │          └── npu.py                            // Training script, which is created by users
   ```
Performing Training

Step 1  Choose Run > Run on the application project page or click the button shown in Figure 5-11 to perform training. The process is as follows: Connect to the remote training machine -> Package the training project and release it to the remote training machine -> Start the training project.

Figure 5-11 Performing training using a shortcut

Step 2  After performing the training, the real-time runtime information is displayed in the Run window at the bottom of the dialogue box, as shown in the following figure.

Figure 5-12 Real-time runtime information

Step 3  If the training succeeds, a corresponding message is displayed. Otherwise, the network_analysis_timestamp.report of the training project is generated in out/reports in the root directory of the project. The report content is as follows:
5.3 MindSpore Model Training Sample

This section demonstrates how to import a MindSpore model training script from ModelZoo in the Ascend Developer Zone to MindStudio for training.

5.3.1 Setting Up the Environment

Environment Setup

Before training a model in MindStudio, you need to prepare the environment as follows:

- Set up the remote operating environment by referring to the MindSpore Installation Guide. Users who use the Kylin OS V10 SP1-AArch64 architecture can set up a remote environment by referring to the CentOS-AArch architecture.
- Install MindStudio by referring to CANN Software Installation Guide (msInstaller).

---

CAUTION

- The MindSpore model training scripts provided in ModelZoo in the Ascend Community are based on the hardware backends equipped with Ascend AI Processors. Make sure to set up the Ascend 910 MindSpore operating environment. After the MindSpore operating environment is set up, configure the software environment variables corresponding to Ascend 910 AI Processors.
- MindSpore is installed in a remote environment while MindStudio is installed on the local host.
5.3.2 Training with the ResNet-50 Sample

The following describes the overall procedure for creating a training project with the ResNet-50 training sample. For details about the project information and related pop-up windows, see 5.2 Procedure.

**Step 1** Create an Ascend Training project.

Configure the project information based on the site requirements. Click Next.

![Figure 5-14 New Project dialog box](image)

**Step 2** In the Create Training Project From Template window shown in Figure 5-15, select ResNet-50 in the Samples area and click Finish.

![Figure 5-15 Selecting a training project](image)
**Step 3** View the ResNet-50 template project window as shown in Figure 5-16.

*Figure 5-16 Template project window*

**Step 4** Run the `download_cifar10.sh` script in the `data/` directory to download the CIFAR-10 dataset. The operation is as follows:

1. Go to the `data/` subdirectory in the project directory.
   ```
   cd {Project Location}/MyTraining1/data
   ```
   Where, `{Project Location}` indicates the user-defined path for storing the project, which is set in Step 1. In this example, the path is `/home/mindspore/AscendProjects`, and `MyTraining1` is the project name set in Step 1.

2. Run the following commands to download the dataset:
   ```
   chmod +x download_cifar10.sh  # Grant the execute permission on the script.
   ./download_cifar10.sh
   ```

   *Figure 5-17* shows the downloaded dataset.

*Figure 5-17 Dataset directory*

```
- data
  - cifar-10-batches-bin
    - cifar-10-binary.tar.gz
    - download_cifar10.sh
```

- **NOTE**
  - You can also download the CIFAR-10 dataset by yourself and copy it to the `data/` directory in the local environment or the remote operating environment. Ensure that the dataset import path in the training script is the same as the uploaded dataset path.
  - To download the dataset by using the script, ensure that the local environment where MindStudio is located is connected to the Internet.

**Step 5** Set the run configurations and run the project.

1. Choose Run > Edit Configurations... or click the menu shown in *Figure 5-18*. The Run Configurations dialog box is displayed.
2. In the Run Configurations dialog box, click + and select Ascend Training. The Run/Debug Configurations dialog box is displayed, as shown in Figure 5-19.

Set run configurations of the training project on the right, as described in Table 5-3.

Table 5-3 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Project name (user-defined). Enter a maximum of 64 characters, starting with a letter. Only letters, digits, hyphens (-), and underscores (_) are allowed.</td>
<td>MyTraining</td>
</tr>
<tr>
<td>Executable</td>
<td>Entry file of the training project.</td>
<td>scripts/run_standalone_train.sh</td>
</tr>
<tr>
<td>Run Mode</td>
<td>Run mode.</td>
<td>Local Run</td>
</tr>
<tr>
<td>SSH Connection</td>
<td>IP address of the remote training server, which is configured by choosing Ascend &gt; SSH Configurations from the main menu.</td>
<td>In this example, Run Mode is set to Local Run. Therefore, this parameter is not displayed.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Command Arguments</td>
<td>Command-line arguments for training.</td>
<td>resnet50 cifar10 ../data/cifar-10-batches-bin</td>
</tr>
<tr>
<td>Environment</td>
<td>Environment variables of the training project.</td>
<td>Optional. Set it as required.</td>
</tr>
<tr>
<td>Variables</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTE**

If you want to use MindInsight to visualize the training process, see 5.4 Visualized Training with MindInsight.

3. Click **OK**, and training project information is created.
4. Choose Run > Run on the application project page or click the button shown in Figure 5-20 to perform training.

**Figure 5-20** Performing training using a shortcut

![Performing training using a shortcut](image)

**Figure 5-21** shows the training process.

**Figure 5-21** Training process display

![Training process display](image)

**Step 6** Perform evaluation using the model file (.ckpt) generated after the training is complete as follows.

1. Choose Run > Edit Configurations... or click the menu shown in Figure 5-22. The Run Configurations dialog box is displayed.

**Figure 5-22** Shortcut menu of the Edit Configurations dialog box

![Shortcut menu of the Edit Configurations dialog box](image)
2. In the **Run Configurations** dialog box, click + and select **Ascend Training**. The **Run/Debug Configurations** dialog box is displayed, as shown in **Figure 5-23**.

**Figure 5-23 Run/Debug Configurations dialog box**

Set run configurations of the training project on the right, as described in **Table 5-4**.

**Table 5-4 Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Project name (user-defined). Enter a maximum of 64 characters, starting with a letter. Only letters, digits, hyphens (-), and underscores (_) are allowed.</td>
<td>MyTraining</td>
</tr>
<tr>
<td>Executable</td>
<td>Entry file of the training project.</td>
<td>scripts/run_eval.sh</td>
</tr>
<tr>
<td>Run Mode</td>
<td>Run mode.</td>
<td>Local Run</td>
</tr>
<tr>
<td>SSH Connection</td>
<td>IP address of the remote training server, which is configured by choosing <strong>Ascend &gt; SSH Configurations</strong> from the main menu.</td>
<td>In this example, <strong>Run Mode</strong> is set to <strong>Local Run</strong>. Therefore, this parameter is not displayed.</td>
</tr>
<tr>
<td>Command Arguments</td>
<td>Command-line arguments for training.</td>
<td>resnet50 cifar10 ../data/cifar-10-batches-bin train/model/resnet-1-1875.ckpt</td>
</tr>
<tr>
<td>Environment Variables</td>
<td>Environment variables of the training project.</td>
<td>Optional. Set it as required.</td>
</tr>
</tbody>
</table>
If you want to use MindInsight to visualize the training process, see 5.4 Visualized Training with MindInsight.

3. Click OK, and training project information is created.

4. Choose Run > Run on the application project page or click the button shown in Figure 5-24 to perform validation.

**Figure 5-24** Performing validation using a shortcut

5. After the evaluation is complete, an .air file is generated to the `ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/ProjectName/Proj...
Step 2  In the **Create Training Project From Template** window shown in **Figure 5-26**, select **MindSpore Project** in the **Templates** area and click **Finish**.

**Figure 5-26 Create Training Project From Template**

Step 3  Import the MindSpore training script sample.

Obtain the training script by referring to **Sample Obtaining**. Extract the script and copy all files in the **lenet** folder to the project directory created in **Step 1**. In this example, the project directory is `{Project Location}/lenet-mindspore`, where `{Project Location}` indicates the project directory path created in **Step 1**. **Figure 5-27** shows the directory structure of the imported project.

**Figure 5-27 Sample project**
Step 4 Prepare a dataset.

Download the LeNet dataset from [http://yann.lecun.com/exdb/mnist/](http://yann.lecun.com/exdb/mnist/) and save it to the `train` and `test` subdirectories in the `data` directory, as shown in Figure 5-28.

Figure 5-28 Dataset directory

![Dataset directory](image)

**NOTE**
- You need to manually create the `train` and `test` subdirectories.
- If the data set is small, copy it to the `data` directory on the local host. If the dataset is large, upload the dataset to the remote operating environment and ensure that the dataset import path in the training script is the same as the path of the uploaded dataset.

Step 5 Set the run configurations and run the project.

1. Choose Run > Edit Configurations... or click the menu shown in Figure 5-29. The Run Configurations dialog box is displayed.

![Shortcut menu of the Edit Configurations dialog box](image)

*Figure 5-29 Shortcut menu of the Edit Configurations dialog box*

2. In the Run Configurations dialog box, click + and select Ascend Training. The Run/Debug Configurations dialog box is displayed, as shown in Figure 5-30.
Set run configurations of the training project on the right, as described in Table 5-5.

### Table 5-5 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Project name (user-defined). Enter a maximum of 64 characters, starting with a letter. Only letters, digits, hyphens (-), and underscores (_) are allowed.</td>
<td>lenet_mindspore</td>
</tr>
<tr>
<td>Executable</td>
<td>Entry file of the training project.</td>
<td>train.py (as used in this sample)</td>
</tr>
<tr>
<td>Run Mode</td>
<td>Run mode.</td>
<td>Remote Run</td>
</tr>
<tr>
<td>SSH Connection</td>
<td>IP address of the remote training server, which is configured by choosing Ascend &gt; SSH Configurations from the main menu.</td>
<td>Select from existing SSH connections.</td>
</tr>
<tr>
<td>Command Arguments</td>
<td>Command-line arguments for training.</td>
<td><code>--data_path='./data' --device_target='Ascend' --ckpt_path='./'</code></td>
</tr>
<tr>
<td>Environment Variables</td>
<td>Environment variables of the training project.</td>
<td>Optional. Set it as required.</td>
</tr>
</tbody>
</table>

**NOTE**

If you want to use MindInsight to visualize the training process, see 5.4 Visualized Training with MindInsight.

3. Click **OK**, and training project information is created.
4. Choose Run > Run on the application project page or click the button shown in Figure 5-31 to perform training.

![Figure 5-31 Performing training using a shortcut](image)

Figure 5-31 shows the training process.

![Figure 5-32 Training process display](image)

**Step 6** After the training is complete, a .ckpt model file is generated. Use the model file for evaluation. For details, see 5.3.2 Training with the ResNet-50 Sample.

---End

### 5.4 Visualized Training with MindInsight

**Overview**

MindInsight provides easy-to-use tuning and debugging capabilities for MindSpore. It records data including scalars, tensors, graphs, computational graphs, model hyperparameters, and training elapsed time, and displays and analyzes data in MindInsight visualization window during training.

**Installation**

MindInsight is an independent component of MindStudio. Install it by referring to the installation guide.

**NOTICE**

- The versions of MindInsight and MindSpore must be the same.
- Kylin OS V10 SP1-AArch64 and NeoKylin OS 7.6-AArch64 do not support the MindInsight component.
The default installation path is `/usr/local/python3.7.5/bin/`. You can also change it as required.

Set the following environment variable:

1. Open the `~/.bashrc` file in the file system:
   ```bash
   vi ~/.bashrc
   ```
   Append the following environment variable to the file:
   ```bash
   export PATH=/usr/local/python3.7.5/bin:$PATH
   ```
   In the preceding command, `/usr/local/python3.7.5/bin` is an example of the installation path. Replace it as required.
   Run the `:wq!` command to save the modification and exit.

2. Run the following command to make the environment variable take effect:
   ```bash
   source ~/.bashrc
   ```

3. Check whether the installation is successful.
   Run the following command:
   ```bash
   mindinsight start
   ```
   If the following message is displayed, the installation is successful:
   ```markdown
   Web address: http://127.0.0.1:8080
   service start state: success
   ```

4. Restart MindStudio to enable MindInsight.

**Procedure**

MindStudio provides a configuration page for MindInsight. You can set MindInsight parameters on the configuration page. You can use MindStudio to create a MindSpore-based training project. After the training project is created, you can start MindInsight at any time to view the visualized training information.

**Step 1** Modify the training script.

Before using MindInsight, modify the training script to import `SummaryCollector` to collect and record the training data, and specify the directory for storing the data. Add the following lines to the training script.

```python
#Add the following code.
from mindspore.train.callback import SummaryCollector
# define summary
summary_dir = "summary_dir"
summary_collector = SummaryCollector(summary_dir=summary_dir,collect_freq=32)
```

In the preceding code, `summary_dir` indicates the path relative to that specified in the training script.

**Step 2** Access the MindInsight management page.

On the toolbar, choose **Ascend > MindInsight**. The MindInsight management page is displayed. Multiple MindInsight training visualization projects can be displayed and managed on the MindInsight management page. Table 5-6 describes the parameters on the MindInsight management page.
Figure 5-33 MindInsight management page

Table 5-6 Attributes on the management page

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run Config</td>
<td>Running configuration information of a training project</td>
</tr>
<tr>
<td>Run Type</td>
<td>Running type of a training project</td>
</tr>
<tr>
<td>MindInsight Status</td>
<td>MindInsight status</td>
</tr>
<tr>
<td></td>
<td>• ENABLE: enabled</td>
</tr>
<tr>
<td></td>
<td>• DISABLE: disabled</td>
</tr>
</tbody>
</table>

You can select a training project to be visualized on the page shown in Figure 5-33 and click Enable to open the training project and configure the parameters of the MindInsight component.

Step 3 Configure parameters for the MindInsight component.

You can configure training visualization parameters on the page shown in Figure 5-34 based on the site requirements.
Figure 5-34 MindInsight Enable parameter configuration page

Table 5-7 Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Mandatory/Optional</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary Base Dir</td>
<td>Root directory for loading training log data</td>
<td>Mandatory</td>
<td>Example: <code>scripts/train/summary_dir</code></td>
</tr>
<tr>
<td>Config</td>
<td>Configuration file and configuration module</td>
<td>Optional</td>
<td>Physical file path ({path}/config.py) or recognizable Python module path (path.to.config.module)</td>
</tr>
<tr>
<td>WorkSpace</td>
<td>Working directory of the MindInsight component</td>
<td>Optional</td>
<td>Default value: <code>SHOME/mindsight</code>. You can run the <code>MindInsight start</code> command to view the current path.</td>
</tr>
<tr>
<td>Port</td>
<td>Data transfer port</td>
<td>Optional</td>
<td>Value range: 1 to 65535. Default value: 8080</td>
</tr>
</tbody>
</table>
### Step 4
Click OK to complete the parameter configuration of the MindInsight component. The page automatically returns to Figure 5-33 and displays the current status of MindInsight.

### Step 5
View visualized training data.

If the value of MindInsight Status is ENABLE, MindInsight visualization is enabled. You can click View to view the training logs of the training project. Open the WebUI and view real-time training data.

Alternatively, click Disable to stop the MindInsight visualization process.

#### NOTE

To view real-time training data, you need to use a browser. If no browser is available, install one.

---End
5.5 Offline Inference After MindSpore Model Conversion

This section describes how to perform inference for models trained by MindSpore on Ascend AI Processors.

**Step 1** Open a project.

Copy the acl_resnet50 project in the `MindStudio/samples/inference` directory in the MindStudio installation path to the specified directory and choose **File > Open...** from the toolbar to open the project.

If the message "ADK or Target is not installed for the project." is displayed, click **OK** and reconfigure ADK.

**Step 2** Prepare data.

Obtain the model file (`resnet50.air`) of the ResNet-50 network by referring to **5.3.2 Training with the ResNet-50 Sample** and upload the obtained file to the MindStudio installation server as the MindStudio installation user.

**Step 3** Convert your model.

Before adding a model file, convert the third-party model to an offline model adapted to the Ascend AI Processor (.om). For details about model conversion, see **4.2 Model Conversion**.

1. On the menu bar, choose **Ascend > Model Convert**.
2. In the **Model Information** window, set parameters by referring to **Figure 5-35**.
Figure 5-35 Model Information

- **Model File**: Use the path of the model file (`resnet50.air`) obtained in Step 2. Select the directory of the last level.
- **Model Name**: Retain its default value `resnet50`. To use another name, you need to change the value of `omModelPath` in `src/sample_process.cpp` of the current project before building.

```cpp
const char* omModelPath = "./model/resnet50.om";
```
- **Input Nodes**: In this example, the data type of the model input node is FP32. Therefore, **Type** in **Input Nodes** must be set to **FP32** (float32).

3. Click **Next** to switch to the **Data Pre-Processing** window. Do not enable **Data Pre-Processing** (disabled by default). This example does not support AIPP. Click **Next** to switch to the **Advanced Options Preview** window, and click **Finish** to finish the conversion.

4. After the model conversion is complete, the `resnet50.om` file is generated in `/modelzoo/resnet50/device` in the MindStudio installation user home directory on the MindStudio installation server.

**Step 4** Add one or more model files.

1. Right-click the project name and choose **Add Model** from the shortcut menu.
2. In the pop-up window, select one or more .om files.
   
   Note the following restrictions when adding models:

   - The MindStudio installation user must have the permission to access the .om files.
   - You can add a maximum of 1024 .om files. The size of a single .om file must be within 1 GB, and the total size of all .om files must be less than 10 GB.
The .om file name can consist of up to 64 characters, including letters, digits, and the following special characters: .-_.

3. Click OK.

You can view the added model files in the Project window.

![Tree View]

**Step 5** Prepare inference data.

1. Log in to the MindStudio installation server as the MindStudio installation user.

2. Switch to the directory for storing the data file, that is, the data directory of the acl_resnet50 project file in **Step 1**. Replace it based on site requirements.

3. Run the following command to run the transferPic.py script to convert the .jpg files to the .bin files and resize the images from 1024 x 683 to 224 x 224. Two .bin files are generated in the data directory.

   ```bash
   python3.7.5 ../script/transfer_pic.py
   ```

   **NOTE**

   If the error message "ModuleNotFoundError: No module named 'PIL'" is displayed during script execution, the Pillow library does not exist. In this case, run the `pip3.7.5 install Pillow --user` command to install the Pillow library.

4. On the MindStudio toolbar, click the refresh icon to refresh the project directory. Then, you can find the .bin files in the `ProjectName/data` directory.

**Figure 5-36 Refresh icon**

![Refresh Icon]

**Step 6** Build and run the project. For details, see 7.3.5 Building and Running an Application Project (in Linux) and 7.3.6 Building and Running an Application Project (in Windows).

If Run Mode is set to Remote Run, install Ascend-CANN-Toolkit on the remote host and specify the paths for dynamic libraries such as ACLlib. For details, see CANN Software Installation Guide.

**Figure 5-37** shows a successful running.
Figure 5-37 Running result

[INFO] start to process file:../data/1.bin
[INFO] model execute success
[INFO] top 1: index[0] value[0.999945]
[INFO] top 2: index[4] value[0.000036]
[INFO] top 3: index[5] value[0.000018]
[INFO] top 4: index[3] value[0.000000]
[INFO] top 5: index[2] value[0.000000]
[INFO] output data success
[INFO] start to process file:../data/2.bin
[INFO] model execute success
[INFO] top 1: index[1] value[0.999242]
[INFO] top 2: index[9] value[0.000595]
[INFO] top 3: index[8] value[0.000121]
[INFO] top 4: index[6] value[0.000040]
[INFO] top 5: index[0] value[0.000001]

----End
6 Script Conversion

6.1 PyTorch GPU2Ascend

6.2 MindConverter

6.1 PyTorch GPU2Ascend

6.1.1 Overview

Many training and online inference scripts are executed on GPU instead of NPU while NPU will be the dominant processor for AI computation power. Due to the architecture differences between NPU and GPU, the GPU-based training and online inference scripts cannot be directly used on NPU before being converted into scripts that support NPU.

NOTE

- msFmkTransplt provides suggestions and converts scripts by the adaptation rules, significantly accelerating script migration and reducing development workload. The scripts in Table 6-1 can be directly executed after being converted. The conversion results of other scripts are for reference only. You need to perform adaptation based on the site requirements.
- Currently, msFmkTransplt only supports the conversion of PyTorch training scripts.

Table 6-1 Supported models

<table>
<thead>
<tr>
<th>No.</th>
<th>Model</th>
<th>Script Link</th>
<th>Dataset</th>
<th>Remarks</th>
</tr>
</thead>
</table>
| 1   | ResNet-18 | Link        | ImageNet 2012 | After a script is converted successfully, you must set the following parameters when running the script:  
<p>|     |         |             |              | • Dataset directory                             |
|     |         |             |              | • --gpu=0: The porting process does not involve the |
| 2   | ResNet-34 | Link        | ImageNet 2012 |                                                  |
| 3   | ResNet-50 | Link        | ImageNet 2012 |                                                  |</p>
<table>
<thead>
<tr>
<th>No.</th>
<th>Model</th>
<th>Script Link</th>
<th>Dataset</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>ResNet-101</td>
<td>Link</td>
<td>ImageNet 2012</td>
<td>modification of parameter variables. Therefore, this parameter needs to be specified to ensure proper script running.</td>
</tr>
<tr>
<td>5</td>
<td>ResNet-152</td>
<td>Link</td>
<td>ImageNet 2012</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Wide_ResNet50_2</td>
<td>Link</td>
<td>ImageNet 2012</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Wide_ResNet101_2</td>
<td>Link</td>
<td>ImageNet 2012</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>VGG11</td>
<td>Link</td>
<td>Imagenet2012</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>VGG11_BN</td>
<td>Link</td>
<td>ImageNet 2012</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>VGG13</td>
<td>Link</td>
<td>ImageNet 2012</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>VGG13_BN</td>
<td>Link</td>
<td>ImageNet 2012</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>VGG16</td>
<td>Link</td>
<td>ImageNet 2012</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>VGG16_BN</td>
<td>Link</td>
<td>ImageNet 2012</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>VGG19</td>
<td>Link</td>
<td>ImageNet 2012</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>VGG19_BN</td>
<td>Link</td>
<td>ImageNet 2012</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>AlexNet</td>
<td>Link</td>
<td>ImageNet 2012</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>MobileNet_V2</td>
<td>Link</td>
<td>ImageNet 2012</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>MNASNet0_5</td>
<td>Link</td>
<td>ImageNet 2012</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>MNASNet0_75</td>
<td>Link</td>
<td>ImageNet 2012</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>MNASNet1_0</td>
<td>Link</td>
<td>ImageNet 2012</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>MNASNet1_3</td>
<td>Link</td>
<td>ImageNet 2012</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>SqueezeNet1_0</td>
<td>Link</td>
<td>ImageNet 2012</td>
<td></td>
</tr>
</tbody>
</table>

**Remarks**

- **--arch=Model name.** This parameter specifies the network architecture to be trained. Generally, it is a lowercase string of the network name.
<table>
<thead>
<tr>
<th>No.</th>
<th>Model</th>
<th>Script Link</th>
<th>Dataset</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>SqueezeNet1_1</td>
<td>Link</td>
<td>ImageNet</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>DenseNet121</td>
<td>Link</td>
<td>ImageNet</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>DenseNet169</td>
<td>Link</td>
<td>ImageNet</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>DenseNet201</td>
<td>Link</td>
<td>ImageNet</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>DenseNet161</td>
<td>Link</td>
<td>ImageNet</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>LSTM (unidirectional)</td>
<td>Link</td>
<td>Timit</td>
<td>After the script is converted successfully, set bidirectional in ctc_config.yaml to False and use the --conf parameter to specify the conf/ctc_config.yaml configuration file when running the script.</td>
</tr>
</tbody>
</table>

### 6.1.2 Generating a Report

**Procedure**

**Step 1** Start script conversion.

You can initiate a script conversion task in any of the following ways:
- Choose **Ascend > Framework Trans > PyTorch GPU2Ascend** from the main menu.
- Right-click the training project and choose **PyTorch GPU2Ascend** from the shortcut menu.
- Click on the toolbar.

**Step 2** Configure the parameters as required.

**Figure 6-1** shows the **PyTorch GPU2Ascend** dialog box.
Figure 6-1 PyTorch GPU2Ascend

Table 6-2 Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Path</td>
<td>Training project path</td>
</tr>
<tr>
<td>Framework</td>
<td>Framework type</td>
</tr>
<tr>
<td>Scene</td>
<td>Application scenario</td>
</tr>
<tr>
<td>Output Path</td>
<td>Output path</td>
</tr>
</tbody>
</table>

Step 3 Click **Transplant** to generate a result report.

After the result report is generated, you can view it in the script conversion window at the lower part.

**NOTE**

The result report is opened in a new dialog box.

----End

Report Directory

```
xxx_msft                  // Directory for storing script conversion results. The default directory is the
                         // directory of the original script. xxx indicates the name of the folder where the original script is stored.
├── out_reports          // Directory of conversion result analysis reports
│   ├── orig_script      // Directory of the original script
│   │   └── Original script file
│   ├── msFmkTranspltResult.json // Conversion result analysis report
│   └── out_todo          // To-do comment result directory. You can modify the script based on
│                     // the comments to complete script conversion.
│     ├── Script file with to-do comments
│     └── msFmkTranspltlog.txt  // Log file generated during script conversion
```
6.1.3 Performing Conversion

A report is generated in a new window of MindStudio, as shown in Figure 6-2.

**Figure 6-2 Report**

<table>
<thead>
<tr>
<th>Parameter/Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FrameWork</td>
<td>Indicates the framework for performing script conversion.</td>
</tr>
<tr>
<td>Total</td>
<td>Indicates the total number of scripts to be converted.</td>
</tr>
<tr>
<td>Selected</td>
<td>Indicates the number of selected scripts to be converted.</td>
</tr>
<tr>
<td>NO.</td>
<td>Indicates the sequence number of a script.</td>
</tr>
<tr>
<td>File</td>
<td>Indicates the conversion code file.</td>
</tr>
<tr>
<td>Line</td>
<td>Indicates the line number of the conversion code.</td>
</tr>
<tr>
<td>Content</td>
<td>Displays original content of the conversion code.</td>
</tr>
<tr>
<td>Description</td>
<td>Describes the conversion details.</td>
</tr>
</tbody>
</table>

You can select or deselect all scripts in the table header.

Selects one or more scripts to be converted.

Applies conversion on the selected one or more scripts.

**Batch Conversion**

**Step 1** Select the scripts to be converted.
Step 2  Click **Apply**.
Step 3  Click **Yes** in the displayed dialog box.

---End

**Individual Conversion**

Step 1  Double-click the script to be converted in Figure 6-2.
Step 2  Click **Apply Auto-Convert** in the code editing window.

Figure 6-3 Converting

---End

**NOTE**

A converted script will be removed from the report window.

# 6.2 MindConverter

## 6.2.1 Overview

MindConverter helps you migrate PyTorch or TensorFlow model scripts to MindSpore rapidly with only slight tweaks according to the conversion report.

## 6.2.2 Prerequisites

MindConverter is a submodule in MindInsight. Please follow the guide to install MindInsight.
### NOTE

- MindConverter depends on the MindSpore package and the corresponding MindSpore environment variables. Ensure that the MindSpore version is the same as the MindInsight version.
- TensorFlow is not the dependency library explicitly declared by MindInsight. To use the graph structure-based script generation tool, you need to manually install TensorFlow 1.15.0.
- To use MindConverter, in addition to a TensorFlow version that can load, train, and infer the exported PB model, you also need to install or upgrade the following third-party libraries:
  - `onnx>=1.8.0`
  - `tf2onnx>=1.7.1`
  - `onnxruntime>=1.5.2`
  - `onnxoptimizer>=0.1.2`
- MindConverter relies on operators supported by MindConverter and MindSpore. Unsupported ONNX operators may not be successfully mapped to MindSpore operators. To solve this problem, you can manually edit or implement the mapping through MindConverter, and contribute to our MindInsight repository.
- Kylin OS V10 SP1-AArch64 does not support the MindInsight component. Therefore, it does not support MindConverter.
- NeoKylin OS 7.6-AArch64 does not support the MindSpore and MindInsight components. Therefore, it does not support MindConverter.
- The Windows OS does not support MindConverter.

### CAUTION

You need to ensure the security and consistency of the model files.

Click [here](#) for more information.

#### 6.2.3 Procedure

**Steps**

**Step 1** Start script conversion.

You can initiate a script conversion task in any of the following ways:

- Click 🔄 on the toolbar.

*Figure 6-4* shows that MindConverter is successfully launched.
**Step 2** Select a script conversion mode.

MindConverter can dynamically read the files that meet the file name extension rules and the number of files based on different modes and frameworks.

1. Script conversion based on the abstract syntax tree (AST)
   - MindConverter can dynamically read the number of files that meet the file name extension rules based on different modes and frameworks.
   - Only PyTorch script conversion is supported in AST mode.
   - The AST mode parses the original PyTorch script into an AST, edits this PyTorch AST to change it to an AST of the corresponding MindSpore scripts, and write the code back based on the MindSpore AST. The script after conversion has the same structure as the original one, including variables, functions, class names and so on.
   - Theoretically, the AST mode supports any model script. However, the conversion results may differ due to the coding style of original scripts.

*Figure 6-5* shows the AST mode. You can click **More settings** to view more parameter configuration items and set the parameters based on the site requirements. See **Table 6-4**.
Figure 6-5 AST mode configuration

Table 6-4 AST mode parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Mandatory/Optional</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>Conversion mode</td>
<td>Mandatory</td>
<td>Must be AST to use the AST mode.</td>
</tr>
<tr>
<td>Framework</td>
<td>Framework</td>
<td>Mandatory</td>
<td>The default value is PyTorch. Currently, only script conversion based on the PyTorch framework is supported.</td>
</tr>
<tr>
<td>Input Path</td>
<td>Path of the PyTorch script to be converted</td>
<td>Mandatory</td>
<td>Example: /home/user/AscendProject/resnet50.py. Set this parameter to the actual path.</td>
</tr>
<tr>
<td>Output Path</td>
<td>Path of the converted script</td>
<td>Mandatory</td>
<td>Example: /home/user/AscendProject/output. Set this parameter to the actual path.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Mandatory/Optional</td>
<td>Example</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------------------</td>
<td>--------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>Report Path</td>
<td>Path of the conversion report</td>
<td>Optional</td>
<td>Example: <code>/home/user/AscendProject/output/report</code>. The default value is <code>{Output Path}/report</code>.</td>
</tr>
</tbody>
</table>

2. Script conversion based on the computational graph
   - The computational graph is a standard descriptive language. It is not affected by coding styles. This mode can ensure more operators converted with less or even no extra tweak to the converted script.
   - For operators supported by MindConverter, the **Graph** mode is recommended because of better conversion result.
   - Both TensorFlow and ONNX model conversion are supported in **Graph** mode.

*Figure 6-6* shows the parameter configuration page of the script generation solution in **Graph** mode.

*Figure 6-6* Graph mode configuration

You can manually enter the path of the file to be converted and the path of the converted file in the **Input Path** and **Output Path** text boxes, or click the icon on the right of the parameter input box to select the corresponding file path. Then, enter the **Shape** and **Input Nodes** information of the file. You can also enable **Auto-filing shape info** to automatically obtain and fill in the **Shape** and **Input Nodes** information of the model file, improving the conversion efficiency.
NOTE

A PyTorch model file can be converted to an ONNX model file to generate the script.

a. Generating scripts based on TensorFlow model files

To port a TensorFlow model script, you need to export the TensorFlow model in PB format and obtain the names of the input and output nodes of the model. For details about how to export the PB model, see 15.8.5 How Do I Export a TensorFlow .pb Model?.

You can convert a single model or multiple models based on the actual number of models. Figure 6-7 shows the TensorFlow single-model script conversion. You can set parameters as required.

Figure 6-7 Graph mode configuration (TensorFlow)

Figure 6-7 shows the TensorFlow multi-model script conversion. You can set parameters as required. See Table 6-5.

Figure 6-8 Graph mode configuration (TensorFlow)
Table 6-5 Graph mode parameter description (TensorFlow)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Mandatory/Optional</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>Conversion mode</td>
<td>Mandatory</td>
<td>Must be <strong>Graph</strong> to use the <strong>Graph</strong> mode.</td>
</tr>
<tr>
<td>Framework</td>
<td>Deep learning framework</td>
<td>Mandatory</td>
<td>The default value is <strong>TensorFlow</strong>. Set this parameter to <strong>TensorFlow</strong> for TensorFlow script conversion.</td>
</tr>
<tr>
<td>Input Path</td>
<td>Path of the TensorFlow model to be converted</td>
<td>Mandatory</td>
<td>Example: <code>/home/user/AscendProject/model/ResNet50.pb</code>. Set this parameter to the actual path.</td>
</tr>
<tr>
<td>Output Path</td>
<td>Path of the converted model</td>
<td>Mandatory</td>
<td>Example: <code>/home/user/AscendProject/output</code>. Set this parameter to the actual path.</td>
</tr>
<tr>
<td>Shape</td>
<td>Expected input shape of the model</td>
<td>Mandatory</td>
<td>Format: NHWC, for example, 1,224,224,3. If multiple models need to be converted, no specific information needs to be entered.</td>
</tr>
<tr>
<td>Input Nodes</td>
<td>Name of the input node</td>
<td>Mandatory</td>
<td>Example: <strong>input_1:0</strong>. If multiple models need to be converted, no specific information needs to be entered.</td>
</tr>
<tr>
<td>Output Nodes</td>
<td>Name of the output node</td>
<td>Mandatory</td>
<td>Example: <strong>probs/Softmax:0</strong>. <strong>probs/Softmax:0</strong> If multiple models need to be converted, no specific information needs to be entered.</td>
</tr>
<tr>
<td>Report Path</td>
<td>Path of the conversion report</td>
<td>Optional</td>
<td>Example: <code>/home/user/AscendProject/output/report</code>. The default value is <code>{Output Path}/report</code>.</td>
</tr>
</tbody>
</table>
b. Generating scripts based on ONNX model files

You can convert a single model or multiple models based on the actual number of models. **Figure 6-9** shows the ONNX single-model script conversion. You can set parameters as required. See **Table 6-6**.

**Figure 6-9 Graph mode configuration (ONNX)**

![Figure 6-9](image)

**Figure 6-10** shows the ONNX multi-model script conversion. You can set parameters as required. See **Table 6-6**.

**Figure 6-10 Graph mode configuration (ONNX)**

![Figure 6-10](image)
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Mandatory/Optional</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conversion Mode</td>
<td>Conversion mode</td>
<td>Mandatory</td>
<td>Must be <strong>Graph</strong> to use the <strong>Graph</strong> mode.</td>
</tr>
<tr>
<td>Framework</td>
<td>Deep learning framework</td>
<td>Mandatory</td>
<td>The default value is <strong>TensorFlow</strong>. Set this parameter to <strong>ONNX</strong> for ONNX script conversion.</td>
</tr>
<tr>
<td>Input Path</td>
<td>Path of the TensorFlow model to be converted</td>
<td>Mandatory</td>
<td>Example: <code>/home/user/AscendProject/model/resnet50.onnx</code>. Set this parameter to the actual path.</td>
</tr>
<tr>
<td>Output Path</td>
<td>Path of the converted model</td>
<td>Mandatory</td>
<td>Example: <code>/home/user/AscendProject/output</code>. Set this parameter to the actual path.</td>
</tr>
<tr>
<td>Shape</td>
<td>Expected input shape of the model</td>
<td>Mandatory</td>
<td>Format: NHWC, for example, <code>1,224,224,3</code>. If multiple models need to be converted, no specific information needs to be entered.</td>
</tr>
<tr>
<td>Input Nodes</td>
<td>Name of the input node</td>
<td>Mandatory</td>
<td>Example: <strong>input_1:0</strong>. If multiple models need to be converted, no specific information needs to be entered.</td>
</tr>
<tr>
<td>Output Nodes</td>
<td>Name of the output node</td>
<td>Mandatory</td>
<td>Example: <strong>probs/Softmax:0</strong>. If multiple models need to be converted, no specific information needs to be entered.</td>
</tr>
<tr>
<td>Report Path</td>
<td>Path of the conversion report</td>
<td>Optional</td>
<td>Example: <code>/home/user/AscendProject/output/report</code>. The default value is <code>{Output Path}/report</code>.</td>
</tr>
</tbody>
</table>
Step 3  Click **Transplant** to convert the model script and generate a conversion report.

After the conversion is complete, you can view the conversion report in the script conversion window or find it in the report path.

---End

## 6.2.4 Using a MindConverter Project Sample

**Procedure**

**Step 1** Create a training project.
- If this is your first login to MindStudio, click **Create new project** on the MindStudio welcome window to go to the **New Project** dialog box.
- If this is not your first login to MindStudio, choose **File > New > Project...** on the menu bar to go to the **New Project** dialog box.

In the navigation tree on the left, choose **Ascend Training**. The training project page is displayed, as shown in **Figure 6-11**. Configure the training project information on the right, as shown in **Table 6-7**. Select **MindSpore** for **Framework** and configure other project information based on the site requirements. After the configuration, click **Next**. The page for selecting the MindConverter sample project is displayed.

**Figure 6-11** Creating a training project

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Project name (user-defined). Enter a maximum of 64 characters, starting with a letter. Only letters, digits, hyphens (-), and underscores (_) are allowed.</td>
<td><strong>MyTraining</strong></td>
</tr>
</tbody>
</table>
### Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Project description (user-defined).</td>
<td>Optional</td>
</tr>
<tr>
<td>ADK Version</td>
<td>Version number of the ADK</td>
<td>Select the current ADK version.</td>
</tr>
<tr>
<td>Project Location</td>
<td>Project path.</td>
<td>Retain the default.</td>
</tr>
</tbody>
</table>

**Step 2**  
*Figure 6-12* shows the page for selecting a MindConverter sample project. Take the ONNX model conversion in **Graph** mode as an example. Select **MindConverter ONNX Graph** and click **Finish**.

When selecting the **MindConverter ONNX Graph** or **MindConverter PyTorch AST** project sample, ensure that the following dependency packages have been installed in the environment.

```
torch == 1.5.0
torchvision == 0.5.0
```

*Figure 6-12* Selecting a MindConverter sample project

**Step 3**  
After the creation is complete, confirm the MindConverter sample project. See *Figure 6-13*. The created project is stored in **$HOME/AscendProjects** by default.

*Figure 6-13* MindConverter sample project window
The directory structure and main files of the created MindConverter sample project are as follows:

```
├── data                            // Directory for storing dataset-related files
│   ├── example                     // Directory for storing the one-click script file
│   │   ├── get_model.py
│   │   └── run_example.sh
│   ├── scripts                     // Directory for storing the project running script file
│   ├── src                          // Directory for storing the project configuration script
│   │   └── eval.py
│   └── train.py
```

**Step 4** Run the MindConverter sample project.

You can run the MindConverter sample project in either of the following ways:

**Method 1:** Use the one-click Python script.

1. Run the `get_model.py` file in the `example` directory. After the execution is complete, the `resnet50.pth` and `resnet50.py` files are generated in the `src` directory.
   
The `resnet50.pth` file can also be directly converted into a `resnet50.py` file using MindConverter.
2. In the `data` directory, run the `./dowanload_cifar10.sh` script to obtain the CIFAR-10 dataset.
3. Run `train.py` to train ResNet-50 in the CIFAR-10 dataset and generate a training model.
4. Run `eval.py` to evaluate the performance of the generated model in the CIFAR-10 test dataset.

**Method 2:** Use the one-click Shell script.

1. In the `example` directory, run the `./run_example.sh` script. After the script is executed, the CIFAR-10 dataset is automatically downloaded, and model conversion and training are performed.
   
   If the CIFAR-10 dataset has been downloaded, the model is automatically converted and trained.
2. Run `eval.py` to evaluate the performance of the generated model in the CIFAR-10 test dataset.

**Step 5** After the training and evaluation are complete, an `.air` file is generated in the `ProjectName/` directory for model conversion and inference in 5.5 Offline Inference After MindSpore Model Conversion.

----End
7 Application Development

7.1 Before You Start

This section describes application development in C/C++. Consult the documents listed in Table 7-1 for reference.

Table 7-1 Reference documents

<table>
<thead>
<tr>
<th>Product</th>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlas 200 DK</td>
<td>• Application Software Development Guide (C&amp;C++)&lt;br&gt;• Log Reference</td>
</tr>
<tr>
<td>Others</td>
<td>• Application Software Development Guide (C&amp;C++) (Inference)&lt;br&gt;• Log Reference (Inference)&lt;br&gt;• mxManufacture User Guide&lt;br&gt;• mxVision User Guide&lt;br&gt;• ATC Instructions (Inference)</td>
</tr>
</tbody>
</table>

7.2 Developing an Application Based on MindX SDKs
7.2.1 Before You Start

- Currently, MindX SDK applies to Ubuntu 18.04, CentOS 7.6, EulerOS V2R8, LinxOS 6.0.9, and KylinOS V10.
- In the Windows OS, MindStudio allows the following projects to be created, but does not support MindX SDK for application and plugin development.
  - MindX SDK Project: creates an empty MindX SDK project.
  - Detection and Classification: creates a MindX SDK sample project.
  - MindX SDK Plugin: creates an empty project for MindX SDK plugin development.
  - MxpiSamplePlugin: creates a sample project for MindX SDK plugin development.

7.2.2 Creating an Application Project

Creating an Application Project

**Step 1** Navigate to the project creation window.
- If it is your first login to MindStudio, click **Create New Project**.
- If not, choose **File > New > Project...** from the menu bar.

**Step 2** In the **New Project** window, select **Ascend App** and set the project parameters, as shown in **Table 7-2**.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Project name (user-defined). Enter a maximum of 64 characters, starting with a letter. Only letters, digits, hyphens (-), and underscores (_) are allowed.</td>
</tr>
<tr>
<td>Description</td>
<td>Project description (user-defined) of up to 256 characters.</td>
</tr>
<tr>
<td>ADK Version</td>
<td>Version number of the ADK.</td>
</tr>
<tr>
<td>Project Location</td>
<td>Project path (user-defined).</td>
</tr>
</tbody>
</table>

**Step 3** Click **Next**. In the **New Project** window, select a project type. See **Figure 7-1**.
Figure 7-1 Developing an application using MindX SDK

- **Empty MindX Project**: creates an empty MindX SDK project that contains only the development framework and does not contain specific code logic.
- **Detection and Classification**: creates a MindX SDK sample project, a template project developed using MindX SDK.

**Step 4** Click **Finish**.

Prepare and edit pipeline files in the project directory by referring to 7.2.5 Visualized Process Orchestration. The MindX SDK sample project provides ready-to-use pipeline files.

- The directory of a project created in **MindX SDK Project** mode is organized as follows.

```
├── Project name
│   ├── build                   // Directory of CMake dependencies
│   │   └── config
│   │       └── logging.conf   // Log configuration file
│   │   └── out
│   │       └── src
│   │           └── CMakeLists.txt // Build script
│   │           └── main.cpp     // Implementation file of the main function, which does not have code
│   │   └── .project              // Project information file, including the project type, project description, type of the target device, and ADK version
│   │                       └── projectName.iml  // Module file automatically created by the IntelliJ IDEA, used for Java application development
│   └── config
│       └── logging.conf       // Log configuration file
│   └── out
│       └── src
```

- The directory of a project created in **Detection and Classification** mode is organized as follows.

```
├── Project name
│   ├── build                   // Directory of CMake dependencies
│   │   └── config
│   │       └── logging.conf   // Log configuration file
│   │   └── data
│   │       └── test.jpg      // Test image
```

MindStudio
User Guide
7 Application Development

Issue 01 (2021-06-24) Copyright © Huawei Technologies Co., Ltd.
 imported models configuration files and labels:

- resnet50 // ResNet-50 model configuration file
  - aipp_resnet50_224_224.aippconfig // AIPP configuration file for model conversion
  - imagenet1000_classid_to_labels.names // Mapping between ImageNet 1000 indexes
- yolov3 // YOLOv3 model configuration file
  - aipp_yolov3_416_416.aippconfig // AIPP configuration file for model conversion
  - coco.names // Mapping between the indexes and labels
  - yolov3_tf-bs1_fp16.cfg // Postprocessing configuration file

- out // Directory of executable files generated after build
- pipeline
  - Sample.pipeline // Pipeline corresponding to object detection
  - SamplePlugin.pipeline // Pipeline corresponding to the custom plugin
- src
  - CMakeLists.txt // Build script
  - main.cpp // Implementation file of the main function
  - project // Project information file, including the project type, project description, type of the target device, and ADK version
  - CMakeLists.txt // Build script that calls the CMakeLists file in the src directory
  - projectName.iml // Module file automatically created by the IntelliJ IDEA, used for Java application development

--- End

Importing an Application Project

**Step 1** Import the project file.
- If you log in to MindStudio for the first time, click **Open or Import** on the MindStudio welcome page, select the project to be imported, and click **OK**.
- On the project creation page, choose **File > Open...** on the top menu bar and select an existing project to open it.

**Step 2** If there is already an active project in the window, the message shown in **Figure 7-2** will be displayed.
- Click **This Window** to open the created project in the current window.
- Click **New Window** to open the created project in a new window.

**Figure 7-2** Prompt for opening a new project

![Prompt for opening a new project](image)

**Step 3** If MindStudio detects that the .project project configuration file does not exist, the window shown in **Figure 7-3** is displayed, helping you create the .project file.

**Figure 7-3** Importing project configurations

![Importing project configurations](image)
The parameters in the window are described as follows. You can select the parameters based on the actual scenario.

- **Type**: Select *Ascend App* (application project) or *Ascend Training* (training project) based on the project type.
- **Project Type**: Select *Ascend ACL App* (AscendCL application project) or *Ascend MindX SDK App* (SDK application project). For MindX SDK-based application development, select *Ascend MindX SDK App*.
- **Project Desc**: project description. The value is *Inference Project* for an application project.

**Step 4** Click **Create** to create the .project file.

If you click **Cancel**, the project configuration file will not be properly created and MindStudio cannot identify the project type. In this case, the dialog box shown in **Figure 7-4** is displayed. After you click **OK** in the dialog box, the imported project file can still be opened. When you import the project file next time, the dialog box shown in **Figure 7-3** is displayed to help you create the project.

**Figure 7-4** Creation failure message

![Unsupported project type. Not all features are available. Please confirm.](image)

----End

7.2.3 Installing the SDK Software Package

The SDK software package can be installed in either of the following ways:

- Online installation
- Offline installation

**Online Installation**

**Step 1** In the project creation window, choose **Ascend > MindX SDK Manager**. **MindX SDK Location** indicates the default SDK installation path. The path is $HOME/Ascend/mindx_sdk for a non-root user or /usr/local/Ascend/mindx_sdk for the root user.

Click **Install SDK**. The **Installation settings** window is displayed.

**Step 2** Set installation parameters in the **Installation settings** window. See **Figure 7-5**.

- **Install Source**: installation source. **Online** indicates online installation, and **Local** indicates local installation. Select **Online**.
- **SDK Version**: version of the SDK software package that can be selected.
- **Install Location**: default installation path. The path is $HOME/Ascend/mindx_sdk for a non-root user or /usr/local/Ascend/mindx_sdk for the root user.

Click **Next**. Confirm the current installation settings, as shown in **Figure 7-6**. Then, click **Next** to start installation.
**Figure 7-5** Installation settings

![SDK Version Setting]

**Figure 7-6** Current settings

![Current settings]

**Step 3** Complete the installation as shown in **Figure 7-7**.
After the installation is complete, click Finish. The MindX SDK Manager window is displayed.

Step 4 In the MindX SDK Manager window, view the information about the installed SDK, as shown in Figure 7-8. Click OK to complete the installation.

The two icons in the window are described as follows:

- ![Uninstall](image) : uninstalls an SDK.
- ![Activate](image) : activates an SDK.

**NOTE**

Note the following SDK activation restrictions:

- Only one SDK can be activated at a time.
- If a new SDK is installed while an activated SDK is in use, the one in use is deactivated and the newly installed one is automatically activated. If you want to use the original one, you need to manually activate it.
Figure 7-8 MindX SDK Manager after successful installation

---End

Offline Installation

**Step 1** Obtain and upload the `mxManufacture` or `mxVision` runfile as required from MindX SDK in the Ascend Community.

**Step 2** In the project creation window, choose *Ascend > MindX SDK Manager*. MindX SDK Location indicates the default SDK installation path. The path is `$HOME/Ascend/mindx_sdk` for a non-root user or `/usr/local/Ascend/mindx_sdk` for the root user.

Click **Install SDK**. The **Installation settings** window is displayed.

**Step 3** Set installation parameters in the **Installation settings** window, as shown in Figure 7-9.

- **Install Source**: installation source. *Online* indicates online installation, and *Local* indicates local installation. Select *Local*.
- **SDK Location**: upload path of the runfiles obtained in **Step 1**. Select the path to the last level.
- **Install Location**: default installation path. The path is `$HOME/Ascend/mindx_sdk` for a non-root user or `/usr/local/Ascend/mindx_sdk` for the root user.
Click **Next**. Confirm the current installation settings, as shown in **Figure 7-10**. Then, click **Next** to start installation.

**Figure 7-9** Installation settings

![Installation settings](image)

**Figure 7-10** Current settings

![Current settings](image)

**Step 4** Complete the installation as shown in **Figure 7-11**.
After the installation is complete, click **Finish**. The **MindX SDK Manager** window is displayed.

**Step 5** In the **MindX SDK Manager** window, view the information about the installed SDK, as shown in Figure 7-12. Click **OK** to complete the installation.

The two icons in the window are described as follows:

- ![Uninstall](image_uninstall.png): uninstalls an SDK.
- ![Activate](image_activate.png): activates an SDK.

**NOTE**

Note the following SDK activation restrictions:

- Only one SDK can be activated at a time.
- If a new SDK is installed while an activated SDK is in use, the one in use is deactivated and the newly installed one is automatically activated. If you want to use the original one, you need to manually activate it.
7.2.4 Developing an Application

7.2.4.1 Typical Service Process

Figure 7-13 shows the typical inference service process. The object coordinates are obtained through object detection, and then the object properties are identified through image classification. There are two image resizing operations in the process. The main difference between the two operations is that the configured aspect ratio parameters are different.

Figure 7-13 Typical service process
### 7.2.4.2 Development Workflow

**NOTE**
- Dump is not supported for SDK-based application development in MindStudio.
- Currently, MindStudio supports only C++.
- The path of the model to be loaded cannot contain spaces.

Figure 7-14 shows the workflow of developing and running an inference service in MindStudio. In-depth development is an advanced SDK feature designed for senior developers.

**Figure 7-14 Inference service development and execution workflow**

#### Step 1 Determine the service process.

Modularize the service process based on functions such as object detection, image classification, and attribute recognition. For details, see 7.2.4.1 Typical Service Process.

#### Step 2 Search for proper plugins.

Match service functions based on the function description and specifications of existing SDK plugins. If the plugins provided by the SDK cannot meet your requirements, you can develop custom plugins. For details, see 7.2.6 Developing a Plugin and sections "Existing Plugin Introduction" and "Plugin Development Description" of the mxManufacture User Guide or mxVision User Guide, as listed in Table 7-1.

#### Step 3 Prepare model files and data.

Add model files and datasets for inference based on the actual application scenario of plugins.

1. Convert your model.

   Before adding a model file, convert the model trained in a third-party framework to an .om offline model adapted to the Ascend AI Processor by referring to 4 Model Conversion. For details about how to convert the model of an SDK sample project, see Step 3 to Step 5 in 7.2.8 Using an SDK Project Sample. For details about model conversion and parameter settings, see the ATC Instructions (Inference) in Table 7-1.
2. Add one or more model files.
   Upload the prepared model files to a user-defined directory of the application project.
3. Prepare inference data.
   Prepare the data required for inference and upload the data to the application project directory.

**Step 4** Orchestrate the process.

For details, see 7.2.5 Visualized Process Orchestration and sections "Process Orchestration" and "Data Flow" of the mxManufacture User Guide or mxVision User Guide, as listed in Table 7-1.

**Step 5** Integrate services.

Use C++ for programming, call the service stream management API MxStreamManager to initialize the stream, load the service stream configuration file (*.pipeline), obtain the output data from the specified stream based on StreamName in the stream configuration file, and destroy the stream.

For details about available APIs, see section "API Reference (C++)" in the mxManufacture User Guide or mxVision User Guide, as listed in Table 7-1.

**Step 6** Build and run the application.

For details, see 7.2.7 Building and Running an Application Project.

----End

7.2.5 Visualized Process Orchestration

The minimum granularity for MindX SDK to implement functions is a plugin. Each plugin implements specific functions, such as image decoding and scaling. Orchestrate these plugins in a proper sequence to implement the required functions. Visualized process orchestration is to develop data stream diagrams in a visualized manner and generate pipeline files for the application framework.

**Step 1** On the project creation page, perform the following operations to start process orchestration:

- On the top menu bar, choose Ascend > MindX SDK Pipeline. The pipeline drawing page is displayed, as shown in Figure 7-15.
- On the top menu bar, choose File > Open... to open the pipeline file you drew, as shown in Figure 7-16. (If the pipeline file you drew has been stored in the project directory, you can also double-click the file in the project directory to open it.)
The drawing page consists of the plugin library on the left, editing area in the center, and plugin attribute display area on the right. For details, see Figure 7-16 and Figure 7-19. Their functions are described in Table 7-3.
<table>
<thead>
<tr>
<th>Area</th>
<th>Description</th>
</tr>
</thead>
</table>
| Plugin library  | ● Dynamically scan the installed MindX SDK to obtain the plugin library and find the corresponding .so file for the generated pipeline. You can expand the plugin library directory and drag the required plugins to the editing area ([Figure 7-17](#)). Alternatively, you can search for the plugins by keyword in the search box and drag the plugins to the editing area ([Figure 7-18](#)).  
● The SDK development kit has a built-in plugin library. After the SDK development kit is installed and the drawing page is displayed, the plugin library on the left is automatically loaded. For details about the built-in plugin library, see section “Existing Plugin Introduction” of the *mxManufacture User Guide* or *mxVision User Guide*, as listed in Table 7-1.  
● If the built-in plugin library cannot meet the development requirements, you can develop plugins and import the generated plugins to the plugin library in the root directory of the project file/lib/plugins directory. If the directory does not exist, you need to create it. For details about how to develop a plugin, see 7.2.6 Developing a Plugin or section "Plugin Development Description" of the *mxManufacture User Guide* or *mxVision User Guide*, as listed in Table 7-1. |

**Figure 7-17** Directly dragging the plugin library directory
<table>
<thead>
<tr>
<th>Area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Figure 7-18</strong> Plugin search by keyword</td>
<td></td>
</tr>
</tbody>
</table>

```plaintext
tppt
rtptdemux
```
<table>
<thead>
<tr>
<th>Area</th>
<th>Description</th>
</tr>
</thead>
</table>
| Editing area | • Drag a plugin from the plugin library to the editing area. Two plugins can be connected to form a data stream. The one-to-one, one-to-many, and many-to-one connection relationships are supported.  
• The configurable attributes of each plugin can be scanned. You can click a plugin in the editing area and customize the attributes (such as the path parameters in the plugin attributes). See Figure 7-19. |

**Figure 7-19** Modifying path parameters

![Figure 7-19 Modifying path parameters](image)

• As shown in Figure 7-16, plugins in purple are input plugins, plugins in pink are output plugins, plugins in blue are the plugins between the input and output plugins, and plugins in gray are the plugins that are not defined in the current plugin library.  
• When you hover your mouse cursor over a plugin, the function description of the plugin is displayed, as shown in Figure 7-20.  

**Figure 7-20** Hovering the mouse over a plugin to view the plugin function description

![Figure 7-20 Hovering the mouse over a plugin to view the plugin function description](image)

• Press Del to delete the selected plugin or connection.
### Area

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
</table>
| Open a pipeline file with content. The **Pipeline Stream Editor** and **Text** options are displayed in the lower left corner of the editing area. See **Figure 7-16**. The graphical display (default) and code display modes are supported. You are advised to use the process orchestration visualization function in graphical display mode.

### Plugin attribute display area

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
</table>
| Click a plugin to view its attributes. See **Figure 7-19**. Plugin attributes are displayed in key-value or subkey-subvalue mode. See **Figure 7-21** and **Figure 7-22**. In subkey-subvalue mode, click ![Delete](image) to delete a key-value pair or click ![Add](image) to add a key-value pair.

**Figure 7-21** Key-value

```
modelPath: _tf.bs1_fp16.om
```

**Figure 7-22** Subkey-subvalue

```
dataSource
postProcessLibPath: postProcessor.so
modelPath: _f.bs1_fp16.om
dataSource: i_imageresize0
labelPath: v3/coco.names
postProcessConfigPath: tf.bs1_fp16.cfg
```

### Table 7-4

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format</td>
<td>Resolves plugin overlapping or formats the plugin process drawn by a user.</td>
</tr>
<tr>
<td>New</td>
<td>Creates a canvas.</td>
</tr>
<tr>
<td>Open</td>
<td>Opens a pipeline file.</td>
</tr>
<tr>
<td>Button</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| Save   | Saves the plugin flowchart as a pipeline file. Click **Save** to display the **Save** dialog box, as shown in **Figure 7-23**. The parameters in **Figure 7-23** are described as follows. You can define the parameters as required.  
  - **Pipeline Filename**: name of the pipeline file.  
  - **Output Path**: path for storing the pipeline file. |

**Figure 7-23 Save**

![Save dialog box](image)

---

The parameters in **Figure 7-23** are described as follows. You can define the parameters as required.

- **Pipeline Filename**: name of the pipeline file.
- **Output Path**: path for storing the pipeline file.
### 7.2.6 Developing a Plugin

**Step 1** Navigate to the project creation window.
- If it is your first login to MindStudio, click **Create New Project**.
- If not, choose **File > New > Project...** from the menu bar.

**Step 2** Select plugins in the plugin library, drag the plugins to the editing area, and connect them according to the service process. If you drag an unwanted plugin or make an unwanted connection, select the plugin or connection and press **Del** to delete the plugin or connection. After the pipeline is drawn, select and right-click all plugins in the pipeline and then choose **Set Stream Name** from the shortcut menu to set the stream name. If there is more than one pipeline, set the stream name for each pipeline.

- **NOTE**
  
  By default, **Stream name** is **stream0** and **Device Id** is **0**. Change the values based on site requirements. If you specify a value for **Device Id**, select all plugins that belong to the same stream and set them to the same device.

**Step 3** Click **Save** to save the settings.

----End
Step 2  In the New Project window, select Ascend App and set the project parameters, as shown in Table 7-5.

Table 7-5 Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Project name (user-defined). Enter a maximum of 64 characters, starting with a letter. Only letters, digits, hyphens (-), and underscores (_) are allowed.</td>
</tr>
<tr>
<td>Description</td>
<td>Project description (user-defined) of up to 256 characters.</td>
</tr>
<tr>
<td>ADK Version</td>
<td>Version number of the ADK.</td>
</tr>
<tr>
<td>Project Location</td>
<td>Project path (user-defined).</td>
</tr>
</tbody>
</table>

Step 3  Click Next. In the New Project window, select a project type. See Figure 7-25.

Figure 7-25 Plugin development project

- MindX SDK Plugin: creates an empty project for MindX SDK plugin development that contains only the development framework and does not contain specific code logic.
- MxpiSamplePlugin: creates a sample project for MindX SDK plugin development.

Step 4  Click Finish.
The directory of a project created in **MindX SDK Plugin** mode is organized as follows.

### NOTE

Set the name of the empty project for MindX SDK plugin development (that is, the project name in the following project directory) to the name of the developed plugin.

```
├── Project name
│   ├── build                       // Directory of CMake dependencies
│   │   └── Project name
│   │       ├── CMakeLists.txt    // Build script
│   │       ├── Project name.h   // Dependent header file of the main function
│   │       └── Project name.cpp // Implementation file of the main function
│   └── lib
│       └── plugins                // Directory for storing the .so files of the developed plugin
│   └── out                         // Directory of executable files generated after build
└── src
    ├── Project name
    │   └── CMakeLists.txt // Build script
    └── .project                    // Project information file, including the project type, project description, type of the target device, and ADK version
```

The directory of a project created in **MxpiSamplePlugin** mode is organized as follows.

```
├── Project name
│   ├── build               // Directory of CMake dependencies
│   │   └── MxpiSamplePlugin
│   │       ├── CMakeLists.txt // Build script
│   │       └── MxpiSamplePlugin.h // Dependent header file of the main function
│   └── lib
│       └── plugins        // Directory for storing the .so files of the developed plugin
│   └── src
│       └── MxpiSamplePlugin
│           └── CMakeLists.txt // Build script that calls the CMakeLists file in the src directory
└── .project                         // Project information file, including the project type, project description, type of the target device, and ADK version
```

### Step 5

For details about the code implementation of the empty project for plugin development, see section "Plugin Development Description" of the *mxManufacture User Guide* or *mxVision User Guide*, as listed in Table 7-1.

For details about the code implementation of the sample project for plugin development, see `MxpiSamplePlugin.cpp` (implementation file of the main function) and `MxpiSamplePlugin.h` (dependent header file of the main function) in the project file.

### Step 6

Develop and compile the plugin. The following uses the plugin development sample project as an example.

1. On the MindStudio project page, choose **Build > Edit Build Configuration...** from the top menu bar. The compilation configuration page is displayed. For details about parameter settings, see **Building an Application Project**.
2. After the compilation is complete, the .so file of the developed plugin is generated in `project directory/lib/plugins`. See **Figure 7-26**.
3. On the top menu bar, choose **Ascend > MindX SDK Pipeline**. The pipeline drawing page is displayed. You can find the plugin `MxpiSamplePlugin` in the plugin library and use it properly (for example, you can drag the plugin to the editing area, connect it to other plugins, and view plugin attributes). See **Figure 7-26**.
7.2.7 Building and Running an Application Project

**NOTE**
- If the existing SDK plugins meet the development requirements, build and run your application project as follows.
- If not, develop custom plugins, create and build the `CMakeLists` file to obtain the plugin dynamic library file. Place the postprocessing library file in the `lib` directory under the project directory, and the plugin library file in the `lib/plugins` directory under the project directory. The custom plugins stored in this directory can be automatically identified during process orchestration. After the preceding operations are complete, build and run the application project.
- Currently, cross compilation is not supported.

**Building an Application Project**

**Step 1** (Optional) Modify the `src/CMakeLists.txt` file in the project directory.
- `include_directories`: Add the directories of the header files to be included.
  
  Example:
  ```
  include_directories(
    directoryPath1
    directoryPath2
  )
  ```

- `link_directories`: Add the directories of the library files to be linked with.
  
  Example:
  ```
  link_directories(
    directoryPath3
    directoryPath4
  )
  ```

- `add_executable`: Add the directory of the `.cpp` file.
  
  Example:
  ```
  add_executable(
    main
    directoryPath5
    directoryPath6
  )
  ```

- `target_link_libraries`: Add the library files on which application building depends.
  
  Example:
target_link_libraries(
main
ascendcl
libName1
libName2)

- **install**: Install the build output **main** to a specified path.

  Example:
  ```cmake
  install(TARGETS main DESTINATION $(CMAKE_RUNTIME_OUTPUT_DIRECTORY))
  ```

**NOTE**

For details about the CMake parameters, visit [https://cmake.org/cmake/help/latest/guide/tutorial/index.html](https://cmake.org/cmake/help/latest/guide/tutorial/index.html) and query the required version.

**Step 2** In the MindStudio project window, choose **Build > Edit Build Configuration**.

1. Configure build parameters.

   If you want to run the built executable file in the operating environment, the options displayed in the **Target OS** drop-down list box are related to the installed ACLlib component. You can view the basic information about the ACLlib component in the **ADK Manager** dialog box. For details about how to open the **ADK Manager** dialog box, see **ADK Manager Overview**.

   - **Linux**: Select **Target OS** and **Target Architecture** according to the **Host OS Arch** field of the ACLlib component in the **ADK Manager** dialog box.

   ![Figure 7-27 Build parameter configuration in the Linux scenario](image)

2. Click **Build** to build the project.

   If no error message is displayed and the message "build successfully" is displayed, the build is successful. Find the generated executable file **main** in the following directory:

   ```
   ├── Project name
   │   ├── out
   │   │   └── main         // Executable
   ```

---End

**Running an Application Project**

**Step 1** In the MindStudio project window, choose **Run > Edit Configurations** to open the run configuration dialog box.

**Step 2** Choose **Ascend App > Project name** to set running parameters of the application project. **Table 7-6** describes the running parameters shown in **Figure 7-28**.
**Figure 7-28** Run parameter configuration

![Run parameter configuration](image)

**Table 7-6** Description of running parameters of an application project

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the run configuration (customizable).</td>
</tr>
<tr>
<td>Executable</td>
<td>Directory of the executable file.</td>
</tr>
<tr>
<td>Run Mode</td>
<td>- Remote Run&lt;br&gt;- Local Run</td>
</tr>
<tr>
<td>SSH Connection</td>
<td>SSH connection. It is available when the <strong>Remote Run</strong> mode is selected. For details, see 12.1 SSH Connection Management. Select the IP address of the remote server that runs the application from the drop-down list box. If the IP address is not added, click <img src="image" alt="add SSH connection" /> to add it. For details, see 12.1.3 Adding a New SSH Connection.</td>
</tr>
<tr>
<td>Command Arguments</td>
<td>Argument passed for application running (customized). The arguments are optional and related to user code implementation.</td>
</tr>
<tr>
<td>Environment Variables</td>
<td>Environment variable configuration. Enter environment variables in the text box or click <img src="image" alt="add environment variable" /> to add more by clicking <img src="image" alt="add environment variable" /> in the displayed dialog box.</td>
</tr>
<tr>
<td>Before launch</td>
<td>Pre-processing before running the application (optional). For example, click <img src="image" alt="add before launch" /> to add <strong>Ascend App Build</strong> to build the executable file before running it.</td>
</tr>
</tbody>
</table>
Step 3 After the configuration is complete, click **Apply** to save the run configuration and click **OK** to close the run configuration dialog box.

Step 4 In the MindStudio project window, choose **Run > Run**....

In the dialog box displayed, select the created run configuration and run the application.

- If no error message is displayed and the message "Running *** finished" is displayed, the execution is complete.
- If an error message is displayed during application execution, view the detailed logs of the operating environment by referring to Log Reference listed in Table 7-1.
  - The default log path is SHOME/ascend/log. For details, see the "Log Files" section.
  - The default log level is ERROR. For details about how to query or change the log level, see the "Log Level Setting" section.

----End

7.2.8 Using an SDK Project Sample

Step 1 Obtain and execute the runfiles by referring to 7.2.3 Installing the SDK Software Package.

Step 2 Create an application project by referring to 7.2.2 Creating an Application Project. Select Detection and Classification as the project type.

Step 3 Download the .pb files of the YOLOv3 and ResNet-50 models from the Ascend Community. See Table 7-7.

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>YOLOv3</td>
<td>1.1</td>
<td><strong>Link</strong>&lt;br&gt;Keyword: YOLOv3&lt;br&gt;Application Level: Released&lt;br&gt;Framework: TensorFlow&lt;br&gt;On the model page, select <strong>Version History</strong> to download the 1.1 version.</td>
</tr>
<tr>
<td>ResNet50</td>
<td></td>
<td><strong>Link</strong>&lt;br&gt;Keyword: ResNet-50&lt;br&gt;Application Level: Released&lt;br&gt;Framework: TensorFlow&lt;br&gt;On the model page, select <strong>Version History</strong> to download the 1.1 version.</td>
</tr>
</tbody>
</table>
Step 4  Convert your model.

Choose **Ascend > Model Converter** on the toolbar in the project creation window.

1.  On the **Model Information** page, set the parameters as follows:
   
   **Figure 7-29** shows example configuration parameters for YOLOv3 model conversion. **Figure 7-30** shows example configuration parameters for ResNet-50 model conversion.

   - **Model File**
     Select the path of the model PB file uploaded to the environment.

   - **Model Name**
     When converting a YOLOv3 model, name it **yolov3_tf_bs1_fp16**. When converting a ResNet-50 model, name it **resnet50_aipp_tf**.

   - **Target SoC Version**
     Sets the target SoC version. The version is the same for YOLOv3 model conversion and ResNet-50 model conversion. Select the version based on the chip type.

   - **Input Format**
     Input data format. The format is the same for YOLOv3 model conversion and ResNet-50 model conversion. Use the default settings.

   - **input/input_data**
     When converting a YOLOv3 model, set **1.416.416.3** for **Shape** and **FP32** for **Type**.
     When converting a ResNet-50 model, set **1.224.224.3** for **Shape** and **FP32** for **Type**.

   - **Output Nodes**: model output node information.
     When converting a YOLOv3 model, select **conv_lbbox/BiasAdd**, **conv_mbbox/BiasAdd**, and **conv_sbbox/BiasAdd**.
     When converting a ResNet-50 model, you do not need to set **Output Nodes**.
Figure 7-29 YOLOv3 model parameters

Figure 7-30 ResNet-50 model parameters
2. Click **Next**. In the **Data Pre-Processing** window, set parameters by referring to **Figure 7-31**.

**Figure 7-31 Data Pre-Processing**

![Data Pre-Processing Window](image)

3. Click **Next**. In the **Advanced Options Preview** window, set parameters by referring to **Figure 7-32**.

Set **Additional Arguments** to the absolute path of the model AIPP configuration file, `$HOME` to the home directory of the user who starts MindStudio, and `MyApp` to the name of the new SDK sample project.

When converting a YOLOv3 model, set the parameters as follows:

```
--insert_op_conf=$HOME/AscendProjects/MyApp/models/yolov3/aipp_yolov3_416_416.aippconfig
```

When converting a ResNet-50 v1.5 model, set the parameters as follows:

```
--insert_op_conf=$HOME/AscendProjects/MyApp/models/resnet50/aipp_resnet50_224_224.aippconfig
```
Step 5  Save the converted model files in the `project_file_root_dir/models/resnet50` and `project_file_root_dir/models/yolov3` directories.

Step 6  Build and run the application project by referring to 7.2.7 Building and Running an Application Project.

If the following information is displayed, the project execution is successful.

```
Results:{"MxpiObject":[{"MxpiClass":["beagle","confidence":0.87109375},"classVec":["dog","confidence":0.99641436299999997,"headerVec":[]}]}
```

-----End

7.3 Developing an Application Based on AscendCL

7.3.1 Creating an Application Project

Creating an Application Project

Step 1  Navigate to the project creation window.

- If it is your first login to MindStudio, click Create New Project.
- If not, choose File > New > Project... from the menu bar.

Step 2  In the New Project window, select Ascend App and set the project parameters, as shown in Table 7-8.
Table 7-8 Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Project name (user-defined). Enter a maximum of 64 characters, starting with a letter. Only letters, digits, hyphens (-), and underscores (_) are allowed.</td>
</tr>
<tr>
<td>Description</td>
<td>Project description (user-defined) of up to 256 characters.</td>
</tr>
<tr>
<td>ADK Version</td>
<td>Version number of the ADK.</td>
</tr>
<tr>
<td>Project Location</td>
<td>Project path (user-defined).</td>
</tr>
</tbody>
</table>

Step 3 Click Next. In the New Project dialog box, select a project type. See Figure 7-33.

Figure 7-33 AscendCL application project

- **Acl Project**: creates an empty AscendCL project that contains only the development framework and does not contain specific code logic.
- **Acl ResNet-50**: creates an AscendCL sample project, a project from the acl_resnet50 sample.
  
  The acl_resnet50 sample is described as follows:
  
  This sample shows how to classify images based on the Caffe ResNet-50 network (single input with batch size = 1).
  
  Convert the model file of the Caffe ResNet-50 network to an .om offline model adapted to the Ascend AI Processor. In the sample, load the .om file, decode, resize, and infer two .jpg images, process the obtained inference
results, and output the class indexes with the top 5 confidence values of each image.

**Figure 7-34** Sample diagram

![Diagram](image)

**Step 4** Click **Finish**.

- The directory of a project created in **Acl Project** mode is organized as follows.

```
├── ProjectName
│   ├── .idea                      //Directory automatically created by the IntelliJ IDEA, used to store the
│   │   ├── build                  //Directory of CMake dependencies
│   │   │    ├── cmake            //Directory of executable files generated after build
│   │   │    └── src
│   │   │        └── acl.json     //Configuration file for system initialization
│   │   │        └── CMakeLists.txt //Build script
│   │   │           └── main.cpp  //Implementation file of the main function, which does not have code
│   │   │       └── .project        //Project information file, including the project type, project description,
│   │   │                       type of the target device, and ADK version
│   │   │               └── CMakeLists.txt //Build script that calls the CMakeLists file in the src directory
│   │   │                    └── projectName.iml //Module file automatically created by the IntelliJ IDEA, used for Java
│   │   │                           application development
│   │   └── .project              //Project information file, including the project type, project description,
│   │       type of the target device, and ADK version
```

- The directory of a project created in **Acl ResNet-50** mode is organized as follows.

```
├── ProjectName
│   ├── .idea                   //Directory automatically created by the IntelliJ IDEA, used to store the
│   │   ├── build
│   │   │    └── src
│   │   │        └── acl.json
│   │   │               └── CMakeLists.txt //Build script that calls the CMakeLists file in the src directory
│   │   │                    └── projectName.iml //Module file automatically created by the IntelliJ IDEA, used for Java
```

---

*Copyright © Huawei Technologies Co., Ltd.*
---End

Importing an Application Project

**Step 1** Import the project file.
- If you log in to MindStudio for the first time, click **Open or Import** on the MindStudio welcome page, select the project to be imported, and click **OK**.
- On the project creation page, choose **File > Open...** on the top menu bar and select an existing project to open it.

**Step 2** If there is already an active project in the window, the message shown in Figure 7-35 will be displayed.
- Click **This Window** to open the created project in the current window.
- Click **New Window** to open the created project in a new window.

**Figure 7-35** Prompt for opening a new project

---End
The parameters in the window are described as follows. You can select the parameters based on the actual scenario.

- **Type**: Select *Ascend App* (application project) or *Ascend Training* (training project) based on the project type.
- **Project Type**: Select *Ascend ACL App* (AscendCL application project) or *Ascend MindX SDK App* (SDK application project) based on site requirements.
- **Project Desc**: project description. The value is *Inference Project* for an application project.

**Step 4** Click **Create** to create the `.project` file.

If you click **Cancel**, the project configuration file will not be properly created and MindStudio cannot identify the project type. In this case, the dialog box shown in Figure 7-37 is displayed. After you click **OK** in the dialog box, the imported project file can still be opened. When you import the project file next time, the dialog box shown in Figure 7-36 is displayed to help you create the project.

**7.3.2 Preparing Model Files and Data**

**Acl Project Mode**

Add model files and image datasets for inference based on the actual application scenario.

**Step 1** Convert your model.

Before adding a model file, convert the model trained in a third-party framework to an `.om` offline model adapted to the Ascend AI Processor by referring to 4 Model Conversion.

**Step 2** Add one or more model files.
1. Right-click the project name and choose **Add Model** from the shortcut menu. In Windows, if a message is displayed indicating that you do not have the permission to add a model, run MindStudio again as the administrator. For details, see 7.5.1 Insufficient Permissions to Add a Model or Add Data In an Application Project In Windows.

2. In the displayed dialog box, select one or more .om files. (Press and hold **Ctrl** to select multiple files.)

   Note the following restrictions when adding models:
   - The MindStudio installation user must have the permission to access the .om files.
   - You can add a maximum of 1024 .om files. The size of a single .om file must be within 1 GB, and the total size of all .om files must be less than 10 GB.
   - The .om file name can consist of up to 64 characters, including letters, digits, and the following special characters: -_.

3. Click **Add**.

   You can view the added model files in the Project window.

   ├── ProjectName
   │   └── model
   │       └──*.om           //Model file

Step 3 Prepare inference data.

Prepare the data required for inference and upload the data to the application project directory.

----End

Acl ResNet-50 Mode

Step 1 Prepare data.

Download the .prototxt model file and .caffemodel pre-trained model file of ResNet-50 from the following links and upload them to the MindStudio installation server as the MindStudio installation user:

- Visit our ModelZoo repository on Gitee and find the download links in the README.md file.
- Visit our modelzoo repository on GitHub and find the download links in the README.md file.

Step 2 Convert your model.

Before adding a model file, convert the model trained in a third-party framework to an .om offline model adapted to the Ascend AI Processor by referring to 4 Model Conversion.

- On the **Model Information** tab page of the **Model Converter** dialog box:
  - Select the .prototxt model file and .caffemodel pre-trained model file obtained in **Step 1**.
  - For **Model Name**, retain its default value resnet50. To use another name, change the value of omModelPath in src/sample_process.cpp of the current project before building.
    ```c++
    const char* omModelPath = "./model/resnet50.om";
    ```
- In the current sample, the data type of the model input node is FP16. Therefore, the **Type** parameter in the **Input Nodes** area can only be set to **FP16 (float16)**.

- During model conversion, on the **AIPP** tab page in the **Model Converter** dialog box, keep the **Data Pre-Processing** toggle switch off (off by default), since the current sample does not support AIPP.

- During model conversion, on the **Advanced Options Preview** tab page, switch on the **Additional Arguments** toggle and add the following command-line argument: **--output_type=FP32**
  - In the current sample, the float32 data type is used for processing the inference result. Therefore, **--output_type=FP32** is added. Otherwise, the processing result is incorrect.

- After the model conversion is complete, the **resnet50.om** file is generated in **/modelzoo/resnet50/device** in the MindStudio installation user home directory on the MindStudio installation server.

**Step 3** Add one or more model files.

1. Right-click the project name and choose **Add Model** from the shortcut menu. In Windows, if a message is displayed indicating that you do not have the permission to add a model, run MindStudio again as the administrator. For details, see 7.5.1 Insufficient Permissions to Add a Model or Add Data In an Application Project In Windows.

2. In the displayed dialog box, select one or more .om files. (Press and hold **Ctrl** to select multiple files.)

   Note the following restrictions when adding models:
   - The MindStudio installation user must have the permission to access the .om files.
   - You can add a maximum of 1024 .om files. The size of a single .om file must be within 1 GB, and the total size of all .om files must be less than 10 GB.
   - The .om file name can consist of up to 64 characters, including letters, digits, and the following special characters: .-_

3. Click **Add**.

   You can view the added model files in the **Project** window.

   ```
   └── ProjectName
       └── model
           └──*.om       //Model file
   ```

**Step 4** Prepare inference data.

- In Linux, prepare the input images by referring to "Prepare input images" under section "Build and Run" in the **README.md** file in the root directory of the created application project.

- In Windows, you can perform the following steps to convert the image format. The following uses the image stored in the **C:\Users\user1\AscendProjects\MyApp1\data** directory as an example.

  a. Open the system CLI.
  b. Run the following command to go to the **C:\Users\user1\AscendProjects\MyApp1\data** directory:

     ```
     cd C:\Users\user1\AscendProjects\MyApp1\data
     ```
7.3.3 (Optional) Adding a Custom DLL

You might need to add third-party dynamic link libraries (DLLs) in your applications. For example, you can use OpenCV to postprocess the YOLOv3 inference result. In the scenario where the development environment and operating environment are deployed on different machines, if the operating environment can be accessed only through MindStudio, remote execution fails because the required dynamic link library cannot be found. To solve this problem, the custom DLL support has been added to MindStudio application development to improve development efficiency.

To add a custom DLL, perform the following steps:

**Step 1** Cross-compile the DLL.

For details about the cross compilation procedure, see the corresponding document of the DLL in use. Note that the cross compiler must match the OS architecture of the operating environment.

**Step 2** Create a directory for storing the DLL and copy or soft link the cross-compiled DLL to the directory. As shown in Figure 7-38, the lib directory is created to store libcustom.so.

![Figure 7-38 DLL directory](image)

Configure the DLL directory environment variable when setting the run configurations of the application project (Running an Application Project). Configure environment variable according to the directory of the main executable file (`project_name/out/main`). As shown in the screenshot, `../mylib` indicates the DLL directory, which is under the project directory. Two configuration modes are provided:

- Enter an environment variable in the Environment Variables text box, as shown in Figure 7-39.

![Figure 7-39 Entering an environment variable](image)

- Click in the dialog box that is displayed, click and add an environment variable.
7.3.4 Developing an Application

**NOTE**
- Currently, MindStudio supports only C++.
- The path of the model to be loaded cannot contain spaces.

**Acl Project Mode**

For details about how to develop an application, see *Application Software Development Guide (C&C++)* listed in Table 7-1.

**Acl ResNet-50 Mode**

- After an application project is created, in the *src* directory you can find the .cpp template code of the application for system initialization, model execution, model uninstallation, resource destruction, and more. You only need to make the following modifications to *sample_process.cpp* based on the model file name, image name, and path (relative path only).
  - Change the image file names and paths as needed.
    ```cpp
    //The .. directory is relative to the directory of the executable file.
    //For example, if the executable file is stored in the out directory, .. indicates the upper-level directory of the out directory.
    string testFile[] = {
        "../data/dog1_1024_683.bin",
        "../data/dog2_1024_683.bin"
    };
    ```
  - Change the name of the .om model file as needed.
    ```cpp
    //The .. directory is relative to the directory of the executable file.
    //For example, if the executable file is stored in the out directory, .. indicates the upper-level directory of the out directory.
    const char* omModelPath = "../model/resnet50.om";
    ```

In the template code, APIs prefixed with *acl* are available to users. For details about the APIs, see "AscendCL API Reference" in *Application Software Development Guide (C&C++)*, as listed in Table 7-1.
7.3.5 Building and Running an Application Project (in Linux)

Building an Application Project

**Step 1** (Optional) Modify the `src/CMakeLists.txt` file in the project directory.

- **include_directories**: Add the directories of the header files to be included.
  
  Example:
  ```
  include_directories(
    directoryPath1
    directoryPath2
  )
  ```

- **link_directories**: Add the directories of the library files to be linked with.
  
  Example:
  ```
  link_directories(
    directoryPath3
    directoryPath4
  )
  ```

- **add_executable**: Add the directory of the .cpp file.
  
  Example:
  ```
  add_executable(
    main
    directoryPath5
    directoryPath6
  )
  ```

- **target_link_libraries**: Add the library files on which application building depends.
  
  Example:
  ```
  target_link_libraries(
    main
    ascendcl
    libName1
    libName2
  )
  ```

- **install**: Install the build output `main` to a specified path.
  
  Example:
  ```
  install(TARGETS main DESTINATION ${CMAKE_RUNTIME_OUTPUT_DIRECTORY})
  ```

**NOTE**

For details about the CMake parameters, visit [https://cmake.org/cmake/help/latest/guide/tutorial/index.html](https://cmake.org/cmake/help/latest/guide/tutorial/index.html) and query the required version.

**Step 2** In the MindStudio project window, choose Build > Edit Build Configuration....

1. Configure build parameters.

   If you want to run the built executable file in the operating environment, the options displayed in the Target OS drop-down list box are related to the installed ACLlib component. You can view the basic information about the ACLlib component in the ADK Manager dialog box. For details about how to open the ADK Manager dialog box, see [ADK Manager Overview](#).

   - **Linux**: Select Target OS and Target Architecture according to the Host OS Arch field of the ACLlib component in the ADK Manager dialog box.
Figure 7-41 Build parameter configuration in the Linux scenario

2. Click **Build** to build the project.

   If no error message is displayed and the message "build successfully" is displayed, the build is successful. Find the generated executable file **main** in the following directory:

   ```
   ├── Project name
   │   ├── out
   │   │   └── main         // Executable file
   ```

   ----End

Running an Application Project

**Step 1** In the MindStudio project window, choose **Run > Edit Configurations...** to open the run configuration dialog box.

**Step 2** Choose **Ascend App > Project name** to set running parameters of the application project. **Table 7-9** describes the running parameters shown in **Figure 7-42**.
Table 7-9 Description of running parameters of an application project

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the run configuration (customizable).</td>
</tr>
<tr>
<td>Executable</td>
<td>Directory of the executable file.</td>
</tr>
<tr>
<td>Run Mode</td>
<td>● Remote Run</td>
</tr>
<tr>
<td></td>
<td>● Local Run</td>
</tr>
<tr>
<td>SSH Connection</td>
<td>SSH connection. It is available when the Remote Run mode is selected. For details, see 12.1 SSH Connection Management. Select the IP address of the remote server that runs the application from the drop-down list box. If the IP address is not added, click to add it. For details, see 12.1.3 Adding a New SSH Connection.</td>
</tr>
<tr>
<td>Command Arguments</td>
<td>Argument passed for application running (customized). The arguments are optional and related to user code implementation.</td>
</tr>
<tr>
<td>Environment Variables</td>
<td>Environment variable configuration. Enter environment variables in the text box or click to add more by clicking in the displayed dialog box.</td>
</tr>
<tr>
<td>Before launch</td>
<td>Pre-processing before running the application (optional). For example, click to add Ascend App Build to build the executable file before running it.</td>
</tr>
</tbody>
</table>

Step 3 After the configuration is complete, click Apply to save the run configuration and click OK to close the run configuration dialog box.

Step 4 In the MindStudio project window, choose Run > Run....

In the dialog box displayed, select the created run configuration and run the application.

- If no error message is displayed and the message "Running *** finished" is displayed, the execution is complete.
- If an error message is displayed during application execution, view the detailed logs of the operating environment by referring to Log Reference listed in Table 7-1.
  - The default log path is SHOME/ascend/log. For details, see the "Log Files" section.
  - The default log level is ERROR. For details about how to query or change the log level, see the "Log Level Setting" section.

---End
7.3.6 Building and Running an Application Project (in Windows)

**NOTE**

To select the Local Build option in Build Mode on the build parameter configuration page and the Local Run option in Run Mode on the run parameter configuration page, install the software package of the Windows version by referring to (Optional) Installing the Windows ACLlib Package. Otherwise, the two options are unavailable.

Building an Application Project

**Step 1** (Optional) Modify the `src/CMakeLists.txt` file in the project directory.

- **include_directories:** Add the directories of the header files to be included.
  
  Example:
  ```c
  include_directories(
      directoryPath1
      directoryPath2
  )
  ```

- **link_directories:** Add the directories of the library files to be linked with.
  
  Example:
  ```c
  link_directories(
      directoryPath3
      directoryPath4
  )
  ```

- **add_executable:** Add the directory of the .cpp file.
  
  Example:
  ```c
  add_executable(
      main
      directoryPath5
      directoryPath6
  )
  ```

- **target_link_libraries:** Add the library files on which application building depends.
  
  Example:
  ```c
  target_link_libraries(
      main
      ascendcl
      libName1
      libName2
  )
  ```

- **install:** Install the build output main to a specified path.
  
  Example:
  ```c
  install(TARGETS main DESTINATION ${CMAKE_RUNTIME_OUTPUT_DIRECTORY})
  ```

**NOTE**

For details about the CMake parameters, visit https://cmake.org/cmake/help/latest/guide/tutorial/index.html and query the required version.

**Step 2** In the MindStudio project window, choose Build > Edit Build Configuration....

1. Configure build parameters. In Figure 7-43, the x86 architecture is used as an example.
2. Click **Build** to build the project.

If no error message is displayed and the message “build successfully” is displayed, the build is successful. View the generated executable file **main** in the following directory:

```
├── Project name
│   ├── out
│   │   └──main  //Executable
```

---End

**Running an Application Project**

**Step 1** In the MindStudio project window, choose Run > Edit Configurations... to open the run configuration dialog box.

**Step 2** Choose Ascend APP > **Project name** to configure run parameters of the application project according to **Table 7-10**.
Figure 7-44 Run parameter configuration

![Run parameter configuration](image)

Table 7-10 Description of run parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the run configuration (customizable).</td>
</tr>
<tr>
<td>Executable</td>
<td>Directory of the executable file.</td>
</tr>
<tr>
<td>Run Mode</td>
<td>• Remote Run</td>
</tr>
<tr>
<td></td>
<td>• Local Run</td>
</tr>
<tr>
<td>SSH Connection</td>
<td>SSH connection. For details, see 12.1 SSH Connection Management.</td>
</tr>
<tr>
<td></td>
<td>Select the IP address of the remote server that runs the application from</td>
</tr>
<tr>
<td></td>
<td>the drop-down list box. If the IP address is not added, click to add it.</td>
</tr>
<tr>
<td></td>
<td>For details, see 12.1.3 Adding a New SSH Connection.</td>
</tr>
<tr>
<td>Command Arguments</td>
<td>Arguments passed for application running (customized). The arguments are</td>
</tr>
<tr>
<td></td>
<td>optional and related to user code implementation.</td>
</tr>
<tr>
<td>Environment</td>
<td>Environment variable configuration. Enter environment variables in the</td>
</tr>
<tr>
<td>Variables</td>
<td>text box or click to add more by clicking in the displayed dialog box.</td>
</tr>
<tr>
<td>Before launch</td>
<td>Preprocessing before running the application (optional). For example, click</td>
</tr>
<tr>
<td></td>
<td>to add Ascend App Build to build the executable file before running it.</td>
</tr>
</tbody>
</table>
Step 3 After the configuration is complete, click **Apply** to save the run configuration and click **OK** to close the run configuration dialog box.

Step 4 In the MindStudio project window, choose **Run > Run...**.

In the dialog box displayed, select the created run configuration and run the application.

- If no error message is displayed and the message "Running *** finished" is displayed, the execution is complete.
- If an error message is displayed during application execution, view the detailed logs of the operating environment by referring to **Log Reference** listed in **Table 7-1**.
  - The default log path is $HOME/ascend/log$. For details, see the "Log Files" section.
  - The default log level is **ERROR**. For details about how to query or change the log level, see the "Log Level Setting" section.

---End

7.3.7 AscendCL Sample Project Usage

This section uses the AscendCL sample project ACL ResNet-50. For more samples, see [https://gitee.com/ascend/samples](https://gitee.com/ascend/samples).

**Function Description**

This sample shows how to classify images based on the Caffe ResNet-50 network (single input with batch size = 1).

Convert the model file of the Caffe ResNet-50 network to an .om offline model adapted to the Ascend AI Processor. In the sample, load the .om file, decode, resize, and infer two .jpg images, process the obtained inference results, and output the class indexes with the top 5 confidence values of each image.
**Figure 7-45** Sample diagram

```
*jpg image
1024 x 683

Convert and resize using
transferPic.py

*bin file
224 x 224

Model Inference (synchronous)

Postprocess inference result

Class indexes of top 5 confidences
```

**Principles**

The following table lists the key functions involved in this sample. For details about the APIs, see [AscendCL API Reference](#).

<table>
<thead>
<tr>
<th>Initialiation</th>
<th>aclInit: initializes AscendCL.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>aclFinalize: deinitializes AscendCL.</td>
</tr>
<tr>
<td>Device Manageme nt</td>
<td>aclrtSetDevice: sets the compute device.</td>
</tr>
<tr>
<td></td>
<td>aclrtGetRunMode: obtains the run mode of the Ascend AI Software Stack. The internal processing process varies according to the run mode.</td>
</tr>
<tr>
<td></td>
<td>aclrtResetDevice: resets the compute device and cleans up all resources associated with the device.</td>
</tr>
<tr>
<td>Context Manageme nt</td>
<td>aclrtCreateContext: creates a context.</td>
</tr>
<tr>
<td></td>
<td>aclrtDestroyContext: destroys a context.</td>
</tr>
<tr>
<td>Stream Manageme nt</td>
<td>aclrtCreateStream: creates a stream.</td>
</tr>
<tr>
<td></td>
<td>aclrtDestroyStream: destroys a stream.</td>
</tr>
<tr>
<td>Memory Manageme nt</td>
<td>aclrtMalloc: allocates device memory.</td>
</tr>
<tr>
<td></td>
<td>aclrtFree: frees device memory.</td>
</tr>
</tbody>
</table>
### Data Transfer

aclrtMemcpy (for running the app on the host):
- Transfers data from the host to the device for decoding.
- Transfers the inference result from the device to the host.

If your application runs on the device, data transfer is not needed.
If the application runs on the developer board, data transfer is not needed.

### Model Inference

- aclmdlLoadFromFileWithMem: loads a model from an .om file.
- aclmdlExecute: performs synchronous model inference.
- aclmdlUnload: unloads a model.

### Data Post-processing

Provides sample code to process the model inference result and display the category indexes with top 5 confidences on the device.
The sample provides a custom API DumpModelOutputResult, which is used to write the model inference result to a file (after the executable file is executed, the inference result file is generated in the directory of the executable file in the operating environment). This API is not called by default. To call this API, add the following code before the OutputModelResult API in sample_process.cpp in advance.

```c++
// print the top 5 confidence values with indexes.use function DumpModelOutputResult
// if want to dump output result to file in the current directory
processModel.DumpModelOutputResult();
processModel.OutputModelResult();
```

### Usage

**Step 1** You can create or import an application project.
- Create an application project: For details, refer to Creating an Application Project to create the Acl Project Sample-ResNet50 sample project.
- Import an application project:
  - Run the `git clone -b master https://gitee.com/ascend/samples.git` command to download the sample code. The acl_resnet50 sample code is named `resnet50_imagenet_classification` in `/samples/cplusplus/level2_simple_inference/1_classification` under the download directory.
  - For details, see Importing an Application Project.

**Step 2** For details about how to prepare model files and data, see Acl ResNet-50 Mode.

**Step 3** For details about how to develop an application, see 7.3.4 Developing an Application.

**Step 4** Build the application project.

In the MindStudio project window, choose Build > Edit Build Configuration....
1. Configure build parameters.
If you want to run the built executable file in the operating environment, the options displayed in the **Target OS** drop-down list box are related to the installed ACLlib component. You can view the basic information about the ACLlib component in the **ADK Manager** dialog box. For details about how to open the **ADK Manager** dialog box, see **ADK Manager Overview**.

- **Linux**: Select **Target OS** and **Target Architecture** according to the **Host OS Arch** field of the ACLlib component in the **ADK Manager** dialog box.

![Figure 7-46 Build parameter configuration in the Linux scenario](image)

2. Click **Build** to build the project.
If no error message is displayed and the message "build successfully" is displayed, the build is successful. Find the generated executable file **main** in the following directory:

```
├── Project name
│   ├── out
│   │   ├──main // Executable file
```

**Step 5** Run the application project.

1. Configure run parameters.
**Figure 7-47** shows a configuration example in Linux. In Windows, the **Local Run** option in **Run Mode** is unavailable. For details, see **7.3.6 Building and Running an Application Project (in Windows)**. After the configuration is complete, click **Apply** to save the run configuration and click **OK** to close the run configuration dialog box.
2. In the MindStudio project window, choose Run > Run.... In the dialog box displayed, select the created run configuration and run the application. Figure 7-48 shows an example of the execution result. The indexes and values vary according to the actual situation of the operating environment.
If an error message is displayed during application execution, view the detailed logs of the operating environment by referring to Log Reference listed in Table 7-1.

- The default log path is $HOME/ascend/log. For details, see the "Log Files" section.
- The default log level is ERROR. For details about how to query or change the log level, see the "Log Level Setting" section.

----End

7.4 Debugging an Application Project
7.4.1 Preparations

CAUTION

In Windows, the debugging function is unavailable.

Installing Dependencies

- To support debugging for cross architectures, install gdb-multiarch on the server (UI host) where MindStudio is installed. The following is a command example for dependency installation on the Ubuntu server:

  \texttt{apt install gdb-multiarch}

  NOTE

  - Run the \texttt{gdb-multiarch} command. If the gdb-multiarch version information is displayed, the installation is successful. If an error occurs indicating that the .so file is missing, manually install the missing .so file.
  - The installed gdb-multiarch and gdbserver software is used only for debugging user-developed programs and has no security risks.

- For remote debugging, install gdbserver on the remote server (AI host) and install ssh-askpass on the server (UI host) where MindStudio is installed. The following are examples of the installation commands in the Ubuntu environment. If your OS is CentOS or EulerOS, install the openssh-askpass and gdb-gdbserver.

  \texttt{apt-get install gdbserver}

  \texttt{apt install ssh-askpass}

  If gdbserver cannot be installed directly, compile gdbserver as follows:

  a. Click \texttt{here} to obtain the gdb source code. \texttt{gdb-8.1.1.tar.gz} is recommended.
  b. Obtain and install the cross compiler as required.
  c. Cross compile gdbserver as follows. \texttt{gdb-8.1.1.tar.gz} is used as an example in the following:

     i. Decompress the gdb source code package and go to the \texttt{gdbserver} subdirectory.

        \texttt{tar xf gdb-8.1.1.tar.gz}

        \texttt{cd gdb-8.1.1/gdb/gdbserver}

     ii. Set compile parameters.

        \texttt{./configure --host=\{compiler\} --prefix=\$HOME/install/gdbserver}

        Replace \texttt{\{compiler\}} with the actual cross compiler. Replace \texttt{\$HOME/install/gdbserver} with the actual installation directory.

     iii. Run the compile and install commands.

        \texttt{make -j}

        \texttt{make install}

  d. Deploy gdbserver.

      Copy the compiled gdbserver to the target running server.
Configuring the Compilation Environment and Operating Environment

Step 1  On the toolbar, choose Build > Edit Build Configuration... to open the compilation configuration window. Select Debug from Build Type, and click Build. The executable file for debugging is generated, and the debug icon 🔄 on the toolbar is turned on. The executable file is generated to the ProjectName\out directory.

Step 2  On the toolbar, choose Run > Edit Configurations... to open the running configuration window.
- Executable: Select the path of the executable file used for compilation to the last level.
- Run Mode: Select Local Run or Remote Run.
- Environment Variables: Enter the path of the DLL. See Figure 7-49.

Figure 7-49 DLL directory

You can also click 📡, and click 📡 on the displayed page. Then, enter the path.

Step 3  If Local Run is selected, click Apply and then OK to complete the configuration. If Remote Run is selected, configure the remote server in SSH Connection. Click Apply and then OK to complete the configuration. For details about SSH connection, see 12.1 SSH Connection Management.

----End

7.4.2 Breakpoint Management

An at-line breakpoint suspends the program execution at a specific point, for checking the variables, memory, and correction of the program logic. Breakpoints can be set in multiple places, for example, at the entry of function execution, or at a specific line of a file.
Adding a Breakpoint

To add a breakpoint, open the code to be debugged, select the line at which you want to add the breakpoint, and click the area next to the line number, as shown in Figure 7-51. After a breakpoint is added, the icon  is displayed next to the line number.

Figure 7-51 Adding a breakpoint

```cpp
int main()
{
    SampleProcess sampleProcess;
    Result ret = sampleProcess.InitResource();
    if (ret != SUCCESS) {
        ERROR_LOG("sample init resource failed");
        return FAILED;
    }
}
```

Deleting a Breakpoint

To delete a breakpoint, click the icon.

Disabling a Breakpoint

To use only some of the breakpoints without deleting the unneeded ones during debugging, disable the unneeded breakpoints as follows:

Right-click  and deselect Enabled in the displayed dialog box. You can see that  changes to  . In this case, the disabled breakpoint does not take effect during debugging next time.

Figure 7-52 Disabling a breakpoint

Enabling a Breakpoint

To enable a disabled breakpoint during debugging, perform the following steps:

Right-click  and select Enabled in the displayed dialog box. You can see that  changes to  . In this case, the newly enabled breakpoint takes effect in next debugging.
The restart of MindStudio or the project does not affect the states or the number of breakpoints set in the current project.

### Viewing Breakpoints

To view the set breakpoints, perform the following steps:

Click **More (Ctrl+Shift+F8)**. The dialog box for viewing breakpoints is displayed, as shown in Figure 7-54.

**Figure 7-54 Viewing the breakpoints**

In the preceding figure, area 1 displays all the breakpoints set in the current code, area 2 indicates the **Disable** or **Enable** state of a breakpoint, and area 3 displays the code.

- In area 1, if the check box of a breakpoint is selected, **Enabled** is selected in area 2, and **●** appears next to the line number in area 3. Alternatively, if the check box of a breakpoint in area 1 is not selected, **Enabled** in area 2 is deselected, and **○** appears next to the line number in area 3.
- To delete a breakpoint, select the breakpoint in area 1 and click **—**. In this case, the icon next to the line number in area 3 is deleted.
- To add a breakpoint, click a code line number in area 3. In this case, information about this breakpoint is automatically added to area 1.
Adding Variables to the Watches Area

Right-click the variable to be viewed and choose Add to Watches from the shortcut menu. The variable is added to the Watches area, as shown in Figure 7-55.

Figure 7-55 Adding variables to the Watches area

You can also click the icons on the toolbar on the left to perform more functions:

- +: adds a variable to the Watches area.
- -: deletes the selected variable.
- //: copies the selected variable.

You can modify the parameter based on the copied variable by right-clicking the copied variable, choosing Edit from the shortcut menu, and entering a new variable name.

- //: opens a new Watches window if there are too many variables or threads in the Watches area. The variables are automatically displayed in this window, as shown in Figure 7-56. If you want to merge the Watches window into the Variables window, click this icon in the upper part of the Watches window again.

Figure 7-56 Opening the Watches window

7.4.3 Debugging

Starting Debugging

1. After the breakpoint is configured, click the debug icon on the toolbar to start debugging.

NOTE

- To enable debugging, MindStudio needs to load the model file, data file, and debugging dependency. The startup process takes about 1 minute.
- If Remote Run is selected, enter the password for logging in to the remote server again after clicking .
2. When the MindStudio page stops at the first breakpoint, the value of the variable is automatically displayed on the right of the line where the code variable is located, as shown in the information in green in Figure 7-57. The Debugger view is displayed in the lower right corner of the project window.

   - The **Frames** area on the left is the program's method calling stack. This area displays the methods that have been called by the program to the breakpoint. The methods at the bottom are called earlier. The information in the red box indicates the code line where the current debugging program stays, for example, line 19 of the main.cpp file.

   - The **Variables** area on the right displays the variable and thread information.

**Figure 7-57 Debug window**

Table 7-11 describes the icons in the Debug window.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🔄</td>
<td>Rerun(Ctrl+F5)</td>
<td>Reruns the debug.</td>
</tr>
<tr>
<td>🚦</td>
<td>Resume program(F9)</td>
<td>Resumes the program running until the next breakpoint.</td>
</tr>
<tr>
<td>🚦</td>
<td>Stop(Ctrl+F2)</td>
<td>Stops the debug.</td>
</tr>
</tbody>
</table>

Runs the code that needs to be executed between two breakpoints. If there is no breakpoint in the subsequent code, the program will be finished once you click the icon again.

---

Copyright © Huawei Technologies Co., Ltd. 2021-06-24
<table>
<thead>
<tr>
<th>Icon</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Icon" /></td>
<td>View Breakpoints (Ctrl+Shift+F8)</td>
<td>Views all breakpoints. Views the set breakpoints and sets attributes to the breakpoints. For details about the displayed window and its functions, see Viewing Breakpoints.</td>
</tr>
<tr>
<td><img src="image2" alt="Icon" /></td>
<td>Mute Breakpoints</td>
<td>Mutes all breakpoints. For details, see Mute Breakpoints.</td>
</tr>
<tr>
<td><img src="image3" alt="Icon" /></td>
<td>Step Over (F8)</td>
<td>Goes to the next code line. Click this icon to make the program execute the next line. If a function is called in the current line, it will be executed and returned. Then, the program will display the frames and variables in the next line. For details, see Step Over.</td>
</tr>
<tr>
<td><img src="image4" alt="Icon" /></td>
<td>Step Into (F7)</td>
<td>Steps into the function. The program executes the next line. If a user-defined function is in this line, the user-defined function is executed (without entering the official library), and the frames and variables are displayed. For details, see Step Into.</td>
</tr>
<tr>
<td><img src="image5" alt="Icon" /></td>
<td>Force Step Into (Alt+Shift+F7)</td>
<td>Forcibly steps into a function. Click this icon to debug any function, including user-defined functions and official library functions.</td>
</tr>
<tr>
<td><img src="image6" alt="Icon" /></td>
<td>Step Out (Shift+F8)</td>
<td>Steps out of a function. If you steps into a function during debugging and the function is normal, click this icon to step out of the function and return to the original position. For details, see Step Out.</td>
</tr>
<tr>
<td><img src="image7" alt="Icon" /></td>
<td>Run to cursor (Alt+F9)</td>
<td>Runs to the cursor. If no breakpoint is set, the program pauses at the line where the cursor is located, and the frames and variables are displayed. For details, see Run to cursor.</td>
</tr>
</tbody>
</table>

**NOTE**
- Only C++ files can be debugged.
- When multiple C++ files are run at the same time, functions such as starting debugging, stopping debugging, single-step debugging, stepping into a function, and stepping out of a function take effect only for the selected files in the current Debug window.
The following describes the functions of each icon with examples.

**Step Over**

1. Set a breakpoint.
2. After debugging is enabled, the **Debug** window stops at the breakpoint.
3. Click 🔄 to execute the next line, as shown in Figure 7-58.

**Figure 7-58 Step Over**

![Before](image1)

![After](image2)

**Step Into**

1. Set a breakpoint.
2. After debugging is enabled, the **Debug** window stops at the breakpoint.
3. Click ↓ to execute the next line. If there is a customized method in the line, run the customized method, as shown in Figure 7-59.
Step Into

During debugging, if you step into a function through **Step Into**, for example, **GetFileList**, and find that the function is normal, you can click \( \uparrow \) to step out of the function and return to the original position, as shown in Figure 7-60.

Step Out
**Figure 7-60 Step Out**

Before

![Before Image]

After

![After Image]

**Run to cursor**

If no breakpoint is set for a code line and you want to debug the code line, click to move the breakpoint to the code line where the cursor is located, as shown in **Figure 7-61**.
Mute Breakpoints

If you want to mute all breakpoints during debugging, click 🗝️. Then all breakpoints are grayed out, as shown in Figure 7-62, and the icons turn gray. Click 🕵️‍♂️ to finish debugging directly.
7.5 FAQ

7.5.1 Insufficient Permissions to Add a Model or Add Data In an Application Project In Windows

Symptom

In Windows, error message “You do not have sufficient privilege to perform this operation.” is displayed when a user adds a model or adds data in an application project (see Figure 7-63).
Figure 7-63 Add Model

Cause

Adding a model or data is to create a symbolic link. To create a symbolic link in Windows, you must have the administrator permissions. If you open MindStudio as a non-administrator, a message is displayed indicating that you do not have the permission.

Solution

Launch MindStudio again as the administrator.
8 Custom Operator Development

8.1 Before You Start
8.2 Development Workflow
8.3 Operator Analysis
8.4 Project Creation
8.5 TBE Operator Development (TensorFlow)
8.6 TBE Operator Development (PyTorch)
8.7 TBE Operator Development (MindSpore)
8.8 AI CPU Operator Development (TensorFlow)
8.9 Other Functions and Operations

8.1 Before You Start

This section describes how to use MindStudio to develop a custom operator in different frameworks, including TBE operator development in MindSpore, PyTorch, and TensorFlow, as well as AI CPU operator development in TensorFlow. The following mainly focuses on MindStudio operations but also touches on important code implementations. For details about the programming, see the corresponding guide listed in the following tabel.

Table 8-1 Reference documents

<table>
<thead>
<tr>
<th>Product</th>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlas 200 DK</td>
<td>TBE Custom Operator Development Guide</td>
</tr>
</tbody>
</table>
MindStudio provides operator development samples in the `/MindStudio/samples/operator_demo_projects` directory in the MindStudio installation path, including:

- TBE operator "Square" of the MindSpore operator project, stored in the `/mindspore_operator_sample` directory
- TBE operator "Add" and AI CPU operator "ReshapeCust" of the TensorFlow operator project, stored in the `/tbe_operator_sample` and `/aicpu_operator_sample` directories, respectively.
- For more examples, see 8.9.4 Custom Operator Samples.

The `Ascend/ascend-toolkit/xxx/opp/op_impl/built-in/ai_core/tbe/impl` directory stores the implementation files of the built-in AI Core operators supported by the Ascend AI Processor. You can refer to these files when implementing the operator code.

### 8.2 Development Workflow

#### 8.2.1 TensorFlow TBE Operator Development Workflow

Figure 8-1 shows the workflow of developing a TensorFlow TBE operator in MindStudio.
1. **Operator analysis**: Determine the operator functionality, inputs, outputs, select an operator development mode, and name the operator type and implementation function.

2. **Project creation**
   Create a TBE operator project in MindStudio. After that, the operator project directory and corresponding file templates are automatically generated. You can develop operators based on these templates.

3. **Operator development**
   - **Operator code implementation**: Describe the implementation process of an operator.
   - **Operator prototype definition**: Define the constraints on the operator to run on Ascend AI Processor, mainly by defining operator inputs, outputs, attributes, and their value ranges, verifying arguments, and inferring shape. The information defined by the prototype is registered with the operator prototype library of GE. During network execution, GE calls the verification API of the operator prototype library to verify operator arguments. If the verification passes, GE infers the output shape and dtype of each node by calling the inference function of the operator prototype library and allocates static memory for the result tensor.
   - **Operator information definition**: Register the operator information to the operator information library, including the input and output dtype and format, and input shape of the operator. During network execution, FE performs basic verification based on the operator information in the operator information library, and determines whether to insert a proper conversion node for the operator. It also finds the corresponding operator implementation file based on the information in the operator information library and builds the operator binary file for execution.
   - **Operator plugin implementation**: If the custom operator is developed based on a third-party framework (such as TensorFlow/Caffe), after developing the custom operator implementation code, you need to
develop plug-ins to map the operator to one that adapts to Ascend AI Processor and register the operator information with GE. When the TensorFlow-/Caffe-based network is running, the plugin information in GE is loaded and called first to parse and map the operators in the TensorFlow/Caffe network to operators that adapt to the Ascend AI Processor.

- **Operator implementation verification**
  
  Run test cases to verify the function and logic of the operator implementation code.
  
  - **UT**: verifies the operator implementation logic in a simulation environment, including implementation code of the operator logic and the operator prototype definition.
  
  - **ST**: verifies the operator functionality in actual hardware with automatically generated test cases.

4. **Operator building**

   Build the operator plug-in implementation file, prototype definition file, and information definition file into the corresponding operator plug-in library, operator prototype library, and operator information library.

5. **Operator deployment**

   Deploy the operator implementation file, built operator plug-in, operator prototype library, and operator information library in the the Ascend AI Processor OPP for running the operator on the network.

### 8.2.2 PyTorch TBE Operator Development Workflow

**Figure 8-2** shows the flow of developing a PyTorch TBE operator in MindStudio.
1. **Operator analysis**: Determine the operator functionality, inputs, outputs, select an operator development mode, and name the operator type and implementation function.

2. **Project creation**
   Create a TBE operator project in MindStudio. After that, the operator project directory and corresponding file templates are automatically generated. You can develop operators based on these templates.

3. **Operator development**
   - **Operator code implementation**: Describe the implementation process of an operator.
   - **Operator prototype definition**: Define the constraints on the operator to run on Ascend AI Processor, mainly by defining operator inputs, outputs, attributes, and their value ranges, verifying arguments, and inferring shape. The information defined by the prototype is registered with the operator prototype library of GE. During network execution, GE calls the verification API of the operator prototype library to verify operator arguments. If the verification passes, GE infers the output **shape** and **dtype** of each node by calling the inference function of the operator prototype library and allocates static memory for the result tensor.
   - **Operator information definition**: Register the operator information to the operator information library, including the input and output **dtype** and **format**, and input **shape** of the operator. During network execution, FE performs basic verification based on the operator information in the operator information library, and determines whether to insert a proper conversion node for the operator. It also finds the corresponding operator implementation file based on the information in the operator information library and builds the operator binary file for execution.
   - **Operator implementation verification**
     Run test cases to verify the function and logic of the operator implementation code.
     - **UT**: verifies the operator implementation logic in a simulation environment, including implementation code of the operator logic and the operator prototype definition.
     - **ST**: verifies the operator functionality in actual hardware with automatically generated test cases.

4. **Operator building**
   Build the operator plug-in implementation file, prototype definition file, and information definition file into the corresponding operator plug-in library, operator prototype library, and operator information library.

5. **Operator deployment**
   Deploy the operator implementation file, built operator plug-in, operator prototype library, and operator information library in the the Ascend AI Processor OPP for running the operator on the network.

6. **PyTorch operator adaptation**
   Adapt PyTorch operators through Neural Processing Unit (NPU) extension. NPU serves the function of memory management, device management, and operator call implementation.
8.2.3 MindSpore TBE Operator Development Workflow

Figure 8-3 shows the flow of developing a TBE operator in MindStudio.

1. **Operator analysis**: Determine the operator functionality, inputs, outputs, select an operator development mode, and name the operator type and implementation function.

2. **Project creation**
   Create a MindSpore TBE operator project in MindStudio. After that, the operator project directory and corresponding file templates are automatically generated. You can develop operators based on these templates.

3. **Operator development**
   - **Operator information registration file**: Describe the implementation of the internal compute logic of the operator through the TBE domain-specific language (DSL). The file describes the basic information about the TBE operator, such as the operator name and supported input and output types. Operator information is the key to guide the backend to select an operator and insert proper type and format transformation for the operator.
   
   - **Operator prototype definition**: Define the frontend API prototype of an operator as the basic unit of a network model, including the operator name, attributes (optional), input and output names, output shape inference method, and output dtype inference method.
   
   - **Operator implementation verification**
     Run test cases to verify the function and logic of the operator implementation code.

     - **UT**: verifies the operator implementation logic in a simulation environment, including implementation code of the operator logic and the operator prototype definition.
     
     - **ST**: verifies the operator functionality in actual hardware with automatically generated test cases.
8.2.4 TensorFlow AI CPU Operator Development Workflow

Figure 8-4 shows the workflow of developing a TensorFlow AI CPU operator in MindStudio.

Figure 8-4 TensorFlow AI CPU operator development workflow

1. **Operator analysis**: Determine the operator functionality, inputs, outputs, select an operator development mode, and name the operator type and implementation function.

2. **Project creation**
   Create a TensorFlow AI CPU operator project in MindStudio. After that, the operator project directory and corresponding file templates are automatically generated. You can develop operators based on these templates.

3. **Operator development**
   - **Operator implementation**: Implement the compute logic of the operator.
   - **Operator prototype definition**: Define the constraints on the operator to run on the Ascend AI Processor, mainly the mathematical meanings of the operator by defining operator inputs, outputs, attributes, and their value ranges, verifying arguments, and inferring shape. The information defined by the prototype is registered with the operator prototype library of GE. During offline model conversion, GE calls the verification API of the operator prototype library to verify operator arguments. If the verification passes, GE infers the output shape and dtype of each node by calling the inference function of the operator prototype library and allocates static memory for the result tensor.
   - **Operator information definition**: Register the operator information to the operator information library, including the OpType, and input and output dtype and name. During network execution, AI CPU Engine performs basic verification and operator matching based on the operator information in the operator information library.
   - **Operator adaptation plug-in implementation**: In the custom operator development scenario based on a third-party framework (such as TensorFlow/Caffe), after developing implementation code of the custom
operator, you need to develop an adaptation plug-in to map the third-party operator to an operator supported by the the Ascend AI Processor.

4. **Operator building**
Build the operator adaptation plug-in implementation file, prototype definition file, and information definition file into the operator plug-in library, operator prototype library, and operator information library.

5. **Operator deployment**
Place the operator implementation file, plug-in library file, prototype library, and information library in the corresponding directories of OPP.

6. **Operator implementation verification**
**ST:** verifies the operator functionality in actual hardware with automatically generated test cases.

### 8.3 Operator Analysis

**Overview**

This section uses the TBE operator "Add" and the AI CPU operator as examples to show the workflow of operator analysis. You need to analyze the TBE operator "Square" of the MindSpore operator project by yourself.

**TBE Operator Analysis**

Before developing an Add operator by using the TBE DSL, you need to determine the operator functionality, inputs, outputs, select an operator development mode, and name the operator type and implementation function.

**Step 1** Specify the operator function and mathematical expression.

The mathematical expression of the Add operator is as follows:

\[ z = x + y \]

The Add operator adds two inputs and returns the result \( z \).

**Step 2** Specify the inputs and output.

- The Add operator has two inputs, \( x \) and \( y \), and outputs the result \( z \).
- The supported input data types include float16, float32, and int32. The output has the same data type as the inputs.
- The operator inputs support all shapes. The output has the same shape as the inputs.
- The operator input supports the following formats: NCHW, NC1HWC0, NHWC, and ND.

**Step 3** Determine the operator development mode and the compute API.

1. The compute process involves only the addition operation. The `te.lang.cce.vadd(lhs, rhs)` API can be used to implement \"x + y\" for preliminary analysis. For details, see *TBE Custom Operator Development Guide*.

2. The `te.lang.cce.vadd(lhs, rhs)` API requires that the two input tensors to have the same shape. Therefore, you need to obtain the larger shape of the two
input tensors, and then call the `te.lang.cce.broadcast(var, shape, output_dtype=None)` API to broadcast the input argument to a specified shape.

**Step 4** Specify the operator implementation file name, operator implementation function name, and **OpType**.

The naming rules are as follows:

- Name **OpType** in upper camel case and indicate the separation of words with a single capitalized letter.
- Name the operator file and operator function in either of the following ways:
  - To create user-defined names, configure `opFile.value` and `opInterface.value` in the **operator information definition file**.
  - If `opFile.value` and `opInterface.value` in the **operator information definition file** are not configured, FE obtains the operator function name by replacing **OpType** as follows:
    - Replace the first uppercase letter with a lowercase letter.
      Example: Abc -> abc
    - Replace each uppercase letter following lowercase letters with an underscore (_) and a lowercase letter.
      Example: AbcDef -> abc_def
    - Uppercase letters following a digit or an uppercase letter are regarded as a character string. If there is a lowercase letter after this string, replace the last uppercase letter in this string with an underscore (_) and a lowercase letter, and replace the other uppercase letters with lowercase letters. If there is no lowercase letter after the string, directly replace the string with lowercase letters.
      Examples: ABCDef -> abc_def; Abc2DEF -> abc2d_ef; ABC2dEF -> abc2d_ef

In this example, **OpType** of the operator is defined as **Add**. Uncapitalize the first letter to obtain the operator implementation file name and implementation function name, that is, **add**.

Based on the preceding analysis, the design specifications of the Add operator are as follows.

**Table 8-2** Design specifications

<table>
<thead>
<tr>
<th><strong>OpType</strong></th>
<th><strong>Add</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operator</strong></td>
<td><strong>Input</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
AI CPU Operator Analysis

Before developing an AI CPU operator, you need to determine the operator function, input, output, development mode, operator type (OpType), implementation function name, and more.

**Step 1** Specify the operator function and mathematical expression.

Take the Add operator as an example. The mathematical expression of the Add operator is as follows:

\[ z = x + y \]

The Add operator adds two inputs and returns a result.

**Step 2** Specify the inputs and output.

- The Add operator has two inputs, \( x \) and \( y \), and outputs the result \( z \).
- The supported input data types include float16, float32, and int32. The output has the same data type as the inputs.
- The operator inputs support all shapes. The output has the same shape as the inputs.
- The operator input supports the following formats: NCHW, NHWC, and ND.

**Step 3** Specify the operator implementation file name and operator type (OpType).

- Name **OpType** in upper camel case and indicate the separation of words with a single capitalized letter.
- Name the operator file in either of the following ways:
  
  Name the operator file after **OpType** as follows:
  
  - Convert the first uppercase letter to a lowercase letter.
  
  Example: Abc -> abc
  
  - Replace each uppercase letter following lowercase letters with an underscore (\_) and a lowercase letter.
  
  Example: AbcDef -> abc_def
Uppercase letters following a digit or an uppercase letter are regarded as a semantic string. If there is a lowercase letter after this string, replace the last uppercase letter in this string with an underscore (_) and a lowercase letter, and convert the other uppercase letters into lowercase letters. If there is no lowercase letter after the string, directly convert the string into lowercase letters.

Examples: ABCDef -> abc_def; Abc2DEf -> abc2d_ef; Abc2DEF -> abc2def; ABC2dEF -> abc2d_ef

In this example, OpType of the operator is defined as Add. You are advised to name the deliverables as follows:

- Code implementation (or kernel implementation) files of the operator: add_kernel.h and add_kernel.cc
- Plugin implementation file: add_kernel_plugin.cpp
- Prototype implementation files: add.h and add.cpp
- Information definition file: add.ini

Based on the preceding analysis, the design specifications of the Add operator are as follows.

<table>
<thead>
<tr>
<th>Table 8-3 Add operator design specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpType</td>
</tr>
<tr>
<td>Operator Input</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Operator Output</td>
</tr>
<tr>
<td>Operator Implementation File Name</td>
</tr>
</tbody>
</table>

8.4 Project Creation

8.4.1 Creating an Operator Project

Overview

This section describes how to create an Add operator project using MindStudio. After the project is created, the operator project directory and corresponding file templates are automatically generated. You can develop operators based on these templates.
Procedure

Step 1 Navigate to the New Project dialog box.
- If this is your first login to MindStudio, click Create new project on the MindStudio welcome window to go to the New Project dialog box.
- If this is not your first login to MindStudio, choose File > New > Project... on the menu bar to go to the New Project dialog box.

Step 2 Create an operator project.
1. On the navigation pane on the left, select Ascend Operator. Configure the operator project details on the right, as described in Table 8-4.

Table 8-4 Project information configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Project name (customizable). Enter a maximum of 64 characters, starting with a letter. Only letters, digits, hyphens (-), and underscores (_) are allowed.</td>
<td>MyOperator</td>
</tr>
<tr>
<td>Description</td>
<td>Project description (customizable).</td>
<td>(Optional)</td>
</tr>
<tr>
<td>ADK Version</td>
<td>Current version number of the ADK.</td>
<td>Current ADK version</td>
</tr>
<tr>
<td>Project Location</td>
<td>Project path.</td>
<td>Retain the default.</td>
</tr>
</tbody>
</table>

2. Click Next. On the window that is displayed, configure operator information, as described in Table 8-5.

Table 8-5 Operator information configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Operator (operator creation mode)</td>
<td>Creates an empty operator project. If this mode is selected, the Operator Type parameter is displayed in the lower part. You need to enter the OpType of the operator to be created by referring to 8.3 Operator Analysis.</td>
<td>Select from the three modes.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| From Template        | Creates an operator project from the IR definition template. The IR definition template file can be in JSON or Excel format. If this mode is selected, the Template File parameter is displayed in the lower part. You need to select the file that defines the IR prototype. **NOTICE**  
  - Ascend_IR_Template.json  
    Obtain the IR definition template file in JSON format from the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/toolkit/python/site-packages/op_gen/json_template` directory.  
  - Ascend_IR_Template.xlsx  
    Obtain the IR definition template file in Excel format from the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/toolkit/tools/msopgen/template` directory.  
    Modify the IR definition of the custom operator on the Op sheet in the template file.  
    For details about IR definition configuration, see IR Definition Configuration. |         |
| From Tensorflow      | Creates an operator project from the TensorFlow prototype definition.  
  To use this mode, download the TensorFlow source code on the server where MindStudio is installed. If this mode is selected, the following parameters are displayed:  
  - Operator Path: directory of the TensorFlow source code. You are advised to set it to `tensorflow/core/ops` for higher search efficiency.  
  - Operator Type: OpType of the operator to be created. |         |
| Plugin Framework     | Sets the original framework of the operator.  
  - MindSpore  
  - PyTorch  
  - TensorFlow | TensorFlow |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| Compute Unit| – If **Plugin Framework** is set to **MindSpore**, only **AI Core/Vector Core** can be selected.  
– If **Plugin Framework** is set to **PyTorch**, only **AI Core/Vector Core** can be selected.  
– If **Plugin Framework** is set to **TensorFlow**, the options are as follows:  
  ▪ **AI Core/Vector Core**: The operator runs on AI Core or Vector Core, indicating the TBE operator.  
  ▪ **AI CPU**: The operator runs on AI CPU, indicating the AI CPU operator.                                      | AI Core/Vector Core |

| Unit Type   | – This parameter is displayed when **Compute Unit** is set to **AI Core/Vector Core**.  
– This parameter is not displayed when **Plugin Framework** is set to **MindSpore**.                                                                 |                     |

3. **Click Finish.**  
If there is already an active project in the window, the message shown in Figure 8-5 is displayed.  
– **Click This Window** to open the created project in the current window.  
– **Click New Window** to open the created project in a new window.  

Figure 8-5 Prompt for opening a new project

<table>
<thead>
<tr>
<th>NOTICE</th>
</tr>
</thead>
</table>
| – The default path of the new project is under **AscendProjects** in the home directory of the installation user.  
– If you need to customize multiple TBE operators, implement them in the same operator project. Right-click the root directory of the operator project and choose **New > Operator** from the shortcut menu to add an operator.  
– In the same operator project, the names of the AI CPU operator and TBE operator cannot be the same. |
Step 3  View the directory structure and main files of the operator project.

The directory of a TensorFlow or PyTorch operator sample project is organized as follows.

```
├── .idea
│   └── build // Files generated after build
├── cmake // Directory of common files related to build
├── cpukernel // Directory of AI CPU operator files. This directory is not created for the
TensorFlow and PyTorch TBE operator projects.
│   ├── impl // Directory of the operator implementation files
│       │   └── reshape_cust_kernel.cc // Operator source file
│       │   └── reshape_cust_kernel.h // Operator header file
│   ├── op_info cfg // Directory of the operator information libraries
│       └── aicpu_kernel
│       └── reshape_cust.ini // Operator information definition file
│   └── CMakeLists.txt // Build script, which is called by the CMakeLists.txt file in the root
directory of the operator project.
│       ├── toolchain.cmake
│       │   └── framework // Directory of the operator plugin implementation files
│       │   └── tf_plugin // Directory of the operator plugin files and build scripts of the TensorFlow
framework. This directory is not generated when Plugin Framework is set to PyTorch.
│       │   └── tensorflow_add_plugin.cc // add indicates the operator type. The Add operator is used as an
example to show the TBE operator development workflow. The AI CPU operator sample equivalent is
reshape_cust.
│       └── CMakeLists.txt // Build script, which is called by the CMakeLists.txt file in the parent
directory.
├── framework // Directory of the operator plugin implementation files
└── CMakeLists.txt // Build script, which is called by the CMakeLists.txt file in the root
directory of the operator project.
├── op_proto // Directory of the operator IR definition files
│   └── add.cc // add indicates the operator type. The Add operator is used as an
example to show the TBE operator development workflow. The AI CPU operator sample equivalent is
reshape_cust.
│   └── add.h // add indicates the operator type. The Add operator is used as an
example to show the TBE operator development workflow. The AI CPU operator sample equivalent is
reshape_cust.
├── CMakeLists.txt // Build script, which is called by the CMakeLists.txt file in the root
directory of the operator project.
│   └── scripts // Project-related script
│       └── tbe // Directory of TBE operator files. This directory is not created for the
TensorFlow AI CPU operator project.
│       ├── impl // Directory of the operator implementation files
│           └── add.py // Operator implementation and registration file
│       ├── op_proto // Directory of the operator information libraries
│           └── {soc_version}
│               └── add.ini
│       └── project // Project information file, including the project type, project description,
target device type, and ADK version
├── CMakeLists.txt
└── MyOperator.iml
```

Directory structure and key files of a MindSpore operator sample project are organized as follows.

```
├── .idea
│   └── mindsplre // Directory of the operator implementation file
│       └── impl // Directory of the operator implementation and registration file
│           └── cus_square_impl.py // Operator implementation and registration file
├── op_proto
│   └── cus_square.py // Operator primitive file
└── project // Project information file, including the project type, project description, target
device type, and ADK version
    └── MyOperator.iml
```
**IR Definition Configuration**

- Excel file parameter description

### Table 8-6 Parameters in IR prototype definition

<table>
<thead>
<tr>
<th>Column Header</th>
<th>Description</th>
<th>Mandat ory or Not</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op</td>
<td>Operator type.</td>
<td>Yes</td>
</tr>
<tr>
<td>Classify</td>
<td>Parameter classification.</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>- Input</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- DYNAMIC_INPUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Output</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- DYNAMIC_OUTPUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Attr</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Parameter name.</td>
<td>Yes</td>
</tr>
<tr>
<td>Type</td>
<td>Parameter data type.</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Value range:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>tensor, int, bool, float, ListInt,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ListFloat, and more</td>
<td></td>
</tr>
<tr>
<td>TypeRange</td>
<td>Tensor data types.</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Value range:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fp16, fp32, double, int8, int16, int32,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>int64, uint8, uint16, uint32, uint64, bool, and more</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Selected from the following equivalents in the MindSpore scenario:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>I8_Default, I16_Default, I32_Default,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>I64_Default, U8_Default, U16_Default,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U32_Default, U64_Default, BOOL_Default, and more</td>
<td></td>
</tr>
<tr>
<td>Required</td>
<td>Whether an input is required:</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>- TRUE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- FALSE</td>
<td></td>
</tr>
<tr>
<td>Doc</td>
<td>Parameter description.</td>
<td>No</td>
</tr>
<tr>
<td>Attr_Default_value</td>
<td>Default value of an attribute.</td>
<td>No</td>
</tr>
<tr>
<td>Format</td>
<td>Tensor formats.</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Value range:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ND, NHWC, NCHW, HWCN, NC1HWC0, FRACTAL_Z, and more</td>
<td></td>
</tr>
</tbody>
</table>
JSON file parameter description

**Table 8-7 JSON parameter description**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>op</td>
<td>String</td>
<td>Operator type.</td>
<td>Yes</td>
</tr>
<tr>
<td>input_desc</td>
<td>List</td>
<td>Input description.</td>
<td>No</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>Input name.</td>
<td></td>
</tr>
<tr>
<td>param_type</td>
<td>String</td>
<td>Parameter classification.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>● required</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>● optional</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>● dynamic</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Defaults to <strong>required</strong></td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>List</td>
<td>Tensor formats.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Selected from:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ND, NHWC, NCHW,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>HWCN, NC1HWC0,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FRACTAL_Z, and</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>more</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>List</td>
<td>Parameter data types.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Selected from:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>float, int32, bool,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>int64, half, unit32,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>and more</td>
<td></td>
</tr>
<tr>
<td>output_desc</td>
<td>List</td>
<td>Output description.</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>Output name.</td>
<td></td>
</tr>
<tr>
<td>param_type</td>
<td>String</td>
<td>Parameter classification.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>● required</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>● optional</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>● dynamic</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Defaults to <strong>required</strong></td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Description</td>
<td>Required</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>format</td>
<td>List</td>
<td>Tensor formats. Selected from: ND, NHWC, NCHW, HWCN, NC1HWC0, FRACTAL_Z, and more</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>List</td>
<td>Parameter data types. Selected from: float, int32, bool, int64, half, unit32, and more</td>
<td></td>
</tr>
<tr>
<td>attr</td>
<td>List</td>
<td>Attribute description.</td>
<td>No</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>Attribute name.</td>
<td></td>
</tr>
<tr>
<td>param_type</td>
<td>String</td>
<td>Parameter classification. ● required ● optional Defaults to required.</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>String</td>
<td>Parameter data type. Selected from: int, bool, float, string, list_int, list_float, and more</td>
<td></td>
</tr>
<tr>
<td>default_value</td>
<td>-</td>
<td>Default value.</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE**

You can configure more than one operator in a single JSON file. A JSON file is a list of elements, with each element representing an operator.

- For operators with the same op field, the operator project will be created based on the operator that comes last on the list.
- If input_desc of an operator has identical name fields, the name field that comes last on the list prevails. This rule also applies to output_desc.
- For input_desc and output_desc, the supported data types set in the type list must be in one-to-one mapping with the supported formats set in the format list. In the event of a shorter format list, ND is used as a substitution.

To create a MindSpore operator project, configure the JSON file by referring to **Table 8-8**.
### Table 8-8 Parameters in the JSON file of the MindSpore operator project

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
<th>Mandatory or Not</th>
</tr>
</thead>
<tbody>
<tr>
<td>op</td>
<td>String</td>
<td>Operator type.</td>
<td>Yes</td>
</tr>
<tr>
<td>input_desc</td>
<td>List</td>
<td>Input description.</td>
<td>No</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>Input name.</td>
<td></td>
</tr>
<tr>
<td>param_type</td>
<td>String</td>
<td>Parameter type.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>● required</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>● optional</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>● dynamic</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Defaults to <strong>required</strong>.</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>List</td>
<td>Parameter data type.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value range: <strong>I8_Default</strong>, <strong>I16_Default</strong>,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>I32_Default</strong>, <strong>I64_Default</strong>, <strong>U8_Default</strong>,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>U16_Default</strong>, <strong>U32_Default</strong>, <strong>U64_Default</strong>,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>BOOL_Default</strong>, and more</td>
<td></td>
</tr>
<tr>
<td>output_desc</td>
<td>List</td>
<td>Output description.</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>Output name.</td>
<td></td>
</tr>
<tr>
<td>param_type</td>
<td>String</td>
<td>Parameter type.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>● required</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>● optional</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>● dynamic</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Defaults to <strong>required</strong>.</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Description</td>
<td>Mandatory or Not</td>
</tr>
<tr>
<td>--------------</td>
<td>------</td>
<td>--------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>type</td>
<td>List</td>
<td>Parameter data type. Value range:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>I8_Default, I16_Default, I32_Default, I64_Default, U8_Default, U16_Default, U32_Default, U64_Default, BOOL_Default, and more</td>
<td></td>
</tr>
<tr>
<td>attr</td>
<td>List</td>
<td>Attribute description.</td>
<td>No</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>Attribute name.</td>
<td></td>
</tr>
<tr>
<td>param_type</td>
<td>String</td>
<td>Parameter type.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>● required</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>● optional</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Defaults to <strong>required</strong>.</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>String</td>
<td>Parameter data type. Value range:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>I8_Default, I16_Default, I32_Default, I64_Default, U8_Default, U16_Default, U32_Default, U64_Default, BOOL_Default, and more</td>
<td></td>
</tr>
<tr>
<td>default_value</td>
<td></td>
<td>Default value.</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE**

You can configure more than one operator in a single JSON file. A JSON file is a list of elements, with each element representing an operator.

- For operators with the same `op` field, the operator project will be created based on the operator that comes last on the list.
- If `input_desc` of an operator has identical `name` fields, the `name` field that comes last on the list prevails. This rule also applies to `output_desc`.
- For the possible values of the `type` field, see the **DataType** enumeration on the [MindSpore website](https://www.mindspore.cn/).
8.4.2 Setting the Python Library

Before operator development, you need to configure the Python library on which the operator project depends by referring to this section.

- (Required) Setting the Globally Dependent Python SDK: You only need to perform the configuration once, which takes effect for all operator projects, including projects created later.

- (Required) Setting the Dependent Python SDK of the Current Project: It must be performed each time when an operator project is created. The configuration takes effect only for the current operator project.

- (Optional) Adding a Custom Python SDK: It needs to be configured when you add a customized Python SDK.

Setting the Globally Dependent Python SDK

**Step 1** In the project dialog box, choose File > Project Structure. The Project Structure dialog box is displayed.

**Step 2** On the left menu bar, choose Platform Settings > SDKs. Add a Python library by referring to Figure 8-6.

**Figure 8-6 Adding the global Python library**

![Figure 8-6 Adding the global Python library](image)

**NOTICE**

Click to expand the drop-down list, if Add Python SDK... is not displayed, install the plug-in of Python Community Edition by referring to Step 11. If you do not have the network permission, configure the proxy agent.
Select Python 3.7.5 from the **Interpreter** drop-down list box by clicking 📊.

**System Interpreter** indicates the local Python interpreter. If the following modes are used, you need to install the dependencies (such as conda and pipenv) and configure the Python image in advance.

1. **Virtualenv Environment**: In the virtual management environment, independent Python libraries and interpreter are created in the directory of the Python virtual environment. After the Python libraries and interpreter are created, use the PIP tool to download the libraries based on the PIP sources configured in the ~/.pip/pip.conf file of the current user in the created directory.

2. **Conda Environment**: The Anaconda Python interpreter is selected and conda is used to create a virtual environment.

3. **Pipenv Environment**: Virtualenv and pip are integrated. The virtual environment can be created and libraries can be managed.

**Step 3** Ensure that the Python libraries related to operator development exist, including te and topi.

The following figure shows an example.

**Figure 8-7** Viewing the Python libraries

![Figure 8-7 Viewing the Python libraries](image)

**Step 4** In the **Project Structure** window, click **Apply**.

**Step 5** Click **OK** to close the **Project Structure** window.
Setting the Dependent Python SDK of the Current Project

During operator code development, if a third-party Python library or custom Python library is linked, perform the following steps to add the corresponding Python SDK to the project:

**Step 1** In the project dialog box, choose File > Project Structure. The Project Structure dialog box is displayed.

**Step 2** On the left menu bar, choose Project Settings > Project.

In Project SDK, select the Python SDK configured in Setting the Globally Dependent Python SDK from the drop-down list box and click Apply, as shown in Figure 8-8.

![Figure 8-8 Selecting the Project SDK](image)

**Step 3** Choose Project Settings > Modules from the menu bar on the left.

On the Dependencies tab page, select the Python SDK configured in 8.4.2 Setting the Python Library from the drop-down list box and click Apply, as shown in Figure 8-9.

![Figure 8-9 Selecting the Modules SDK](image)
Step 4 Click OK to close the Project Structure window.

A message is displayed on the menu bar at the bottom of the project, as shown in Figure 8-10.

Figure 8-10 Update page

Step 5 After the update is successful, the Python code can be automatically interpreted, as shown in Figure 8-11. Click the imported Python module code and press Ctrl +Q. The corresponding tips are displayed. You can press Ctrl and click to open the corresponding module definition file.

Figure 8-11 Example of auto interpretation of Python code

NOTE

If the problem persists, close the project and open it again.

----End

(Optional) Adding a Custom Python SDK

During operator code development, if a third-party Python library or custom Python library is linked, perform the following steps to add the corresponding Python SDK to the project:

Step 1 In the project dialog box, choose File > Project Structure. The Project Structure dialog box is displayed.

Step 2 On the left menu bar, choose Platform Settings > SDKs.

Step 3 Select the corresponding Python libraries and click + in the Classpath column to add a customized Python library, as shown in Figure 8-12.

Notes:
8.5 TBE Operator Development (TensorFlow)

8.5.1 Operator Code Implementation

Call TBE DSL APIs to implement the Add operator in the `tbe/impl/add.py` file, including operator function definition, operator argument verification, and compute process implementation, scheduling, and building.

Code Template Introduction

```python
# Import the dependent Python modules.
import te.lang.cce as tbe
from te import tvm
from te.platform.fusion_manager import fusion_manager

# Operator compute function
@fusion_manager.register("add")
def add_compute(x, y, z, kernel_name="add"):  
    To do: Implement the operator by referring to the TBE Operator Development Guide.

    res = tbe.XXX(x, y)
    return res
```

---

*Figure 8-12* Adding a user-defined Python library

**Step 4** Click OK to close the Project Structure window.

---

- In the new Python library, ensure that the default Python libraries related to operator development exist, including `te` and `topi`.
- The Python libraries on the Platform Settings tab page are globally configured and takes effect for all operator projects.
# Operator definition function

def add(x, y, z, kernel_name="add"):
    
    To do: Implement the operator by referring to the
    TBE Operator Development Guide.
    
    # Input placeholder
    data_x = tvm.placeholder(x.get("shape"), dtype=x.get("dtype"), name="data_x")
    data_y = tvm.placeholder(y.get("shape"), dtype=y.get("dtype"), name="data_y")
    
    # Call the operator compute function.
    res = add_compute(data_x, data_y, z, kernel_name)
    
    # Auto schedule
    with tvm.target.cce():
        schedule = tbe.auto_schedule(res)
    
    # Build
    config = {"name": kernel_name,
              "tensor_list": [data_x, data_y, res]}
    tbe.build(schedule, config)

- The Python modules for operator development are as follows:
  - `te.lang.cce`: imports the SDL APIs supported by TBE, including common ones such as `vmuls`, `vadds`, and `matmul`.
    For details about the API definition, see the Python functions in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/
    {arch}-linux/atc/python/site-packages/te/te/lang/cce` directory.
  - `te.tvm`: imports the code generation mechanism of TVM.
    For details about the API definition, see the Python functions in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/
    {arch}-linux/atc/python/site-packages/te/te/tvm` directory. For details about the usage, visit [https://docs.tvm.ai/](https://docs.tvm.ai/).
  - `te.platform.fusion_manager.fusion_manager`: implements automatic UB fusion for operators.
    For details about the API definition, see the definition of the `fusion_manager` function in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/
    {arch}-linux/atc/python/site-packages/te/te/platform/fusion_manager.py` file.

- The template generates a compute function declaration named `operatorName_compute`.
  - If `From Template` or `From TensorFlow` is selected during operator project creation, the input and output parameters and attributes are automatically generated based on the prototype definition.
  - If `Empty` is selected during operator project creation, an input and an output without attributes are generated by default.

- The template generates with the declaration and part of the implementation of the definition function named `operatorName`. The sample code in the implementation function template contains the following functions:
  - Obtain the shape and data type of the input tensor, verify the input parameters, place the input tensor, and call the compute function of the operator for computing, scheduling, and building.
Implementation of the Operator Definition Function

You need to implement the operator compute function based on the template code generated by MindStudio. In addition, you need to add the verification code of the operator inputs, outputs, or attributes to the operator definition function. The two inputs may have different shapes. This scenario is supported by the Add operator, but not supported by the operator compute API `te.lang.cce.vadd()`. Therefore, the two input shapes need to be broadcast and verified, so that faults can be located during operator build. The modified code is as follows.

![Notice]

If you remotely start MindStudio on Windows, the following code may fail to be copied. For details about how to solve the problem, see 15.8.1 What Do I Do If the Copied Content Cannot Be Pasted to the Editor Window Occasionally When MindStudio Is Opened Remotely on Windows?

```python
from __future__ import absolute_import
from functools import reduce
from te import tvm
from te.platform.fusion_manager import fusion_manager
import te.lang.cce as tbe
from te.utils import para_check
from te.utils import shape_util

SHAPE_SIZE_LIMIT = 2147483648

# Implement the compute logic of the Add operator.
@fusion_manager.register("add")
def add_compute(input_x, input_y, output_z, kernel_name="add"):
    # Convert the shape to a list.
    shape_x = shape_util.shape_to_list(input_x.shape)
    shape_y = shape_util.shape_to_list(input_y.shape)

    # Assign the larger value of each dimension of shape_x and shape_y to shape_max.
    shape_x, shape_y, shape_max = shape_util.broadcast_shapes(shape_x, shape_y, param_name_input1="input_x",
                                                              param_name_input2="input_y")

    shape_size = reduce(lambda x, y: x * y, shape_max[:])
    if shape_size > SHAPE_SIZE_LIMIT:
        raise RuntimeError("the shape is too large to calculate")

    # Broadcast the shape of input_x as shape_max.
    input_x = tbe.broadcast(input_x, shape_max)
    input_y = tbe.broadcast(input_y, shape_max)

    # Add input_x and input_y.
    res = tbe.vadd(input_x, input_y)

    return res

# Operator definition function
@para_check.check_op_params(para_check.REQUIRED_INPUT, para_check.REQUIRED_INPUT,
                            para_check.REQUIRED_OUTPUT, para_check.KERNEL_NAME)
def add(input_x, input_y, output_z, kernel_name="add"):
    # Obtain the shape and data type of the operator input tensor.
    shape_x = input_x.get("shape")
    shape_y = input_y.get("shape")

    # Verify the operator input type.
    check_tuple = ("float16", "float32", "int32")
    input_data_type = input_x.get("dtype").lower() if input_data_type else "int32"
    para_check.check_dtype(input_data_type, check_tuple, param_name="input_x")
```
# Assign the larger value of each dimension of shape_x and shape_y to shape_max.
shape_x, shape_y, shape_max = shape_util.broadcast_shapes(shape_x, shape_y,
    param_name_input1="input_x",
    param_name_input2="input_y")

# If the shape length is 1, assign a value directly. If the shape length is not 1, shape needs to be tiled
# and the last dimension needs to be removed. For the shape with the last dimension of 1 and the shape
# without the last dimension, if their formats are the same, for example, 2 x 3 = 2 x 3 x 1, the last dimension
# can be removed to improve the scheduling efficiency.
if shape_x[-1] == 1 and shape_y[-1] == 1 and shape_max[-1] == 1:
    shape_x = shape_x if len(shape_x) == 1 else shape_x[:-1]
    shape_y = shape_y if len(shape_y) == 1 else shape_y[:-1]
    shape_max = shape_max if len(shape_max) == 1 else shape_max[:-1]

# Call the placeholder API of TVM to create a placeholder for each input tensor, returning a tensor
# object, respectively.
data_x = tvm.placeholder(shape_x, name="data_1", dtype=input_data_type)
data_y = tvm.placeholder(shape_y, name="data_2", dtype=input_data_type)

# Call the compute implementation function.
res = add_compute(data_x, data_y, output_z, kernel_name)

# Auto schedule
with tvm.target.cce():
    schedule = tbe.auto_schedule(res)
# Build configuration
config = {"name": kernel_name,
          "tensor_list": (data_x, data_y, res)}
tbe.build(schedule, config)

1. The operator definition function declaration contains the operator input
   information, output information, and kernel name.
def add(input_x, input_y, output_z, kernel_name="add"):

   - **input_x**, **input_y**: two input tensors of the Add operator. Each tensor
     must be defined in dictionary format, including information such as
     shape and data type. The number of input tensors must be consistent
     with that defined in the operator information definition file (tbe/np_info_cfg/ai_core/add.ini).

   - **output_z**: output tensor. The tensor information must be defined in
     dictionary format, including the shape, data type, and more. This field is
     reserved.

     The number of output tensors must be consistent with that defined in the
     operator information definition file (tbe/op_info_cfg/ai_core/add.ini).

   - **kernel_name**: unique name of the operator in the kernel, that is, the
     name of the generated binary file and operator description file. The value
     can contain a maximum of 200 characters, which must be a combination
     of letters, digits, and underscores (_), beginning with a letter or
     underscore (_).

2. Verify the operator input and infer the output shape.

   The Add operator needs to verify shapes of the two input tensors. Only data
   types float16, float32, and int32 are supported. Since the two input tensors of
   the Add operator may have different shapes, shape_util.broadcast_shapes() needs
   to be implemented to generate and verify the broadcast shape.

3. Create placeholders of the two input tensors.

   Call the placeholder API of TVM to create a placeholder for each input
   tensor, returning a tensor object respectively.
**NOTICE**

tensor_list described in 5 is a list of tensor objects returned by calls to the tvm.placeholder API. Therefore, these objects cannot be replaced in subsequent computation.

4. Call the add_compute function.
   ```python
   res = add_compute(data_x, data_y, output_z, kernel_name)
   ```
data_x and data_y are the tensor objects generated in 3.
For details about how to implement the compute function, see Implementation of the Compute Function.

5. Implement operator scheduling and building.
tensor_list:
   ```python
   "tensor_list": (data_x, data_y, res)
   ```
The arguments are the two input tensors and one output tensor.

**Implementation of the Compute Function**

You need to customize the compute function of an operator based on the compute logic. The compute function of the Add operator is implemented as follows.

```python
@fusion_manager.register("add")
def add_compute(input_x, input_y, output_z, kernel_name="add"):
    # Convert the shape to a list.
    shape_x = shape_util.shape_to_list(input_x.shape)
    shape_y = shape_util.shape_to_list(input_y.shape)

    # Assign the larger value of each dimension of shape_x and shape_y to shape_max.
    shape_x, shape_y, shape_max = shape_util.broadcast_shapes(shape_x, shape_y,
                                                              param_name_input1="input_x",
                                                              param_name_input2="input_y")

    shape_size = reduce(lambda x, y: x * y, shape_max[:])
    if shape_size > SHAPE_SIZE_LIMIT:
        raise RuntimeError("the shape is too large to calculate")

    # Broadcast the shape of input_x as shape_max.
    input_x = tbe.broadcast(input_x, shape_max)
    input_y = tbe.broadcast(input_y, shape_max)

    # Add input_x and input_y.
    res = tbe.vadd(input_x, input_y)

    return res
```

1. The add_compute function is declared as follows.
   ```python
   @fusion_manager.register("add")
   def add_compute(input_x, input_y, output_z, kernel_name="add")
   ```
The decorator @fusion_manager.register ("add") is required in the DSL operator development mode. It is used to support automatic UB fusion for operators during network running, so that the compute function of the current custom operator can be automatically fused with other operators in the UB according to UB fusion patterns.

Note the following:
- input_x, input_y: indicate the arguments passed to the compute function, that is, the placeholders for the input tensors declared in 3, including information such as the shape and data type.
MindStudio
User Guide

8 Custom Operator Development

2.

–

output_z: indicates the dictionary returned by the call to the operator API
function in 1.

–

kernel_name: indicates the operator name in the kernel.

Implement the compute logic of the Add operator.
The Add operator requires that shapes of the two tensors to be added be the
same. Therefore, the tbe.broadcast API is called to broadcast the two input
tensors to the same shape, and then the tbe.vadd API is called to add the
input tensors and return the result tensor.

Verifying Operator Building
Step 1 At the bottom of the Python file of the operator, add the main function to call the
operator, and build the operator implementation file by using MindStudio for
simple syntax verification of the single-operator code. A code example is as
follows.
# Call the operator.
if __name__ == '__main__':
input_output_dict = {"shape": (5, 6, 7),"format": "ND","ori_shape": (5, 6, 7),"ori_format": "ND", "dtype":
"float16"}
add(input_output_dict, input_output_dict, input_output_dict, kernel_name="add")

Step 2 Right-click "tbe/impl/add.py and choose Run 'add' from the shortcut menu to
build the operator.
If no build error is reported and a kernel_meta folder containing the following
files is generated in the tbe/impl directory, the operator code can be built and run
properly.
●

Binary file of the operator (.o)

●

Operator description file (.json): defines operator attributes and resources
required for running the operator.

----End

8.5.2 Operator Prototype Definition
Go to the op_proto/ directory, write the IR implementation files add.h and add.cc,
and register the operator with the operator prototype library. During network
execution, GE calls the verification API of the operator prototype library to verify
operator arguments. If the verification passes, GE infers the output shape and
dtype of each node by calling the inference function of the operator prototype
library and allocates static memory for the result tensor.

Implementing add.h
MindStudio generates the operator registration code to the add.h file. You can
modify the code as required. The prototype definition of the Add operator is as
follows.
#ifndef GE_OPS_OP_PROTO_ADD_H_
#define GE_OPS_OP_PROTO_ADD_H_
#include "graph/operator_reg.h"
namespace ge {
REG_OP(Add)
.INPUT(x1,
TensorType({DT_FLOAT, DT_INT32, DT_INT64, DT_FLOAT16, DT_INT16, DT_INT8, DT_UINT8,
DT_DOUBLE, DT_COMPLEX128,

Issue 01 (2021-06-24)

Copyright © Huawei Technologies Co., Ltd.

251


● Add in REG_OP(Add) indicates the operator type that is registered with the Ascend AI Processor. The operator type must be consistent with that in REGISTER_CUSTOM_OP(“Add”) in 8.5.4 Operator Plugin Implementation.

● .INPUT and .OUTPUT indicate the names and data types of the input and output tensors of the operator. The input and output sequence must be consistent with the function parameter sequence in 8.5.1 Operator Code Implementation as well as that in Operator Information Definition.

add.cc Implementation

You need to implement the InferShape and Verify functions in add.cc.

- The Verify function, that is, IMPLEMT_VERIFIER(Add, AddVerify) in the following sample code, is used to check whether the data types of the two inputs of the Add operator are the same.

- The InferShape function that is, IMPLEMT_COMMON_INFERFUNC(AddInferShape) in the following sample code, is used to infer the output tensor description of the operator. In this way, the memory can be statically allocated for all tensors during network execution, avoiding the overhead caused by dynamic memory allocation.

The implementation code of the add.cc file is as follows.

```c++
namespace ge {

bool InferShapeAndTypeAdd(Operator& op, 
const string& inputName1, 
const string& inputName2, const string& outputName) 
{
    TensorDesc vOutputDesc = op.GetOutputDescByName(outputName.c_str());

    DataType inputDtype = op.GetInputDescByName(inputName1.c_str()).GetDataType();
    Format inputFormat = op.GetInputDescByName(inputName1.c_str()).GetFormat();
    // Exchange the shape dimensions.
    ge::Shape shapeX = op.GetInputDescByName(inputName1.c_str()).GetShape();
    ge::Shape shapeY = op.GetInputDescByName(inputName2.c_str()).GetShape();
    std::vector<int64_t> dimsX = shapeX.GetDims();
    std::vector<int64_t> dimsY = shapeY.GetDims();
    if (dimsX.size() < dimsY.size()) {
        std::vector<int64_t> dimsTmp = dimsX;
        dimsX = dimsY;
        dimsY = dimsTmp;
    }

    // Pad the smaller shape dimension with 1.
    if (dimsX.size() != dimsY.size()) {
        int dec = dimsX.size() - dimsY.size();
        if (dimsX.size() > dimsY.size()) {  // If the shape of the first input is larger than the second input.
            for (int i = 0; i < dec; i++) {
                std::vector<int64_t> dimsXNew = dimsX;
                dimsXNew.push_back(1);
                dimsX = dimsXNew;
            }
        }
        else {  // If the shape of the second input is larger than the first input.
            for (int i = 0; i < dec; i++) {
                std::vector<int64_t> dimsYNew = dimsY;
                dimsYNew.push_back(1);
                dimsY = dimsYNew;
            }
        }
    }

    // Assign the shapes of the outputs.
    std::vector<int64_t> dimsOutput;
    if (dimsX.size() == dimsY.size()) {
        for (int i = 0; i < dimsX.size(); i++) {
            if (dimsX[i] == 1) {
                dimsOutput.push_back(1);
            }
            else {
                dimsOutput.push_back(dimsX[i] + 1);
            }
        }
    }
    else {
        for (int i = 0; i < dimsX.size(); i++) {
            if (dimsX[i] == 1) {
                dimsOutput.push_back(1);
            }
            else {
                dimsOutput.push_back(dimsX[i] + 1);
            }
        }
        for (int i = 0; i < dimsY.size(); i++) {
            if (dimsY[i] == 1) {
                dimsOutput.push_back(1);
            }
            else {
                dimsOutput.push_back(dimsY[i] + 1);
            }
        }
    }

    // Assign the data types of the outputs.
    vOutputDesc.SetDataFormat(inputFormat);
    vOutputDesc.SetDataType(inputDtype);
    if (vOutputDesc.GetShape() != dimsOutput) {
        return false;
    }
    return true;
}
```

```c++
} // namespace ge
```
for (int i = 0; i < dec; i++) {
    dimsY.insert(dimsY.begin(), (int64_t)1);
}

// Set the output shape dimension.
std::vector<int64_t> dimVec;
for (size_t i = 0; i < dimsX.size(); i++) {
    if ((dimsX[i] != dimsY[i]) && (dimsX[i] != 1) && (dimsY[i] != 1)) {
        return false;
    }
    int64_t dims = dimsX[i] > dimsY[i] ? dimsX[i] : dimsY[i];
    dimVec.push_back(dims);
}
ge::Shape outputShape = ge::Shape(dimVec);
vOutputDesc.SetShape(outputShape);
vOutputDesc.SetDataType(inputDtype);
vOutputDesc.SetFormat(inputFormat);
op.UpdateOutputDesc(outputName.c_str(), vOutputDesc);
return true;
}

//----------------Add-------------------
IMPLEMT_VERIFIER(Add, AddVerify)
{
    if (op.GetInputDescByName("x1").GetDataType() != op.GetInputDescByName("x2").GetDataType()) {
        return GRAPH_FAILED;
    }
    return GRAPH_SUCCESS;
}

// Obtain the processing function of the output tensor description.
IMPLEMT_COMMON_INFERFUNC(AddInferShape)
{
    if(InferShapeAndTypeAdd(op, "x1", "x2", "y")) {
        return GRAPH_SUCCESS;
    }
    return GRAPH_FAILED;
}

// Registered inference function. Pass the OpType as the first argument.
COMMON_INFER_FUNC_REG(Add, AddInferShape);

// Registered verification function. Pass the OpType as the first argument.
VERIFY_FUNC_REG(Add, AddVerify);
//----------------Add-------------------

8.5.3 Operator Information Definition

You need to register the operator information to the operator information library by configuring the operator information file. The operator information library mainly reflects the physical implementation restrictions of an operator on the Ascend AI Processor, including the input and output data types, formats, and input shape of the operator. During network execution, FE performs basic verification based on the operator information in the operator information library, and determines whether to insert a proper conversion node for the operator. It also finds the corresponding operator implementation file based on the information in the operator information library and builds the operator binary file for execution.
Go to the \texttt{tbe/op_info_cfg/ai_core/soc\_version} directory and configure the operator information library file \texttt{add.ini}.

\textit{soc\_version} indicates the current Ascend AI Processor version, for example, \texttt{ascend310} and \texttt{ascend910}.

You need to modify the \texttt{add.ini} file automatically generated by MindStudio. The following shows the operator information definition of the modified Add operator.

```plaintext
[Add]
input0.name=x1
input0.dtype=float16,float16,float16,float16,float,float,float,float,int32,int32,int32,int32
input0.format=NCHW,NC1HWC0,NHWC,ND,NCHW,NC1HWC0,NHWC,ND,NCHW,NC1HWC0,NHWC,ND
input0.shape=all
input0.paramType=required
input1.name=x2
input1.dtype=float16,float16,float16,float16,float,float,float,float,int32,int32,int32,int32
input1.format=NCHW,NC1HWC0,NHWC,ND,NCHW,NC1HWC0,NHWC,ND,NCHW,NC1HWC0,NHWC,ND
input1.shape=all
input1.paramType=required
output0.name=y
output0.dtype=float16,float16,float16,float16,float,float,float,float,int32,int32,int32,int32
output0.format=NCHW,NC1HWC0,NHWC,ND,NCHW,NC1HWC0,NHWC,ND,NCHW,NC1HWC0,NHWC,ND
output0.shape=all
output0.paramType=required
opFile.value=add
opInterface.value=add
```

Table 8-9 describes the configuration options. The table lists only common configuration options of the operator information library. For details about more configuration options, see: "Operator Information Library Definition" in 	extit{TBE Custom Operator Development Guide}.

<table>
<thead>
<tr>
<th>Configuration Option</th>
<th>Add Operator Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[OpType]</td>
<td>[Add]</td>
<td>Operator type, which is included in square brackets to mark the start of operator information. The operator type is \texttt{Add} according to \textit{8.3 Operator Analysis}.</td>
</tr>
<tr>
<td>input0.name</td>
<td>x1</td>
<td>Name of the first input tensor of the Add operator. According to \textit{8.3 Operator Analysis}, the name of the first input tensor is \texttt{x1}. Therefore, set this parameter to \texttt{x1}, which must be the same as the name in \textit{Operator Prototype Definition}.</td>
</tr>
<tr>
<td>input0.dtype</td>
<td>\texttt{float16, float16, float16, float16, float, float, float, float, int32, int32, int32, int32}</td>
<td>Data type and data layout format supported by the input tensor. According to \textit{8.3 Operator Analysis}, the Add operator</td>
</tr>
<tr>
<td>Input Parameter</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>input0.format</td>
<td>NC HW, NC1HWC0, NCHW, ND, NCHW, NC1HWC0, NHWC, ND, NCHW, NC1HWC0, NHWC, ND</td>
<td></td>
</tr>
<tr>
<td>supports input types (float16, float32, and int32) and data layout formats (NCHW, NC1HWC0, NHWC, and ND). Note: If the input tensor supports multiple dtype and format combinations, dtype and format must be configured in pairs and in sequence. List all the combinations supported by the operator and separate them with commas (,).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>input0.shape</td>
<td>all</td>
<td></td>
</tr>
<tr>
<td>Shape of the input tensor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>input0.paramType</td>
<td>required</td>
<td></td>
</tr>
<tr>
<td>Type of the input tensor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• <strong>dynamic</strong>: indicates that the number of inputs is subject to change, which can be one or more.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• <strong>optional</strong>: indicates that there could be one input or no input at all.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• <strong>required</strong>: indicates that there is one and only one input. Set input0 of the Add operator to <strong>required</strong>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>input1.name</td>
<td>x2</td>
<td></td>
</tr>
<tr>
<td>The Add operator has two inputs, while the generated configuration template has only one input. Therefore, you need to add the configuration of input1. Name of the second input tensor of the Add operator. According to <strong>8.3 Operator Analysis</strong>, the name of the second input tensor is <strong>x2</strong>. Therefore, set this parameter to <strong>x2</strong>, which must be the same as the name in <strong>8.5.2 Operator Prototype Definition</strong>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>input1.dtype</td>
<td>input1.dtype=float16, float16, float16, float16, float16, float16, float16, float, float, float, int32, int32, int32, int32</td>
<td></td>
</tr>
<tr>
<td>Data type and data layout format supported by the input tensor. According to <strong>8.3 Operator Analysis</strong>, the Add operator supports input types (float16, float32, and int32) and data layout formats (NCHW, NC1HWC0, NHWC, and ND).</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### input1.format
WC,ND,NCHW,NC1HWC0,NHWC,ND

**Note:** If the input tensor supports multiple dtype and format combinations, dtype and format must be configured in pairs and in sequence. List all the combinations supported by the operator and separate them with commas (,).

### input1.shape
all
Shape of the input tensor.

### input1.paramType
required
Type of the input tensor.
- **dynamic**: indicates that the number of inputs is subject to change, which can be one or more.
- **optional**: indicates that there could be one input or no input at all.
- **required**: indicates that there is one and only one input.

Set input1 of the Add operator to **required**.

### output0.name
y
Name of the output tensor of the Add operator. According to **8.3 Operator Analysis**, the output name of the operator is y. Therefore, set this parameter to y, which must be the same as the name in **8.5.2 Operator Prototype Definition**.

### output0.dtype
output0.dtype=float16,float16,float16,float16,float16,float16,float16,int32,int32
Data type and data layout format supported by the output tensor. According to **8.3 Operator Analysis**, the Add operator supports output types (float16, float32, and int32) and data layout formats (NCHW, NC1HWC0, NHWC, ND, NHCW, NC1HWC0, NHWC, ND).

### output0.format
output0.format=NCHW,NC1HWC0,NHWC,ND,NCHW,NC1HWC0,NHWC,ND

**Note:** If the output tensor supports multiple dtype and format combinations, dtype and format must be configured in pairs and in sequence. List all the combinations supported by the operator and separate them with commas (,).
<table>
<thead>
<tr>
<th>parameter</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>output0.shape</td>
<td>all</td>
<td>Shape of the output tensor.</td>
</tr>
<tr>
<td>output0.paramType</td>
<td>required</td>
<td>Type of the output tensor:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>dynamic</strong>: indicates that the number of outputs is subject to change, which can be one or more.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>optional</strong>: indicates that there could be one output or no output at all.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>required</strong>: indicates that there is one and only one output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set <code>output0</code> of the Add operator to <strong>required</strong>.</td>
</tr>
<tr>
<td>opFile.value</td>
<td>add</td>
<td>Name of the operator code implementation file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>According to Step 4, the name of the operator implementation code complies with the rules of converting <code>OpType</code> to the code implementation name. Therefore, this parameter does not need to be specified.</td>
</tr>
<tr>
<td>opInterface.value</td>
<td>add</td>
<td>Name of the definition function in the operator implementation file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The name of the operator definition function complies with the rules of converting <code>OpType</code> to the operator definition function name as shown in Step 4. Therefore, this parameter does not need to be specified.</td>
</tr>
</tbody>
</table>

### 8.5.4 Operator Plugin Implementation

You need to develop the operator adaptation plugin to interpret and map the operator in the TensorFlow network to one adapted to the the Ascend AI Processor. MindStudio automatically generates the plugin code of the Add operator to the `framework/tf_plugin/tensorflow_add_plugin.cc` file, as shown in Figure 8-13.
Figure 8-13 Plugin code implementation template

```cpp
#include "register/register.h"

namespace domi {
    REGISTER_CUSTOM_OP("Add")
        .FrameworkType(TENSORFLOW)
        .OriginOpType({ge::AscendString("Add")})
        .ParseParamsByOperatorFn(AutoMappingFn)
        .ImplyType(ImplyType::TVM);
}
```

- Include the header file.
  ```cpp```
  // Include the atc/include/register/register.h file in the ADK to use the operator registration class and call the operator registration APIs.
  #include "register/register.h"
```

- Register the plugin.
  ```cpp```
  namespace domi {
      REGISTER_CUSTOM_OP("Add")
          .FrameworkType(TENSORFLOW)
          .OriginOpType({ge::AscendString("Add")})
          .ParseParamsByOperatorFn(AutoMappingFn)
          .ImplyType(ImplyType::TVM);  // Add it manually.
  }
```

- **REGISTER_CUSTOM_OP**: operator type registered with GE. According to 8.3 Operator Analysis, the operator type is **Add**.
- **FrameworkType**: framework type. The source framework type is **TensorFlow**.
- **OriginOpType**: operator type in the TensorFlow framework.
- **ParseParamsByOperatorFn**: function for registering models to be parsed. The **AutoMappingFn** function is used to automatically parse models.
- **ImplyType**: operator implementation type. **ImplyType::TVM** indicates that the operator is a TBE operator. Add it manually.

### 8.5.5 UT

#### 8.5.5.1 Overview

MindStudio provides an upgraded UT solution based on the GTest framework, simplifying UT case development.

UT is one of the methods to verify a single-operator, which aims to:

- Test correctness of operator code and verify consistency between the desired and actual inputs and outputs.
- Ensure the operator code coverage. The selected scenario combination must cover all branches of the operator code (the coverage should reach 100%), to reduce the failure rate for building operator code in different scenarios.
8.5.5.2 APIs

8.5.5.2.1 OpUT

Overview

Functions as a base class for the UT framework. It provides the following two APIs to define and run the test cases.

- add_case
- add_precision_case

Definition of the OpUT Test Class

- Prototype
  \[
  \text{OpUT}(\text{op\_type}, \text{op\_module\_name}=\text{None}, \text{op\_func\_name}=\text{None})
  \]

- Parameters
  - \text{op\_type}: operator type.
  - \text{op\_module\_name}: operator module name, that is, the name and path of the operator implementation file, for example, \text{impl.add} (the file path is \text{impl/\text{add.py}}). Defaults to \text{None}. The argument is generated automatically based on \text{op\_type}. For example, for a BiasAdd operator, the generated module name is \text{impl.bias\_add}. For details about how the argument of \text{op\_module\_name} is generated based on \text{op\_type}, see Step 4.
  - \text{op\_func\_name}: operator function name in the operator implementation file. Defaults to \text{None}. The argument is generated automatically based on \text{op\_type}. For details about how the argument of \text{op\_func\_name} is generated based on \text{op\_type}, see Step 4. For example, for an Add operator, the corresponding API must exist in \text{impl/\text{add.py}}. The API definition is as follows.

```python
@check_op_params(REQUIRED_INPUT, REQUIRED_INPUT, REQUIRED_OUTPUT, KERNEL_NAME)
def add(input_x, input_y, output_z, kernel_name="add"):
```

add_case

- Prototype
  \[
  \text{OpUT.add\_case}(\text{support\_soc}=\text{None}, \text{case}=\text{None})
  \]

- Description
  Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an .o file.

- Parameters
  - \text{support\_soc}: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the atc/data/platform_config directory in the
ATC installation path. **support_soc** can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to **all** or **None**, all SoCs are supported.

- **case**: a dictionary. An example is as follows.

```python
{
    "params": [
        {
            "shape": (32, 64),
            "ori_shape": (32, 64),
            "format": "ND",
            "ori_format": (32, 64),
            "dtype": "float16"
        },
        {
            "shape": (32, 64),
            "ori_shape": (32, 64),
            "format": "ND",
            "ori_format": (32, 64),
            "dtype": "float16"
        }
    ],
    "case_name": "test_add_case_1",
    "expect": "success"
}
```

The key fields in the dictionary include:

- **params**: This field is passed through to the operator API during test case running.

- **case_name**: (optional) name of a test case. If it is not specified, the test framework generates the argument of **case_name** automatically in the following format:

```
test_[op_type]_auto_case_name_[case_count]
```

Example: **test_Add_auto_case_name_1**

- **expect**: expected result. Defaults to **success**. It can also be set to **RuntimeError**, indicating an expected exception.

### add_precision_case

- **Prototype**

```
OpUT.add_precision_case(support_soc=None, case)
```

- **Description**

Adds test cases for operator build and precision test.

- **Parameters**

  - **support_soc**: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the **atc/data/platform_config** directory in the ATC installation path. **support_soc** can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to **all** or **None**, all SoCs are supported.

  - **case**: a dictionary. An example is as follows.

```python
{
    "params": [
        {
            "shape": (32, 64),
            "ori_shape": (32, 64),
```
The key fields in the dictionary include:

- **params**: This field is passed through to the operator API during test case running.

- **case_name**: (optional) name of a test case. If it is not specified, the test framework generates the argument of case_name automatically in the following format:
  
  test_[op_type]_auto_case_name_[case_count]

  Example: `test_Add_auto_case_name_1`

- **calc_expect_func**: expected result generation function.

- **precision_standard**: user-defined precision standard. If this field is not set, the following default precision is used for comparison with the expected data:
  
  - **float16**: dual-0.1% error limit, that is, the error ratio is within 0.1% and the relative error is within 0.1%.
  
  - **float32**: dual-0.01% error limit, that is, the error ratio is within 0.01% and the relative error is within 0.01%.

### Examples

#### Example 1

```python
from op_test_frame.ut import OpUT

# ut_case is a keyword of the UT framework and cannot be modified.
ut_case = OpUT("Add", "impl.add", "add")

def np_add(x1, x2, y):
    y = (x1.get("value") + x2.get("value"), )
    return y

ut_case.add_precision_case(case={
    "params": [
        {
            "shape": (32, 64),
            "ori_shape": (32, 64),
            "format": "ND",
            "ori_format": "ND",
            "dtype": "float16",
            "param_type": "input"
        },
        {
            "shape": (32, 64),
            "ori_shape": (32, 64),
            "format": "ND",
            "ori_format": "ND",
            "dtype": "float16",
            "param_type": "output"
        }
    ],
    "case_name": "test_add_case_1",
    "calc_expect_func": np_add # A function
    "precision_standard": precision_info.PrecisionStandard(0.001, 0.001)  # Optional field
}
```
if __name__ == '__main__':
    ut_case.run("Ascend910",None,"ca","/home/allan/Ascend/toolkit/tools/simulator")

Note:

**ut_case.run**: For details, see run API.

In the preceding example, the values of the operator inputs are not set. By default, the framework calls `np.random.uniform(value_range, size=shape).astype(dtype)` to automatically generate the input data for each input.

The value_range is defaulted to `[0.1, 1.0]`. The values of shape and dtype are consistent with those set in params. value_range can also be specified as follows.

```python
{
    "shape": (32, 64),
    "ori_shape": (32, 64),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float16",
    "param_type": "input",
    "value_range": [2.0, 3.0]
}
```

You can also specify input values as follows.

```python
{
    "shape": (32, 64),
    "ori_shape": (32, 64),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float16",
    "param_type": "input",
    "value": np.zeros((32, 64), np.float16)
}
```

**Example 2**

The following is an example of the user-defined precision standard.

```python
from op_test_frame.common import precision_info
from op_test_frame.ut import OpUT

ut_case = OpUT("Add")
ut_case.add_precision_case(case={
    "params": [
        {
            "shape": (32, 64),
            "ori_shape": (32, 64),
            "format": "ND",
            "ori_format": "ND",
            "dtype": "float16",
        }
    ]
})
```
"param_type": "input"
},
{
  "shape": [32, 64],
  "ori_shape": [32, 64],
  "format": "ND",
  "ori_format": "ND",
  "dtype": "float16",
  "param_type": "input"
},
{
  "shape": [32, 64],
  "ori_shape": [32, 64],
  "format": "ND",
  "ori_format": "ND",
  "dtype": "float16",
  "param_type": "output"
}
],
"case_name": "test_add_case_1",
"calc_expect_func": np_add,
"precision_standard": precision_info.PrecisionStandard(0.1, 0.1)  # Uses the standard precision of one tenth and compares it with the expected data.
})

run API

- Prototype

   OpUT.run(soc, case_name=None, simulator_mode=None, simulator_lib_path=None)

- Description

   Runs test cases.

- Parameters

  - `soc`: the the Ascend AI Processor to run test cases. The value is the name of the corresponding .ini file in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/atc/data/platform_config` directory. `support_soc` can be a string or a tuple or list of strings (indicating multiple SoCs are supported).

  - `case_name`: the test case to run. Set this parameter to the value of `case_name` in `add_case` or `add_precision_case`.

  - `simulator_lib_path`: path of the simulator libraries.

      The path structure is as follows.

      ```
      simulator_lib_path/
      Ascend910/
      lib/
      libpv_model.so
      ...
      Ascend310/
      lib/
      libpv_model.so
      ...
      ```

- NOTICE

   If MindStudio is used to run UT cases, `OpUT.run` does not need to be called manually.
8.5.5.2.2 BroadcastOpUT

Overview

Inherits the capabilities of OpUT.

BroadcastOpUT defines test cases for Broadcast operators with dual inputs and single output, for example, the Add and Mul operators. It also provides more convenient APIs for these operators. For example, when creating the test case file for operator build, you do not need to set the format in some simple scenarios.

Definition of the BroadcastOpUT Test Class

- **Prototype**
  
  ```python
  BroadcastOpUT(op_type, op_module_name=None, op_func_name=None)
  ```

- **Parameters**
  - `op_type`: operator type.
  - `op_module_name`: operator module name, that is, the name and path of the operator implementation file, for example, `impl.add` (the file path is `impl/add.py`). Defaults to `None`. The argument is generated automatically based on `op_type`. For example, for a BiasAdd operator, the generated module name is `impl.bias_add`. For details about how the argument of `op_module_name` is generated based on `op_type`, see Step 4.
  - `op_func_name`: operator function name in the operator implementation file. Defaults to `None`. The argument is generated automatically based on `op_type`. For details about how the argument of `op_func_name` is generated based on `op_type`, see Step 4. For example, for an Add operator, the corresponding API must exist in `impl/add.py`. The API definition is as follows.

  ```python
  @check_op_params(REQUIRED_INPUT, REQUIRED_INPUT, REQUIRED_OUTPUT, KERNEL_NAME)
  def add(input_x, input_y, output_z, kernel_name="add"):
  ```

  `add_broadcast_case`

- **Prototype**

  ```python
  BroadcastOpUT.add_broadcast_case(self, soc, input_1_info, input_2_info, output_info=None, expect=op_status.SUCCESS, case_name=None)
  ```

- **Description**

  Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an .o file.

- **Parameters**
  - `soc`: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the `atc/data/platform_config` directory in the ATC installation path. `support_soc` can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to `all` or `None`, all SoCs are supported.
  - `input_1_info`: information about the first input of the operator, coming in either of the following format:
- [dtype, shape, format, ori_shape, ori_format]
- [dtype, shape, format]: In this format, the values of ori_shape and ori_format are the same as those of shape and format.
  - input_2_info: information about the second input of the operator, with the same information of input_1_info.
  - output_info: defaults to None. This parameter does not need to be set.
  - expect: expected build result. Defaults to op_status.SUCCESS. It can also be set to RuntimeError, indicating an expected exception.
  - case_name: defaults to None. The test framework automatically generates the argument of case_name.

Example
```
ut_case.add_broadcast_case("all", ["float16", (32, 32), "ND"], ["float16", (32, 32), "ND"])
```

### add_broadcast_case_simple

- Prototype
  ```python
  BroadcastOpUT.add_broadcast_case_simple(self, soc, dtypes, shape1, shape2, expect=op_status.SUCCESS, case_name=None)
  ```
- Description
  Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an .o file. This API is simpler than add_broadcast_case.
- Parameters
  - soc: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the {Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/atc/data/platform_config directory. support_soc can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to all or None, all SoCs are supported.
  - dtypes: types of the data to be tested. If multiple data types are set, multiple test cases are added at a time.
  - shape1: shape of the first input of the operator.
  - shape2: shape of the second input of the operator.
  - expect: expected build result. Defaults to op_status.SUCCESS. It can also be set to RuntimeError, indicating an expected exception.
  - case_name: defaults to None. The test framework automatically generates the argument of case_name.

Compared with the add_broadcast_case API, this API sets the formats of all inputs to ND.

Example
```
ut_case.add_broadcast_case_simple(["Ascend910", "Ascend310"], ["float16", "float32"], (32, 32), (32, 32))
```
The preceding test case implements the same function as calling the `add_case` API.

```python
ut_case.add_case(support_soc=['Ascend910', 'Ascend310'], case={
    "params": [{
        "shape": (32, 32),
        "ori_shape": (32, 32),
        "format": "ND",
        "ori_format": "ND",
        "dtype": "float16"
    },
    {"shape": (32, 32),
        "ori_shape": (32, 32),
        "format": "ND",
        "ori_format": "ND",
        "dtype": "float16"
    },
    {"shape": (32, 32),
        "ori_shape": (32, 32),
        "format": "ND",
        "ori_format": "ND",
        "dtype": "float16"
    }
])

ut_case.add_case(support_soc=['Ascend910', 'Ascend310'], case={
    "params": [{
        "shape": (32, 32),
        "ori_shape": (32, 32),
        "format": "ND",
        "ori_format": "ND",
        "dtype": "float32"
    },
    {"shape": (32, 32),
        "ori_shape": (32, 32),
        "format": "ND",
        "ori_format": "ND",
        "dtype": "float32"
    },
    {"shape": (32, 32),
        "ori_shape": (32, 32),
        "format": "ND",
        "ori_format": "ND",
        "dtype": "float32"
    }
    ]
})
```

### 8.5.5.2.3 ElementwiseOpUT

#### Overview

Inherits the capabilities of **OpUT**.

**ElementwiseOpUT** defines test cases for element-wise operators with single input and single output, for example, the Abs and Square operators.

It also provides more convenient APIs for these operators. For example, when creating the test case file for operator build, you do not need to set the format in some simple scenarios.

#### Definition of the ElementwiseOpUT Test Class

- **Prototype**

  ```python
  ElementwiseOpUT(op_type, op_module_name=None, op_func_name=None)
  ```
- **Parameters**
  - **op_type**: operator type.
  - **op_module_name**: operator module name, that is, the name and path of the operator implementation file, for example, `impl.add` (the file path is `impl/add.py`). Defaults to `None`. The argument is generated automatically based on `op_type`. For example, for a MaximumGrad operator, the generated module name is `impl.maximum_grad`. For details about how the argument of `op_module_name` is generated based on `op_type`, see Step 4.
  - **op_func_name**: operator function name in the operator implementation file. Defaults to `None`. The argument is generated automatically based on `op_type`. For details about how the argument of `op_func_name` is generated based on `op_type`, see Step 4. For example, for an Add operator, the corresponding API must exist in `impl/add.py`. The API definition is as follows.

```python
@check_op_params(REQUIRED_INPUT, REQUIRED_INPUT, REQUIRED_OUTPUT, KERNEL_NAME)
def add(input_x, input_y, output_z, kernel_name="add"):
```

### add_elewise_case

- **Prototype**
  ```python
  BroadcastOpUT.add_elewise_case(self, soc, param_info, expect=op_status.SUCCESS, case_name=None)
  ```

- **Description**
  Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an .o file.

- **Parameters**
  - **soc**: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/ata/data/platform_config` directory. `support_soc` can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to `all` or `None`, all SoCs are supported.
  - **param_info**: information about the operator input, coming in either of the following format:
    - `[dtype, shape, format, ori_shape, ori_format]`
    - `[dtype, shape, format]`: In this format, the values of `ori_shape` and `ori_format` are the same as those of `shape` and `format`.
  - **expect**: expected build result. Defaults to `op_status.SUCCESS`. It can also be set to `RuntimeError`, indicating an expected exception.
  - **case_name**: defaults to `None`. The test framework automatically generates the argument of `case_name`.

### Example

```python
ut_case.add_elewise_case("Ascend910", ["float16", (32, 32), "ND"])  
```

The preceding test case implements the same function as calling the `add_case` API.

```python
ut_case.add_case(support_soc="Ascend910", case=
  "params": [{
    "shape": (32, 32),
  }])
```

---

MindStudio
User Guide
8 Custom Operator Development

Copyright © Huawei Technologies Co., Ltd.

Issue 01 (2021-06-24)  Copyright © Huawei Technologies Co., Ltd.  267
add_elewise_case_simple

- **Prototype**
  ```python
  BroadcastOpUT.add_elewise_case_simple(self, soc, dtypes, shape, expect=op_status.SUCCESS, case_name=None)
  ```
- **Description**
  Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an .o file.
- **Parameters**
  - **soc**: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/atc/data/platform_config` directory. `support_soc` can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to `all` or `None`, all SoCs are supported.
  - **dtypes**: types of the data to be tested. If multiple data types are set, multiple test cases are added at a time.
  - **shape**: shape of the operator input.
  - **expect**: expected build result. Defaults to `op_status.SUCCESS`. It can also be set to `RuntimeError`, indicating an expected exception.
  - **case_name**: defaults to `None`. The test framework automatically generates the argument of `case_name`.

Compared with the `add_elewise_case` API, this API sets the formats of all inputs to `ND`.

**Example**

```python
ut_case.add_elewise_case_simple("Ascend910", ["float16", "float32"], [32, 32])
```

The preceding test case implements the same function as calling the `add_case` API.

```python
ut_case.add_case(support_soc="Ascend910", case={
  "params": [{
    "shape": (32, 32),
    "ori_shape": (32, 32),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float16"
  }, {
    "shape": (32, 32),
    "ori_shape": (32, 32),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float16"
  }]
})
```
8.5.5.2.4 ReduceOpUT

Overview

Inherits the capabilities of OpUT.

ReduceOpUT defines test cases for Reduce operators, for example, the ReduceSum and ReduceMean operators. It also provides more convenient APIs for these operators. For example, when creating the test case file for operator build, you do not need to set the format in some simple scenarios.

Definition of the ReduceOpUT Test Class

- Prototype
  ReduceOpUT(op_type, op_module_name=None, op_func_name=None)

- Parameters
  - op_type: operator type.
  - op_module_name: operator module name, that is, the name and path of the operator implementation file, for example, impl.add (the file path is impl/add.py). Defaults to None. The argument is generated automatically based on op_type. For example, for a ReduceSum operator, the generated module name is impl.reduce_sum. For details about how the argument of op_module_name is generated based on op_type, see Step 4.
  - op_func_name: operator function name in the operator implementation file. Defaults to None. The argument is generated automatically based on op_type. For details about how the argument of op_func_name is generated based on op_type, see Step 4. For example, for an Add operator, the corresponding API must exist in impl/add.py. The API definition is as follows.

```python
@check_op_params(REQUIRED_INPUT, REQUIRED_INPUT, REQUIRED_OUTPUT, KERNEL_NAME)
def add(input_x, input_y, output_z, kernel_name="add"):
```

add_reduce_case

- Prototype
  BroadcastOpUT.add_reduce_case(self, soc, input_info, axes,
keep_dim=False, expect=op_status.SUCCESS, case_name=None)
● Description

Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an .o file.

● Parameters

- **soc**: used to test whether the test case file supports the corresponding Ascend AI Processor. The value is the name of the corresponding .ini file in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/atc/data/platform_config` directory. `support_soc` can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to all or None, all SoCs are supported.

- **input_info**: information about the operator input, coming in either of the following format:
  - `[dtype, shape, format, ori_shape, ori_format]`
  - `[dtype, shape, format]`: In this format, the values of `ori_shape` and `ori_format` are the same as those of `shape` and `format`.

- **axes**: dimensions to reduce.

- **keep_dims**: If True, retains reduced dimensions with length 1. Either True or False.

- **expect**: expected build result. Defaults to `op_status.SUCCESS`. It can also be set to `RuntimeError`, indicating an expected exception.

- **case_name**: defaults to None. The test framework automatically generates the argument of `case_name`.

Example

```python
ut_case.add_reduce_case("Ascend910", ["float16", (32, 32), "ND"], [0], False)
```

The preceding test case implements the same function as calling the `add_case` API.

```python
ut_case.add_case(support_soc="Ascend910", case={
    "params": [
        {
            "shape": (32, 32),
            "ori_shape": (32, 32),
            "format": "ND",
            "ori_format": "ND",
            "dtype": "float16"
        },
        {
            "shape": (32),
            "ori_shape": (32),
            "format": "ND",
            "ori_format": "ND",
            "dtype": "float16"
        },
    ],
    [0], False
})
```

### add_reduce_case_simple

- **Prototype**

  `BroadcastOpUT.add_reduce_case_simple(self, soc, dtypes, shape, axes, keep_dim=False, expect=op_status.SUCCESS, case_name=None)`

- **Description**

  Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an .o file.
- **Parameters**
  - **soc**: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/(arch)-linux/atc/data/platform_config` directory. `support_soc` can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to `all` or `None`, all SoCs are supported.
  - **dtypes**: types of the data to be tested. If multiple data types are set, multiple test cases are added at a time.
  - **shape**: shape of the operator input.
  - **axes**: dimensions to reduce.
  - **keep_dims**: If `True`, retains reduced dimensions with length 1. Either `True` or `False`.
  - **expect**: expected build result. Defaults to `op_status.SUCCESS`. It can also be set to `RuntimeError`, indicating an expected exception.
  - **case_name**: defaults to `None`. The test framework automatically generates the argument of `case_name`.

Compared with the `add_reduce_case` API, this API sets the formats of all inputs to ND.

**Example**

```python
ut_case.add_reduce_case_simple("Ascend910", ["float16", "float32"], [32, 32], [1,], True)
```

The preceding test case implements the same function as calling the `add_case` API.

```python
ut_case.add_case(support_soc="Ascend910", case={
  "params": [{
      "shape": (32, 32),
      "ori_shape": (32, 32),
      "format": "ND",
      "ori_format": "ND",
      "dtype": "float16"
    }, { 
      "shape": (32, 1),
      "ori_shape": (32, 1),
      "format": "ND",
      "ori_format": "ND",
      "dtype": "float16"
    }], [1,], True)
})
```

```python
ut_case.add_case(support_soc="Ascend910", case={
  "params": [{
      "shape": (32, 32),
      "ori_shape": (32, 32),
      "format": "ND",
      "ori_format": "ND",
      "dtype": "float32"
    }, { 
      "shape": (32, 1),
      "ori_shape": (32, 1),
      "format": "ND",
      "ori_format": "ND",
      "dtype": "float32"
    }], [1,], True)
})
```
8.5.5.3 Creating and Running UT Cases

Prerequisites

The custom operator has been developed, including the operator implementation code and operator prototype definition. For details, see 8.5.1 Operator Code Implementation, 8.5.2 Operator Prototype Definition, and 8.5.3 Operator Information Definition.

**NOTE**
CentOS7.8 ARM containers do not support the UT of the operator implementation code.

Procedure

**Step 1** Create UT cases.

1. A UT case file can be created in any of the following ways:
   Right-click the root directory of the operator project and choose **New Cases > TBE UT Case** from the shortcut menu.
   
   If UT cases of the operator exist, right-click the **testcases** directory or **testcases > ut**, and choose **New Cases > TBE UT Case** from the shortcut menu to create UT cases.

2. In the **Create UT for an Operator** window, choose the target operator, and click **OK**, as shown in the following figure.

   ![Create UT for an Operator window](image)

   **NOTE**
   If the UT cases of the operator already exist, the message “testcases/ut/ops_test/xx already exists. Do you want to overwrite?” is displayed.
   You can click **Overwrite** or **Cancel**.

   After the creation is complete, the **testcases** folder is generated in the root directory of the operator project. The directory structure is as follows.

   ```
   ├── MyOperator            // Root directory of the project
   │   ├── testcases
   │   │   ├── libs                  // GTest framework. It is a third-party dependency and can be ignored.
   │   │   ├── ut
   │   │   │   ├── ops_test
   │   │   │   │   ├── add
   │   │   │   │   │   ├── CMakeLists.txt        // Build script
   │   │   │   │   │   │   └ test_add_impl.py      // Test case file for the operator implementation code
   │   │   │   │   │   │   └ test_add_proto.cc    // Test case file for the operator prototype definition code
   │   │   │   │   │   │   └ CMakeLists.txt      // Build script
   │   │   │   │   │   └ test_main.cc            // Main entry for test case calling
   │   │   └ CMakeLists.txt
   │
   ├── MyOperator            // Root directory of the project
   │   ├── testcases
   │   │   ├──.libs
   │   │   └ ut
   │   │   └ ops_test
   │   │   └ add
   │   │   └ CMakeLists.txt
   │   └ testcases
   └ CMakeLists.txt
   ```

**Step 2** For the CentOS container, NeoKylin OS, and Kylin OS, add the following information in bold to the **./testcases/ops_test/add/CMakeLists.txt** file. For other OSs, skip this step.
add_definitions(-D_GLIBCXX_USE_CXX11_ABI=0)
set(CMAKE_CXX_FLAGS "-std=c++11")
set(PROJECT_DIR "$ENV{PROJECT_PATH}")
set(GTEST_DIR ${PROJECT_DIR}/testcases/libs/gtest)
set(ADK_DIR "$ENV{ADK_PATH}")
set(ATC_DIR ${ADK_DIR}/atc)
set(OP_PROTO_SRC_DIR ${PROJECT_DIR}/op_proto)
message(STATUS "ATC_DIR=${ATC_DIR}")
enable_testing()

include_directories(
  "$GTEST_DIR/include"
  "$ATC_DIR/include"
  "$OP_PROTO_SRC_DIR"
)

aux_source_directory(${OP_PROTO_SRC_DIR} OP_PROTO_SOURCE_SRCS)
file(GLOB OP_PROTO_TEST_FILES **proto.cc)

link_directories(
  "$ATC_DIR/lib64"
  "$GTEST_DIR"
  "/usr/local/gcc7.3.0/lib64/"// Set this parameter to the actual lib64 path of gcc7.3.0 if the user
configures the Dockerfile.
)

set(CUSTOM_OBJECT_NAME "add_proto_test")

add_executable(${CUSTOM_OBJECT_NAME}
  ${PROJECT_DIR}/testcases/ut/ops_test/test_main.cc ${OP_PROTO_SOURCE_SRCS} 
  {OP_PROTO_TEST_FILES})
target_link_libraries(${CUSTOM_OBJECT_NAME} gtest c_sec alog pthread error_manager graph register)

▌▌ NOTE

The GCC version must be 7.3.0 or later for the UT. If the GCC version does not meet the
requirement, upgrade the GCC.

Step 3 Write UT cases in Python for the operator implementation code.

In the testcases/ut/ops_test/add/test_add_impl.py file, write UT cases in Python for
the operator implementation code. Execute the operator and compare the
compute result with the expect result to test the operator logic.

import sys
from op_test_frame.ut import BroadcastOpUT  # Import the UT class based on the operator type.

ut_case = BroadcastOpUT("Add")  # Instantiate the UT case. ut_case is a keyword of the UT framework
and cannot be modified. Add is the type of the operator.

def calc_expect_func(input_x, input_y, output_z):  # Define the function for generating the expected data.
    res = input_x["value"] + input_y["value"]
    return [res, ]  # Return the expected data.

# Add test cases.
ut_case.add_precision_case("all", {
    "params": [{"dtype": "float16", "format": "ND", "ori_format": "ND", "ori_shape": (32,), "shape": (32,),
    "param_type": "input"},
    {"dtype": "float16", "format": "ND", "ori_format": "ND", "ori_shape": (32,), "shape": (32,),
    "param_type": "input"},
    {"dtype": "float16", "format": "ND", "ori_format": "ND", "ori_shape": (32,), "shape": (32,),
    "param_type": "output"}]
})
# If multiple test cases are defined, multiple ut_case.add_precision_case functions need to be defined.
1. Import the UT class based on the operator type. For details, see 8.5.5.2 APIs.
2. Instantiate the UT case. For details about how to use OpUT, see Definition of the OpUT Test Class.
3. Define the function for generating the expected data.
4. Add test cases.
   For details about how to use each test API, see 8.5.5.2 APIs.
   To compare the result with the expectation, call add_precision_case.

**Step 4** Write UT cases in C++ for the operator prototype definition.

In the testcases/ut/ops_test/add/test_add_proto.cc file, write the UT case in C++ for the operator prototype definition. Define the operator instance, update the input and output of the operator, call InferShapeAndType, and verify the execution process as well as the result of InferShapeAndType.

1. Include the GTest framework and the header file of the operator IR definition.
   
   C++ UT cases use the GTest framework. Therefore, you need to include the GTest framework. The operator prototype is defined in the prototype definition header file. Therefore, you need to include the .h file of the prototype definition.
   
   ```
   // Include the GTest framework.
   #include <gtest/gtest.h>
   // Include the basic vector library.
   #include <vector>
   // Include the header file of the operator IR definition.
   #include "add.h"
   ```

2. Define a test class.
   
   C++ UT cases use the GTest framework. Therefore, a class needs to be defined to inherit the gtest class.
   
   ```
   #include <gtest/gtest.h>
   #include <vector>
   #include "add.h"

   class AddTest : public testing::Test {
   protected:
     static void SetUpTestCase() {
       std::cout << "add test SetUp" << std::endl;
     }

     static void TearDownTestCase() {
       std::cout << "add test TearDown" << std::endl;
     }
   };
   ```
   The test class name can be user-defined and suffixed with Test.

3. Write test cases.
   
   Write a test case function for each scenario. In this case, an operator instance needs to be constructed, including the operator name, shape, and data type.
Call the `InferShapeAndType` function and compare the inferred shape and dtype with the expected results.

The following is an example.

```cpp
TEST_F(AddTest, add_test_case_1) {
    // Define the operator instance, input shape, and input type, which are carried by the TensorDesc instance.
    ge::op::Add add_op;  // `Add` is the operator type, which must be the same as the OpType argument in REG_OP (OpType) in the operator definition file.
    ge::TensorDesc tensorDesc;
    ge::Shape shape({2, 3, 4});
    tensorDesc.SetDataType(ge::DT_FLOAT16);
    tensorDesc.SetShape(shape);

    // Update the operator inputs. The input name must be the same as that in the prototype definition file (.h). For example, x1 and x2 are inputs of the Add operator.
    add_op.UpdateInputDesc("x1", tensorDesc);
    add_op.UpdateInputDesc("x2", tensorDesc);
    // Call the InferShapeAndType function. The InferShapeAndType() API is a fixed API. During case execution, the shape inference function in the operator prototype definition is automatically called.
    auto ret = add_op.InferShapeAndType();
    // Verify whether the calling process is successful.
    EXPECT_EQ(ret, ge::GRAPH_SUCCESS);

    // Obtain the operator output and compare the shape and type. The name of the operator output must be the same as that in the prototype definition file (.h). For example, the operator output is y.
    auto output_desc = add_op.GetOutputDesc("y");
    EXPECT_EQ(output_desc.GetDataType(), ge::DT_FLOAT16);
    std::vector<int64_t> expected_output_shape = {2, 3, 4};
    EXPECT_EQ(output_desc.GetShape().GetDims(), expected_output_shape);
}
```

If the shapes of inputs are different, define multiple TensorDesc objects, as follows.

```cpp
gce::op::Operator1 operator1_op;  // Operator1 indicates the operator type.
    ge::TensorDesc tensorDesc1;
    ge::TensorDesc tensorDesc2;
    ge::Shape shape1({2, 3, 4});
    ge::Shape shape2({3, 4, 5});
    tensorDesc1.SetDataType(ge::DT_FLOAT16);
    tensorDesc1.SetShape(shape1);
    tensorDesc2.SetDataType(ge::DT_FLOAT16);
    tensorDesc2.SetShape(shape2);

    // Update the operator inputs.
    operator1_op.UpdateInputDesc("x1", tensorDesc1);
    operator1_op.UpdateInputDesc("x2", tensorDesc2);
```

**Step 5** Run the UT cases of the operator implementation file.

You can run the UT cases of all operators in the current project or run the UT cases of a single-operator.

- Right-click the `testcases/ut/ops_test` folder and choose Run The Operator 'All' UT Impl with coverage from the shortcut menu to run the test cases of the operator implementation code in the folder.
- Right-click the `testcases/ut/ops_test/operator_name` folder and choose Run The Operator 'operator_name' UT Impl with coverage to run test cases of implementation code of a single-operator.

When the case runs for the first time, the corresponding configuration dialog box is displayed. Configure the parameters and click Run. For details about how to modify the run configurations, see 12.3.1 Editing a Run Configuration.
Table 8-10 Run configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the run configuration (user-defined).</td>
</tr>
<tr>
<td>Test Type</td>
<td>ut_impl is recommended.</td>
</tr>
<tr>
<td>Compute Unit</td>
<td>Compute unit.</td>
</tr>
<tr>
<td></td>
<td>- AI Core/Vector Core</td>
</tr>
<tr>
<td></td>
<td>- AI CPU</td>
</tr>
<tr>
<td></td>
<td>You can select different compute units to switch between the AI Core/Vector</td>
</tr>
<tr>
<td></td>
<td>Core and AI CPU UT configuration pages.</td>
</tr>
<tr>
<td>SoC Version</td>
<td>Current the Ascend AI Processor version.</td>
</tr>
<tr>
<td>Target</td>
<td>Target environment.</td>
</tr>
<tr>
<td></td>
<td>- Simulator_Function: functional simulation environment.</td>
</tr>
<tr>
<td></td>
<td>- Simulator_TMMModel: The scheduling pipeline of operator execution is quickly</td>
</tr>
<tr>
<td></td>
<td>displayed, and the actual operator computing is not performed.</td>
</tr>
<tr>
<td></td>
<td>Currently, the environment is supported in the Ascend 310 AI Processor and</td>
</tr>
<tr>
<td></td>
<td>Ascend 910 AI Processor only.</td>
</tr>
<tr>
<td>Operator Name</td>
<td>Test cases.</td>
</tr>
<tr>
<td></td>
<td>- all: all test cases</td>
</tr>
<tr>
<td></td>
<td>- Other values: test cases of a specific operator</td>
</tr>
<tr>
<td>ADK Machine</td>
<td>SSH connection of the device where the ADK tool is located. For details about</td>
</tr>
<tr>
<td></td>
<td>how to add an SSH connection, see 12.1 SSH Connection Management.</td>
</tr>
<tr>
<td></td>
<td>NOTE</td>
</tr>
<tr>
<td></td>
<td>Only Windows OSs are supported.</td>
</tr>
<tr>
<td>Case Names</td>
<td>Test cases to run, that is, the Python UT cases of the operator implementation</td>
</tr>
<tr>
<td></td>
<td>code. You can select or deselect all test cases.</td>
</tr>
</tbody>
</table>

- Check the execution result.
  a. After the execution is complete, view the execution result in the Run log print window at the bottom.
  b. In the Run window, click the URL in index.html to view the UT case coverage, as shown in Figure 8-14. In the URL, localhost indicates the server IP of MindStudio.
To view the test results of UT cases, you need to use a browser. If no browser is available, install one.

**Figure 8-14** Viewing the UT case coverage

The preceding figure shows that the UT case coverage of the operator is 100% and the UT verification is passed.

If "Page'http://***.html'requested without authorization, you can copy URL and open it in browser to trust it." is displayed, rectify the fault by referring to 15.8.3 How Do I Allow Unsigned Access Requests?

c. Click the operator on the HTML page to navigate to the UT case coverage dialog box, as shown in **Figure 8-15**. The green and red labels are used to indicate the coverage rates.

**Figure 8-15** UT coverage details

d. View the outputs.

- If **Target** is set to **Simulator_TMModel** during running information configuration, you can view the execution pipeline, as follows.
Before the processor performs computation, Vector Unit, Cube Unit, MTE1, MTE2, MTE3, and more units are initialized. As a result, data is generated on each unit before data movement progress is displayed in the timeline.

**Step 6** Run the UT cases defined in the operator prototype.

You can run the UT cases of all operators in the current project or run the UT cases of a single-operator.

- Right-click the `testcases/ut/ops_test` folder and choose **Run The Operator 'All' UT Proto** from the shortcut menu to run the test cases of the operator prototype definition code in the folder.

- Right-click the `testcases/ut/ops_test/operator_name` folder and choose **Run The Operator 'operator_name' UT Proto** from the shortcut menu to run test cases of a single-operator prototype definition code.

When the case runs for the first time, the corresponding configuration dialog box is displayed. Configure the parameters and click **Run**. For details about how to modify the run configurations, see **12.3.1 Editing a Run Configuration**.

<table>
<thead>
<tr>
<th>Table 8-11 Run configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameter</strong></td>
</tr>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Test Type</td>
</tr>
</tbody>
</table>
| Operator Name | Test cases.  
 - `all`: all test cases  
 - Other values: test cases of a specific operator |
| Case Names | Select test cases to be executed, that is, the test cases defined in `TEST_F`. You can select or deselect all test cases. |

After the execution is complete, view the execution result in the log print window at the bottom. The result shows the test case count, including successful execution and failed execution, as shown in the following figure.
8.5.6 Operator Project Building

Overview

The custom operator project is built into a custom OPP runfile `custom_opp_Target OS_Target Architecture.run`.

Build the operator plug-in implementation file, operator prototype definition file, and operator information definition file into the corresponding operator plug-in, operator prototype library, and operator information library. **Figure 8-16** shows the build process.

**Figure 8-16 Build flowchart**

![Build flowchart](image)

**NOTE**

Windows OSs do not support local build.
Build Procedure

**Step 1** In the MindStudio project window, select the operator project.

**Step 2** Choose **Build > Edit Build Configuration...** on the top menu bar.

**Step 3** Set parameters in the displayed **Build Configurations** dialog box, as shown in the following figure. For details, see **Table 8-12**.

![Build Configurations dialog box](image)

**Figure 8-17** Build Configurations dialog box

**Table 8-12** Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build Configuration</td>
<td>Name of the build configuration. Defaults to <strong>Build-Configuration</strong>.</td>
</tr>
<tr>
<td>Build Mode</td>
<td>Build mode.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Remote Build:</strong></td>
</tr>
<tr>
<td></td>
<td><strong>NOTICE</strong> g++ 7.5.0 is required for remote build.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Local Build:</strong></td>
</tr>
<tr>
<td></td>
<td>The operator project is built on the MindStudio server. You can quickly</td>
</tr>
<tr>
<td></td>
<td>locate the implementation code in MindStudio based on the build log for</td>
</tr>
<tr>
<td></td>
<td>troubleshooting. In this mode, you need to configure the cross compilation</td>
</tr>
<tr>
<td></td>
<td>environment by referring to <strong>2.4.5 Configuring the Compilation Environment</strong>.</td>
</tr>
<tr>
<td>SSH Connection</td>
<td>This parameter is available only in <strong>Remote Build</strong> mode. Select an SSH</td>
</tr>
<tr>
<td></td>
<td>configuration from the drop-down list box. If no connection is added,</td>
</tr>
<tr>
<td></td>
<td>click <strong>+</strong> to add it. For details, see <strong>12.1 SSH Connection Management</strong>.</td>
</tr>
<tr>
<td>Environment variables</td>
<td>This parameter is available only in <strong>Remote Build</strong> mode. Set the environment variables.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Target OS</td>
<td>This parameter is available only in <strong>Local Build</strong> mode.</td>
</tr>
<tr>
<td></td>
<td>● For Ascend EP, select the host-side OS where the Ascend AI Processor is located.</td>
</tr>
<tr>
<td></td>
<td>● For Ascend RC, select the OS of the board environment.</td>
</tr>
<tr>
<td>Target Architecture</td>
<td>This parameter is available only in <strong>Local Build</strong> mode. Select the OS architecture of the target environment.</td>
</tr>
</tbody>
</table>

If remote build is used, you need to configure the SSH connection and environment variables of the remote server.

- Configuring the SSH connection
  For details about how to add an SSH connection, see 12.1 SSH Connection Management.

- Configuring the environment variables
  Set the environment variable `ASCEND_TENSOR_COMPLIER_INCLUDE`.
  `ASCEND_TENSOR_COMPLIER_INCLUDE=/home/xxx/Ascend/ascend-toolkit/latest/atc/include`
  Replace `/home/xxx/Ascend/ascend-toolkit/latest/` with the actual installation path of the ADK.

**Step 4** Click **Build** to build the project.

**Step 5** View the build result in the lower part of MindStudio. The custom OPP runfile `custom_opp_Target OS_Target Architecture.run` is generated in the `cmake-build` directory of the operator project.

For the value of **Target OS** and **Target Architecture**, see **Step 3**.

--- End

## 8.5.7 Local Operator Deployment

**NOTICE**

- This section describes how to deploy the custom OPP runfile. During the execution of **8.5.9 ST**, an OPP runfile is automatically generated and deployed in the `opp` directory in the development environment. You can change the deploy path by referring to this section. If you have run operator ST by referring to **8.5.9 ST**, you can safely skip this section.

- Before deploying operators, ensure that the custom OPP runfile `custom_opp_Target OS_Target Architecture.run` is generated by referring to **8.5.6 Operator Project Building**.

- Windows OSs do not support local operator deployment.
Step 1  On the menu bar of the MindStudio project window, choose Ascend > Deploy. The operator deployment dialog box is displayed.

Step 2  Select Deploy Locally and click Deploy.

If the following information is displayed on the Output tab page, the custom operator is successfully deployed.

![Output screen capture showing successful deployment](image)

After the custom OPP is successfully installed, the custom operator is deployed in the /opp directory in the 
(Ascend-CANN-Toolkit installation path)/ascend-toolkit/{version}/{arch}-linux.

The directory structure is as follows.

```
directory structure
```

---End
8.5.8 Remote Operator Deployment

Deploy the custom OPP runfile `custom_opp_Target_OS_Target_Architecture.run` to the system OPP in the hardware environment with Ascend AI Processor to construct necessary conditions for operators execution on the network.

Step 1 In the MindStudio project window, select the operator project.

Step 2 On the top menu bar, choose **Ascend > Deploy**. The operator deployment dialog box is displayed.

Choose **Deploy Remotely > SSH Connection** from the drop-down list box to configure SSH connection. If the no SSH connection is not added in advance, click + to add it.

Step 3 Configure the environment variable.

You can use either of the following methods:

- Configure the environment variables on the host of the hardware device where the Ascend AI Processor is located. Deploy the operator on the host as the running user in MindStudio. Before deploying the operator, ensure the following environment variable is configured on the host.
  
  a. Run the following command in the `$HOME/.bashrc` file on the host as the running user:

     ```bash
     export ASCEND_OPP_PATH=home/xxx/Ascend/ascend-toolkit/latest/opp
     
     home/xxx/Ascend/ascend-toolkit/latest is the OPP installation path. Replace it with the actual path.
     
     b. Run the following command to make the environment variable to take effect:

     ```bash
     source ~/.bashrc
     ```

  - Add the environment variable in **Environment Variables**. Type `ASCEND_OPP_PATH=/home/xxx/Ascend/ascend-toolkit/latest/opp` in the **Environment Variables** field.

    `home/xxx/Ascend/ascend-toolkit/latest` is the OPP installation path. Replace it with the actual path.

    You can also click the icon next to the text box and enter a value in the displayed dialog box.

    – Type `ASCEND_OPP_PATH` in the **Name** field.

    – Type `home/xxx/Ascend/ascend-toolkit/latest/opp` in the **Value** field.

**Figure 8-18** Operator deployment

- Deploy Locally
- Deploy Remotely

SSH Connection

Environment Variables

[Image of the operator deployment interface]
Step 4  Select the target server for operator deployment and click **Deploy**.

Step 5  Deploy the operators. Operator deployment is equivalent to installing the custom OPP generated in **8.5.6 Operator Project Building**. After the deployment, the operator is deployed in the OPP installation path on the host. The default path is `/usr/local/Ascend/opp/`.

Figure 8-19 Operator deployment log messages

```
2020-03-09 21:49:59 Verifying archive integrity... All good.
2020-03-09 21:49:59 Uncompressing version:1.0
2020-03-09 21:49:59 [runtime] [2020-03-10 09:49:57] [INFO] copy uninstall sh success
2020-03-09 21:49:59 [ops custom] upgrade framework
2020-03-09 21:49:59 [runtime] [2020-03-10 09:49:57] [INFO] replace old ops framework files ...
2020-03-09 21:49:59 [runtime] [2020-03-10 09:49:57] copy new ops framework files ...
2020-03-09 21:49:59 [ops custom] upgrade op proto
2020-03-09 21:49:59 [runtime] [2020-03-10 09:49:57] [INFO] replace old ops op proto files ...
2020-03-09 21:49:59 [runtime] [2020-03-10 09:49:57] copy new ops op proto files ...
2020-03-09 21:49:59 [ops custom] upgrade op impl
2020-03-09 21:49:59 [runtime] [2020-03-10 09:49:57] [INFO] replace old ops op_impl files ...
2020-03-09 21:49:59 [runtime] [2020-03-10 09:49:57] copy new ops op_impl files ...
2020-03-09 21:49:59 [ops custom] change mode ...
2028-03-09 21:49:59 SUCCESS
2020-03-09 21:49:59 2028-03-10 00:40:59 - [INFO] Deploying the operator info to the remote host finished.
```

After the custom OPP is deployed on the host, the directory structure is similar to the following:

```
|-- opp      // OPP directory
|   |-- op_impl
|       |-- built-in
|       |-- custom
|       |   |-- ai_core
|       |       |-- config
|       |       |   |-- aic_ops_info.json // Custom operator info library file
|       |   |-- custom_impl // Custom operator implementation code
|           |-- add.py
|   |-- framework
|       |-- built-in
|       |-- custom
|       |   |-- caffe
|       |       |-- tensorflow // Operator plug-in library for TensorFlow adaption
|       |       |   |-- libcust_tf_parsers.so
|       |   |-- op_proto
|       |       |-- built-in
|       |       |-- custom
|       |       |   |-- libcust_op_proto.so // Prototype library file of the custom operator
|       |   |-- vector_core // Reserved directory, which can be ignored

----End

8.5.9 ST

Overview

MindStudio provides an upgraded ST framework to automatically generate test cases, verify operator functionality and compute accuracy in real hardware environment, and generate an execution report. Feature details are as follows:

- Generates an operator test case definition file based on the operator information library.
- Generates test data of different shapes and dtypes and AscendCL-based test cases from the operator test case definition file.
- Builds the operator project, deploys the operators in the system OPP, and runs test cases in the hardware environment to verify the operator functionality.
- Generates an ST report (st_report.json) that displays information about test cases and phase-by-phase execution states.
- Generates a test function, compares the expected operator output and the actual operator output, and displays the comparison result to verify the compute accuracy.

Prerequisites
- Custom operators have been developed. For details, see 8.5.1 Operator Code Implementation, 8.5.2 Operator Prototype Definition, 8.5.3 Operator Information Definition, and 8.5.4 Operator Plugin Implementation.
- MindStudio has been connected to a hardware device.

Generating an ST Case Definition File

**Step 1** Create ST cases.

Three portals are available.
- Right-click the root directory of the operator project and choose New Cases > ST Case from the shortcut menu.
- Right-click the operator information definition file {project name}/tbe/op_info_cfg/ai_core/{SoC version}/xx.ini, and choose New Cases > ST Case.
- If ST cases of the operator exist, right-click the testcases or testcases > st directory, and choose New Cases > ST Case from the shortcut menu to add ST cases.

**Step 2** In the Create ST Cases for an Operator dialog box, select the operator for which the ST case needs to be created.

See the following figure.

![Create ST Cases for an Operator dialog box](image)

**Operator Name**: Select an operator name from the Operator drop-down list.

**SoC Version**: Select the version of your the Ascend AI Processor from the SoC Version drop-down list.

- If Import operator info from a model is not selected, click OK and an operator test case definition file with empty shapes is generated. The following figure shows the Design Cases dialog box.
You need to configure the shape information to generate test data and test cases. Configure the rest fields as required. For details, see **Field Description of the Operator Test Case Definition File**.

- If you select **Import operator info from a model** and upload a TensorFlow model file (.pb) that contains the operator, the top-layer shape of the obtained model is displayed.

You can also modify the shape information of the first-layer input in **Input Nodes Shape**.

After you click **OK**, the tool automatically dumps the shape information of the selected operator based on the shape information of the first layer and generates the corresponding operator test case definition file.

This file is used for generating test data and test cases. You can modify related fields. For details, see **Field Description of the Operator Test Case Definition File**.

**NOTE**

To use this function, you need to install the TensorFlow framework in the operating environment. If the Windows OS is used, you need to install the TensorFlow framework on the local Windows host.

**Step 3** To compare the expected data with the benchmark data, define and configure a function for generating the expected data of the operator.

1. **Customize a test function for generating the expected operator result.**

   The function is implemented on a framework such as TensorFlow or Caffe and has the same functionality as the custom operator. The function runs on the CPU and generates benchmark data. The operator accuracy is tested by comparing the benchmark data with the output data. The function for generating expected data is implemented using the Python language. Multiple expected operator data generation functions can be implemented in a Python file. Keep the inputs, outputs, and attributes (including the format, type, and shape) of this function consistent with those of the custom operator.

2. **Edit the test case definition file.**

   Configure the function in the test case definition file. You can configure it in the **Design** view or the **Text** view.
– In the Design view, select a destination path of the Python file as the script path in Expected Result Verification dialog box. In Script Function, enter the name of the function that generates expected operator data.

![Expected Result Verification](image)

**NOTE**

In Script Function, you can choose to enter the function or leave it empty.

- If the name of the function is entered, the function is called to generate the benchmark data during ST.
- If the function name is left empty, the function with the same name as the custom operator is automatically matched to generate benchmark data during ST. If no function with the same name exists, a message indicating match failure is displayed.

– In the Text view, if the calc_expect_func_file parameter is added, the value is the file path and name of the function generating the expected operator data. For example:

```json
```

Where, `/home/teste/test_*.py` indicates the implementation file of the test function, and `function` indicates the function name. Separate the file path and function name with a colon (`:`).

Example: The `test_add.py` file is the expected data generation file of the Add operator. The function implementation is as follows:

```python
def calc_expect_func(input_x, input_y, out):
    res = input_x["value"] + input_y["value"]
    return [res, ]
```

**Note:** You need to complete the expected data generation function of the operator based on the developed custom operator. The input parameters of the operator's expected data generation function must be the same as those of the operator in the operator code implementation.

**Step 4** Click Save to save the modification to the operator test case definition file.

The operator test case definition file (named as `OpType_case_timestamp.json`) is stored in the `testcases/st/OpType/{SoC Version}` directory under the root directory of the operator project.

**NOTE**

Strictly follow the naming rules of the operator test case definition file. Do not name irrelevant files in this format to the `testcases/st/OpType/{SoC Version}` directory under the root directory of the operator project. Otherwise, file parsing errors may occur.

----End
Running ST Cases

**Step 1** Right-click the ST case definition file generated in Generating an ST Case Definition File (testcases > st > add > {SoC Version} > xxxx.json) and choose Run The Operator 'xxx' ST Case from the shortcut menu.

### Table 8-13 Run configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Set the name of the run configuration (user-defined).</td>
</tr>
<tr>
<td>Test Type</td>
<td>Select <code>st_cases</code>.</td>
</tr>
</tbody>
</table>
| Execute Mode     | ● Remote Execute  
                  ● Local Execute  
                  NOTE Local Execute does not apply to Windows OSs. |
| SSH Connection   | If Execute Mode is set to Remote Execute, select an SSH connection from the drop-down list box.  
                  You can also click `➕` to add a new SSH connection. For details about how to add an SSH connection, see 12.1 SSH Connection Management. |
| ADK Machine      | SSH connection of the device where the ADK tool is located. For details about how to add an SSH connection, see 12.1 SSH Connection Management.  
                  NOTE This parameter applies only to Windows OSs. |
| Environment Variables | ● Add environment variables in the text box, formatted as:  
                  `PATH_1=Path 1; PATH_2=Path 2`  
                  Use semicolons (`;`) to separate multiple environment variables.  
                  ● You can also click the icon next to the text box and enter a value in the displayed dialog box.  
                  - Type `PATH_1` in the Name field.  
                  - Type the value (Path 1) in the Value field.  
                  If you select Instead system environment variables, the system environment variables are displayed. |
<p>| Operator Name    | Select the operator to test. |
| SoC Version      | Select the SoC version. |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executable File Name</td>
<td>Select the test case definition file to run from the drop-down list box. If the ST is performed on the AI CPU operator, <em>(AI CPU)</em> is displayed in front of the test case file.</td>
</tr>
<tr>
<td>Target OS</td>
<td>For Ascend EP, select the OS of the host of the hardware device powered by the the Ascend AI Processor. For Ascend RC, select the OS of the board environment.</td>
</tr>
<tr>
<td>Target Architecture</td>
<td>Select the OS architecture of the target environment.</td>
</tr>
<tr>
<td>Case Names</td>
<td>Specify the names of the cases to be executed. <strong>Note:</strong> All cases are selected by default. You can deselect unnecessary cases.</td>
</tr>
<tr>
<td>Advanced Options</td>
<td>Specify advanced options.</td>
</tr>
<tr>
<td>ACT Log Level</td>
<td>Select an ACT log level.</td>
</tr>
<tr>
<td></td>
<td>- INFO</td>
</tr>
<tr>
<td></td>
<td>- DEBUG</td>
</tr>
<tr>
<td></td>
<td>- WARNING</td>
</tr>
<tr>
<td></td>
<td>- ERROR</td>
</tr>
<tr>
<td></td>
<td>- NULL</td>
</tr>
<tr>
<td>Precision Mode</td>
<td>Set the precision mode. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>- force_fp176</td>
</tr>
<tr>
<td></td>
<td>- allow_mix_precision</td>
</tr>
<tr>
<td></td>
<td>- allow_fp32_to_fp16</td>
</tr>
<tr>
<td></td>
<td>- must_keep_origin_dtype</td>
</tr>
<tr>
<td>Device Id</td>
<td>Set the ID of the device that runs the ST. Specify the ID of the AI processor in use.</td>
</tr>
<tr>
<td>Enable Profiling</td>
<td>Enable profiling to obtain the performance data of the operator on the the Ascend AI Processor. To use this function, you need to configure the path of the msprof tool in the operating environment to the <em>PATH</em> environment variable. The msprof tool works under <em>toolkit/tools/profiler/bin/msprof</em> of the Toolkit directory.</td>
</tr>
</tbody>
</table>
NOTE
- The ST supports the setting and query of the board log level. For details, see 12.6 Log Management.
- The Windows OS does not support the Local Execute function.

Step 2 Set the host running user.
Add a host running user in the SSH Connection dialog box. The user must be with the HwHiAiUser group. For details about how to add an SSH connection, see 12.1 SSH Connection Management.

Step 3 Configure the environment variables of related components in the operating environment.
- On a remote device:
  - For Ascend EP, you need to configure the environment variables of the component installation paths on the host of the device.
    Configure the installation paths of the ACLlib, FwkACLlib, and Driver components in the ~/.bashrc file as the host running user.
    i. Open the .bashrc file of the running user.
    vi ~/.bashrc
    ii. Append the following lines to the file:
    ```bash
    export ASCEND_DRIVER_PATH=/usr/local/Ascend/driver
    export ASCEND_HOME=/usr/local/Ascend/ascend-toolkit/latest
    export ASCEND_AICPU_PATH=${ASCEND_HOME}/<target architecture>
    ```
    If the remote device is in an inference environment:
    ```bash
    export LD_LIBRARY_PATH=${ASCEND_DRIVER_PATH}/lib64:$\{ASCEND_HOME\}/acllib/lib64:$LD_LIBRARY_PATH
    ```
    If the remote device is in a training environment:
    ```bash
    export LD_LIBRARY_PATH=${ASCEND_DRIVER_PATH}/lib64/driver:$\{ASCEND_DRIVER_PATH\}/lib64/common:$\{ASCEND_HOME\}/fwkacllib/lib64:$LD_LIBRARY_PATH
    ```
    If the preceding environment variables exist, check if the installation path is correct.
    Replace <target architecture> with the OS architecture, for example, x86_64-linux or arm64-linux.
    iii. Save the file and exit.
    ```bash
    :wq
    ```
    iv. Make the configuration take effect.
    ```bash
    source ~/.bashrc
    ```
- Add the environment variable in Environment Variables.
  Set the environment variables in the Environment Variables area as described in Step 1.
  - Add environment variables in the text box.
    ASCEND_DRIVER_PATH=/usr/local/Ascend/driver;
    ASCEND_HOME=/usr/local/Ascend/ascend-toolkit/latest;
ASCEND_AICPU_PATH=${ASCEND_HOME}/<target architecture>-linux;

If the remote device is in an inference environment:
LD_LIBRARY_PATH=${ASCEND_DRIVER_PATH}/lib64:${ASCEND_HOME}/acllib/lib64:$LD_LIBRARY_PATH;

If the remote device is in a training environment:
LD_LIBRARY_PATH=${ASCEND_DRIVER_PATH}/lib64/driver:${ASCEND_DRIVER_PATH}/lib64/common:${ASCEND_HOME}/fwkacllib/lib64:$LD_LIBRARY_PATH;

Modify the environment variables based on the actual installation paths of the driver and ADK and the architecture used by the remote OS.

- You can also click the icon next to the text box and enter a value in the displayed dialog box.
  
Type the environment variable name in the **Name** field and the value in the **Value** field.

**Step 4** Click **Run**.

MindStudio generates test data and test code in `/testcases/st/out/<operator name>` under the operator project root directory based on the operator test case definition file, builds an executable file, and executes test cases on the specified hardware backend. The report about comparison of execution result and benchmark is printed to the **Output** window and the **st_report.json** file is generated in `/testcases/st/out/<operator name>` under the operator project root directory. For details about the **st_report.json** file, see **Table 8-14**.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>run_cmd</td>
<td>Command.</td>
</tr>
<tr>
<td>report_list</td>
<td>List of reports of test cases.</td>
</tr>
<tr>
<td>trace_detail</td>
<td>Execution details.</td>
</tr>
<tr>
<td>st_case_info</td>
<td>Test information.</td>
</tr>
<tr>
<td>expect_data_path</td>
<td>Path of the expected result.</td>
</tr>
<tr>
<td>case_name</td>
<td>Test case name.</td>
</tr>
<tr>
<td>input_data_path</td>
<td>Path of the input data.</td>
</tr>
<tr>
<td>planned_output_data_paths</td>
<td>Path of the actual result.</td>
</tr>
<tr>
<td>op_params</td>
<td>Operator parameters.</td>
</tr>
<tr>
<td>stage_result</td>
<td>Execution result by phase.</td>
</tr>
<tr>
<td>case_name</td>
<td>Test name.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>status</td>
<td>Test result.</td>
</tr>
</tbody>
</table>

----End

**More Functions**

- **Switching between the Design and Text views**
  Double-click the operator test case definition file in the `testcases/st/OpType/{SoC Version}` folder or right-click the file and choose `OpType_case_timestamp.json` from the shortcut menu. The window is displayed with Design and Text views, as shown in the following figure.

  If there is no incorrect configuration information in the Design view, the changes will be automatically synchronized to the Text view when you switch to the Text view. The reverse is also true.

- **Adding a test case**
  On the operator test case definition page, click Add. The dialog box shown in the following figure is displayed, prompting you to input the Case Name.

  **Case Name** is a string combination of digits, letters, and underscores (_). Click OK to add the created case to the bottom of the Design Cases dialog box. The fields of the new case are empty and need to be configured. For
details about the configuration rules, see Field Description of the Operator Test Case Definition File.

● Deleting a test case

In the operator test case definition dialog box, click ☑️ on the right of a case to delete the case.

● Copying and adding an operator input

On the operator test case definition page, click ☑️ on the right of Input[xx] to copy Input[xx] as a new Input[xx]. Alternatively, click + to add an operator input. Modify the parameters based on the site requirements.

NOTE

If the operator ST project supports dynamic multi-shape operators, when the operator input of x1 is copied, the Name parameter must be named in the x10, x11, x12, ..., format. When the operator input of x2 is copied, the Name parameter must be named in the x20, x21, x22, ..., format.

● Deleting an operator input

In the operator test case definition dialog box, click ☑️ on the right of Input[xx] to delete Input[xx].
● Running one or more test cases

In the operator test case definition dialog box, select one or more cases and click Run.

● Modifying and running ST source code

a. Generating source code
   After the ST is successful, the ST source code is generated in ./testcases/st/out/OpName/src.

b. Modifying source code
   You can modify the source code as required to implement custom requirements and functions.

c. Running source code
   Right-click ./testcases/st/out/<operator name> and choose Run St Source from the shortcut menu to run the modified source code.

⚠️ CAUTION

If you run the ST from another entry, the modified source code will be overwritten. Therefore, back up the source code in advance.
### Field Description of the Operator Test Case Definition File

**Table 8-15** Operator test case definition file (.json)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test.xxx.xx</td>
<td>(Required) A string. Test case name.</td>
</tr>
<tr>
<td>Input[xx]</td>
<td>(Required) Operator input.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTICE</strong></td>
</tr>
<tr>
<td></td>
<td>The number of parameter values in each</td>
</tr>
<tr>
<td></td>
<td>input of an operator must be the same.</td>
</tr>
<tr>
<td></td>
<td>Otherwise, test case generation will fail.</td>
</tr>
<tr>
<td></td>
<td>For example, if the Input[01] supports two</td>
</tr>
<tr>
<td></td>
<td>formats, the Input[02] also supports two formats.</td>
</tr>
<tr>
<td></td>
<td>Similarly, the parameter values of type,</td>
</tr>
<tr>
<td></td>
<td>shape, data_distribute, and value_range in all</td>
</tr>
<tr>
<td></td>
<td>Input[xx] must be the same.</td>
</tr>
<tr>
<td></td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>Operator input name.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>Format</td>
<td>(Required) A string or a 1D array of strings. Input format. Must not be empty. Value range:</td>
</tr>
<tr>
<td></td>
<td>- NCHW</td>
</tr>
<tr>
<td></td>
<td>- NHWC</td>
</tr>
<tr>
<td></td>
<td>- ND: any format</td>
</tr>
<tr>
<td></td>
<td>- NC1HWC0: the 5D format defined by Huawei. C0 is closely related to the micro-architecture, and the value is equal to a size of a cube unit, for example, 16. C1 divides the C dimension by C0, that is, C1 = C/C0. When the division is not exact, the last data segment is padded to C0.</td>
</tr>
<tr>
<td></td>
<td>- FRACTAL_Z: format of the convolution weight.</td>
</tr>
<tr>
<td></td>
<td>- FRACTAL_NZ: fractal format defined by Huawei. The data format of the output matrix is NW1H1H0W0 for Cube Unit computation. The matrix is divided into (H1 x W1) fractals in column-major order, which looks like an N-shape layout. Each fractal consists of (H0 x W0) elements in row-major order, resembling a Z-shaped layout. Thus the NW1H1H0W0 data format is referred to as an Nz format. (H0 x W0) indicates the size of a fractal, as shown in the following figure.</td>
</tr>
</tbody>
</table>
|       | - RESERVED: reserved. If this value is used, type needs to be set to UNDEFINED, indicating that the
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>corresponding operator input is optional.</td>
</tr>
<tr>
<td>-</td>
<td>OriginFormat (Optional) Original tensor format. If this parameter is not</td>
</tr>
<tr>
<td></td>
<td>carried, the original tensor format is retained.</td>
</tr>
<tr>
<td>-</td>
<td>ShapeRange (Optional) Value range of a dynamic shape. The default value is</td>
</tr>
<tr>
<td></td>
<td>[1, -1]. For example, if Shape is [16, -1, 16, -1], -1 indicates an</td>
</tr>
<tr>
<td></td>
<td>unknown shape. If ShapeRange is [1, 128], [1, -1], [1, 128] indicates that</td>
</tr>
<tr>
<td></td>
<td>the value of the first -1 in Shape ranges from 1 to 128. [1, -1] indicates</td>
</tr>
<tr>
<td></td>
<td>that the value of the second -1 in Shape ranges from 1 to infinity.</td>
</tr>
<tr>
<td>-</td>
<td>TypicalShape Fixed shape, which is used to generate available test data.</td>
</tr>
<tr>
<td>-</td>
<td>Type (Required) A string or a 1D array of strings. Input data type.</td>
</tr>
<tr>
<td></td>
<td>• bool</td>
</tr>
<tr>
<td></td>
<td>• int8</td>
</tr>
<tr>
<td></td>
<td>• uint8</td>
</tr>
<tr>
<td></td>
<td>• int16</td>
</tr>
<tr>
<td></td>
<td>• uint16</td>
</tr>
<tr>
<td></td>
<td>• int32</td>
</tr>
<tr>
<td></td>
<td>• int64</td>
</tr>
<tr>
<td></td>
<td>• uint32</td>
</tr>
<tr>
<td></td>
<td>• uint64</td>
</tr>
<tr>
<td></td>
<td>• float16</td>
</tr>
<tr>
<td></td>
<td>• float</td>
</tr>
<tr>
<td></td>
<td>• UNDEFINED: applies to an optional input.</td>
</tr>
<tr>
<td>-</td>
<td>Shape (Required) An int or a 1D or 2D array of ints. Input shape.</td>
</tr>
<tr>
<td>-</td>
<td>OriginShape (Optional) Original tensor shape. If this field is not carried,</td>
</tr>
<tr>
<td></td>
<td>the original tensor shape is retained.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td><strong>ValueRange</strong> (Required)</td>
</tr>
<tr>
<td></td>
<td>An int or float, or a 1D or 2D array of ints or floats.</td>
</tr>
<tr>
<td></td>
<td>Value range. Must not be empty.</td>
</tr>
<tr>
<td></td>
<td>Formatted as <code>[min_value, max_value]</code>, with <code>min_value ≤ max_value</code>.</td>
</tr>
<tr>
<td></td>
<td><strong>DataDistribute</strong> (Required)</td>
</tr>
<tr>
<td></td>
<td>A string or a 1D array of strings.</td>
</tr>
<tr>
<td></td>
<td>Data distribution modes for generating test data:</td>
</tr>
<tr>
<td></td>
<td>● <strong>uniform</strong>: returns random values that are evenly distributed.</td>
</tr>
<tr>
<td></td>
<td>● <strong>normal</strong>: returns random values of the normal distribution (Gaussian distribution).</td>
</tr>
<tr>
<td></td>
<td>● <strong>beta</strong>: returns random values of Beta distribution.</td>
</tr>
<tr>
<td></td>
<td>● <strong>laplace</strong>: returns random values of Laplace distribution.</td>
</tr>
<tr>
<td></td>
<td>● <strong>triangular</strong>: returns random values of the triangular distribution.</td>
</tr>
<tr>
<td></td>
<td>● <strong>relu</strong>: returns random values that are evenly distributed and activated by the ReLU function.</td>
</tr>
<tr>
<td></td>
<td>● <strong>sigmoid</strong>: returns random values that are evenly distributed and activated by the sigmoid function.</td>
</tr>
<tr>
<td></td>
<td>● <strong>softmax</strong>: returns random values that are evenly distributed and activated by softmax function.</td>
</tr>
<tr>
<td></td>
<td>● <strong>tanh</strong>: returns random values that are evenly distributed and activated by tanh function.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output[xx]</th>
<th>- (Required)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Operator output.</td>
</tr>
</tbody>
</table>

**NOTICE**

The number of values in **Output** must be the same as that in **Input**. Otherwise, test case generation fails.

For example, if **Input** supports two formats, the **Output** also supports two formats.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| -     | **Format** (Required)  
A string or a 1D array of strings. 
Output format. Must not be empty. 
Value range:  
- **NCHW**  
- **NHWC**  
- **ND**: any format  
- **NC1HWC0**: the 5D format defined by Huawei. **C0** is closely related to the micro-architecture, and the value is equal to a size of a cube unit, for example, **16**. **C1** divides the C dimension by **C0**, that is, **C1 = C/C0**. When the division is not exact, the last data segment is padded to **C0**.  
- **FRACTAL_Z**: format of the convolution weight.  
- **FRACTAL_NZ**: fractal format defined by Huawei. The data format of the output matrix is **NW1H1H0W0** for Cube Unit computation. The matrix is divided into (H1 x W1) fractals in column-major order, which looks like an N-shape layout. Each fractal consists of (H0 x W0) elements in row-major order, resembling a Z-shaped layout. Thus the NW1H1H0W0 data format is referred to as an Nz format. (H0 x W0) indicates the size of a fractal, as shown in the following figure. |
| -     | **OriginFormat** (Optional) Original tensor format. If this parameter is not carried, the original tensor format is retained. |

Fractal Matrix Size

![Fractal Matrix Size Figure](image-url)
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Type  | (Required) A string or a 1D array of strings. Output data type.  
|       |   • bool  
|       |   • int8  
|       |   • uint8  
|       |   • int16  
|       |   • uint16  
|       |   • int32  
|       |   • int64  
|       |   • uint32  
|       |   • uint64  
|       |   • float16  
|       |   • float  
| Shape | (Required) An int or a 1D or 2D array of ints. Output shape.  
| OriginShape | (Optional) Original tensor shape. If this field is not carried, the original tensor shape is retained.  
| Attr  | -  
| Name  | Required if attr is configured. A string. Attribute name. Must not be empty.  
| Type  | Required if attr is configured. Attribute data type. A string. Output data type.  
|       |   • bool  
|       |   • int  
|       |   • float  
|       |   • string  
|       |   • list_bool  
|       |   • list_int  
|       |   • list_float  
|       |   • list_string  
|       |   • list_list_int  


### Field Description

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value</td>
<td>Required if <code>attr</code> is configured. Must not be <code>null</code>. A string. Attribute value. Varies according to <code>type</code>. - <code>bool</code>: true/false - <code>int</code>: 10 - <code>float</code>: 1.0 - <code>string</code>: &quot;NCHW&quot; - <code>list_bool</code>: [false, true] - <code>list_int</code>: [1, 224, 224, 3] - <code>list_float</code>: [1.0, 0.0] - <code>list_string</code>: [&quot;str1&quot;, &quot;str2&quot;] - <code>list_list_int</code>: [[1, 3, 5, 7], [2, 4, 6, 8]]</td>
</tr>
</tbody>
</table>

### Expected Result Verification

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(Optional)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Script Path</td>
<td>Path of the test function for generating expected operator result.</td>
</tr>
<tr>
<td></td>
<td>Script Function</td>
<td>Name of the test function for generating expected operator result.</td>
</tr>
</tbody>
</table>

**Note**

- When `inputx.paramType` in the operator information library definition file (.ini) is set to `optional`, set `format` of `inputx` to `UNDEFINED` or `RESERVED`, and `type` to `UNDEFINED` in the generated operator test cases.

- When `dynamicShapeSupport.flag` is set to `true` in the operator information definition file (.ini), the `ShapeRange` and `TypicalShape` parameters are added to the generated operator test case. The value of `ShapeRange` is `[[1,-1]]`, indicating that the `Shape` parameter can be set to any value. If the value of the `Shape` parameter contains `-1` or `[-2]`, you need to provide a fixed shape value in the `TypicalShape` parameter for actual tests. If the value of the `Shape` parameter contains `-1`, for example, `(200, -1)`, the length of the second axis is unknown. If the value contains `[-2]`, the dimension is unknown.

- In the operator information library definition file (.ini), if a tensor implementation uses a Huawei-developed format and is different from the original format, you need to manually enter the original format and shape in `OriginFormat` and `OriginShape` to convert the format and shape into those compatible with the offline model.

  - `OriginFormat` is an array specifying the formats supported by the original operator. The array must have the same length as the `Format` array.
  
  - The value of `OriginShape` must correspond to the value of `Shape` and match the values of `Format` and `OriginFormat`. 
8.6 TBE Operator Development (PyTorch)

8.6.1 Operator Code Implementation

Call TBE DSL APIs to implement the Add operator in the `tbe/impl/add.py` file, including operator function definition, operator argument verification, and compute process implementation, scheduling, and building.

Code Template Introduction

MindStudio generates the code to `tbe/impl/add.py`.

```python
# Import the dependent Python modules.
import te.lang.cce as tbe
from te import tvm
from te.platform.fusion_manager import fusion_manager

# Operator compute function
@fusion_manager.register("add")
def add_compute(x, y, z, kernel_name="add"):
    """
    To do: Implement the operator by referring to the TBE Operator Development Guide.
    """
    res = tbe.XXX(x, y)
    return res

# Operator definition function
def add(x, y, z, kernel_name="add"):
    """
    To do: Implement the operator by referring to the TBE Operator Development Guide.
    """
    # Input placeholder
data_x = tvm.placeholder(x.get("shape"), dtype=x.get("dtype"), name="data_x")
data_y = tvm.placeholder(y.get("shape"), dtype=y.get("dtype"), name="data_y")

    # Call the operator compute function.
    res = add_compute(data_x, data_y, z, kernel_name)

    # Auto schedule
    with tvm.target.cce():
        schedule = tbe.auto_schedule(res)

    # Build
    config = {"name": kernel_name,
              "tensor_list": [data_x, data_y, res]}
    tbe.build(schedule, config)
```

- The Python modules for operator development are as follows:
  - `te.lang.cce`: imports the SDL APIs supported by TBE, including common ones such as `vmuls`, `vadds`, and `matmul`.
  - `te.tvm`: imports the code generation mechanism of TVM.

For details about the API definition, see the Python functions in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/
{arch}-linux/atc/python/site-packages/te/te/lang/cce` directory.
For details about the API definition, see the Python functions in the
\{Ascend-CANN-Toolkit installation path\}/ascend-toolkit/\{version\}/
\{arch\}-linux/atc/python/site-packages/te/te/tvm directory. For details
about the usage, visit https://docs.tvm.ai/.

- `te.platform.fusion_manager.fusion_manager`: implements automatic
  UB fusion for operators.

  For details about the API definition, see the definition
  of the `fusion_manager` function in the \{Ascend-CANN-Toolkit installation
  path\}/ascend-toolkit/\{version\}/\{arch\}-linux/atc/python/site-
  packages/te/te/platform/fusion_manager.py file.

- The template generates a compute function declaration named
  `operatorName_compute`.

  - If `From Template` or `From TensorFlow` is selected during operator
    project creation, the input and output parameters and attributes are
    automatically generated based on the prototype definition.

  - If `Empty` is selected during operator project creation, an input and an
    output without attributes are generated by default.

- The template generates with the declaration and part of the implementation
  of the definition function named `operatorName`. The sample code in the
  implementation function template contains the following functions:

  - Obtain the shape and data type of the input tensor, verify the input
    parameters, place the input tensor, and call the compute function of the
    operator for computing, scheduling, and building.

### Implementation of the Operator Definition Function

You need to implement the operator compute function based on the template
code generated by MindStudio. In addition, you need to add the verification code
of the operator inputs, outputs, or attributes to the operator definition function.
The two inputs may have different shapes. This scenario is supported by the Add
operator, but not supported by the operator compute API `te.lang.cce.vadd( )`.
Therefore, the two input shapes need to be broadcast and verified, so that faults
can be located during operator build. The modified code is as follows.

```python
from __future__ import absolute_import
from functools import reduce
from te import tvm
from te.platform.fusion_manager import fusion_manager
import te.lang.cce as tbe
from te.utils import para_check
from te.utils import shape_util

# Implement the compute logic of the Add operator.
```

---

**NOTICE**

If you remotely start MindStudio on Windows, the following code may fail to be
copied. For details about how to solve the problem, see 15.8.1 What Do I Do If
the Copied Content Cannot Be Pasted to the Editor Window Occasionally
When MindStudio Is Opened Remotely on Windows?
@fusion_manager.register("add")
def add_compute(input_x, input_y, output_z, kernel_name="add"):
    # Convert the shape to a list.
    shape_x = shape_util.shape_to_list(input_x.shape)
    shape_y = shape_util.shape_to_list(input_y.shape)

    # Assign the larger value of each dimension of shape_x and shape_y to shape_max.
    shape_x, shape_y, shape_max = shape_util.broadcast_shapes(shape_x, shape_y,
        param_name_input1="input_x",
        param_name_input2="input_y")

    shape_size = reduce(lambda x, y: x * y, shape_max[:])
    if shape_size > SHAPE_SIZE_LIMIT:
        raise RuntimeError("the shape is too large to calculate")

    # Broadcast the shape of input_x as shape_max.
    input_x = tbe.broadcast(input_x, shape_max)
    input_y = tbe.broadcast(input_y, shape_max)

    # Add input_x and input_y.
    res = tbe.vadd(input_x, input_y)

    return res

# Operator definition function
@para_check.check_op_params(para_check.REQUIRED_INPUT, para_check.REQUIRED_INPUT,
    para_check.REQUIRED_OUTPUT, para_check.KERNEL_NAME)
def add(input_x, input_y, output_z, kernel_name="add"):
    # Obtain the shape and data type of the operator input tensor.
    shape_x = input_x.get("shape")
    shape_y = input_y.get("shape")

    # Verify the operator input type.
    check_tuple = ("float16", "float32", "int32")
    input_data_type = input_x.get("dtype").lower()
    para_check.check_dtype(input_data_type, check_tuple, param_name="input_x")

    # Assign the larger value of each dimension of shape_x and shape_y to shape_max.
    shape_x, shape_y, shape_max = shape_util.broadcast_shapes(shape_x, shape_y,
        param_name_input1="input_x",
        param_name_input2="input_y")

    # If the shape length is 1, assign a value directly. If the shape length is not 1, shape needs to be tiled
    # and the last dimension needs to be removed. For the shape with the last dimension of 1 and the shape
    # without the last dimension, if their formats are the same, for example, 2 x 3 = 2 x 3 x 1, the last dimension
    # can be removed to improve the scheduling efficiency.
    if shape_x[-1] == 1 and shape_y[-1] == 1 and shape_max[-1] == 1:
        shape_x = shape_x if len(shape_x) == 1 else shape_x[:-1]
        shape_y = shape_y if len(shape_y) == 1 else shape_y[:-1]
        shape_max = shape_max if len(shape_max) == 1 else shape_max[:-1]

    # Call the placeholder API of TVM to create a placeholder for each input tensor, returning a tensor
    # object, respectively.
    data_x = tvm.placeholder(shape_x, name="data_1", dtype=input_data_type)
    data_y = tvm.placeholder(shape_y, name="data_2", dtype=input_data_type)

    # Call the compute implementation function.
    res = add_compute(data_x, data_y, output_z, kernel_name)

    # Auto schedule
    with tvm.target.cce():
        schedule = tbe.auto_schedule(res)
    # Build configuration
    config = {"name": kernel_name,
        "tensor_list": [data_x, data_y, res]}
    tbe.build(schedule, config)

1. The operator definition function declaration contains the operator input
   information, output information, and kernel name.
   def add(input_x, input_y, output_z, kernel_name="add"):
- **input_x, input_y**: two input tensors of the Add operator. Each tensor must be defined in dictionary format, including information such as shape and data type. The number of input tensors must be consistent with that defined in the **operator information definition file** (*tbe/op_info_cfg/ai_core/add.ini*).

- **output_z**: output tensor. The tensor information must be defined in dictionary format, including the shape, data type, and more. This field is reserved. The number of output tensors must be consistent with that defined in the **operator information definition file** (*tbe/op_info_cfg/ai_core/add.ini*).

- **kernel_name**: unique name of the operator in the kernel, that is, the name of the generated binary file and operator description file. The value can contain a maximum of 200 characters, which must be a combination of letters, digits, and underscores (_), beginning with a letter or underscore (_).

2. Verify the operator input and infer the output shape.

   The Add operator needs to verify shapes of the two input tensors. Only data types float16, float32, and int32 are supported. Since the two input tensors of the Add operator may have different shapes, `shape_util.broadcast_shapes()` needs to be implemented to generate and verify the broadcast shape.

3. Create placeholders of the two input tensors.

   Call the **placeholder** API of TVM to create a placeholder for each input tensor, returning a tensor object respectively.

   **NOTICE**

   `tensor_list` described in 5 is a list of tensor objects returned by calls to the **tvm.placeholder** API. Therefore, these objects cannot be replaced in subsequent computation.

4. Call the **add_compute** function.
   ```python
   res = add_compute(data_x, data_y, output_z, kernel_name)
   ```

   `data_x` and `data_y` are the tensor objects generated in 3.

   For details about how to implement the compute function, see **Implementation of the Compute Function**.

5. Implement operator scheduling and building.

   `tensor_list`:
   ```python
   "tensor_list": (data_x, data_y, res)
   ```

   The arguments are the two input tensors and one output tensor.

**Implementation of the Compute Function**

You need to customize the compute function of an operator based on the compute logic. The compute function of the Add operator is implemented as follows.

```python
@fusion_manager.register("add")
def add_compute(input_x, input_y, output_z, kernel_name="add"): 
    # Convert the shape to a list.
    shape_x = shape_util.shape_to_list(input_x.shape)
    ... # Compute logic
```

---

---

---
shape_y = shape_util.shape_to_list(input_y.shape)

# Assign the larger value of each dimension of shape_x and shape_y to shape_max.
shape_x, shape_y, shape_max = shape_util.broadcast_shapes(shape_x, shape_y,
    param_name_input1="input_x",
    param_name_input2="input_y")

shape_max = reduce(lambda x, y: x * y, shape_max[:]
if shape_size > SHAPE_SIZE_LIMIT:
raise RuntimeError("the shape is too large to calculate")

# Broadcast the shape of input_x as shape_max.
input_x = tbe.broadcast(input_x, shape_max)
input_y = tbe.broadcast(input_y, shape_max)

# Add input_x and input_y.
res = tbe.vadd(input_x, input_y)

return res

1. The **add_compute** function is declared as follows.

```python
@fusion_manager.register("add")
def add_compute(input_x, input_y, output_z, kernel_name="add")
```

The decorator `@fusion_manager.register ("add")` is required in the DSL operator development mode. It is used to support automatic UB fusion for operators during network running, so that the compute function of the current custom operator can be automatically fused with other operators in the UB according to UB fusion patterns.

Note the following:
- **input_x, input_y**: indicate the arguments passed to the compute function, that is, the placeholders for the input tensors declared in 3, including information such as the shape and data type.
- **output_z**: indicates the dictionary returned by the call to the operator API function in 1.
- **kernel_name**: indicates the operator name in the kernel.

2. Implement the compute logic of the Add operator.

The Add operator requires that shapes of the two tensors to be added be the same. Therefore, the `tbe.broadcast` API is called to broadcast the two input tensors to the same shape, and then the `tbe.vadd` API is called to add the input tensors and return the result tensor.

### Verifying Operator Building

**Step 1** At the bottom of the Python file of the operator, add the **main** function to call the operator, and build the operator implementation file by using MindStudio for simple syntax verification of the single-operator code. A code example is as follows.

```python
if __name__ == '__main__':
    input_output_dict = {"shape": (5, 6, 7),"format": "ND","ori_shape": (5, 6, 7),"ori_format": "ND", "dtype": "float16")
    add(input_output_dict, input_output_dict, input_output_dict, kernel_name="add")
```

**Step 2** Right-click `tbe_impl/add.py` and choose **Run 'add'** from the shortcut menu to build the operator.

If no build error is reported and a `kernel_meta` folder containing the following files is generated in the `tbe/impl` directory, the operator code can be built and run properly.
- Binary file of the operator (.o)
- Operator description file (.json): defines operator attributes and resources required for running the operator.

--- End

8.6.2 Operator Prototype Definition

Go to the **op_proto/** directory, write the IR implementation files **add.h** and **add.cc**, and register the operator with the operator prototype library. During network execution, GE calls the verification API of the operator prototype library to verify operator arguments. If the verification passes, GE infers the output **shape** and **dtype** of each node by calling the inference function of the operator prototype library and allocates static memory for the result tensor.

Implementing add.h

MindStudio generates the operator registration code to the **add.h** file. You can modify the code as required. The prototype definition of the Add operator is as follows.

```c
#ifndef GE_OPS_OP_PROTO_ADD_H_
define GE_OPS_OP_PROTO_ADD_H_
#include "graph/operator_reg.h"
namespace ge {
REG_OP(Add)
.INPUT(x1,
  TensorType([DT_FLOAT, DT_INT32, DT_INT64, DT_FLOAT16, DT_INT16, DT_INT8, DT_UINT8,
             DT_DOUBLE, DT_COMPLEX128,
             DT_COMPLEX64, DT_STRING]))
.INPUT(x2,
  TensorType([DT_FLOAT, DT_INT32, DT_INT64, DT_FLOAT16, DT_INT16, DT_INT8, DT_UINT8,
             DT_DOUBLE, DT_COMPLEX128,
             DT_COMPLEX64, DT_STRING]))
.OUTPUT(y,
  TensorType([DT_FLOAT, DT_INT32, DT_INT64, DT_FLOAT16, DT_INT16, DT_INT8, DT_UINT8,
              DT_DOUBLE, DT_COMPLEX128,
              DT_COMPLEX64, DT_STRING]))
.OP_END_FACTORY_REG(Add)
}
#endif //GE_OPS_OP_PROTO_ADD_H_
```

add.cc Implementation

You need to implement the **InferShape** and **Verify** functions in **add.cc**.

- The **Verify** function, that is, **IMPLEMT_VERIFIER(Add, AddVerify)** in the following sample code, is used to check whether the data types of the two inputs of the Add operator are the same.

- The **InferShape** function that is, **IMPLEMT_COMMON_INFERFUNC(AddInferShape)** in the following sample code, is used to infer the output tensor description of the operator. In this way, the memory can be statically allocated for all tensors during network execution, avoiding the overhead caused by dynamic memory allocation.

The implementation code of the **add.cc** file is as follows.

```c
#include "./add.h" // IR registration header file
#include <vector>   // Vector templates can be used and APIs of class vector can be called.
```
#include <string>        // The string class is part of the C++ standard library. With the header file included, string objects can be constructed and string APIs can be called.

namespace ge {
  bool InferShapeAndTypeAdd(Operator& op,
   const string& inputName1,
   const string& inputName2, const string& outputName)
  {
    TensorDesc vOutputDesc = op.GetOutputDescByName(outputName.c_str());
    DataType inputDtype = op.GetInputDescByName(inputName1.c_str()).GetDataType();
    Format inputFormat = op.GetInputDescByName(inputName1.c_str()).GetFormat();
    // Exchange the shape dimensions.
    ge::Shape shapeX = op.GetInputDescByName(inputName1.c_str()).GetShape();
    ge::Shape shapeY = op.GetInputDescByName(inputName2.c_str()).GetShape();
    std::vector<int64_t> dimsX = shapeX.GetDims();
    std::vector<int64_t> dimsY = shapeY.GetDims();
    if (dimsX.size() < dimsY.size()) {
      std::vector<int64_t> dimsTmp = dimsX;
      dimsX = dimsY;
      dimsY = dimsTmp;
    }
    // Pad the smaller shape dimension with 1.
    if (dimsX.size() != dimsY.size()) {
      int dec = dimsX.size() - dimsY.size();
      for (int i = 0; i < dec; i++) {
        dimsY.insert(dimsY.begin(), (int64_t)1);
      }
    }
    // Set the output shape dimension.
    std::vector<int64_t> dimVec;
    for (size_t t = 0; t < dimsX.size(); t++) {
      if ((dimsX[t] != dimsY[t]) || (dimsX[t] != 1) || (dimsY[t] != 1)) {
        return false;
      }
      int64_t dims = dimsX[t] > dimsY[t] ? dimsX[t] : dimsY[t];
      dimVec.push_back(dims);
    }
    ge::Shape outputShape = ge::Shape(dimVec);
    vOutputDesc.SetShape(outputShape);
    vOutputDesc.SetDataType(inputDtype);
    vOutputDesc.SetFormat(inputFormat);
    op.UpdateOutputDesc(outputName.c_str(), vOutputDesc);
    return true;
  }

  //----------------Add-------------------
  IMPLEMENT_VERIFIER(Add, AddVerify)
  {
    if (op.GetInputDescByName("x1").GetDataType() != op.GetInputDescByName("x2").GetDataType()) {
      return GRAPH_FAILED;
    }
    return GRAPH_SUCCESS;
  }

  // Obtain the processing function of the output tensor description.
  IMPLEMENT_COMMON_INFERFUNC(AddInferShape)
  {
    if(InferShapeAndTypeAdd(op, "x1", "x2", "y")) {
      return GRAPH_SUCCESS;
    }
    return GRAPH_FAILED;
  }
}
8.6.3 Operator Information Definition

You need to register the operator information to the operator information library by configuring the operator information file. The operator information library mainly reflects the physical implementation restrictions of an operator on the Ascend AI Processor, including the input and output data types, formats, and input shape of the operator. During network execution, FE performs basic verification based on the operator information in the operator information library, and determines whether to insert a proper conversion node for the operator. It also finds the corresponding operator implementation file based on the information in the operator information library and builds the operator binary file for execution.

Go to the tbe/op_info_cfg/ai_core/soc_version directory and configure the operator information library file add.ini.

**soc_version** indicates the current Ascend AI Processor version, for example, ascend310 and ascend910.

You need to modify the add.ini file automatically generated by MindStudio. The following shows the operator information definition of the modified Add operator.

```
[Add]
input0.name=x1
input0.dtype=float16,float16,float16,float16,float,float,float,float,int32,int32,int32,int32
input0.format=NCHW,NC1HWC0,NHWC,ND,NCHW,NC1HWC0,NHWC,ND,NCHW,NC1HWC0,NHWC,ND
input0.shape=all
input0.paramType=required
input1.name=x2
input1.dtype=float16,float16,float16,float16,float,float,float,float,int32,int32,int32,int32
input1.format=NCHW,NC1HWC0,NHWC,ND,NCHW,NC1HWC0,NHWC,ND,NCHW,NC1HWC0,NHWC,ND
input1.shape=all
input1.paramType=required
output0.name=y
output0.dtype=float16,float16,float16,float16,float,float,float,float,int32,int32,int32,int32
output0.format=NCHW,NC1HWC0,NHWC,ND,NCHW,NC1HWC0,NHWC,ND,NCHW,NC1HWC0,NHWC,ND
output0.shape=all
output0.paramType=required
opFile.value=add
opInterface.value=add
```

**Table 8-16** describes the configuration options. The table lists only common configuration options of the operator information library. For details about more configuration options, see:

"Operator Information Library Definition" in TBE Custom Operator Development Guide.
Table 8-16 Operator information definition of the Add operator

<table>
<thead>
<tr>
<th>Configuration Option</th>
<th>Add Operator Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[OpType]</td>
<td>[Add]</td>
<td>Operator type, which is included in square brackets to mark the start of operator information. The operator type is Add according to 8.3 Operator Analysis.</td>
</tr>
<tr>
<td>input0.name</td>
<td>x1</td>
<td>Name of the first input tensor of the Add operator. According to 8.3 Operator Analysis, the name of the first input tensor is x1. Therefore, set this parameter to x1, which must be the same as the name in Operator Prototype Definition.</td>
</tr>
<tr>
<td>input0.dtype</td>
<td>input0.dtype=float16, float16, float, float16, float, float, float, float, int32, int32</td>
<td>Data type and data layout format supported by the input tensor. According to 8.3 Operator Analysis, the Add operator supports input types (float16, float32, and int32) and data layout formats (NCHW, NC1HWC0, NHWC, and ND). Note: If the input tensor supports multiple dtype and format combinations, dtype and format must be configured in pairs and in sequence. List all the combinations supported by the operator and separate them with commas (,).</td>
</tr>
<tr>
<td>input0.format</td>
<td>input0.format=NC1HWC0, NHWC, ND</td>
<td></td>
</tr>
<tr>
<td>input0.shape</td>
<td>all</td>
<td>Shape of the input tensor.</td>
</tr>
<tr>
<td>input0.paramType</td>
<td>required</td>
<td>Type of the input tensor.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>dynamic</strong>: indicates that the number of inputs is subject to change, which can be one or more.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>optional</strong>: indicates that there could be one input or no input at all.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>required</strong>: indicates that there is one and only one input. Set input0 of the Add operator to required.</td>
</tr>
</tbody>
</table>
### The Add operator

The Add operator has two inputs, while the generated configuration template has only one input. Therefore, you need to add the configuration of `input1`. Name of the second input tensor of the Add operator. According to **8.3 Operator Analysis**, the name of the second input tensor is `x2`. Therefore, set this parameter to `x2`, which must be the same as the name in **8.5.2 Operator Prototype Definition**.

<table>
<thead>
<tr>
<th>parameter</th>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>input1.name</code></td>
<td><code>x2</code></td>
<td>The Add operator has two inputs, while the generated configuration template has only one input. Therefore, you need to add the configuration of <code>input1</code>. Name of the second input tensor of the Add operator. According to <strong>8.3 Operator Analysis</strong>, the name of the second input tensor is <code>x2</code>. Therefore, set this parameter to <code>x2</code>, which must be the same as the name in <strong>8.5.2 Operator Prototype Definition</strong>.</td>
</tr>
<tr>
<td><code>input1.dtype</code></td>
<td><code>float16,float16,float16,float16,float16,int32,int32,int32,int32,int32</code></td>
<td>Data type and data layout format supported by the input tensor. According to <strong>8.3 Operator Analysis</strong>, the Add operator supports input types (float16, float32, and int32) and data layout formats (NCHW, NC1HWC0, NHWC, and ND). Note: If the input tensor supports multiple dtype and format combinations, dtype and format must be configured in pairs and in sequence. List all the combinations supported by the operator and separate them with commas (,).</td>
</tr>
<tr>
<td><code>input1.format</code></td>
<td><code>NC1HWC0,NHWC,ND</code></td>
<td>Shape of the input tensor.</td>
</tr>
<tr>
<td><code>input1.shape</code></td>
<td><code>all</code></td>
<td>Shape of the input tensor.</td>
</tr>
<tr>
<td><code>input1.paramType</code></td>
<td><code>required</code></td>
<td>Type of the input tensor. <strong>dynamic</strong>: indicates that the number of inputs is subject to change, which can be one or more. <strong>optional</strong>: indicates that there could be one input or no input at all. <strong>required</strong>: indicates that there is one and only one input. Set input1 of the Add operator to <code>required</code>.</td>
</tr>
</tbody>
</table>

---

---

---

---

---
<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>output0.name</td>
<td>y</td>
<td>Name of the output tensor of the Add operator. According to Section 8.3 Operator Analysis, the output name of the operator is ( y ). Therefore, set this parameter to ( y ), which must be the same as the name in Section 8.5.2 Operator Prototype Definition.</td>
</tr>
<tr>
<td>output0.dtype</td>
<td>output0.dtype=float16,float16,float16,float16,float16,float16,float16,float16,int32,int32,int32</td>
<td>Data type and data layout format supported by the output tensor. According to Section 8.3 Operator Analysis, the Add operator supports output types ( \text{float16}, \text{float32}, \text{and int32} ) and data layout formats ( \text{NCHW}, \text{NC1HWC0}, \text{NHWC}, \text{ND} ). Note: If the output tensor supports multiple dtype and format combinations, dtype and format must be configured in pairs and in sequence. List all the combinations supported by the operator and separate them with commas (,).</td>
</tr>
<tr>
<td>output0.format</td>
<td>output0.format=NCHW,NC1HWC0,NHWC,ND,NCHW,NC1HWC0,NHWC,ND</td>
<td>Shape of the output tensor.</td>
</tr>
</tbody>
</table>
| output0.shape     | all   | Type of the output tensor:  
  - **dynamic**: indicates that the number of outputs is subject to change, which can be one or more.  
  - **optional**: indicates that there could be one output or no output at all.  
  - **required**: indicates that there is one and only one output.  
  Set output0 of the Add operator to **required**. |
| output0.paramType | required | Name of the operator code implementation file. According to Step 4, the name of the operator implementation code complies with the rules of converting OpType to the code implementation name. Therefore, this parameter does not need to be specified. |
| opFile.value      | add   | Name of the operator code implementation file. According to Step 4, the name of the operator implementation code complies with the rules of converting OpType to the code implementation name. Therefore, this parameter does not need to be specified. |
**8.6.4 UT**

### 8.6.4.1 Overview

MindStudio provides an upgraded UT solution based on the GTest framework, simplifying UT case development.

UT is one of the methods to verify a single-operator, which aims to:

- Test correctness of operator code and verify consistency between the desired and actual inputs and outputs.
- Ensure the operator code coverage. The selected scenario combination must cover all branches of the operator code (the coverage should reach 100%), to reduce the failure rate for building operator code in different scenarios.

**NOTE**

For details about definition of the test classes, see the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/toolkit/python/site-packages/op_test_frame/ut/op_ut.py` directory.

### 8.6.4.2 APIs

#### 8.6.4.2.1 OpUT

**Overview**

Functions as a base class for the UT framework. It provides the following two APIs to define and run the test cases.

- `add_case`
- `add_precision_case`

**Definition of the OpUT Test Class**

- **Prototype**
  
  `OpUT(op_type, op_module_name=None, op_func_name=None)`

- **Parameters**
op_type: operator type.

op_module_name: operator module name, that is, the name and path of the operator implementation file, for example, implAdd (the file path is impl/add.py). Defaults to None. The argument is generated automatically based on op_type. For example, for a BiasAdd operator, the generated module name is impl.bias_add. For details about how the argument of op_module_name is generated based on op_type, see Step 4.

op_func_name: operator function name in the operator implementation file. Defaults to None. The argument is generated automatically based on op_type. For details about how the argument of op_func_name is generated based on op_type, see Step 4. For example, for an Add operator, the corresponding API must exist in impl/add.py. The API definition is as follows.

```python
@check_op_params(REQUIRED_INPUT, REQUIRED_INPUT, REQUIRED_OUTPUT, KERNEL_NAME)
def add(input_x, input_y, output_z, kernel_name="add"):
```

### add_case

- **Prototype**
- **OpUT.add_case(support_soc=None, case=None)**

- **Description**
  Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an .o file.

- **Parameters**
  - **support_soc**: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the `atc/data/platform_config` directory in the ATC installation path. support_soc can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to all or None, all SoCs are supported.

  - **case**: a dictionary. An example is as follows.

    ```json
    {
    "params": [
    {
    "shape": (32, 64),
    "ori_shape": (32, 64),
    "format": "ND",
    "ori_format": (32, 64),
    "dtype": "float16"
    },
    {
    "shape": (32, 64),
    "ori_shape": (32, 64),
    "format": "ND",
    "ori_format": (32, 64),
    "dtype": "float16"
    }
    ],
    "case_name": "test_add_case_1",
    "expect": "success"
    }
    ```

  The key fields in the dictionary include:

  - **params**: This field is passed through to the operator API during test case running.
- **case_name**: (optional) name of a test case. If it is not specified, the test framework generates the argument of `case_name` automatically in the following format:
  
  ```
test_[op_type]_auto_case_name_[case_count]
  ```
  
  Example: `test_Add_auto_case_name_1`

- **expect**: expected result. Defaults to `success`. It can also be set to `RuntimeError`, indicating an expected exception.

### `add_precision_case`

- **Prototype**
  
  ```
  OpUT.add_precision_case(support_soc=None, case)
  ```

- **Description**
  Adds test cases for operator build and precision test.

- **Parameters**
  - **support_soc**: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the `atc/data/platform_conf` directory in the ATC installation path. `support_soc` can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to `all` or `None`, all SoCs are supported.
  
  - **case**: a dictionary. An example is as follows.
    ```
    { 
      "params": [ 
        { 
          "shape": (32, 64),
          "ori_shape": (32, 64),
          "format": "ND",
          "ori_format": "ND",
          "dtype": "float16",
          "param_type": "input"
        }, 
        { 
          "shape": (32, 64),
          "ori_shape": (32, 64),
          "format": "ND",
          "ori_format": "ND",
          "dtype": "float16",
          "param_type": "output"
        }
      ],
      "case_name": "test_add_case_1",
      "calc_expect_func": np_add # A function
      "precision_standard": precision_info.PrecisionStandard(0.001, 0.001) # Optional field
    }
    ```

  The **key** fields in the dictionary include:

  - **params**: This field is passed through to the operator API during test case running.
  
  - **case_name**: (optional) name of a test case. If it is not specified, the test framework generates the argument of `case_name` automatically in the following format:
    
    ```
test_[op_type]_auto_case_name_[case_count]
    ```
    
    Example: `test_Add_auto_case_name_1`
calc_expect_func: expected result generation function.

cpyrecision_standard: user-defined precision standard. If this field is
not set, the following default precision is used for comparison with
the expected data:
  - float16: dual-0.1% error limit, that is, the error ratio is within
    0.1% and the relative error is within 0.1%.
  - float32: dual-0.01% error limit, that is, the error ratio is within
    0.01% and the relative error is within 0.01%.

● Examples

Example 1
from op_test_frame.ut import OpUT
# ut_case is a keyword of the UT framework and cannot be modified.

test_case = OpUT("Add", "impl.add", "add")

def np_add(x1, x2, y):
  y = (x1.get("value") + x2.get("value"), )
  return y

test_case.add_precision_case(case={
  "params": [
    {
      "shape": (32, 64),
      "ori_shape": (32, 64),
      "format": "ND",
      "ori_format": "ND",
      "dtype": "float16",
      "param_type": "input"
    },
    {
      "shape": (32, 64),
      "ori_shape": (32, 64),
      "format": "ND",
      "ori_format": "ND",
      "dtype": "float16",
      "param_type": "input"
    },
    {
      "shape": (32, 64),
      "ori_shape": (32, 64),
      "format": "ND",
      "ori_format": "ND",
      "dtype": "float16",
      "param_type": "output"
    }
  ],
  "case_name": "test_add_case_1",
  "calc_expect_func": np_add
})

if __name__ == '__main__':
  test_case.run("Ascend910",None,"ca",/home/allan/Ascend/toolkit/tools/simulator")

Note:

ut_case.run: For details, see run API.

In the preceding example, the values of the operator inputs are not set. By
default, the framework calls np.random.uniform(value_range,
size=shape).astype(dtype) to automatically generate the input data for each
input.
The value_range is defaulted to [0.1, 1.0]. The values of shape and dtype are consistent with those set in params. value_range can also be specified as follows.

```json
{
    "shape": (32, 64),
    "ori_shape": (32, 64),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float16",
    "param_type": "input",
    "value_range": [2.0, 3.0]
}
```

You can also specify input values as follows.

```json
{
    "shape": (32, 64),
    "ori_shape": (32, 64),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float16",
    "param_type": "input",
    "value": np.zeros((32, 64), np.float16)
}
```

**Example 2**

The following is an example of the user-defined precision standard.

```python
from op_test_frame.common import precision_info
from op_test_frame.ut import OpUT

ut_case = OpUT("Add")
ut_case.add_precision_case(case={
    "params": [
        {
            "shape": (32, 64),
            "ori_shape": (32, 64),
            "format": "ND",
            "ori_format": "ND",
            "dtype": "float16",
            "param_type": "input"
        },
        {
            "shape": (32, 64),
            "ori_shape": (32, 64),
            "format": "ND",
            "ori_format": "ND",
            "dtype": "float16",
            "param_type": "input"
        },
        {
            "shape": (32, 64),
            "ori_shape": (32, 64),
            "format": "ND",
            "ori_format": "ND",
            "dtype": "float16",
            "param_type": "output"
        }
    ],
    "case_name": "test_add_case_1",
    "calc_expect_func": np_add,
    "precision_standard": precision_info.PrecisionStandard(0.1, 0.1) # Uses the standard precision of one tenth and compares it with the expected data.
})
```

run API

- Prototype
OpUT.run(soc, case_name=None, simulator_mode=None, simulator_lib_path=None)

- **Description**
  Runs test cases.

- **Parameters**
  - **soc**: the the Ascend AI Processor to run test cases. The value is the name of the corresponding .ini file in the 
    `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/atc/data/platform_config`
directory. `support_soc` can be a string or a tuple or list of strings (indicating multiple SoCs are supported).
  - **case_name**: the test case to run. Set this parameter to the value of `case_name` in `add_case` or `add_precision_case`.
  - **simulator_lib_path**: path of the simulator libraries.
    The path structure is as follows.
    
    ```
    simulator_lib_path/
    Ascend910/
    lib/
    libpv_model.so
    ...
    Ascend310/
    lib/
    libpv_model.so
    ...
    ```

**NOTICE**

If MindStudio is used to run UT cases, OpUT.run does not need to be called manually.

### 8.6.4.2.2 BroadcastOpUT

**Overview**

Inherits the capabilities of OpUT.

BroadcastOpUT defines test cases for Broadcast operators with dual inputs and single output, for example, the Add and Mul operators. It also provides more convenient APIs for these operators. For example, when creating the test case file for operator build, you do not need to set the format in some simple scenarios.

**Definition of the BroadcastOpUT Test Class**

- **Prototype**
  
  ```
  BroadcastOpUT(op_type, op_module_name=None, op_func_name=None)
  ```

- **Parameters**
  - **op_type**: operator type.
  - **op_module_name**: operator module name, that is, the name and path of the operator implementation file, for example, impl_add (the file path is impl/add.py). Defaults to None. The argument is generated automatically based on op_type. For example, for a BiasAdd operator, the
generated module name is **impl.bias_add**. For details about how the argument of **op_module_name** is generated based on **op_type**, see Step 4.

- **op_func_name**: operator function name in the operator implementation file. Defaults to None. The argument is generated automatically based on **op_type**. For details about how the argument of **op_func_name** is generated based on **op_type**, see Step 4. For example, for an Add operator, the corresponding API must exist in **impl/add.py**. The API definition is as follows.

```python
@check_op_params(REQUIRED_INPUT, REQUIRED_INPUT, REQUIRED_OUTPUT, KERNEL_NAME)
def add(x, y, out, kernel_name="add"):
```

**add_broadcast_case**

- **Prototype**

  ```python
  BroadcastOpUT.add_broadcast_case(self, soc, input_1_info, input_2_info, output_info=None, expect=op_status.SUCCESS, case_name=None)
  ```

- **Description**

  Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an .o file.

- **Parameters**

  - **soc**: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the `atc/data/platform_config` directory in the ATC installation path. **support_soc** can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to all or None, all SoCs are supported.

  - **input_1_info**: information about the first input of the operator, coming in either of the following format:
    - [dtype, shape, format, ori_shape, ori_format]
    - [dtype, shape, format]: In this format, the values of ori_shape and ori_format are the same as those of shape and format.

  - **input_2_info**: information about the second input of the operator, with the same information of **input_1_info**.

  - **output_info**: defaults to None. This parameter does not need to be set.

  - **expect**: expected build result. Defaults to op_status.SUCCESS. It can also be set to RuntimeError, indicating an expected exception.

  - **case_name**: defaults to None. The test framework automatically generates the argument of **case_name**.

**Example**

```python
ut_case.add_broadcast_case("all", ["float16", (32, 32), "ND"], ["float16", (32, 32), "ND"], ["float16", (32, 32), "ND"];

ut_case.add_broadcast_case("all", ["float16", (32, 32), "ND"], ["float16", (32, 32), "ND"], ["float16", (32, 32), "ND"];

# Expected exception UT case
ut_case.add_broadcast_case("all", ["float16", (31, 32), "ND"], ["float16", (32, 32), "ND"],
expect=RuntimeError)
```
add_broadcast_case_simple

- **Prototype**
  
  `BroadcastOpUT.add_broadcast_case_simple(self, soc, dtypes, shape1, shape2, expect=op_status.SUCCESS, case_name=None)`

- **Description**

  Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an .o file. This API is simpler than `add_broadcast_case`.

- **Parameters**
  - `soc`: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/atc/data/platform_config` directory. `support_soc` can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to `all` or `None`, all SoCs are supported.
  - `dtypes`: types of the data to be tested. If multiple data types are set, multiple test cases are added at a time.
  - `shape1`: shape of the first input of the operator.
  - `shape2`: shape of the second input of the operator.
  - `expect`: expected build result. Defaults to `op_status.SUCCESS`. It can also be set to `RuntimeError`, indicating an expected exception.
  - `case_name`: defaults to `None`. The test framework automatically generates the argument of `case_name`.

Compared with the `add_broadcast_case` API, this API sets the formats of all inputs to `ND`.

**Example**

```python
ut_case.add_broadcast_case_simple(['Ascend910', 'Ascend310'], ['float16', 'float32'], (32, 32), (32, 32))
```

The preceding test case implements the same function as calling the `add_case` API.

```python
ut_case.add_case(support_soc=['Ascend910', 'Ascend310'], case={
    "params": [
        {
            "shape": (32, 32),
            "ori_shape": (32, 32),
            "format": "ND",
            "ori_format": "ND",
            "dtype": "float16"
        },
        {
            "shape": (32, 32),
            "ori_shape": (32, 32),
            "format": "ND",
            "ori_format": "ND",
            "dtype": "float16"
        },
        {
            "shape": (32, 32),
            "ori_shape": (32, 32),
            "format": "ND",
            "ori_format": "ND",
            "dtype": "float16"
        }
    ]
})
```

```python
ut_case.add_case(support_soc=['Ascend910', 'Ascend310'], case={
    "params": [
        {
            "shape": (32, 32),
            "ori_shape": (32, 32),
            "format": "ND",
            "ori_format": "ND",
            "dtype": "float16"
        }
    ]
})
```
8.6.4.2.3 ElementwiseOpUT

Overview

Inherits the capabilities of OpUT.

ElementwiseOpUT defines test cases for element-wise operators with single input and single output, for example, the Abs and Square operators.

It also provides more convenient APIs for these operators. For example, when creating the test case file for operator build, you do not need to set the format in some simple scenarios.

Definition of the ElementwiseOpUT Test Class

- Prototype

  ElementwiseOpUT(op_type, op_module_name=None, op_func_name=None)

- Parameters

  - **op_type**: operator type.
  
  - **op_module_name**: operator module name, that is, the name and path of the operator implementation file, for example, `impl.add` (the file path is `impl/add.py`). Defaults to **None**. The argument is generated automatically based on **op_type**. For example, for a MaximumGrad operator, the generated module name is `impl.maximum_grad`. For details about how the argument of `op_module_name` is generated based on **op_type**, see **Step 4**.

  - **op_func_name**: operator function name in the operator implementation file. Defaults to **None**. The argument is generated automatically based on **op_type**. For details about how the argument of `op_func_name` is generated based on **op_type**, see **Step 4**. For example, for an Add operator, the corresponding API must exist in `impl/add.py`. The API definition is as follows.

```python
@check_op_params(REQUIRED_INPUT, REQUIRED_INPUT, REQUIRED_OUTPUT, KERNEL_NAME)
def add(input_x, input_y, output_z, kernel_name="add"):
```
add_elewise_case

- Prototype

`BroadcastOpUT.add_elewise_case(self, soc, param_info, expect=op_status.SUCCESS, case_name=None)`

- Description

Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an .o file.

- Parameters

- `soc`: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/arch-linux/atc/data/platform_config` directory. `support_soc` can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to `all` or `None`, all SoCs are supported.

- `param_info`: information about the operator input, coming in either of the following format:

  - `[dtype, shape, format, ori_shape, ori_format]`
  - `[dtype, shape, format]`: In this format, the values of `ori_shape` and `ori_format` are the same as those of `shape` and `format`.

- `expect`: expected build result. Defaults to `op_status.SUCCESS`. It can also be set to `RuntimeError`, indicating an expected exception.

- `case_name`: defaults to `None`. The test framework automatically generates the argument of `case_name`.

Example

```python
ut_case.add_elewise_case("Ascend910", ["float16", (32, 32), "ND"])
```

The preceding test case implements the same function as calling the `add_case` API.

```json
ut_case.add_case(support_soc="Ascend910", case={
  "params": [
    { "shape": (32, 32),  
      "ori_shape": (32, 32),  
      "format": "ND",  
      "ori_format": "ND",  
      "dtype": "float16"  
    },  
    { "shape": (32, 32),  
      "ori_shape": (32, 32),  
      "format": "ND",  
      "ori_format": "ND",  
      "dtype": "float16"  
    }
  ]
})
```

add_elewise_case_simple

- Prototype

`BroadcastOpUT.add_elewise_case_simple(self, soc, dtypes, shape, expect=op_status.SUCCESS, case_name=None)`

- Description

Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an .o file.
- **Parameters**
  - **soc**: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/ {version}/{arch}-linux/atc/data/platform_config` directory. `support_soc` can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to **all** or **None**, all SoCs are supported.
  - **dtypes**: types of the data to be tested. If multiple data types are set, multiple test cases are added at a time.
  - **shape**: shape of the operator input.
  - **expect**: expected build result. Defaults to **op_status.SUCCESS**. It can also be set to **RuntimeError**, indicating an expected exception.
  - **case_name**: defaults to **None**. The test framework automatically generates the argument of **case_name**.

Compared with the **add_elewise_case** API, this API sets the formats of all inputs to **ND**.

**Example**

```python
ut_case.add_elewise_case_simple("Ascend910", ["float16", "float32"], [32, 32])
```

The preceding test case implements the same function as calling the **add_case** API.

```python
ut_case.add_case(support_soc="Ascend910", case={
  "params": [{
    "shape": (32, 32),
    "ori_shape": (32, 32),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float16"
  }, {
    "shape": (32, 32),
    "ori_shape": (32, 32),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float16"
  }]
})
```

```python
ut_case.add_case(support_soc="Ascend910", case={
  "params": [{
    "shape": (32, 32),
    "ori_shape": (32, 32),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float32"
  }, {
    "shape": (32, 32),
    "ori_shape": (32, 32),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float32"
  }]
})
```

### 8.6.4.2.4 ReduceOpUT

#### Overview

Inherits the capabilities of **OpUT**.
ReduceOpUT defines test cases for Reduce operators, for example, the ReduceSum and ReduceMean operators. It also provides more convenient APIs for these operators. For example, when creating the test case file for operator build, you do not need to set the format in some simple scenarios.

Definition of the ReduceOpUT Test Class

- **Prototype**
  
  `ReduceOpUT(op_type, op_module_name=None, op_func_name=None)`

- **Parameters**
  
  - **op_type**: operator type.
  - **op_module_name**: operator module name, that is, the name and path of the operator implementation file, for example, `impl/add` (the file path is `impl/add.py`). Defaults to `None`. The argument is generated automatically based on **op_type**. For example, for a ReduceSum operator, the generated module name is `impl.reduce_sum`. For details about how the argument of **op_module_name** is generated based on **op_type**, see Step 4.
  
  - **op_func_name**: operator function name in the operator implementation file. Defaults to `None`. The argument is generated automatically based on **op_type**. For details about how the argument of **op_func_name** is generated based on **op_type**, see Step 4. For example, for an Add operator, the corresponding API must exist in `impl/add.py`. The API definition is as follows.

```python
@check_op_params(REQUIRED_INPUT, REQUIRED_INPUT, REQUIRED_OUTPUT, KERNEL_NAME)
def add(input_x, input_y, output_z, kernel_name="add"):
    ...
```

- **add_reduce_case**
  
  - **Prototype**
    
    `BroadcastOpUT.add_reduce_case(self, soc, input_info, axes, keep_dim=False, expect=op_status.SUCCESS, case_name=None)`

  - **Description**
    
    Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an .o file.

  - **Parameters**
    
    - **soc**: used to test whether the test case file supports the corresponding Ascend AI Processor. The value is the name of the corresponding .ini file in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/atc/data/platform_config` directory. **support_soc** can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to **all** or **None**, all SoCs are supported.
    
    - **input_info**: information about the operator input, coming in either of the following format:
      
      - `[dtype, shape, format, ori_shape, ori_format]`
      - `[dtype, shape, format]`: In this format, the values of **ori_shape** and **ori_format** are the same as those of **shape** and **format**.
      
      - **axes**: dimensions to reduce.
If `keep_dims` is `True`, retains reduced dimensions with length 1. Either `True` or `False`.

- `expect`: expected build result. Defaults to `op_status.SUCCESS`. It can also be set to `RuntimeError`, indicating an expected exception.

- `case_name`: defaults to `None`. The test framework automatically generates the argument of `case_name`.

Example

```python
ut_case.add_reduce_case("Ascend910", ["float16", (32, 32), "ND"], [0,], False)
```

The preceding test case implements the same function as calling the `add_case` API.

```python
ut_case.add_case(support_soc="Ascend910", case={
    "params": [[
        "shape": (32, 32),
        "ori_shape": (32, 32),
        "format": "ND",
        "ori_format": "ND",
        "dtype": "float16"
    ], [
        "shape": (32,)
        "ori_shape": (32,)
        "format": "ND",
        "ori_format": "ND",
        "dtype": "float16"
    ]], [0,], False)
})
```

`add_reduce_case_simple`

- **Prototype**

  ```python
  BroadcastOpUT.add_reduce_case_simple(self, soc, dtypes, shape, axes,
  keep_dim=False, expect=op_status.SUCCESS, case_name=None)
  ```

- **Description**

  Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an .o file.

- **Parameters**
  - `soc`: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/atc/data/platform_config` directory. `support_soc` can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to `all` or `None`, all SoCs are supported.
  - `dtypes`: types of the data to be tested. If multiple data types are set, multiple test cases are added at a time.
  - `shape`: shape of the operator input.
  - `axes`: dimensions to reduce.
  - `keep_dim`: If `True`, retains reduced dimensions with length 1. Either `True` or `False`.
  - `expect`: expected build result. Defaults to `op_status.SUCCESS`. It can also be set to `RuntimeError`, indicating an expected exception.
  - `case_name`: defaults to `None`. The test framework automatically generates the argument of `case_name`. 


Compared with the **add_reduce_case** API, this API sets the formats of all inputs to **ND**.

**Example**

```python
ut_case.add_reduce_case_simple("Ascend910", ["float16", "float32"], [32, 32], [1], True)
```

The preceding test case implements the same function as calling the **add_case** API.

```python
ut_case.add_case(support_soc="Ascend910", case={
  "params": [{
    "shape": (32, 32),
    "ori_shape": (32, 32),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float16"
  }, {
    "shape": (32, 1),
    "ori_shape": (32, 1),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float16"
  }], [1], True])
```

8.6.4.3 Creating and Running UT Cases

**Prerequisites**

The custom operator has been developed, including the operator implementation code and operator prototype definition. For details, see **8.5.1 Operator Code Implementation**, **8.5.2 Operator Prototype Definition**, and **8.5.3 Operator Information Definition**.

**NOTE**

CentOS7.8 ARM containers do not support the UT of the operator implementation code.

**Procedure**

**Step 1** Create UT cases.

1. A UT case file can be created in any of the following ways:

   Right-click the root directory of the operator project and choose **New Cases > TBE UT Case** from the shortcut menu.
If UT cases of the operator exist, right-click the testcases directory or testcases > ut, and choose New Cases > TBE UT Case from the shortcut menu to create UT cases.

2. In the Create UT for an Operator window, choose the target operator, and click OK, as shown in the following figure.

![Operator Name](image)

**NOTE**

If the UT cases of the operator already exist, the message “testcases/ut/ops_test/xx already exists. Do you want to overwrite?” is displayed.

You can click Overwrite or Cancel.

After the creation is complete, the testcases folder is generated in the root directory of the operator project. The directory structure is as follows.

```
MyOperator            // Root directory of the project
│   ├──  testcases
│   │   ├── libs                  // GTest framework. It is a third-party dependency and can be ignored.
│   │   ├──  ut
│   │   │   ├── ops_test
│   │   │   │   ├── add
│   │   │   │   │   ├── CMakeLists.txt        // Build script
│   │   │   │   │   ├── test_add_impl.py      // Test case file for the operator implementation code
│   │   │   │   │   ├── test_add_proto.cc    // Test case file for the operator prototype definition code
│   │   │   │   ├── CMakeLists.txt             // Build script
│   │   │   │   ├── test_main.cc              // Main entry for test case calling
│   │   │   └ CMakeLists.txt
```

**Step 2** For the CentOS container, NeoKylin OS, and Kylin OS, add the following information in bold to the ./testcases/ops_test/add/CMakeLists.txt file. For other OSs, skip this step.

```cmake
add_definitions(-D_GLIBCXX_USE_CXX11_ABI=0)
set(CMAKE_CXX_FLAGS "-std=c++11")
set(PROJECT_DIR "$ENV{PROJECT_PATH}")
set(GTEST_DIR ${PROJECT_DIR}/testcases/libs/gtest)
set(ADK_DIR "$ENV{ADK_PATH}")
set(ATC_DIR ${ADK_DIR}/atc)
set(OP_PROTO_SRC_DIR ${PROJECT_DIR}/op_proto)
message(STATUS "ATC_DIR=${ATC_DIR}" )
enable_testing()

include_directories(
    "${GTEST_DIR}/include"
    "${ATC_DIR}/include"
    "${OP_PROTO_SRC_DIR}" )

aux_source_directory(${OP_PROTO_SRC_DIR} OP_PROTO_SOURCE_SRCS)
file(GLOB OP_PROTO_TEST_FILES **proto.cc)

link_directories(
    "${ATC_DIR}/lib64"
    "${GTEST_DIR}" 
    "/usr/local/gcc7.3.0/lib64/" )
```

// Set this parameter to the actual lib64 path of gcc7.3.0 if the user
Step 3  Write UT cases in Python for the operator implementation code.

In the testcases/ut/ops_test/add/test_add_impl.py file, write UT cases in Python for the operator implementation code. Execute the operator and compare the compute result with the expect result to test the operator logic.

```python
import sys
from op_test_frame.ut import BroadcastOpUT  # Import the UT class based on the operator type.

ut_case = BroadcastOpUT("Add")  # Instantiate the UT case. ut_case is a keyword of the UT framework and cannot be modified. Add is the type of the operator.

def calc_expect_func(input_x, input_y, output_z):    # Define the function for generating the expected data.
    res = input_x["value"] + input_y["value"]
    return [res, ]    # Return the expected data.

# Add test cases.
ut_case.add_precision_case("all", {  
    "params": ["dtype": "float16", "format": "ND", "ori_format": "ND", "ori_shape": (32,), "shape": (32,), "param_type": "Input"],
    ("dtype": "float16", "format": "ND", "ori_format": "ND", "ori_shape": (32,), "shape": (32,), "param_type": "Output"),
    "calc_expect_func": calc_expect_func}),
)
# If multiple test cases are defined, multiple ut_case.add_precision_case functions need to be defined.
ut_case.add_precision_case("all", {
    "params": ["dtype": "float16", "format": "ND", "ori_format": "ND", "ori_shape": (16,2), "shape": (16,2), "param_type": "Input"],
    ("dtype": "float16", "format": "ND", "ori_format": "ND", "ori_shape": (16,2), "shape": (16,2), "param_type": "Output"),
    "calc_expect_func": calc_expect_func
}),
```

1. Import the UT class based on the operator type. For details, see 8.5.5.2 APIs.
2. Instantiate the UT case. For details about how to use OpUT, see Definition of the OpUT Test Class.
3. Define the function for generating the expected data.
4. Add test cases.
   For details about how to use each test API, see 8.5.5.2 APIs.
   To compare the result with the expectation, call add_precision_case.

Step 4  Write UT cases in C++ for the operator prototype definition.

In the testcases/ut/ops_test/add/test_add_proto.cc file, write the UT case in C++ for the operator prototype definition. Define the operator instance, update the
input and output of the operator, call InferenceShapeAndType, and verify the execution process as well as the result of InferenceShapeAndType.

1. Include the GTest framework and the header file of the operator IR definition.

C++ UT cases use the GTest framework. Therefore, you need to include the GTest framework. The operator prototype is defined in the prototype definition header file. Therefore, you need to include the .h file of the prototype definition.

```
#include <gtest/gtest.h>
#include <vector>
#include "add.h"
```

2. Define a test class.

C++ UT cases use the GTest framework. Therefore, a class needs to be defined to inherit the gtest class.

```
#include <gtest/gtest.h>
#include <vector>
#include "add.h"

class AddTest : public testing::Test {
protected:
  static void SetUpTestCase() {
    std::cout << "add test SetUp" << std::endl;
  }

  static void TearDownTestCase() {
    std::cout << "add test TearDown" << std::endl;
  }
};
```

The test class name can be user-defined and suffixed with Test.

3. Write test cases.

Write a test case function for each scenario. In this case, an operator instance needs to be constructed, including the operator name, shape, and data type. Call the InferenceShapeAndType function and compare the inferred shape and dtype with the expected results.

The following is an example.

```
TEST_F(AddTest, add_test_case_1) {
  // Define the operator instance, input shape, and input type, which are carried by the TensorDesc instance.
  ge::op::Add add_op;  // Add is the operator type, which must be the same as the OpType argument in REG_OP(OpType) in the operator definition file.
  ge::TensorDesc tensorDesc;
  ge::Shape shape({2, 3, 4});
  tensorDesc.SetDataType(ge::DT_FLOAT16);
  tensorDesc.SetShape(shape);

  // Update the operator inputs. The input name must be the same as that in the prototype definition file (.h). For example, x1 and x2 are inputs of the Add operator.
  add_op.UpdateInputDesc("x1", tensorDesc);
  add_op.UpdateInputDesc("x2", tensorDesc);

  // Call the InferenceShapeAndType function. The InferenceShapeAndType() API is a fixed API. During case execution, the shape inference function in the operator prototype definition is automatically called.
  auto ret = add_op.InferShapeAndType();
  // Verify whether the calling process is successful.
  EXPECT_EQ(ret, ge::GRAPH_SUCCESS);

  // Obtain the operator output and compare the shape and type. The name of the operator output must be the same as that in the prototype definition file (.h). For example, the operator output is y.
  auto output_desc = add_op.GetOutputDesc("y");
```

If the shapes of inputs are different, define multiple TensorDesc objects, as follows.

```cpp
ge::op::Operator1 operator1_op;  // Operator1 indicates the operator type.
ge::TensorDesc tensorDesc1;
ge::TensorDesc tensorDesc2;
ge::Shape shape1({2, 3, 4});
ge::Shape shape2({3, 4, 5});
tensorDesc1.SetDataType(ge::DT_FLOAT16);
tensorDesc1.SetShape(shape1);
tensorDesc2.SetDataType(ge::DT_FLOAT16);
tensorDesc2.SetShape(shape2);

// Update the operator inputs.
operator1_op.UpdateInputDesc("x1", tensorDesc1);
operator1_op.UpdateInputDesc("x2", tensorDesc2);
```

**Step 5** Run the UT cases of the operator implementation file.

You can run the UT cases of all operators in the current project or run the UT cases of a single-operator.

- Right-click the `testcases/ut/ops_test` folder and choose Run _The Operator 'All' UT Impl with coverage_ from the shortcut menu to run the test cases of the operator implementation code in the folder.
- Right-click the `testcases/ut/ops_test/operator_name` folder and choose Run _The Operator 'operator_name' UT Impl with coverage_ to run test cases of implementation code of a single-operator.

When the case runs for the first time, the corresponding configuration dialog box is displayed. Configure the parameters and click Run. For details about how to modify the run configurations, see **12.3.1 Editing a Run Configuration**.

### Table 8-17 Run configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the run configuration (user-defined).</td>
</tr>
<tr>
<td>Test Type</td>
<td><strong>ut_impl</strong> is recommended.</td>
</tr>
<tr>
<td>Compute Unit</td>
<td>Compute unit.</td>
</tr>
<tr>
<td></td>
<td>- AI Core/Vector Core</td>
</tr>
<tr>
<td></td>
<td>- AI CPU</td>
</tr>
<tr>
<td></td>
<td>You can select different compute units to switch</td>
</tr>
<tr>
<td></td>
<td>between the AI Core/Vector Core and AI CPU UT</td>
</tr>
<tr>
<td></td>
<td>configuration pages.</td>
</tr>
<tr>
<td>SoC Version</td>
<td>Current the Ascend AI Processor version.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Target</td>
<td>Target environment.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Simulator Function</strong>: functional simulation environment.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Simulator_TMMModel</strong>: The scheduling pipeline of operator execution is quickly displayed, and the actual operator computing is not performed.</td>
</tr>
<tr>
<td></td>
<td><strong>Currently, the environment is supported in the Ascend 310 AI Processor and Ascend 910 AI Processor only.</strong></td>
</tr>
<tr>
<td>Operator Name</td>
<td>Test cases.</td>
</tr>
<tr>
<td></td>
<td>- <strong>all</strong>: all test cases</td>
</tr>
<tr>
<td></td>
<td>- Other values: test cases of a specific operator</td>
</tr>
<tr>
<td>ADK Machine</td>
<td>SSH connection of the device where the ADK tool is located. For details about how to add an SSH connection, see 12.1 SSH Connection Management.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong></td>
</tr>
<tr>
<td></td>
<td>Only Windows OSs are supported.</td>
</tr>
<tr>
<td>Case Names</td>
<td>Test cases to run, that is, the Python UT cases of the operator implementation code. You can select or deselect all test cases.</td>
</tr>
</tbody>
</table>

- Check the execution result.
  a. After the execution is complete, view the execution result in the Run log print window at the bottom.
  b. In the Run window, click the URL in `index.html` to view the UT case coverage, as shown in Figure 8-20. In the URL, localhost indicates the server IP of MindStudio.

**NOTE**

To view the test results of UT cases, you need to use a browser. If no browser is available, install one.

**Figure 8-20** Viewing the UT case coverage

<table>
<thead>
<tr>
<th>Module</th>
<th>statements</th>
<th>missing</th>
<th>excluded</th>
<th>coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>/home/xxxxx/MindStudio/samples/demv_projects/custom/lib/iml/add.py</td>
<td>43</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>Total</td>
<td>43</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
</tbody>
</table>

The preceding figure shows that the UT case coverage of the operator is 100% and the UT verification is passed.

If "Page'\http://***.html'requested without authorization, you can copy URL and open it in browser to trust it." is displayed, rectify the fault by referring to 15.8.3 How Do I Allow Unsigned Access Requests?
c. Click the operator on the HTML page to navigate to the UT case coverage dialog box, as shown in Figure 8-21. The green and red labels are used to indicate the coverage rates.

**Figure 8-21 UT coverage details**

```
Coverage for /home/username/MindStudio/samples/demo_projects/custom/tbe/impl/add.py : 100%
43 statements 43 run 0 missing 0 excluded
```

A NOTE

Before the processor performs computation, Vector Unit, Cube Unit, MTE1, MTE2, MTE3, and more units are initialized. As a result, data is generated on each unit before data movement progress is displayed in the timeline.

**Step 6** Run the UT cases defined in the operator prototype.

You can run the UT cases of all operators in the current project or run the UT cases of a single-operator.

- Right-click the `testcases/ut/ops_test` folder and choose **Run The Operator 'All' UT Proto** from the shortcut menu to run the test cases of the operator prototype definition code in the folder.

- Right-click the `testcases/ut/ops_test/operator_name` folder and choose **Run The Operator 'operator_name' UT Proto** from the shortcut menu to run test cases of a single-operator prototype definition code.

When the case runs for the first time, the corresponding configuration dialog box is displayed. Configure the parameters and click **Run**. For details about
how to modify the run configurations, see 12.3.1 Editing a Run Configuration.

Table 8-18 Run configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the run configuration (user-defined).</td>
</tr>
<tr>
<td>Test Type</td>
<td>ut_proto is recommended.</td>
</tr>
<tr>
<td>Operator Name</td>
<td>Test cases.</td>
</tr>
<tr>
<td></td>
<td>- all: all test cases</td>
</tr>
<tr>
<td></td>
<td>- Other values: test cases of a specific operator</td>
</tr>
<tr>
<td>Case Names</td>
<td>Select test cases to be executed, that is, the test cases defined in TEST_F.</td>
</tr>
<tr>
<td></td>
<td>You can select or deselect all test cases.</td>
</tr>
</tbody>
</table>

After the execution is complete, view the execution result in the log print window at the bottom. The result shows the test case count, including successful execution and failed execution, as shown in the following figure.

--- End

8.6.5 Operator Project Building

Overview

The custom operator project is built into a custom OPP runfile custom_opp_Target OS_Target Architecture.run.

Build the operator plug-in implementation file, operator prototype definition file, and operator information definition file into the corresponding operator plug-in,
operator prototype library, and operator information library. Figure 8-22 shows the build process.

**Figure 8-22 Build flowchart**

- **Plug-in implementation** (tensorflow_\*\_plugin.cpp) → Build → **Operator plug-in** (libcust_tf\_parsers.so)
- **Operator prototype definition** (.h/cpp) → Build → **Operator prototype library** (libcust_op\_proto.so)
- **Operator info definition** (.ini) → Build → **Operator info library** (aic<\soc\_version>-ops\_info.json)

**NOTE**

Windows OSs do not support local build.

**Build Procedure**

**Step 1** In the MindStudio project window, select the operator project.

**Step 2** Choose **Build > Edit Build Configuration...** on the top menu bar.

**Step 3** Set parameters in the displayed **Build Configurations** dialog box, as shown in the following figure. For details, see **Table 8-19**.

**Table 8-19 Parameter description**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build Configuration</td>
<td>Name of the build configuration. Defaults to Build-Configuration.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Build Mode      | Build mode.  
  - **Remote Build:**  
    NOTICE  
    g++ 7.5.0 is required for remote build.  
  - **Local Build:**  
    The operator project is built on the MindStudio server. You can quickly locate the implementation code in MindStudio based on the build log for troubleshooting. In this mode, you need to configure the cross compilation environment by referring to [2.4.5 Configuring the Compilation Environment](#). |
| SSH Connection  | This parameter is available only in **Remote Build** mode. Select an SSH configuration from the drop-down list box. If no connection is added, click + to add it. For details, see [12.1 SSH Connection Management](#). |
| Environment variables | This parameter is available only in **Remote Build** mode. Set the environment variables. |
| Target OS       | This parameter is available only in **Local Build** mode.  
  - For Ascend EP, select the host-side OS where the Ascend AI Processor is located.  
  - For Ascend RC, select the OS of the board environment. |
| Target Architecture | This parameter is available only in **Local Build** mode. Select the OS architecture of the target environment. |

If remote build is used, you need to configure the SSH connection and environment variables of the remote server.

- Configuring the SSH connection  
  For details about how to add an SSH connection, see [12.1 SSH Connection Management](#).
- Configuring the environment variables  
  Set the environment variable `ASCEND_TENSOR_COMPILER_INCLUDE`.  
  ```bash
  ASCEND_TENSOR_COMPILER_INCLUDE=/home/xxx/Ascend/ascend-toolkit/latest/atc/include
  ```  
  Replace `/home/xxx/Ascend/ascend-toolkit/latest/atc/include` with the actual installation path of the ADK.

**Step 4** Click **Build** to build the project.
Step 5  View the build result in the lower part of MindStudio. The custom OPP runfile `custom_opp_Target OS_Target Architecture.run` is generated in the `cmake-build` directory of the operator project.

For the value of `Target OS` and `Target Architecture`, see Step 3.

-----End

8.6.6 Local Operator Deployment

**NOTICE**

Windows OSs do not support local operator deployment.

Step 1  On the menu bar of the MindStudio project window, choose Ascend > Deploy.

The operator deployment dialog box is displayed.

Step 2  Select Deploy Locally and click Deploy.

If the following information is displayed on the Output tab page, the custom operator is successfully deployed.

After the custom OPP is successfully installed, the custom operator is deployed in the `/opp` directory in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux`.

The directory structure is as follows.

```
| opp    // OPP directory
|   | op_impl
|   |   | built-in
|   |   | custom
|   |   |   | ai_core
|   |   |   |   | tbe
|   |   |   |   |   | config
|   |   |   |   |   |   | soc_version // Ascend AI Processor version
|   |   |   |   |   |   | aic-soc_version-ops-info.json // Custom operator info library file
|   |   |   |   |   |   | custom_impl // Custom operator implementation code
|   |   |   |   |   |   | add.py
|   |   |   |   |   |   | vector_core // Reserved directory, which can be ignored
|   |   |   |   |   | framework
```
8.6.7 Remote Operator Deployment

Deploy the custom OPP runfile `custom_opp_Target OS_Target Architecture.run` to the system OPP in the hardware environment with Ascend AI Processor to construct necessary conditions for operators execution on the network.

**Step 1** In the MindStudio project window, select the operator project.

**Step 2** On the top menu bar, choose Ascend > Deploy. The operator deployment dialog box is displayed.

Choose Deploy Remotely > SSH Connection from the drop-down list box to configure SSH connection. If the no SSH connection is not added in advance, click + to add it.

**Step 3** Configure the environment variable.

You can use either of the following methods:

- Configure the environment variables on the host of the hardware device where the Ascend AI Processor is located.
  
  Deploy the operator on the host as the running user in MindStudio. Before deploying the operator, ensure the following environment variable is configured on the host.
  
  a. Run the following command in the $HOME/.bashrc file on the host as the running user:
     
     ```
     export ASCEND_OPP_PATH=`home/xxx/Ascend/ascend-toolkit/latest/opp`
     
     home/xxx/Ascend/ascend-toolkit/latest is the OPP installation path. Replace it with the actual path.
     ```
  
  b. Run the following command to make the environment variable to take effect:
     
     ```
     source ~/.bashrc
     ```

- Add the environment variable in Environment Variables.
  
  Type `ASCEND_OPP_PATH=/home/xxx/Ascend/ascend-toolkit/latest/opp` in the Environment Variables field.
  
  home/xxx/Ascend/ascend-toolkit/latest is the OPP installation path. Replace it with the actual path.
  
  You can also click the icon next to the text box and enter a value in the displayed dialog box.
    
    - Type `ASCEND_OPP_PATH` in the Name field.
- Type `home/xxx/Ascend/ascend-toolkit/latest/opp` in the Value field.

**Figure 8-24** Operator deployment

![Operator deployment](image)

**Step 4** Select the target server for operator deployment and click **Deploy**.

**Step 5** Deploy the operators. Operator deployment is equivalent to installing the custom OPP generated in **8.5.6 Operator Project Building**. After the deployment, the operator is deployed in the OPP installation path on the host. The default path is `/usr/local/Ascend/opp/`.

**Figure 8-25** Operator deployment log messages

```plaintext
2020-03-09 21:49:59 Verifying archive integrity... All good.
2020-03-09 21:49:59 Uncompressing version:1.0
2020-03-09 21:49:59 [info] copy uninstall sh success
2020-03-09 21:49:59 [ops custom upgrade framework]
2020-03-09 21:49:59 [runtime] [2020-03-10 09:49:57] [info] replace old ops framework files ...
2020-03-09 21:49:59 [ops custom upgrade op preto]
2020-03-09 21:49:59 [runtime] [2020-03-10 09:49:57] [info] replace old ops op proto files ...
2020-03-09 21:49:59 [ops custom upgrade op impl]
2020-03-09 21:49:59 [runtime] [2020-03-10 09:49:57] [info] replace old ops op impl files ...
2020-03-09 21:49:59 [ops custom changemode...]
2020-03-09 21:49:59 SUCCESS
2020-03-09 21:49:59 Deploying the operator info to the remote host finished.
```

After the custom OPP is deployed on the host, the directory structure is similar to the following:

```
opp      // OPP directory
├── op_impl
│   ├── built-in
│   │   └── custom
│   │       └── aic_ops_info.json     // Custom operator info library file
│   │       └── custom_impl               // Custom operator implementation code
│   │           └── add.py
│   │   └── config
│   │       ├── add.py
│   │       └── custom_impl               // Custom operator implementation code
│   │           └── add.py
│   └── framework
│       ├── built-in
│       │   └── custom
│       │       └── caffe
│       │           └── op_proto
│       │           └── built-in
│       │               └── custom
│       │                   └── libcust_op_proto.so // Prototype library file of the custom operator
│       └── custom
│           └── caffe
```

----End
8.6.8 Operator Adaptation in PyTorch

**NOTE**
Currently, MindStudio does not support operator adaptation in PyTorch. For details about operator adaptation, see *PyTorch Operator Development Guide*.

Perform the following operations to adapt operators in PyTorch. For details, see *PyTorch Operator Development Guide*.

- Installing the PyTorch environment dependencies
- Compiling and installing the PyTorch framework.
- Adapting a TBE operator in the PyTorch framework

8.6.9 ST

**Overview**
MindStudio provides an upgraded ST framework to automatically generate test cases, verify operator functionality and compute accuracy in real hardware environment, and generate an execution report. Feature details are as follows:

- Generates an operator test case definition file based on the operator information library.
- Generates test data of different shapes and dtypes and AscendCL-based test cases from the operator test case definition file.
- Builds the operator project, deploys the operators in the system OPP, and runs test cases in the hardware environment to verify the operator functionality.
- Generates an ST report (**st_report.json**) that displays information about test cases and phase-by-phase execution states.
- Generates a test function, compares the expected operator output and the actual operator output, and displays the comparison result to verify the compute accuracy.

**Prerequisites**
- Custom operators have been developed. For details, see 8.6.1 Operator Code Implementation, 8.6.2 Operator Prototype Definition, 8.6.3 Operator Information Definition, and 8.6.8 Operator Adaptation in PyTorch.
- MindStudio has been connected to a hardware device.

**Generating an ST Case Definition File**

**Step 1** Create an ST case.

Three portals are available.

- Right-click the root directory of the operator project and choose **New Cases > ST Case** from the shortcut menu.
- Right-click the operator information definition file **(project name)**/tbe/op_info cfg/ai_core/**(SoC version)**/xx.ini, and choose **New Cases > ST Case**.
● If ST cases of the operator exist, right-click the testcases or testcases > st directory, and choose New Cases > ST Case from the shortcut menu to add ST cases.

**Step 2** In the Create ST Cases for an Operator dialog box, select the operator for which the ST case needs to be created.

See the following figure.

![Operator Name and SoC Version](image)

Select an operator name from the Operator drop-down list.

Select the version of your the Ascend AI Processor from the SoC Version drop-down list.

● If Import operator info from a model is not selected, click OK and an operator test case definition file with empty shapes is generated. The following figure shows the Design Cases dialog box.

![Design Cases](image)

You need to configure the shape information to generate test data and test cases. Configure the rest fields as required. For details, see Field Description of the Operator Test Case Definition File.

● If you select Import operator info from a model and upload an ONNX model file (.onnx) that contains the operator, the top-layer shape of the obtained model is displayed.

You can also modify the shape information of the top-layer input in Input Nodes Shape.

After you click OK, the tool automatically dumps the shape information of the selected operator based on the shape information of the first layer and generates the corresponding operator test case definition file.
This file is used for generating test data and test cases. You can modify related fields. For details, see Field Description of the Operator Test Case Definition File.

**NOTE**

To use this function, you need to install the ONNX library in the operating environment. If the Windows OS is used, you need to install the ONNX library on the local Windows host.

**Step 3** To compare the expected data with the benchmark data, define and configure a function for generating the expected data of the operator.

1. Customize a test function for generating the expected operator result.
   
   The function is implemented on a framework such as TensorFlow or Caffe and has the same functionality as the custom operator. The function runs on the CPU and generates benchmark data. The operator accuracy is tested by comparing the benchmark data with the output data. The function for generating expected data is implemented using the Python language. Multiple expected operator data generation functions can be implemented in a Python file. Keep the inputs, outputs, and attributes (including the format, type, and shape) of this function consistent with those of the custom operator.

2. Edit the test case definition file.
   
   Configure the function in the test case definition file. You can configure it in the Design view or the Text view.
   
   - In the Design view, select a destination path of the Python file as the script path in Expected Result Verification dialog box. In Script Function, enter the name of the function that generates expected operator data.

   ![Expected Result Verification](image)

   **NOTE**

   In Script Function, you can choose to enter the name of the function that generates expected operator data or leave it empty.

   - If the name of the function is entered, the function is called to generate the benchmark data during ST.
   - If the function name is left empty, the function with the same name as the custom operator is automatically matched to generate benchmark data during ST. If no function with the same name exists, a message indicating match failure is displayed.

   - In the Text view, if the calc_expect_func_file parameter is added, the value is the file path and name of the function generating the expected operator data. For example:
Where, `/home/teste/test_*.py` indicates the implementation file of the test function, and `function` indicates the function name. Separate the file path and function name with a colon (:).

Example: The `test_add.py` file is the expected data generation file of the Add operator. The function implementation is as follows:

```python
def calc_expect_func(input_x, input_y, out):
    res = input_x["value"] + input_y["value"]
    return [res, ]
```

**Note:** You need to complete the expected data generation function of the operator based on the developed custom operator. The input parameters of the operator's expected data generation function must be the same as those of the operator in the operator code implementation.

**Step 4** Click **Save** to save the modification to the operator test case definition file.

The operator test case definition file (named as `OpType_case_timestamp.json`) is stored in the `testcases/st/OpType/{SoC Version}` directory under the root directory of the operator project.

**NOTE**

Strictly follow the naming rules of the operator test case definition file. Do not name irrelevant files in this format to the `testcases/st/OpType/{SoC Version}` directory under the root directory of the operator project. Otherwise, file parsing errors may occur.

---End

### Running ST Cases

**Step 1** Right-click the ST case definition file generated in *Generating an ST Case Definition File* (testcases > st > add > `{SoC Version}` > `xxxx.json`) and choose Run The Operator 'xxx' ST Case from the shortcut menu.

**Table 8-20 Run configuration**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the run configuration (user-defined).</td>
</tr>
<tr>
<td>Test Type</td>
<td>Select <strong>st_cases</strong>.</td>
</tr>
<tr>
<td>Execute Mode</td>
<td>• Remote Execute</td>
</tr>
<tr>
<td></td>
<td>• Local Execute</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong> Local Execute does not apply to Windows OSs.</td>
</tr>
<tr>
<td>SSH Connection</td>
<td>If <strong>Execute Mode</strong> is set to Remote Execute, select an SSH connection from the drop-down list box. You can also click ![+] to add a new SSH connection. For details about how to add an SSH connection, see 12.1 SSHE Connection Management.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ADK Machine</td>
<td>SSH connection of the device where the ADK tool is located. For details about how to add an SSH connection, see 12.1 SSH Connection Management.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong> This parameter applies only to Windows OSs.</td>
</tr>
<tr>
<td>Environment Variables</td>
<td>● Add environment variables in the text box. PATH_1=Path 1; PATH_2=Path 2. Use semicolons (;) to separate multiple environment variables.</td>
</tr>
<tr>
<td></td>
<td>● You can also click the icon next to the text box and enter a value in the displayed dialog box.</td>
</tr>
<tr>
<td></td>
<td>– Type PATH_1 in the <strong>Name</strong> field.</td>
</tr>
<tr>
<td></td>
<td>– Type the value (Path 1) in the <strong>Value</strong> field.</td>
</tr>
<tr>
<td></td>
<td>If you select Instead system environment variables, the system environment variables are displayed.</td>
</tr>
<tr>
<td>Operator Name</td>
<td>Select the operator to test.</td>
</tr>
<tr>
<td>SoC Version</td>
<td>Select the SoC version.</td>
</tr>
<tr>
<td>Executable File Name</td>
<td>Select the test case definition file to run from the drop-down list box. If the ST is performed on the AI CPU operator, (AI CPU) is displayed</td>
</tr>
<tr>
<td></td>
<td>in front of the test case file.</td>
</tr>
<tr>
<td>Target OS</td>
<td>For Ascend EP, select the OS of the host of the hardware device powered by the Ascend AI Processor.</td>
</tr>
<tr>
<td></td>
<td>For Ascend RC, select the OS of the board environment.</td>
</tr>
<tr>
<td>Target Architecture</td>
<td>Select the OS architecture of the target environment.</td>
</tr>
<tr>
<td>Case Names</td>
<td>Specify the names of the cases to be executed.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: All cases are selected by default. You can deselect unnecessary cases.</td>
</tr>
<tr>
<td>Advanced Options</td>
<td>Specify advanced options.</td>
</tr>
<tr>
<td>ACT Log Level</td>
<td>Select an ACT log level.</td>
</tr>
<tr>
<td></td>
<td>● INFO</td>
</tr>
<tr>
<td></td>
<td>● DEBUG</td>
</tr>
<tr>
<td></td>
<td>● WARNING</td>
</tr>
<tr>
<td></td>
<td>● ERROR</td>
</tr>
<tr>
<td></td>
<td>● NULL</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Precision Mode</td>
<td>Set the precision mode. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>● force_fp176</td>
</tr>
<tr>
<td></td>
<td>● allow_mix_precision</td>
</tr>
<tr>
<td></td>
<td>● allow_fp32_to_fp16</td>
</tr>
<tr>
<td></td>
<td>● must_keep_origin_dtype</td>
</tr>
<tr>
<td>Device Id</td>
<td>Set the ID of the device that runs the ST. Specify the ID of the AI processor in use.</td>
</tr>
<tr>
<td>Enable Profiling</td>
<td>Enable profiling to obtain the performance data of the operator on the Ascend AI Processor. To use this function, you need to configure</td>
</tr>
<tr>
<td></td>
<td>the path of the msprof tool in the operating environment to the PATH environment variable.</td>
</tr>
<tr>
<td></td>
<td>The msprof tool works under toolkit/tools/profiler/bin/msprof of the Toolkit directory.</td>
</tr>
</tbody>
</table>

**Note**

- The ST supports the setting and query of the board log level. For details, see **12.6 Log Management**.
- The Windows OS does not support the Local Execute function.

**Step 2**

Set the host running user.

Add a host running user in the SSH Connection dialog box. The user must be with the HwHiAiUser group. For details about how to add an SSH connection, see **12.1 SSH Connection Management**.

**Step 3**

Configure the environment variables of related components in the operating environment.

- On a remote device:
  - For Ascend EP, you need to configure the environment variables of the component installation paths on the host of the device.
    Configure the installation paths of the ACLlib, FwkACLlib, and Driver components in the ~/.bashrc file as the host running user.
    1. Open the .bashrc file of the running user.
    2. Append the following lines to the file:
       ```
       export ASCEND_DRIVER_PATH=/usr/local/Ascend/driver
       export ASCEND_HOME=/usr/local/Ascend//ascend-toolkit/latest
       export ASCEND_AICPU_PATH=${ASCEND_HOME}/<target architecture>
       ```
    3. If the remote device is in an inference environment:
       ```
       export LD_LIBRARY_PATH=${ASCEND_DRIVER_PATH}/lib64:${ASCEND_HOME}/acllib/lib64:SLD_LIBRARY_PATH
       ```
If the remote device is in a training environment:

```bash
export
LD_LIBRARY_PATH=${ASCEND_DRIVER_PATH}/lib64/driver:$
{ASCEND_DRIVER_PATH}/lib64/common:${ASCEND_HOME}/
fwkacllib/lib64:$LD_LIBRARY_PATH
```

If the preceding environment variables exist, check if the installation path is correct.

Replace `<target architecture>` with the OS architecture, for example, `x86_64-linux` or `arm64-linux`.

iii. Save the configuration and exit.

```
:wq
```

iv. Make the environment variables take effect.

```
source ~/.bashrc
```

- Add the environment variable in Environment Variables.

Set the environment variables in the Environment Variables area as described in Step 1.

- Add environment variables in the text box.

```bash
ASCEND_DRIVER_PATH=/usr/local/Ascend/driver;
ASCEND_HOME=/usr/local/Ascend/ascend-toolkit/latest;
ASCEND_AICPU_PATH=${ASCEND_HOME}/<target architecture>-linux;
```

If the remote device is in an inference environment:

```bash
LD_LIBRARY_PATH=${ASCEND_DRIVER_PATH}/lib64:${ASCEND_HOME}/acllib/
lib64:$LD_LIBRARY_PATH;
```

Modify the environment variables based on the actual installation paths of the driver and ADK and the architecture used by the remote OS.

- You can also click the icon next to the text box and enter a value in the displayed dialog box.

Type the environment variable name in the Name field and the value in the Value field.

**Step 4** Click Run.

MindStudio generates test data and test code in `/testcases/st/out/<operator name>` under the operator project root directory based on the operator test case definition file, builds an executable file, and executes test cases on the specified hardware backend. The report about comparison of execution result and benchmark is printed to the Output window and the `st_report.json` file is generated in `/testcases/st/out/<operator name>` under the operator project root directory. For details about the `st_report.json` file, see Table 8-21.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>run_cmd</td>
<td>Command.</td>
</tr>
<tr>
<td>report_list</td>
<td>List of reports of test cases.</td>
</tr>
</tbody>
</table>

Table 8-21 `st_report.json` description
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>trace_detail</td>
<td>Execution details.</td>
</tr>
<tr>
<td>st_case_info</td>
<td>Test information.</td>
</tr>
<tr>
<td>expect_data_path</td>
<td>Path of the expected result.</td>
</tr>
<tr>
<td>case_name</td>
<td>Test case name.</td>
</tr>
<tr>
<td>input_data_path</td>
<td>Path of the input data.</td>
</tr>
<tr>
<td>planned_output_data_paths</td>
<td>Path of the actual result.</td>
</tr>
<tr>
<td>op_params</td>
<td>Operator parameters.</td>
</tr>
<tr>
<td>stage_result</td>
<td>Execution result by phase.</td>
</tr>
<tr>
<td>case_name</td>
<td>Test name.</td>
</tr>
<tr>
<td>status</td>
<td>Test result.</td>
</tr>
</tbody>
</table>

### More Functions

- Switching between the Design and Text views

  Double-click the operator test case definition file in the `testcases/st/OpType/{SoC Version}` folder or right-click the file and choose `OpType_case_timestamp.json` from the shortcut menu. The window is displayed with Design and Text views, as shown in the following figure.

  If there is no incorrect configuration information in the Design view, the changes will be automatically synchronized to the Text view when you switch to the Text view. The reverse is also true.
● Adding a test case

On the operator test case definition page, click **Add**. The dialog box shown in the following figure is displayed, prompting you to input the **Case Name**.

**Case Name** is a string combination of digits, letters, and underscores (_). Click **OK** to add the created case to the bottom of the **Design Cases** dialog box. The fields of the new case are empty and need to be configured. For details about the configuration rules, see **Field Description of the Operator Test Case Definition File**.

● Copying and adding an operator input

On the operator test case definition page, click **on the right of **Input[xx]** to copy **Input[xx]** as a new **Input[xx]**. Alternatively, click **+** to add an operator input. Modify the parameters based on the site requirements.

**NOTE**

If the operator ST project supports dynamic multi-shape operators, when the operator input of x1 is copied, the **Name** parameter must be named in the x10, x11, x12, ..., format. When the operator input of x2 is copied, the **Name** parameter must be named in the x20, x21, x22, ..., format.
● Deleting an operator input

In the operator test case definition dialog box, click on the right of Input[xx] to delete Input[xx].

● Deleting a test case

In the operator test case definition dialog box, click on the right of a case to delete the case.

● Running one or more test cases

In the operator test case definition dialog box, select one or more cases and click Run.
- Modifying and running ST source code
  a. Generating source code
     After the ST is successful, the ST source code is generated in `./testcases/st/out/OpName/src`.
  b. Modifying source code
     You can modify the source code as required to implement custom requirements and functions.
  c. Running source code
     Right-click `./testcases/st/out/<operator name>` and choose Run St Source from the shortcut menu to run the modified source code.

**CAUTION**

If you run the ST from another entry, the modified source code will be overwritten. Therefore, back up the source code in advance.

**Field Description of the Operator Test Case Definition File**

**Table 8-22** Operator test case definition file (.json)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test_xxx_xx</td>
<td>(Required) A string. Test case name.</td>
</tr>
</tbody>
</table>

MindStudio User Guide

8 Custom Operator Development

Copyright © Huawei Technologies Co., Ltd.
<table>
<thead>
<tr>
<th><strong>Parameter</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Input[xx]</td>
<td>(Required) Operator input. <strong>NOTICE</strong> The number of parameter values in each input of an operator must be the same. Otherwise, test case generation will fail. For example, if the Input[01] supports two formats, the Input[02] also supports two formats. Similarly, the parameter values of type, shape, data_distribute, and value_range in all Input[xx] must be the same.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>- Format</td>
<td>(Required)</td>
</tr>
<tr>
<td></td>
<td>A string or a 1D array of strings. Input format. Must not be empty. Value range:</td>
</tr>
<tr>
<td></td>
<td>● <strong>NCHW</strong></td>
</tr>
<tr>
<td></td>
<td>● <strong>NHWC</strong></td>
</tr>
<tr>
<td></td>
<td>● <strong>ND</strong>: any format</td>
</tr>
<tr>
<td></td>
<td>● <strong>NC1HWCO</strong>: the 5D format defined by Huawei. <strong>C0</strong> is closely related to the micro-architecture, and the value is equal to a size of a cube unit, for example, <strong>16</strong>. <strong>C1</strong> divides the C dimension by <strong>C0</strong>, that is, <strong>C1 = C/C0</strong>. When the division is not exact, the last data segment is padded to <strong>C0</strong>.</td>
</tr>
<tr>
<td></td>
<td>● <strong>FRACTAL_Z</strong>: format of the convolution weight.</td>
</tr>
<tr>
<td></td>
<td>● <strong>FRACTAL_NZ</strong>: fractal format defined by Huawei. The data format of the output matrix is <strong>NW1H1H0W0</strong> for Cube Unit computation. The matrix is divided into (H1 x W1) fractals in column-major order, which looks like an N-shape layout. Each fractal consists of (H0 x W0) elements in row-major order, resembling a Z-shaped layout. Thus the <strong>NW1H1H0W0</strong> data format is referred to as an <strong>Nz</strong> format. <strong>(H0 x W0)</strong> indicates the size of a fractal, as shown in the following figure.</td>
</tr>
<tr>
<td></td>
<td>● <strong>RESERVED</strong>: reserved. If this value is used, <strong>type</strong> needs to be set to <strong>UNDEFINED</strong>, indicating that the</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>OriginFormat</td>
<td>(Optional) Original tensor format. If this parameter is not carried, the original tensor format is retained.</td>
</tr>
<tr>
<td>ShapeRange</td>
<td>(Optional) Value range of a dynamic shape. The default value is [1, -1]. For example, if Shape is [16, -1, 16, -1], -1 indicates an unknown shape. If ShapeRange is [[1, 128], [1, -1]], [1, 128] indicates that the value of the first -1 in Shape ranges from 1 to 128. [1, -1] indicates that the value of the second -1 in Shape ranges from 1 to infinity.</td>
</tr>
<tr>
<td>TypicalShape</td>
<td>Fixed shape, which is used to generate available test data.</td>
</tr>
<tr>
<td>Type</td>
<td>(Required) A string or a 1D array of strings. Input data type.</td>
</tr>
<tr>
<td>Shape</td>
<td>(Required) An int or a 1D or 2D array of ints. Input shape.</td>
</tr>
<tr>
<td>OriginShape</td>
<td>(Optional) Original tensor shape. If this field is not carried, the original tensor shape is retained.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>-</td>
<td>ValueRange (Required)</td>
</tr>
<tr>
<td></td>
<td>An int or float, or a 1D or 2D array of ints or floats.</td>
</tr>
<tr>
<td></td>
<td>Value range. Must not be empty. Formatted as $[\text{min_value}, \text{max_value}]$, with $\text{min_value} \leq \text{max_value}$.</td>
</tr>
<tr>
<td>-</td>
<td>DataDistribute (Required)</td>
</tr>
<tr>
<td></td>
<td>A string or a 1D array of strings. Data distribution modes for generating test data:</td>
</tr>
<tr>
<td></td>
<td>● uniform: returns random values that are evenly distributed.</td>
</tr>
<tr>
<td></td>
<td>● normal: returns random values of the normal distribution (Gaussian distribution).</td>
</tr>
<tr>
<td></td>
<td>● beta: returns random values of Beta distribution.</td>
</tr>
<tr>
<td></td>
<td>● laplace: returns random values of Laplace distribution.</td>
</tr>
<tr>
<td></td>
<td>● triangular: returns random values of the triangular distribution.</td>
</tr>
<tr>
<td></td>
<td>● relu: returns random values that are evenly distributed and activated by the ReLU function.</td>
</tr>
<tr>
<td></td>
<td>● sigmoid: returns random values that are evenly distributed and activated by the sigmoid function.</td>
</tr>
<tr>
<td></td>
<td>● softmax: returns random values that are evenly distributed and activated by softmax function.</td>
</tr>
<tr>
<td></td>
<td>● tanh: returns random values that are evenly distributed and activated by tanh function.</td>
</tr>
<tr>
<td>Output[xx]</td>
<td>- (Required)</td>
</tr>
<tr>
<td></td>
<td>Operator output.</td>
</tr>
<tr>
<td></td>
<td>NOTICE</td>
</tr>
<tr>
<td></td>
<td>The number of values in Output must be the same as that in Input. Otherwise, test case generation fails.</td>
</tr>
</tbody>
</table>
|           | For example, if Input supports two formats, the Output also supports two formats.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format</td>
<td>(Required) A string or a 1D array of strings. Output format. Must not be empty. Value range:</td>
</tr>
<tr>
<td></td>
<td>- <strong>NCHW</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>NHWC</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>ND</strong>: any format</td>
</tr>
<tr>
<td></td>
<td>- <strong>NC1HWC0</strong>: the 5D format defined by Huawei. <strong>C0</strong> is closely related to the micro-architecture, and the value is equal to a size of a cube unit, for example, <strong>16</strong>. <strong>C1</strong> divides the C dimension by <strong>C0</strong>, that is, <strong>C1 = C/C0</strong>. When the division is not exact, the last data segment is padded to <strong>C0</strong>.</td>
</tr>
<tr>
<td></td>
<td>- <strong>FRACTAL_Z</strong>: format of the convolution weight.</td>
</tr>
<tr>
<td></td>
<td>- <strong>FRACTAL_NZ</strong>: fractal format defined by Huawei. The data format of the output matrix is <strong>NW1H1H0W0</strong> for Cube Unit computation. The matrix is divided into (H1 x W1) fractals in column-major order, which looks like an N-shape layout. Each fractal consists of (H0 x W0) elements in row-major order, resembling a Z-shaped layout. Thus the NW1H1H0W0 data format is referred to as an Nz format. (H0 x W0) indicates the size of a fractal, as shown in the following figure.</td>
</tr>
</tbody>
</table>

![Fractal Matrix Size](image)

<p>| OriginFormat | (Optional) Original tensor format. If this parameter is not carried, the original tensor format is retained. |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>(Required)</td>
</tr>
<tr>
<td></td>
<td>A string or a 1D array of strings.</td>
</tr>
<tr>
<td></td>
<td>Output data types:</td>
</tr>
<tr>
<td></td>
<td>● bool</td>
</tr>
<tr>
<td></td>
<td>● int8</td>
</tr>
<tr>
<td></td>
<td>● uint8</td>
</tr>
<tr>
<td></td>
<td>● int16</td>
</tr>
<tr>
<td></td>
<td>● uint16</td>
</tr>
<tr>
<td></td>
<td>● int32</td>
</tr>
<tr>
<td></td>
<td>● int64</td>
</tr>
<tr>
<td></td>
<td>● uint32</td>
</tr>
<tr>
<td></td>
<td>● uint64</td>
</tr>
<tr>
<td></td>
<td>● float16</td>
</tr>
<tr>
<td></td>
<td>● float</td>
</tr>
<tr>
<td>Shape</td>
<td>(Required)</td>
</tr>
<tr>
<td></td>
<td>An int or a 1D or 2D array of ints.</td>
</tr>
<tr>
<td></td>
<td>Output shape.</td>
</tr>
<tr>
<td>OriginShape</td>
<td>(Optional)</td>
</tr>
<tr>
<td></td>
<td>Original tensor shape.</td>
</tr>
<tr>
<td></td>
<td>If this field is not carried, the original tensor shape is retained.</td>
</tr>
<tr>
<td>Attr</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>(Optional)</td>
</tr>
<tr>
<td>Name</td>
<td>Required if attr is configured.</td>
</tr>
<tr>
<td></td>
<td>A string.</td>
</tr>
<tr>
<td></td>
<td>Attribute name. It cannot be empty.</td>
</tr>
<tr>
<td>Type</td>
<td>Required if attr is configured.</td>
</tr>
<tr>
<td></td>
<td>Attribute data type.</td>
</tr>
<tr>
<td></td>
<td>A string.</td>
</tr>
<tr>
<td></td>
<td>Output data types:</td>
</tr>
<tr>
<td></td>
<td>● bool</td>
</tr>
<tr>
<td></td>
<td>● int</td>
</tr>
<tr>
<td></td>
<td>● float</td>
</tr>
<tr>
<td></td>
<td>● string</td>
</tr>
<tr>
<td></td>
<td>● list_bool</td>
</tr>
<tr>
<td></td>
<td>● list_int</td>
</tr>
<tr>
<td></td>
<td>● list_float</td>
</tr>
<tr>
<td></td>
<td>● list_string</td>
</tr>
<tr>
<td></td>
<td>● list_list_int</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| Value     | Required if `attr` is configured. Must not be `null`. A string. Attribute value. Varies according to type.  
- `bool`: true/false  
- `int`: 10  
- `float`: 1.0  
- `string`: "NCHW"  
- `list_bool`: [false, true]  
- `list_int`: [1, 224, 224, 3]  
- `list_float`: [1.0, 0.0]  
- `list_string`: ["str1", "str2"]  
- `list_list_int`: [[1, 3, 5, 7], [2, 4, 6, 8]] |

| Expected Result Verification | - | (Optional) |

| - | Script Path | Path of the test function for generating expected operator result. |
| - | Script Function | Name of the test function for generating expected operator result. |

**NOTE**

- When `inputx.paramType` in the operator information library definition file (.ini) is set to `optional`, set `format` of `inputx` to `UNDEFINED` or `RESERVED`, and `type` to `UNDEFINED` in the generated operator test cases.

- When `dynamicShapeSupport.flag` is set to `true` in the operator information definition file (.ini), the `ShapeRange` and `TypicalShape` parameters are added to the generated operator test case. The value of `ShapeRange` is `[[1,-1]]`, indicating that the `Shape` parameter can be set to any value. If the value of the `Shape` parameter contains `-1` or `-2`, you need to provide a fixed shape value in the `TypicalShape` parameter for actual tests. If the value of the `Shape` parameter contains `-1`, for example, `(200, -1)`, the length of the second axis is unknown. If the value contains `-2`, the dimension is unknown.

- In the operator information library definition file (.ini), if a tensor implementation uses a Huawei-developed format and is different from the original format, you need to manually enter the original format and shape in `OriginFormat` and `OriginShape` to convert the format and shape into those compatible with the offline model.

- `OriginFormat` is an array specifying the formats supported by the original operator. The array must have the same length as the `Format` array.

- The value of `OriginShape` must correspond to the value of `Shape` and match the values of `Format` and `OriginFormat`. 
8.7 TBE Operator Development (MindSpore)

8.7.1 Operator Implementation and Information Registration File

Call the TBE DSL APIs to write the code of operator implementation and information file (mindspore/impl/cus_square_impl.py in the operator project) for the Square operator, including the code of implementing and registering the TBE operator. For details, see “Custom Operators (Ascend) > Implementing a TBE Operator and Registering the Operator Information” in MindSpore Tutorial.

**NOTE**

- To develop a TBE operator in MindSpore, you need to install MindSpore in the operating environment. Click [here](#) for more details.
- NeoKylin OS 7.6-AArch64 does not support the UT and ST tests developed by MindSpore TBE.

8.7.2 Operator Prototype Definition

Write the operator primitive file in the op_proto/cus_square.py file in the operator project and register the operator information in the operator prototype library. During network execution, GE calls the verification API of the operator prototype library to verify operator arguments. If the verification passes, GE infers the output shape and dtype of each node by calling the inference function of the operator prototype library and allocates static memory for the result tensor. For details, see "Custom Operators (Ascend) > Registering the Operator Primitive" in MindSpore Tutorial.

8.7.3 UT

8.7.3.1 Overview

MindStudio provides an upgraded UT solution based on the GTest framework, simplifying UT case development.

UT is one of the methods to verify a single-operator, which aims to:

- Test correctness of operator code and verify consistency between the desired and actual inputs and outputs.
- Ensure the operator code coverage. The selected scenario combination must cover all branches of the operator code (the coverage should reach 100%), to reduce the failure rate for building operator code in different scenarios.

**NOTE**

For details about definition of the test classes, see the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/toolkit/python/site-packages/ut_test_frame/ut/op_ut.py` directory.
8.7.3.2 APIs

8.7.3.2.1 OpUT

Overview
Functions as a base class for the UT framework. It provides the following two APIs to define and run the test cases.

- add_case
- add_precision_case

Definition of the OpUT Test Class

- Prototype
  `OpUT(op_type, op_module_name=None, op_func_name=None)`

- Parameters
  - `op_type`: operator type.
  - `op_module_name`: operator module name, that is, the name and path of the operator implementation file, for example, `impl.add` (the file path is `impl/add.py`). Defaults to `None`. The argument is generated automatically based on `op_type`. For example, for a BiasAdd operator, the generated module name is `impl.bias_add`. For details about how the argument of `op_module_name` is generated based on `op_type`, see Step 4.
  - `op_func_name`: operator function name in the operator implementation file. Defaults to `None`. The argument is generated automatically based on `op_type`. For details about how the argument of `op_func_name` is generated based on `op_type`, see Step 4. For example, for an Add operator, the corresponding API must exist in `impl/add.py`. The API definition is as follows.

  ```
  @check_op_params(REQUIRED_INPUT, REQUIRED_INPUT, REQUIRED_OUTPUT, KERNEL_NAME)
  def add(input_x, input_y, output_z, kernel_name="add"):
  ```

add_case

- Prototype
  `OpUT.add_case(support_soc=None, case=None)`

- Description
  Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an .o file.

- Parameters
  - `support_soc`: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the `atc/data/platform_config` directory in the ATC installation path. `support_soc` can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to `all` or `None`, all SoCs are supported.
  - `case`: a dictionary. An example is as follows.

```
The key fields in the dictionary include:

- **params**: This field is passed through to the operator API during test case running.

- **case_name**: (optional) name of a test case. If it is not specified, the test framework generates the argument of **case_name** automatically in the following format:
  
  test_[op_type]_auto_case_name_[case_count]
  
  Example: **test_Add_auto_case_name_1**

- **expect**: expected result. Defaults to success. It can also be set to **RuntimeError**, indicating an expected exception.

### add_precision_case

- **Prototype**
  
  ```python
  OpUT.add_precision_case(support_soc=None, case)
  ```
  
- **Description**
  
  Adds test cases for operator build and precision test.

- **Parameters**
  
  - **support_soc**: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the `atc/data/platform_config` directory in the ATC installation path. **support_soc** can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to all or None, all SoCs are supported.
  
  - **case**: a dictionary. An example is as follows.
  
    ```json
    { 
      "params": [ 
        { 
          "shape": (32, 64),
          "ori_shape": (32, 64),
          "format": "ND",
          "ori_format": (32, 64),
          "dtype": "float16"
        },
        { 
          "shape": (32, 64),
          "ori_shape": (32, 64),
          "format": "ND",
          "ori_format": (32, 64),
          "dtype": "float16"
        }
      ],
      "case_name": "test_add_case_1",
      "expect": "success"
    }
    ```
The key fields in the dictionary include:

- **params**: This field is passed through to the operator API during test case running.
- **case_name**: (optional) name of a test case. If it is not specified, the test framework generates the argument of **case_name** automatically in the following format:
  
  
  ```
  test_[op_type]_auto_case_name_[case_count]
  ```

  Example: **test_Add_auto_case_name_1**

- **calc_expect_func**: expected result generation function.

- **precision_standard**: user-defined precision standard. If this field is not set, the following default precision is used for comparison with the expected data:
  
  - **float16**: dual-0.1% error limit, that is, the error ratio is within 0.1% and the relative error is within 0.1%.
  - **float32**: dual-0.01% error limit, that is, the error ratio is within 0.01% and the relative error is within 0.01%.

### Examples

**Example 1**

from op_test_frame.ut import OpUT

# ut_case is a keyword of the UT framework and cannot be modified.

```python
def np_add(x1, x2, y):
    y = (x1.get("value") + x2.get("value"), )
    return y

ut_case.add_precision_case(case={
    "params": [
        {
            "shape": (32, 64),
            "ori_shape": (32, 64),
            "format": "ND",
            "ori_format": "ND",
            "dtype": "float16",
            "param_type": "input"
        },
        {
            "shape": (32, 64),
            "ori_shape": (32, 64),
            "format": "ND",
            "ori_format": "ND",
            "dtype": "float16",
            "param_type": "input"
        },
    "shape": (32, 64),
    "ori_shape": (32, 64),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float16",
    "param_type": "input"
    },
    {"shape": (32, 64),
    "ori_shape": (32, 64),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float16",
    "param_type": "output"}
})
```

```
"format": "ND",
"ori_format": "ND",
"dtype": "float16",
"param_type": "output"
}
,"case_name": "test_add_case_1",
"calc_expect_func": np_add
})
if __name__ == '__main__':
    ut_case.run("Ascend910",None,"ca","/home/allan/Ascend/toolkit/tools/simulator")

Note:

ut_case.run: For details, see run API.
In the preceding example, the values of the operator inputs are not set. By
default, the framework calls np.random.uniform(value_range,
size=shape).astype(dtype) to automatically generate the input data for each
input.
The value_range is defaulted to [0.1, 1.0]. The values of shape and dtype are consistent with those set in params. value_range can also be specified as follows.

{  
"shape": (32, 64),
"ori_shape": (32, 64),
"format": "ND",
"ori_format": "ND",
"dtype": "float16",
"param_type": "input",
"value_range": [2.0, 3.0]
}

You can also specify input values as follows.

{  
"shape": (32, 64),
"ori_shape": (32, 64),
"format": "ND",
"ori_format": "ND",
"dtype": "float16",
"param_type": "input",
"value": np.zeros((32, 64), np.float16)
}

Example 2

The following is an example of the user-defined precision standard.

```python
from op_test_frame.common import precision_info
from op_test_frame.ut import OpUT

ut_case = OpUT("Add")
ut_case.add_precision_case(case={
"params": [
{
"shape": (32, 64),
"ori_shape": (32, 64),
"format": "ND",
"ori_format": "ND",
"dtype": "float16",
"param_type": "input"
}],
{
"shape": (32, 64),
"ori_shape": (32, 64),
"format": "ND",
"ori_format": "ND",
"dtype": "float16",
"param_type": "input"
}
```

MindStudio
User Guide
8 Custom Operator Development

Copyright © Huawei Technologies Co., Ltd.
run API

- **Prototype**
  
  `OpUT.run(soc, case_name=None, simulator_mode=None, simulator_lib_path=None)`

- **Description**
  
  Runs test cases.

- **Parameters**
  
  - `soc`: the the Ascend AI Processor to run test cases. The value is the name of the corresponding .ini file in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/arch-linux/atc/data/platform_config` directory. `support_soc` can be a string or a tuple or list of strings (indicating multiple SoCs are supported).
  
  - `case_name`: the test case to be executed. Set this parameter to the value of `case_name` in `add_case` or `add_precision_case`.
  
  - `simulator_lib_path`: path of the simulator libraries.

  The path structure is as follows.

  ```
  simulator_lib_path/
  Ascend910/
  lib/
  libpv_model.so
  .....
  Ascend310/
  lib/
  libpv_model.so
  ...
  ```

- **NOTICE**

  If MindStudio is used to run UT cases, `OpUT.run` does not need to be called manually.

### 8.7.3.2.2 BroadcastOpUT

#### Overview

Inherits the capabilities of OpUT.
BroadcastOpUT defines test cases for Broadcast operators with dual inputs and single output, for example, the Add and Mul operators. It also provides more convenient APIs for these operators. For example, when creating the test case file for operator build, you do not need to set the format in some simple scenarios.

Definition of the BroadcastOpUT Test Class

- **Prototype**
  
  `BroadcastOpUT(op_type, op_module_name=None, op_func_name=None)`

- **Parameters**
  
  - `op_type`: operator type.
  - `op_module_name`: operator module name, that is, the name and path of the operator implementation file, for example, `impl.add` (the file path is `impl/add.py`). Defaults to `None`. The argument is generated automatically based on `op_type`. For example, for a BiasAdd operator, the generated module name is `impl.bias_add`. For details about how the argument of `op_module_name` is generated based on `op_type`, see [Step 4](#).
  
  - `op_func_name`: operator function name in the operator implementation file. Defaults to `None`. The argument is generated automatically based on `op_type`. For details about how the argument of `op_func_name` is generated based on `op_type`, see [Step 4](#). For example, for an Add operator, the corresponding API must exist in `impl/add.py`. The API definition is as follows.

    ```python
    @check_op_params(REQUIRED_INPUT, REQUIRED_INPUT, REQUIRED_OUTPUT, KERNEL_NAME)
    def add(input_x, input_y, output_z, kernel_name="add"):
    ```

  
  - `addBroadcastCase`

    - **Prototype**
      
      `BroadcastOpUT.add_broadcast_case(self, soc, input_1_info, input_2_info, output_info=None, expect=op_status.SUCCESS, case_name=None)`

    - **Description**
      
      Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an .o file.

    - **Parameters**
      
      - `soc`: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the `atc/data/platform_config` directory in the ATC installation path. `support_soc` can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to `all` or `None`, all SoCs are supported.
      
      - `input_1_info`: information about the first input of the operator, coming in either of the following format:

        - `[dtype, shape, format, ori_shape, ori_format]`
        
        - `[dtype, shape, format]: In this format, the values of `ori_shape` and `ori_format` are the same as those of `shape` and `format`.
      
      - `input_2_info`: information about the second input of the operator, with the same information of `input_1_info`. 
- **output_info**: defaults to **None**. This parameter does not need to be set.
- **expect**: expected build result. Defaults to op_status.SUCCESS. It can also be set to RuntimeError, indicating an expected exception.
- **case_name**: defaults to **None**. The test framework automatically generates the argument of case_name.

Example

```python
ut_case.add_broadcast_case("all", ["float16", (32, 32), "ND"], ["float16", (32, 32), "ND")
```

# Expected exception UT case
```
ut_case.add_broadcast_case("all", ["float16", (31, 32), "ND"], ["float16", (32, 32), "ND"],
expect=RuntimeError)
```

### add_broadcast_case_simple

- **Prototype**
  ```python
  BroadcastOpUT.add_broadcast_case_simple(self, soc, dtypes, shape1, shape2, expect=op_status.SUCCESS, case_name=None)
  ```
- **Description**
  Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an .o file. This API is simpler than add_broadcast_case.
- **Parameters**
  - **soc**: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/atc/data/platform_config` directory. **support_soc** can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to all or **None**, all SoCs are supported.
  - **dtypes**: types of the data to be tested. If multiple data types are set, multiple test cases are added at a time.
  - **shape1**: shape of the first input of the operator.
  - **shape2**: shape of the second input of the operator.
  - **expect**: expected build result. Defaults to op_status.SUCCESS. It can also be set to RuntimeError, indicating an expected exception.
  - **case_name**: defaults to **None**. The test framework automatically generates the argument of case_name.

Compared with the add_broadcast_case API, this API sets the formats of all inputs to **ND**.

Example

```python
ut_case.add_broadcast_case_simple(["Ascend910", "Ascend310"], ["float16", "float32"], (32, 32), (32, 32))
```

The preceding test case implements the same function as calling the add_case API.

```python
ut_case.add_case(support_soc=["Ascend910", "Ascend310"], case={
  "params": {
    "shape": (32, 32),
    "ori_shape": (32, 32),
    "format": "ND",
  }
```


8.7.3.2.3 ElementwiseOpUT

Overview

Inherits the capabilities of OpUT.

ElementwiseOpUT defines test cases for element-wise operators with single input and single output, for example, the Abs and Square operators. It also provides more convenient APIs for these operators. For example, when creating the test case file for operator build, you do not need to set the format in some simple scenarios.

Definition of the ElementwiseOpUT Test Class

- Prototype
  
  ElementwiseOpUT(op_type, op_module_name=None, op_func_name=None)

- Parameters
  
  - op_type: operator type.
  
  - op_module_name: operator module name, that is, the name and path of the operator implementation file, for example, impl.add (the file path is impl/add.py). Defaults to None. The argument is generated automatically based on op_type. For example, for a MaximumGrad
operator, the generated module name is `impl.maximum_grad`. For
details about how the argument of `op_module_name` is generated based
on `op_type`, see **Step 4**.

- **op_func_name**: operator function name in the operator implementation
  file. Defaults to `None`. The argument is generated automatically based on
  `op_type`. For details about how the argument of `op_func_name` is
  generated based on `op_type`, see **Step 4**. For example, for an Add
  operator, the corresponding API must exist in `impl/add.py`. The API
definition is as follows.

```python
@check_op_params(REQUIRED_INPUT, REQUIRED_INPUT, REQUIRED_OUTPUT, KERNEL_NAME)
def add(input_x, input_y, output_z, kernel_name="add"):
```

### add_elewise_case

- **Prototype**
  
  ```python
  BroadcastOpUT.add_elewise_case(self, soc, param_info,
  expect=op_status.SUCCESS, case_name=None)
  ```

- **Description**
  
  Adds test cases for operator build, tests whether an operator meets the
  related specifications, and builds an .o file.

- **Parameters**

  - **soc**: used to test whether the test case file supports the corresponding
    the Ascend AI Processor. The value is the name of the corresponding .ini
    file in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/
    {version}/{arch}-linux/atc/data/platform_config` directory. `support_soc`
    can be a string or a tuple or list of strings (indicating multiple SoCs are
    supported). If this parameter is set to **all** or **None**, all SoCs are supported.

  - **param_info**: information about the operator input, coming in either of
    the following format:

    - `[dtype, shape, format, ori_shape, ori_format]`
    - `[dtype, shape, format]`: In this format, the values of `ori_shape` and
      `ori_format` are the same as those of `shape` and `format`.

  - **expect**: expected build result. Defaults to `op_status.SUCCESS`. It can also
    be set to `RuntimeError`, indicating an expected exception.

  - **case_name**: defaults to `None`. The test framework automatically
    generates the argument of `case_name`.

- **Example**

  ```python
  ut_case.add_elewise_case("Ascend910", ["float16", (32, 32), "ND"])  
  ```

  The preceding test case implements the same function as calling the
  `add_case` API.

```python
ut_case.add_case(support_soc="Ascend910", case=
"params": [{
  "shape": (32, 32),
  "ori_shape": (32, 32),
  "format": "ND",
  "ori_format": "ND",
  "dtype": "float16"
},
  {"shape": (32, 32),
  "ori_shape": (32, 32),
  "format": "ND"},
```

---

**add_elewise_case**

- Prototype
  
  ```python
  BroadcastOpUT.add_elewise_case(self, soc, param_info,
  expect=op_status.SUCCESS, case_name=None)
  ```

- Description
  
  Adds test cases for operator build, tests whether an operator meets the
  related specifications, and builds an .o file.

- Parameters

  - **soc**: used to test whether the test case file supports the corresponding
    the Ascend AI Processor. The value is the name of the corresponding .ini
    file in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/
    {version}/{arch}-linux/atc/data/platform_config` directory. `support_soc`
    can be a string or a tuple or list of strings (indicating multiple SoCs are
    supported). If this parameter is set to **all** or **None**, all SoCs are supported.

  - **param_info**: information about the operator input, coming in either of
    the following format:

    - `[dtype, shape, format, ori_shape, ori_format]`
    - `[dtype, shape, format]`: In this format, the values of `ori_shape` and
      `ori_format` are the same as those of `shape` and `format`.

  - **expect**: expected build result. Defaults to `op_status.SUCCESS`. It can also
    be set to `RuntimeError`, indicating an expected exception.

  - **case_name**: defaults to `None`. The test framework automatically
    generates the argument of `case_name`.

- **Example**

  ```python
  ut_case.add_elewise_case("Ascend910", ["float16", (32, 32), "ND"])  
  ```

  The preceding test case implements the same function as calling the
  `add_case` API.

```python
ut_case.add_case(support_soc="Ascend910", case=
"params": [{
  "shape": (32, 32),
  "ori_shape": (32, 32),
  "format": "ND",
  "ori_format": "ND",
  "dtype": "float16"
},
  {"shape": (32, 32),
  "ori_shape": (32, 32),
  "format": "ND"},
```
add_elewise_case_simple

- **Prototype**

  `BroadcastOpUT.add_elewise_case_simple(self, soc, dtypes, shape, expect=op_status.SUCCESS, case_name=None)`

- **Description**

  Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an .o file.

- **Parameters**

  - `soc`: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/atc/data/platform_config` directory. `support_soc` can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to all or None, all SoCs are supported.

  - `dtypes`: types of the data to be tested. If multiple data types are set, multiple test cases are added at a time.

  - `shape`: shape of the operator input.

  - `expect`: expected build result. Defaults to `op_status.SUCCESS`. It can also be set to `RuntimeError`, indicating an expected exception.

  - `case_name`: defaults to `None`. The test framework automatically generates the argument of `case_name`.

  Compared with the `add_elewise_case` API, this API sets the formats of all inputs to `ND`.

- **Example**

  ```python
  ut_case.add_elewise_case_simple("Ascend910", ["float16", "float32"], [32, 32])
  ```

  The preceding test case implements the same function as calling the `add_case` API.

  ```python
  ut_case.add_case(support_soc="Ascend910", case={
  "params": [(
  "shape": (32, 32),
  "ori_shape": (32, 32),
  "format": "ND",
  "ori_format": "ND",
  "dtype": "float16"
  ),
  "shape": (32, 32),
  "ori_shape": (32, 32),
  "format": "ND",
  "ori_format": "ND",
  "dtype": "float16"
  )]
  })
  ```

  ```python
  ut_case.add_case(support_soc="Ascend910", case={
  "params": [(
  "shape": (32, 32),
  "ori_shape": (32, 32),
  "format": "ND",
  "ori_format": "ND",
  "dtype": "float32"
  )]
  })
  ```
8.7.3.2.4 ReduceOpUT

Overview

Inherits the capabilities of OpUT.

ReduceOpUT defines test cases for Reduce operators, for example, the ReduceSum and ReduceMean operators. It also provides more convenient APIs for these operators. For example, when creating the test case file for operator build, you do not need to set the format in some simple scenarios.

Definition of the ReduceOpUT Test Class

- Prototype
  ReduceOpUT(op_type, op_module_name=None, op_func_name=None)
- Parameters
  - op_type: operator type.
  - op_module_name: operator module name, that is, the name and path of the operator implementation file, for example, impl.add (the file path is impl/add.py). Defaults to None. The argument is generated automatically based on op_type. For example, for a ReduceSum operator, the generated module name is impl.reduce_sum. For details about how the argument of op_module_name is generated based on op_type, see Step 4.
  - op_func_name: operator function name in the operator implementation file. Defaults to None. The argument is generated automatically based on op_type. For details about how the argument of op_func_name is generated based on op_type, see Step 4. For example, for an Add operator, the corresponding API must exist in impl/add.py. The API definition is as follows.
  ```python
  @check_op_params(REQUIRED_INPUT, REQUIRED_INPUT, REQUIRED_OUTPUT, KERNEL_NAME)
  def add(input_x, input_y, output_z, kernel_name="add"):
  ```

add_reduce_case

- Prototype
  BroadcastOpUT.add_reduce_case(self, soc, input_info, axes, keep_dim=False, expect=op_status.SUCCESS, case_name=None)
- Description
  Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an .o file.
- Parameters
  - soc: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini
file in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/ {version}/{arch}-linux/atc/data/platform_config` directory. `support_soc` can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to `all` or `None`, all SoCs are supported.

- **input_info**: information about the operator input, coming in either of the following format:
  - `[dtype, shape, format, ori_shape, ori_format]`
  - `[dtype, shape, format]`: In this format, the values of `ori_shape` and `ori_format` are the same as those of `shape` and `format`.

- **axes**: dimensions to reduce.
- **keep_dims**: If `True`, retains reduced dimensions with length 1. Either `True` or `False`.
- **expect**: expected build result. Defaults to `op_status.SUCCESS`. It can also be set to `RuntimeError`, indicating an expected exception.
- **case_name**: defaults to `None`. The test framework automatically generates the argument of `case_name`.

Example

```python
ut_case.add_reduce_case("Ascend910", ["float16", (32, 32), "ND"], [0,], False)
```

The preceding test case implements the same function as calling the `add_case` API.

```python
ut_case.add_case(support_soc="Ascend910", case={
  "params": [{
    "shape": (32, 32),
    "ori_shape": (32, 32),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float16"
  }, {
    "shape": (32,),
    "ori_shape": (32,),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float16"
  }], [0,], False)
})
```

## add_reduce_case_simple

- **Prototype**

  `BroadcastOpUT.add_reduce_case_simple(self, soc, dtypes, shape, axes, keep_dim=False, expect=op_status.SUCCESS, case_name=None)`

- **Description**

  Adds test cases for operator build, tests whether an operator meets the related specifications, and builds an `.o` file.

- **Parameters**

  - **soc**: used to test whether the test case file supports the corresponding the Ascend AI Processor. The value is the name of the corresponding .ini file in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/ {version}/{arch}-linux/atc/data/platform_config` directory. `support_soc` can be a string or a tuple or list of strings (indicating multiple SoCs are supported). If this parameter is set to `all` or `None`, all SoCs are supported.
**dtypes**: types of the data to be tested. If multiple data types are set, multiple test cases are added at a time.

**shape**: shape of the operator input.

**axes**: dimensions to reduce.

**keep_dims**: If True, retains reduced dimensions with length 1. Either True or False.

**expect**: expected build result. Defaults to op_status.SUCCESS. It can also be set to RuntimeError, indicating an expected exception.

**case_name**: defaults to None. The test framework automatically generates the argument of case_name.

Compared with the add_reduce_case API, this API sets the formats of all inputs to ND.

Example

```python
ut_case.add_reduce_case_simple("Ascend910", ["float16", "float32"], [32, 32], [1,], True)
```

The preceding test case implements the same function as calling the add_case API.

```python
ut_case.add_case(support_soc="Ascend910", case={
  "params": [{
    "shape": (32, 32),
    "ori_shape": (32, 32),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float16"
  }, {
    "shape": (32, 1),
    "ori_shape": (32, 1),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float16"
  }, [1,], True]
})
```

```python
ut_case.add_case(support_soc="Ascend910", case={
  "params": [{
    "shape": (32, 32),
    "ori_shape": (32, 32),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float32"
  }, {
    "shape": (32, 1),
    "ori_shape": (32, 1),
    "format": "ND",
    "ori_format": "ND",
    "dtype": "float32"
  }, [1,], True]
})
```

### 8.7.3.3 Creating and Running UT Cases

**Prerequisites**

The deliverables of the operator project have been generated in MindSpore. For details, see 8.7.1 Operator Implementation and Information Registration File and 8.7.2 Operator Prototype Definition.
**Procedure**

**Step 1** Create UT cases.

1. A UT case file can be created in any of the following ways:
   
   Right-click the root directory of the operator project and choose **New Cases > TBE UT Case** from the shortcut menu.
   
   If UT cases of the operator exist, right-click the **testcases** directory or **testcases > ut**, and choose **New Cases > TBE UT Case** from the shortcut menu to create UT cases.
   
2. In the operator selection window, select the target operator **cus_square** and click **OK**.

   **NOTE**
   
   If the UT cases of the operator already exist, the message "testcases/ut/ops_test/xx already exists. Do you want to overwrite?" is displayed.
   
   You can click **Overwrite** or **Cancel**.

   After the creation is complete, the **testcases** folder is generated in the root directory of the operator project. The directory structure is as follows.

   ```
   ├── MyOperator            // Root directory of the project
   │   ├──  testcases
   │   │       ├──  ut
   │   │           ├──  ops_test
   │   │                ├── test_cus_square_impl.py      // Test case file for the operator implementation code
   ```

**Step 2** Write UT cases in Python for the operator implementation code.

In the **testcases/ut/ops_test/test_cus_square_impl.py** file, write UT cases in Python for the operator implementation code. Execute the operator and compare the compute result with the expect result to test the operator logic.

```python
import sys
import numpy as np
from op_test_frame.ut import BroadcastOpUT    # Import the UT class based on the operator type.

# Instantiate the UT case. **ut_case** is a keyword of the UT framework and cannot be modified.
# Set the value of **op_func_name** the same as the name of operator function in **cus_square_impl.py**.
ut_case = BroadcastOpUT("cus_square_impl", op_func_name="CusSquareImpl")

# Customize a function for generating expected data.
def calc_expect_func(input_x, output_y):
    res = np.square(input_x["value"])  
    return [res, ]

# Add test cases.
ut_case.add_precision_case("all", {
    "params": [{"dtype": "float16", "format": "ND", "ori_format": "ND", "ori_shape": (32,), "shape": (32,),
    "param_type": "input"},
    {"dtype": "float16", "format": "ND", "ori_format": "ND", "ori_shape": (32,), "shape": (32,),
    "param_type": "output"}],
    "calc_expect_func": calc_expect_func
})

# If multiple test cases are defined, multiple **ut_case.add_precision_case** functions need to be defined.
ut_case.add_precision_case("all", {
    "params": [{"dtype": "float16", "format": "ND", "ori_format": "ND", "ori_shape": (16,2), "shape": (16,2),
```
1. Import the UT class based on the operator type. For details, see 8.7.3.2 APIs.
2. Instantiate the UT case. For details about how to use OpUT, see Definition of the OpUT Test Class.
3. Define the function for generating the expected data.
4. Add test cases.
   For details about how to use each test API, see 8.7.3.2 APIs.
   To compare the result with the expectation, call add_precision_case.

**Step 3** Run the UT cases of the operator implementation file.

You can run the UT cases of all operators in the current project or run the UT cases of a single-operator.

- Right-click the testcases/ut/ops_test folder and choose Run The Operator 'All' UT Impl with coverage from the shortcut menu to run the test cases of the operator implementation code in the folder.
- Right-click the testcases/ut/ops_test/test_cus_square_impl.py folder and choose Run The Operator 'operator_name' UT Impl with coverage to run test cases of implementation code of a single-operator.

When the case runs for the first time, the corresponding configuration dialog box is displayed. Configure the parameters and click Run. For details about how to modify the run configurations, see 12.3.1 Editing a Run Configuration.

### Table 8-23 Run configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the run configuration (user-defined).</td>
</tr>
<tr>
<td>Test Type</td>
<td>ut_impl is recommended.</td>
</tr>
<tr>
<td>Compute Unit</td>
<td>Compute unit.</td>
</tr>
<tr>
<td></td>
<td>- AI Core/Vector Core</td>
</tr>
<tr>
<td></td>
<td>- AI CPU (not available)</td>
</tr>
<tr>
<td></td>
<td>You can select different compute units to switch between the AI Core/Vector</td>
</tr>
<tr>
<td></td>
<td>Core and AI CPU UT configuration pages.</td>
</tr>
<tr>
<td>SoC Version</td>
<td>Current Ascend AI Processor version.</td>
</tr>
<tr>
<td></td>
<td>Only Ascend 910 AI Processors are supported in MindSpore.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Target          | Target environment.  
- **Simulator Function**: functional simulation environment.  
- **Simulator TMModel**: The scheduling pipeline of operator execution is quickly displayed, and the actual operator computing is not performed. Currently, only Ascend 910 AI Processors support this function. |
| Operator Name   | Test cases.  
- **all**: all test cases  
- Other values: test cases of a specific operator |
| ADK Machine     | SSH connection of the device where the ADK tool is located. For details about how to add an SSH connection, see [12.1 SSH Connection Management](#). 
**NOTE**  
Only Windows OSs are supported. |
| Case Names      | Test cases to run, that is, the Python UT cases of the operator implementation code. You can select or deselect all test cases. |

- Check the execution result.
  a. After the execution is complete, view the execution result in the Run log print window at the bottom.
  b. In the Run window, click the URL in `index.html` to view the UT case coverage, as shown in Figure 8-26. In the URL, `localhost` indicates the server IP of MindStudio.

*NOTE*
To view the test results of UT cases, you need to use a browser. If no browser is available, install one.

**Figure 8-26** Viewing the UT case coverage
![Coverage report: 100%](image)

The preceding figure shows that the UT case coverage of the operator is 100% and the UT verification is passed.

If the message shown in Figure 8-27 is displayed, rectify the fault by referring to [15.8.3 How Do I Allow Unsigned Access Requests?](#).
c. Click the operator on the HTML page to navigate to the UT case coverage dialog box, as shown in Figure 8-28. The green and red labels are used to indicate the coverage rates.

Figure 8-28 UT coverage details

```
Coverage for /home/yourdirectory/mindspore_operator/sample/mindspore/impl/cus_square_impl.py: 100%
21 statements 21 run
```

```python
1. from future import absolute_import
2. from te import two
3. from topi import generic
4. import test_long_case
5. from topi.cce import util
6. from mindspore.ops.op_info_register import op_info_register, TENSOR_IN, Data
```

Figure 8-27 Error message

![Error message]

---

8.7.4 ST

Overview

MindStudio provides an upgraded ST framework to automatically generate test cases and verify MindSpore operator functionality and compute accuracy in a real hardware environment. The framework supports the following features:

- If **Target** is set to **Simulator_TMMModel** during running information configuration, you can view the execution pipeline, as follows.

- **NOTE**

  Before the processor performs computation, Vector Unit, Cube Unit, MTE1, MTE2, MTE3, and more units are initialized. As a result, data is generated on each unit before data movement progress is displayed in the timeline.
Generates an operator test case definition file (.json) based on the operator implementation and information file (*.impl.py), as the input of operator ST.

Generates the ST test data and test case execution code based on the operator test case definition file, for executing the test cases in the hardware environment.

Prerequisites

- The custom operator has been developed. For details, see 8.7.1 Operator Implementation and Information Registration File and 8.7.2 Operator Prototype Definition.
- MindSpore has been installed on the server and Ascend 910 AI Processor is available.

Generating an ST Case Definition File

**Step 1** Create ST cases.

Three portals are available.

- Right-click the root directory of the operator project and choose **New Cases > ST Case** from the shortcut menu.
- Right-click the operator implementation and operator information file `{project name}/mindspore/impl/cus_square_impl.py`, and choose **New Cases > ST Case**.
- If ST cases of the operator exist, right-click the testcases or testcases > st directory, and choose **New Cases > ST Case** from the shortcut menu to add ST cases.

**Step 2** In the **Create ST Cases for an Operator** dialog box, select the operator for which the ST case needs to be created.

The following figure shows an example.

![Operator selection dialog](image)

Select an operator name from the **Operator** drop-down list box.

Click **OK**. A test case definition file with empty shapes is generated. The following figure shows the **Design Cases** dialog box.

![Design Cases dialog](image)
You need to configure the shape information to generate test data and test cases. Configure the rest fields as required. Click **Add** to add test cases. For details about each field, see Table 8-25.

**Step 3** Click **Save** to save the modification to the operator test case definition file.

The operator test case definition file (named as `OpType_case_timestamp.json`) is stored in the `testcases/st/OpType/` directory under the root directory of the operator project.

**NOTE**

Strictly follow the naming rule of the operator test case definition file. Do not name irrelevant files in this format to the `testcases/st/OpType/` directory under the root directory of the operator project. Otherwise, file parsing errors may occur.

---End

**Running ST Cases**

**Step 1** Right-click the generated ST case definition file (`testcases > st > cus_square > xxxx.json`) and choose **Run The Operator 'xxx' ST Case** from the shortcut menu.
### Table 8-24 Run configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Set the name of the run configuration (user-defined).</td>
</tr>
<tr>
<td>Test Type</td>
<td>Select <strong>st_cases</strong>.</td>
</tr>
</tbody>
</table>
| Execute Mode   | • **Remote Execute**  
|                | • **Local Execute**  
|                | **NOTE**  
|                | **Local Execute** does not apply to Windows OSs. |
| SSH Connection | If **Execute Mode** is set to **Remote Execute**, select an SSH connection from the drop-down list box.  
|                | You can also click **[+]** to add a new SSH connection. For details about how to add an SSH configuration, see **12.1 SSH Connection Management**. |
| ADK Machine    | SSH connection of the device where the ADK tool is located. For details about how to add an SSH connection, see **12.1 SSH Connection Management**.  
|                | **NOTE**  
|                | This parameter applies only to Windows OSs. |
| Operator Name  | Select the operator to test. |
| Executable File Name | Select the test case definition file to run from the drop-down list box. |
| Target OS      | For Ascend EP, select the host-side OS where the Ascend AI Processor is located.  
|                | For Ascend RC, select the OS of the board environment.  
|                | Currently, only the EP form is supported. |
| Target Architecture | Select the OS architecture of the target environment. |
| Case Names     | Select the test cases to run.  
<p>|                | All cases are selected by default. You can deselect unnecessary cases. |
| Advanced Options | Specify advanced options. |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT Log Level</td>
<td>Select an ACT log level.</td>
</tr>
<tr>
<td></td>
<td>• INFO</td>
</tr>
<tr>
<td></td>
<td>• DEBUG</td>
</tr>
<tr>
<td></td>
<td>• WARNING</td>
</tr>
<tr>
<td></td>
<td>• ERROR</td>
</tr>
<tr>
<td></td>
<td>• NULL</td>
</tr>
<tr>
<td>Precision Mode</td>
<td>Set the precision mode. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>• force_fp176</td>
</tr>
<tr>
<td></td>
<td>• allow_mix_precision</td>
</tr>
<tr>
<td></td>
<td>• allow_fp32_to_fp16</td>
</tr>
<tr>
<td></td>
<td>• must_keep_origin_dtype</td>
</tr>
<tr>
<td>Device Id</td>
<td>Set the ID of the device that runs the ST. Specify the ID of the AI processor in use.</td>
</tr>
</tbody>
</table>

*NOTE*
- The ST supports the setting and query of the board log level. For details, see 12.6 Log Management.
- Windows OSs do not support the Local Execute function.

**Step 2** Click Run.

MindStudio generates test data and test code in `/testcases/st/out/OpType` under the operator project root director based on the operator test case definition file, and executes test cases on the specified hardware backend. The execution result will be printed to the Output window.

****End****

**Table 8-25** MindSpore Operator test case definition file (.json)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>case_name</td>
<td>Test case name.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>st_mode</td>
<td>ST test mode. Must be &quot;ms_python_train&quot;, indicating a MindSpore operator project.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>op</td>
<td>Operator type. Must not be left empty.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------</td>
<td>----------</td>
</tr>
<tr>
<td>input_desc</td>
<td>Operator input description.</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td><strong>NOTICE</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The parameters in <code>input_desc</code> must have the same number of values. Otherwise, test case generation fails.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>For example, if input 0 supports two formats, input 1 should also support two formats.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The rule also applies to <code>type</code>, <code>shape</code>, <code>data_distribute</code>, and values of <code>value_range</code> of the inputs.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>Input data type.</td>
<td>String or 1D array of strings</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>- <code>bool</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <code>int8</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <code>uint8</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <code>int16</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <code>uint16</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <code>int32</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <code>int64</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <code>uint32</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <code>uint64</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <code>float16/fp16/half</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <code>float32/float</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <code>double</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <code>UNDEFINED</code>: reserved. If this value is used, <code>format</code> needs to be set to <code>RESERVED</code>, indicating that the corresponding operator input is optional.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>shape</td>
<td>Input shape.</td>
<td>int 1D or 2D array</td>
<td>Yes</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>data_distribute</td>
<td>Data distribution modes for generating test data.</td>
<td>String or 1D array of strings</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>- <strong>uniform</strong>: returns random values that are evenly distributed.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>normal</strong>: returns random values of the normal distribution (Gaussian distribution).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>beta</strong>: returns random values of Beta distribution.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>laplace</strong>: returns random values of Laplace distribution.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>triangular</strong>: returns random values of the triangular distribution.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>relu</strong>: returns random values that are evenly distributed and activated by the ReLU function.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>sigmoid</strong>: returns random values that are evenly distributed and activated by the sigmoid function.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>softmax</strong>: returns random values that are evenly distributed and activated by softmax function.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>tanh</strong>: returns random values that are evenly distributed and activated by tanh function.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>value_range</td>
<td>Value range. Must not be left empty.</td>
<td>Int or float 1D or 2D array</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Formatted as <code>[min_value, max_value]</code>, with <code>min_value &lt;= max_value</code>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>output_desc</td>
<td>Operator output description. NOTICE The values of output_desc must one-to-one map to those in input_desc. Otherwise, test case generation fails. For example, if input x supports two formats, the output should also support two formats.</td>
<td>String or 1D array of strings</td>
<td>Yes</td>
</tr>
<tr>
<td>type</td>
<td>Output data type.</td>
<td>int16</td>
<td>String or 1D array of strings</td>
</tr>
<tr>
<td></td>
<td>• bool</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• int8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• uint8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• int16</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• uint16</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• int32</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• int64</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• uint32</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• uint64</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• float16/fp16/half</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• float32/float</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• double</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• UNDEFINED: reserved. If this value is used, format needs to be set to RESERVED, indicating that the corresponding operator input is optional.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>shape</td>
<td>Output shape.</td>
<td>int 1D or 2D array</td>
<td>Yes</td>
</tr>
<tr>
<td>attr</td>
<td></td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>name</td>
<td>Attribute name. Must not be left empty.</td>
<td>String</td>
<td>Required if attr is configured.</td>
</tr>
<tr>
<td>type</td>
<td>Attribute type.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· bool</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· int</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· float</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· string</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· list_bool</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· list_int</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· list_float</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· list_string</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· list_list_int</td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>Attribute value. Varies according to type.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· bool: true/false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· int: 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· float: 1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· string: &quot;NCHW&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· list_bool: [false, true]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· list_int: [1, 224, 224, 3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· list_float: [1.0, 0.0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· list_string: [&quot;str1&quot;, &quot;str2&quot;]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· list_list_int: [[1, 3, 5, 7], [2, 4, 6, 8]]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>String</td>
<td>Required if attr is configured. Must not be null.</td>
<td></td>
</tr>
</tbody>
</table>
8.8 AI CPU Operator Development (TensorFlow)

8.8.1 Operator Code Implementation

Overview

The implementation of an AI CPU operator consists of:

- Header file (.h) that declares the operator class. The custom operator class inherits the CpuKernel base class.
- Source file (.cc) that has rewritten the Compute function in the operator class to implement the operator compute logic.

Header File Code Module

Declare the operator class in the cpukernel/impl/reshape_cust_kernel.h file of the operator project. The code module is as follows.

```cpp
#ifndef _RESHAPE_CUST_KERNELS_H_
#define _RESHAPE_CUST_KERNELS_H_
#include "cpu_kernel.h"        // Defines CpuKernel base class and registration macro.
// Include the following header files as needed:
#include "cpu_tensor.h"        // Defines Tensor and related methods.
#include "cpu_tensor_shape.h"  // Defines TensorShape class and related methods.
#include "cpu_types.h"         // Defines data types and formats.
#include "cpu_attr_value.h"    // Defines AttrValue and related methods.
namespace aicpu {        // Defines the namespace aicpu.
class ReshapeCustCpuKernel : public CpuKernel {        // Specifies that the operator class inherits the CpuKernel base class.
public:
    ~ReshapeCustCpuKernel() = default;
    virtual uint32_t Compute(CpuKernelContext &ctx) override;    // Declares the Compute function, which has been rewritten.
};
} // namespace aicpu
#endif
```

- Include related header files.
  - cpu_kernel.h defines the AI CPU operator base class CpuKernel and the kernel registration macro.
  - cpu_tensor.h defines the AI CPU Tensor class and related methods.
  - cpu_tensor_shape.h defines the AI CPU TensorShape class and related methods.
  - cpu_types.h defines the AI CPU data types and formats.
  - cpu_attr_value.h defines the AttrValue class and related methods.
- Declare the operator class, which is derived from the CpuKernel class. Declare the overloading function Compute, which needs to be implemented in the operator implementation file. For details, see Source File Code Module. The operator class must be defined within the namespace aicpu. The namespace name is fixed to aicpu and cannot be modified.
Source File Code Module

Implement the operator compute logic in the `cpukernel/impl/reshape_cust_kernel.cc` file of the operator project. The code module is as follows.

```cpp
#include "reshape_cust_kernel.h"  // Includes the header files declaring the ReshapeCust operator class.

namespace {
    const char *RESHAPE_CUST = "ReshapeCust";  // The operator OpType is ReshapeCust class.
}

namespace aicpu {  // Defines the namespace aicpu.
    uint32_t ReshapeCustCpuKernel::Compute(CpuKernelContext &ctx)  // Implements the Compute function of the custom operator class.
    {
        return 0;
    }

}  // namespace aicpu

1. Include related header files.

   The header file `reshape_cust_kernel.h` is declared in Header File Code Module. If the following header files have been included in Header File Code Module, you do not need to include them again.
   - `cpu_kernel.h` defines the AI CPU operator base class `CpuKernel` and the kernel registration macro.
   - `cpu_tensor.h` defines the AI CPU `Tensor` class and related methods.
   - `cpu_tensor_shape.h` defines the AI CPU `TensorShape` class and related methods.
   - `cpu_types.h` defines the AI CPU data types and formats.
   - `cpu_attr_value.h` defines the `AttrValue` class and related methods.

2. Define the namespace and declare that the constant pointer points to the operator OpType.

   See the following example.

   ```cpp
   namespace {
        const char *RESHAPE_CUST = "ReshapeCust";
   }

   ReshapeCust is the operator OpType, and RESHAPE_CUST is the declared constant pointer that points to the operator OpType.

3. Define the namespace aicpu, implement the Compute function of custom operator within the namespace aicpu, and define the compute logic of the operator.

   The namespace name is fixed to aicpu. The base class and related definitions are all within the aicpu namespace.

   - Declare the Compute function.
     ```cpp
     uint32_t ReshapeCustCpuKernel::Compute(CpuKernelContext &ctx)
     ```

     ReshapeCustCpuKernel is the custom operator class defined in the header file. The formal parameter `CpuKernelContext` is the context of the CPU kernel, including the input and output tensors and attributes of the operator.

     - According to your operator development requirements, edit the Compute function body to obtain the input tensors, organize the compute logic
based on the inputs, obtain the output result, and set the output into the output tensor.

4. Register the kernel implementation of the operator:

```c
REGISTER_CPU_KERNEL(RESHAPE_CUST, ReshapeCustCpuKernel);
```

- **SAMPLE** is a string pointer that points to the operator **OpType** defined in 2.
- **ReshapeCustCpuKernel** is the name of the custom operator class.

## Compute Function Implementation

The key code for implementing an AI CPU operator is the implementation code of the Compute function. The ReshapeCust operator copies the input tensor data to the output tensor. The shape information of the output tensor is inferred by **InferShape** defined in the operator prototype. A sample code of operator implementation is as follows.

```c
namespace aicpu {
    uint32_t ReshapeCustCpuKernel::Compute(CpuKernelContext &ctx) {
        Tensor *input_tensor = ctx.Input(0);    // Obtain information such as the input shape and data based on the obtained input_tensor.
        if (input_tensor == nullptr) {
            return -1;
        }
        Tensor *output_tensor = ctx.Output(0);    // Obtain information such as the output shape and data based on the obtained output_tensor.
        if (output_tensor == nullptr) {
            return -1;
        }
        // Obtain the data address of the input tensor.
        auto input_data = input_tensor->GetData();
        if (input_data == nullptr) {
            return -1;
        }
        // Obtain the data address of the output tensor.
        auto output_data = output_tensor->GetData();
        if (output_data == nullptr) {
            return -1;
        }
        // Copy the data of the input tensor to the output tensor.
        uint64_t data_size = input_tensor->GetDataSize();
        memcpy(output_data, input_data, data_size);
        return 0;
    }
}
```

## 8.8.2 Operator Prototype Definition

Go to the **op_proto** directory of the operator project, compile the IR implementation files **reshape_cust.h** and **reshape_cust.cc**, and register the operator with the operator prototype library. During network execution, GE calls the verification API of the operator prototype library to verify operator arguments. If the verification passes, GE infers the output **shape** and **dtype** of each node by calling the inference function of the operator prototype library and allocates static memory for the result tensor.
reshape_cust.h

MindStudio generates the operator registration code to the reshape_cust.h file. You can modify the code as required. The prototype definition of the ReshapeCust operator is as follows.

```c
#ifndef GE_OP_INTERP_RESHAPE_CUST_H
#define GE_OP_INTERP_RESHAPE_CUST_H
#include "graph/operator_reg.h"
namespace ge {
REG_OP(ReshapeCust)
  .INPUT(tensor, TensorType({DT_BOOL, DT_FLOAT16, DT_FLOAT, DT_INT8, DT_INT32, DT_UINT32, DT_UINT8, DT_INT64, DT_UINT64, DT_INT16, DT_UINT16, DT_DOUBLE, DT_COMPLEX64, DT_COMPLEX128, DT_QINT8, DT_QUINT8, DT_QINT16, DT_QUINT16, DT_QINT32}))
  .INPUT(shape, TensorType({DT_INT32, DT_INT64}))
  .OUTPUT(output, TensorType({DT_BOOL, DT_FLOAT16, DT_FLOAT, DT_INT8, DT_INT32, DT_UINT32, DT_UINT8, DT_INT64, DT_UINT64, DT_INT16, DT_UINT16, DT_DOUBLE, DT_COMPLEX64, DT_COMPLEX128, DT_QINT8, DT_QUINT8, DT_QINT16, DT_QUINT16, DT_QINT32}))
  .OP_END_FACTORY_REG(ReshapeCust)
}
#endif // GE_OP_INTERP_RESHAPE_CUST_H
```

- **Add in REG_OP(ReshapeCust)** indicates the operator type that is registered with the Ascend AI Processor. The operator type must be consistent with that in REGISTER_CUSTOM_OP("ReshapeCust") in **8.8.4 Operator Plugin Implementation**.
- **.INPUT** and **.OUTPUT** indicate the names and data types of the input and output tensors of the operator. The input and output sequence must be consistent with the function parameter sequence in **8.8.1 Operator Code Implementation** as well as that in **8.8.3 Operator Information Definition**.

reshape_cust.cc Implementation

The key of prototype definition is to infer the output shape. The principle of the output shape of the ReshapeCust operator is as follows: Obtain the input tensor and the target shape, check whether the element count of the input tensor is the same as that of the target shape. If yes, the target shape is set to the output shape.

MindStudio has generated a code template file in the reshape_cust.cc file. You can modify the file as required. The implementation code of the reshape_cust.cc file is as follows.

```c
#include "reshape_cust.h"  //IR registration header file
#include <vector>          //Vector templates can be used and APIs of class vector can be called.
#include <string>          //The string class is part of the C++ standard library. With the header file included, string objects can be constructed and string APIs can be called.
#include <iostream>        //The iostream class is part of the C++ standard library. With the header file included, input and output stream APIs can be called.
namespace {
// Obtain the target shape.
template <typename T>
std::vector<int64_t> AsInt64(const T *data, int64_t data_size) {
  std::vector<int64_t> ret(data_size);
  for (int64_t i = 0; i < data_size; ++i) {
    ret[i] = data[i];
  }
  return ret;
}
```
// Obtain the element count based on the shape.
int64_t GetElementNum(const std::vector<int64_t> &shape) {
    int64_t ret = 1;
    for (size_t i = 0; i < shape.size(); ++i) {
        ret *= shape[i];
    }
    return ret;
}

namespace ge {

IMPLEMT_COMMON_INFERFUNC(ReshapeCustInferShape) {
    TensorDesc tensordesc_tensor = op.GetInputDesc("tensor");
    TensorDesc tensordesc_shape = op.GetInputDesc("shape");
    TensorDesc tensordesc_output = op.GetOutputDesc("output");
    Tensor shape_tensor;
    // Obtain the value of the target shape.
    if (op.GetInputConstData("shape", shape_tensor) == GRAPH_SUCCESS) {
        DataType shape_type = tensordesc_shape.GetDataType();
        std::vector<int64_t> shape_values;
        if (shape_type == DT_INT32) {
            auto shape_data = reinterpret_cast<const int32_t *>(shape_tensor.GetData());
            shape_values = AsInt64<int32_t>(shape_data, shape_tensor.GetSize());
        } else {
            auto shape_data = reinterpret_cast<const int64_t *>(shape_tensor.GetData());
            shape_values = AsInt64<int64_t>(shape_data, shape_tensor.GetSize());
        }
        // Check whether the target shape is valid.
        std::vector<int64_t> input_shape = tensordesc_tensor.GetShape().GetDims();
        int64_t input_element_num = GetElementNum(input_shape);
        int64_t shape_element_num = GetElementNum(shape_values);
        if (input_element_num != shape_element_num) {
            return GRAPH_FAILED;
        }
        // Set the shape of the output tensor.
        tensordesc_output.SetShape(Shape(input_shape));
        tensordesc_output.SetOriginShape(Shape(input_shape));
    }
    tensordesc_output.SetDataType(tensordesc_tensor.GetDataType());

    std::vector<std::pair<int64_t, int64_t>> range;
    auto status = op.GetInputDesc("tensor").GetShapeRange(range);
    if (status != GRAPH_SUCCESS) {
        return GRAPH_FAILED;
    }
    tensordesc_output.SetShapeRange(range);
    (void)op.UpdateOutputDesc("output", tensordesc_output);
    return GRAPH_SUCCESS;
}

COMMON_INFER_FUNC_REG(ReShapeCust, ReshapeCustInferShape);

8.8.3 Operator Information Definition

Principle

The operator information library is one of the deliverables for operator
development. It mainly describes the implementation specifications of an operator
on the Ascend AI Processor, including the input and output types and names
of the operator. During network execution, basic verification is performed based on the operator information in the operator information library, in addition to operator mapping.

**Description**

Go to the `cpukernel/op_info_cfg/aicpu_kernel` directory and configure the operator information library file `reshape_cust.ini`. You need to modify the `reshape_cust.ini` file automatically generated by MindStudio. The following shows the operator information definition of the modified ReshapeCust operator.

```plaintext
[ReshapeCust]
opInfo.engine=DNN_VM_AICPU
opInfo.flagPartial=False
opInfo.computeCost=100
opInfo.flagAsync=False
opInfo.opKernelLib=CUSTAICPUKernel
opInfo.kernelSo=libcust_aicpu_kernels.so
opInfo.functionName=RunCpuKernel
opInfo.workspaceSize=1024
```

**Table 8-26** describes the parameters. Only the common configuration options of the operator information library are listed. For details about other configuration options, see section "Operator Information Library Definition" in *AI CPU Custom Operator Development Guide*.

**Table 8-26** Operator information definition of the ReshapeCust operator

<table>
<thead>
<tr>
<th>Information</th>
<th>Configuration Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[OpType]</td>
<td>[ReshapeCust]</td>
<td>Operator type, which is included in brackets to mark the start of operator information. The operator type is <strong>ReshapeCust</strong> according to 8.3 Operator Analysis.</td>
</tr>
<tr>
<td>opInfo.engine</td>
<td>DNN_VM_AICPU</td>
<td>Operator engine. Fixed to <strong>DNN_VM_AICPU</strong> for AI CPU custom operators.</td>
</tr>
<tr>
<td>opInfo.flagPartial</td>
<td>False</td>
<td>Reserved, fixed to <strong>False</strong>.</td>
</tr>
<tr>
<td>opInfo.computeCost</td>
<td>100</td>
<td>Reserved, fixed to <strong>100</strong>.</td>
</tr>
<tr>
<td>opInfo.flagAsync</td>
<td>False</td>
<td>Reserved, fixed to <strong>False</strong>.</td>
</tr>
<tr>
<td>opInfo.opKernelLib</td>
<td>CUSTAICPUKernel</td>
<td><strong>kernelLib</strong> called by the operator. Fixed to <strong>CUSTAICPUKernel</strong> for AI CPU custom operators.</td>
</tr>
<tr>
<td>opInfo.kernelSo</td>
<td>libcust_aicpu_kernels.so</td>
<td>Name of the .so file generated after the AI CPU operator is built.</td>
</tr>
</tbody>
</table>
8.8.4 Operator Plugin Implementation

You need to develop the operator adaptation plugin to interpret and map the operator in the TensorFlow network to one adapted to the Ascend AI Processor. MindStudio automatically generates the plugin code of the ReshapeCust operator in the `framework/tf_plugin/tensorflow_reshape_cust_plugin.cc` file.

- Include the header file.
  ```
  #include "register/register.h"
  ```

- Register the plug-in.
  ```
  using namespace ge;    // Add it manually.
  namespace domi { 
  // register op info to GE
  REGISTER_CUSTOM_OP("ReshapeCust")
  .FrameworkType(TENSORFLOW) // type: CAFFE, TENSORFLOW
  .OriginOpType("ReshapeCust") // name in tf module
  .ParseParamsByOperatorFn(AutoMappingByOpFn);
  .ImplyType(ImplyType::AI_CPU); // Add it manually.
  } // namespace domi
  ```
  - **REGISTER_CUSTOM_OP**: operator type registered with GE. According to the AI CPU Operator Analysis, the operator type is **ReshapeCust**.
  - **FrameworkType**: framework type. The source framework type is TensorFlow.
  - **OriginOpType**: operator type in the TensorFlow framework.
  - **ParseParamsByOperatorFn**: function for registering models to be parsed. The `AutoMappingByOpFn` function is used to automatically parse models.
  - **ImplyType**: implementation type of the operator. **ImplyType::AI_CPU** indicates that the operator is an AI CPU operator. Add it manually.

8.8.5 Operator Project Building

**Overview**

After the operator deliverable development is complete, you need to build the operator project to generate a custom OPP runfile (.run). The build workflow includes:
● Build the AI CPU operator code implementation files (.h and .cc) into `libcust_aicpu_kernels.so`.
● Build the AI CPU operator information library definition file (.ini) into `cust_aicpu_kernel.json`.
● Build the operator prototype definition files (.h and .cc) into `libcust_op_proto.so`.
● Build the operator adaptation plugin implementation file (.cc) into `libcust_tf_parsers.so`.

**NOTE**

Windows OSs do not support local build.

**Build Procedure**

**Step 1** In the MindStudio project window, select the operator project.

**Step 2** Choose **Build > Edit Build Configuration...** on the top menu bar.

**Step 3** Set parameters in the displayed **Build Configurations** dialog box, as shown in the following figure. For details, see **Table 8-27**.

The AI CPU operator project supports remote and local build.

**Figure 8-29 Remote Build**

![Remote Build Configuration](image)

**Figure 8-30 Local Build**

![Local Build Configuration](image)

**Table 8-27 Parameter description**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build Configuration</td>
<td>Name of the build configuration. Defaults to <strong>Build-Configuration</strong>.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Build Mode</td>
<td>Build mode.</td>
</tr>
<tr>
<td></td>
<td>● <strong>Remote Build:</strong> NOTICE g++ 7.5.0 is required for remote build.</td>
</tr>
<tr>
<td></td>
<td>● <strong>Local Build:</strong> The operator project is built on the MindStudio server. You can quickly locate the implementation code in MindStudio based on the build log for troubleshooting. In this mode, you need to configure the cross compilation environment by referring to <a href="#">2.4.5 Configuring the Compilation Environment</a>.</td>
</tr>
<tr>
<td>SSH Connection</td>
<td>This parameter is available only in <strong>Remote Build</strong> mode. Select an SSH configuration from the drop-down list box. If no connection is added, click <strong>+</strong> to add it. For details, see <a href="#">12.1 SSH Connection Management</a>.</td>
</tr>
<tr>
<td>Environment Variables</td>
<td>This parameter is available only in <strong>Remote Build</strong> mode. Configure the <strong>ASCEND_OPP_PATH</strong> and <strong>TOOLCHAIN_DIR</strong> environment variables for the remote environment.</td>
</tr>
<tr>
<td>Target OS</td>
<td>This parameter is available only in <strong>Local Build</strong> mode. For Ascend EP, select the host-side OS where the Ascend AI Processor is located. For Ascend RC, select the OS of the board environment.</td>
</tr>
<tr>
<td>Target Architecture</td>
<td>This parameter is available only in <strong>Local Build</strong> mode. Select the OS architecture of the target environment.</td>
</tr>
</tbody>
</table>

If remote build is used, you need to configure the SSH connection and environment variables of the remote server.

- Configuring the SSH connection
  For details about how to add an SSH connection, see [12.1 SSH Connection Management](#).
- Configuring the environment variables
  Set the environment variables **ASCEND_OPP_PATH**, **TOOLCHAIN_DIR**, **ASCEND_TENSOR_COMPILER_INCLUDE**, and **ASCEND_AICPU_PATH** in the Environment Variables text box.

```bash
ASCEND_OPP_PATH=/home/xxx/Ascend/ascend-toolkit/latest/opp;
TOOLCHAIN_DIR=/home/xxx/Ascend/ascend-toolkit/latest/toolkit/toolchain/hcc;
```
Step 4  Click **Build** to build the project.

Step 5  View the build result in the lower part of MindStudio. The custom OPP runfile `custom_opp_Target OS_Target Architecture.run` is generated in the `cmake-build` directory of the operator project.

For the value of **Target OS** and **Target Architecture**, see Step 3.

### 8.8.6 Local Operator Deployment

**NOTICE**

- This section describes how to deploy the custom OPP runfile. During the execution of 8.8.8 ST, an OPP runfile is automatically generated and deployed in the `opp` directory in the development environment. You can change the deploy path by referring to this section. If you have run operator ST by referring to 8.8.8 ST, you can safely skip this section.

- Before deploying operators, ensure that the custom OPP runfile `custom_opp_Target OS_Target Architecture.run` is generated by referring to 8.8.5 Operator Project Building.

- Windows OSs do not support local operator deployment.

Step 1  On the menu bar of the MindStudio project window, choose **Ascend > Deploy**. The operator deployment dialog box is displayed.

Step 2  Select **Deploy Locally** and click **Deploy**.

If the following information is displayed on the **Output** tab page, the custom operator is successfully deployed.
After the custom OPP is successfully installed, the custom operator is deployed in the /opp directory in the 
(Ascend-CANN-Toolkit installation path)/ascend-toolkit/{version}/{arch}-linux.

The directory structure is as follows.

```plaintext
├── opp      // OPP directory
│   ├── op_impl
│       ├── built-in
│       │   └── custom
│       │       ├── ai_core
│       │       │   └── cpu
│       │       │       └── aicpu_kernel/
│       │       │       └── custom_impl       // Custom operator implementation code
│       │       │       └── libcust_aicpu_kernels.so
│       │       └── config
│       │           └── cust_aicpu_kernel.json  // Custom operator info library file
│       └── vector_core   // Reserved directory, which can be ignored
│   ├── framework
│       ├── built-in
│       │   └── custom
│       │       ├── caffe
│       │       │   └── tensorflow        // Operator plug-in library for TensorFlow adaption
│       │       │       └── libcust_tf_parsers.so
│       │       └── npu_supported_ops.json   // File applicable to Ascend 910 AI Processors only, which can be ignored if Ascend 310 AI Processors are used
│       └── op_proto
│           ├── built-in
│           │   └── custom
│           │       └── op_proto.so    // Prototype library file of the custom operator
│           └── libcust_op_proto.so  // Prototype library file of the custom operator
```

8.8.7 Remote Operator Deployment

Deploy the custom OPP runfile custom_opp_Target OS_Target Architecture.run to the system OPP in the hardware environment with Ascend AI Processor to construct necessary conditions for operators execution on the network.

**Step 1** In the MindStudio project window, select the operator project.

**Step 2** On the top menu bar, choose Ascend > Deploy. The operator deployment dialog box is displayed.

Choose Deploy Remotely > SSH Connection from the drop-down list box to configure SSH connection. If the no SSH connection is not added in advance, click [+] to add it.

**Step 3** Configure the environment variable.

You can use either of the following methods:

- Configure the environment variables on the host of the hardware device where the Ascend AI Processor is located.
  Deploy the operator on the host as the running user in MindStudio. Before deploying the operator, ensure the following environment variable is configured on the host:

  a. Run the following command in the $HOME/.bashrc file on the host as the running user:

     ```bash
     export ASCEND_OPP_PATH=/home/xxx/Ascend/ascend-toolkit/latest/opp
     ```

     Replace `/home/xxx/Ascend/ascend-toolkit/latest/opp` with the actual path.
b. Run the following command to make the environment variable to take effect:

```
source ~/.bashrc
```

- Add the environment variable in **Environment Variables**. Type `ASCEND_OPP_PATH=/home/xxx/Ascend/ascend-toolkit/latest/opp` in the **Environment Variables** field.

  `home/xxx/Ascend/ascend-toolkit/latest` is the OPP installation path. Replace it with the actual path.

  You can also click the icon next to the text box and enter a value in the displayed dialog box.

  - Type `ASCEND_OPP_PATH` in the **Name** field.
  - Type `home/xxx/Ascend/ascend-toolkit/latest/opp` in the **Value** field.

---

**Figure 8-31** Operator deployment

- Select the target server for operator deployment and click **Deploy**.

**Step 5** Deploy the operators. Operator deployment is equivalent to installing the custom OPP generated in [8.6.5 Operator Project Building](#). After the deployment, the operator is deployed in the OPP installation path on the host. The default path is `/usr/local/Ascend/opp/`.

**Figure 8-32** Operator deployment log messages

After the custom OPP is deployed on the host, the directory structure is similar to the following:

```
├── opp      // OPP directory
│   ├── op_impl
│       ├── built-in
│       ├── custom
│           ├── ai_core
│           ├── cpu
```
### 8.8.8 ST

#### Overview

MindStudio provides an upgraded ST framework to automatically generate test cases, verify operator functionality and compute accuracy in real hardware environment, and generate an execution report. Feature details are as follows:

- Generates an operator test case definition file based on the operator information library.
- Generates test data of different shapes and dtypes and AscendCL-based test cases from the operator test case definition file.
- Builds the operator project, deploys the operators in the system OPP, and runs test cases in the hardware environment to verify the operator functionality.
- Generates an ST report (st_report.json) that displays information about test cases and phase-by-phase execution states.
- Generates a test function, compares the expected operator output and the actual operator output, and displays the comparison result to verify the compute accuracy.

#### Prerequisites

- Custom operator has been developed. For details, see 8.8.1 Operator Code Implementation, 8.8.2 Operator Prototype Definition, 8.8.3 Operator Information Definition, and 8.8.4 Operator Plugin Implementation.
- MindStudio has been connected to a hardware device.

#### Generating an ST Case Definition File

**Step 1** Create ST cases.

Three portals are available.

- Right-click the root directory of the operator project and choose New Cases > ST Case from the shortcut menu.
- Right-click the operator information definition file `{project name}/cpukernel/op_info_cfg/aicpu_kernel/xx.ini`, and choose New Cases > ST Case.
• If ST cases of the operator exist, right-click the **testcases** or **testcases > st** directory, and choose **New Cases > ST Case** from the shortcut menu to add ST cases.

**Step 2** In the **Create ST Cases for an Operator** dialog box, select the operator for which the ST case needs to be created.

![Operator Name and SoC Version Drop-down Lists](image)

Select an operator name from the **Operator Name** drop-down list.

Select the version of your the Ascend AI Processor from the **SoC Version** drop-down list. If the ST is performed on the AI CPU operator, **aicpu_kernel** is selected by default.

• If **Import operator info from a model** is not selected, click **OK** and an operator test case definition file with empty shapes is generated. The following figure shows the **Design Cases** dialog box.

![Design Cases Dialog Box](image)

You need to configure the shape information to generate test data and test cases. Configure the rest fields as required. For details, see **Field Description of the Operator Test Case Definition File**.
• If you select **Import operator info from a model** and upload a TensorFlow model file (.pb) that contains the operator or a model file in ONNX format, the top-layer shape of the obtained model is displayed.

You can also modify the shape information of the first-layer input in **Input Nodes Shape**.

After you click **OK**, the tool automatically dumps the shape information of the selected operator based on the shape information of the first layer and generates the corresponding operator test case definition file.

**NOTE**

To use this function, you need to install the TensorFlow framework and ONNX library in the operating environment. If the Windows OS is used, you need to install the TensorFlow framework and ONNX library on the local Windows host.

This file is used for generating test data and test cases. You can modify related fields. For details, see **Field Description of the Operator Test Case Definition File**.

**Step 3** To compare the expected data with the benchmark data, define and configure a function for generating the expected data of the operator.

1. Customize a test function for generating the expected operator result.

   The function is implemented on a framework such as TensorFlow or Caffe and has the same functionality as the custom operator. The function runs on the CPU and generates benchmark data. The operator accuracy is tested by comparing the benchmark data with the output data. The function for generating expected data is implemented using the Python language. Multiple expected operator data generation functions can be implemented in a Python file. Keep the inputs, outputs, and attributes (including the format, type, and shape) of this function consistent with those of the custom operator.

2. Edit the test case definition file.

   Configure the function in the test case definition file. You can configure it in the **Design** view or the **Text** view.

   – In the **Design** view, select a destination path of the Python file as the script path in **Expected Result Verification** dialog box. In **Script Function**, enter the name of the function that generates expected operator data.
In Script Function, you can choose to enter the function or leave it empty.

- If the name of the function is entered, the function is called to generate the benchmark data during ST.
- If the function name is left empty, the function with the same name as the custom operator is automatically matched to generate benchmark data during ST. If no function with the same name exists, a message indicating match failure is displayed.

In the Text view, if the `calc_expect_func_file` parameter is added, the value is the file path and name of the function generating the expected operator data. For example:

```
```

Where, `/home/teste/test_*.py` indicates the implementation file of the test function, and `function` indicates the function name. Separate the file path and function name with a colon (:).

Example: The `test_add.py` file is the expected data generation file of the Add operator. The function implementation is as follows:

```python
def calc_expect_func(input_x, input_y, out):
    res = input_x['value'] + input_y['value']
    return [res, ]
```

Note: You need to complete the expected data generation function of the operator based on the developed custom operator. The input parameters of the operator's expected data generation function must be the same as those of the operator in the operator code implementation.

**Step 4**  Click Save to save the modification to the operator test case definition file.

The operator test case definition file (named as `OpType_case_timestamp.json`) is stored in the `testcases/st/OpType/aicpu_kernel` directory under the root directory of the operator project.

---

**NOTE**

Strictly follow the naming rules of the operator test case definition file. Do not name irrelevant files in this format to the `testcases/st/OpType/aicpu_kernel` directory under the root directory of the operator project. Otherwise, file parsing errors may occur.

---End

**Running ST Cases**

**Step 1**  Right-click the ST case definition file generated in Generating an ST Case Definition File (testcases > st > add > aicpu_kernel > xxxx.json) and choose Run The Operator 'xxx' ST Case from the shortcut menu.

**Table 8-28** Run configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the run configuration (user-defined).</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Test Type</td>
<td>Select <strong>st_cases</strong>.</td>
</tr>
<tr>
<td>Execute Mode</td>
<td>• <strong>Remote Execute</strong></td>
</tr>
<tr>
<td></td>
<td>• <strong>Local Execute</strong></td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Local Execute</strong> does not apply to Windows OSs.</td>
</tr>
<tr>
<td>SSH Connection</td>
<td>If <strong>Execute Mode</strong> is set to <strong>Remote Execute</strong>, select an SSH connection</td>
</tr>
<tr>
<td></td>
<td>from the drop-down list box.</td>
</tr>
<tr>
<td></td>
<td>You can also click + to add a new SSH connection.</td>
</tr>
<tr>
<td></td>
<td>For details about how to add an SSH connection, see 12.1 SSH Connection</td>
</tr>
<tr>
<td></td>
<td>Management.</td>
</tr>
<tr>
<td>ADK Machine</td>
<td>SSH connection of the device where the ADK tool is located. For details</td>
</tr>
<tr>
<td></td>
<td>about how to add an SSH connection, see 12.1 SSH Connection Management.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong></td>
</tr>
<tr>
<td></td>
<td>This parameter applies only to Windows OSs.</td>
</tr>
<tr>
<td>Environment Variables</td>
<td>• Add environment variables in the text box.</td>
</tr>
<tr>
<td></td>
<td><code>PATH_1=Path1; PATH_2=Path2</code></td>
</tr>
<tr>
<td></td>
<td>Use semicolons (;) to separate multiple environment variables.</td>
</tr>
<tr>
<td></td>
<td>• You can also click the icon next to the text box and enter a value in</td>
</tr>
<tr>
<td></td>
<td>the displayed dialog box.</td>
</tr>
<tr>
<td></td>
<td>– Type <code>PATH_1</code> in the <strong>Name</strong> field.</td>
</tr>
<tr>
<td></td>
<td>– Type the value (<code>Path 1</code>) in the <strong>Value</strong> field.</td>
</tr>
<tr>
<td></td>
<td>If you select <strong>Instead system environment variables</strong>, the system</td>
</tr>
<tr>
<td></td>
<td>environment variables are displayed.</td>
</tr>
<tr>
<td>Operator Name</td>
<td>Select the operator to test.</td>
</tr>
<tr>
<td>SoC Version</td>
<td>Select the SoC version.</td>
</tr>
<tr>
<td>Executable File Name</td>
<td>Select the test case definition file to run from the drop-down list box.</td>
</tr>
<tr>
<td></td>
<td>If the ST is performed on the AI CPU operator, (AI CPU) is displayed in</td>
</tr>
<tr>
<td></td>
<td>front of the test case file.</td>
</tr>
<tr>
<td>Target OS</td>
<td>For Ascend EP, select the host-side OS where the Ascend AI Processor is</td>
</tr>
<tr>
<td></td>
<td>located.</td>
</tr>
<tr>
<td></td>
<td>For Ascend RC, select the OS of the board environment.</td>
</tr>
<tr>
<td>Target Architecture</td>
<td>Select the OS architecture of the target environment.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Case Names</td>
<td>Specify the name of the case to be executed.</td>
</tr>
<tr>
<td><strong>Note</strong>: All cases are selected by default. You can deselect unnecessary cases.</td>
<td></td>
</tr>
<tr>
<td>Advanced Options</td>
<td>Specify advanced options.</td>
</tr>
<tr>
<td>ACT Log Level</td>
<td>Select an ACT log level.</td>
</tr>
<tr>
<td>● INFO</td>
<td></td>
</tr>
<tr>
<td>● DEBUG</td>
<td></td>
</tr>
<tr>
<td>● WARNING</td>
<td></td>
</tr>
<tr>
<td>● ERROR</td>
<td></td>
</tr>
<tr>
<td>● NULL</td>
<td></td>
</tr>
<tr>
<td>Precision Mode</td>
<td>Set the precision mode. Possible values are:</td>
</tr>
<tr>
<td>● force_fp176</td>
<td></td>
</tr>
<tr>
<td>● allow_mix_precision</td>
<td></td>
</tr>
<tr>
<td>● allow_fp32_to_fp16</td>
<td></td>
</tr>
<tr>
<td>● must_keep_origin_dtype</td>
<td></td>
</tr>
<tr>
<td>Device Id</td>
<td>Set the ID of the device that runs the ST. Specify the ID of the AI processor in use.</td>
</tr>
<tr>
<td>Enable Profiling</td>
<td>Enable profiling to obtain the performance data of the operator on the the Ascend AI Processor. To use this function, you need to configure the path of the msprof tool in the operating environment to the PATH environment variable. The msprof tool works under toolkit/tools/profiler/bin/msprof of the Toolkit directory. <strong>NOTE</strong> Currently, only the Ascend 310 chip is supported.</td>
</tr>
</tbody>
</table>

**NOTE**

- The ST supports the setting and query of the board log level. For details, see 12.6 Log Management.
- The Windows OS does not support the Local Execute function.

**Step 2** Set the host running user.

Add a host running user in the SSH Connection dialog box. The user must be with the HwHiAiUser group. For details about how to add an SSH connection, see 12.1 SSH Connection Management.

**Step 3** Configure the environment variables of related components in the operating environment.

- On a remote device:
For Ascend EP, you need to configure the environment variables of the component installation paths on the host of the device.

Configure the installation paths of the ACLib, FwkACLlib, and Driver components in the ~/.bashrc file as the host running user.

i. Open the .bashrc file of the running user.
   
   vi ~/.bashrc

ii. Append the following lines to the file:

   export ASCEND_DRIVER_PATH=/usr/local/Ascend/driver
   export ASCEND_HOME=/usr/local/Ascend/ascend-toolkit/latest
   export ASCEND_AICPU_PATH=${ASCEND_HOME}/<target architecture>

   If the remote device is in an inference environment:

   export LD_LIBRARY_PATH=${ASCEND_DRIVER_PATH}/lib64:$ASCEND_HOME/acllib/lib64:$LD_LIBRARY_PATH

   If the remote device is in a training environment:

   export
   LD_LIBRARY_PATH=${ASCEND_DRIVER_PATH}/lib64/driver:${ASCEND_DRIVER_PATH}/lib64/common:${ASCEND_HOME}/fwkacllib/lib64:$LD_LIBRARY_PATH

   If the preceding environment variables exist, check if the installation path is correct.

   Replace <target architecture> with the OS architecture, for example, x86_64-linux or arm64-linux.

iii. Save the file and exit.

   :wq

iv. Make the configuration take effect.

   source ~/.bashrc

- Add the environment variable in Environment Variables.

  Set the environment variables in the Environment Variables area as described in Step 1.

- Add environment variables in the text box.

  ASCEND_DRIVER_PATH=/usr/local/Ascend/driver;
  ASCEND_HOME=/usr/local/Ascend/ascend-toolkit/latest;
  ASCEND_AICPU_PATH=${ASCEND_HOME}/<target architecture>-linux;

  If the remote device is in an inference environment:

  LD_LIBRARY_PATH=${ASCEND_DRIVER_PATH}/lib64:$ASCEND_HOME/acllib/lib64:$LD_LIBRARY_PATH;

  If the remote device is in a training environment:

  LD_LIBRARY_PATH=${ASCEND_DRIVER_PATH}/lib64/driver:$ASCEND_DRIVER_PATH/lib64/common:$ASCEND_HOME/fwkacllib/lib64:$LD_LIBRARY_PATH;

  Modify the environment variables based on the actual installation paths of the driver and ADK and the architecture used by the remote OS.

- You can also click the icon next to the text box and enter a value in the displayed dialog box.

  Type the environment variable name in the Name field and the value in the Value field.

**Step 4** Click Run.
MindStudio generates test data and test code in `/testcases/st/out/<operator name>` under the operator project root directory based on the operator test case definition file, builds an executable file, and executes test cases on the specified hardware backend. The report about comparison of execution result and benchmark is printed to the Output window and the st_report.json file is generated in `/testcases/st/out/<operator name>` under the operator project root directory. For details about the st_report.json file, see Table 8-29.

Table 8-29 st_report.json description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>run_cmd</td>
<td>Command.</td>
</tr>
<tr>
<td>report_list</td>
<td>List of reports of test cases.</td>
</tr>
<tr>
<td>trace_detail</td>
<td>Execution details.</td>
</tr>
<tr>
<td>st_case_info</td>
<td>Test information.</td>
</tr>
<tr>
<td>expect_data_path</td>
<td>Path of the expected result.</td>
</tr>
<tr>
<td>case_name</td>
<td>Test case name.</td>
</tr>
<tr>
<td>input_data_path</td>
<td>Path of the input data.</td>
</tr>
<tr>
<td>planned_output_data_paths</td>
<td>Path of the actual result.</td>
</tr>
<tr>
<td>op_params</td>
<td>Operator parameters.</td>
</tr>
<tr>
<td>stage_result</td>
<td>Execution result by phase.</td>
</tr>
<tr>
<td>case_name</td>
<td>Test name.</td>
</tr>
<tr>
<td>status</td>
<td>Test result.</td>
</tr>
</tbody>
</table>

More Functions

- Switching between the Design and Text views
  Double-click the operator test case definition file in the testcases/st/OpType/aicpu_kernel folder or right-click the file and choose OpType_case_timestamp.json from the shortcut menu. The window is displayed with Design and Text views, as shown in the following figure.

If there is no incorrect configuration information in the Design view, the changes will be automatically synchronized to the Text view when you switch to the Text view. The reverse is also true.
Adding a test case

On the operator test case definition page, click Add. The dialog box shown in the following figure is displayed, prompting you to input the Case Name.

Case Name is a string combination of digits, letters, and underscores (_). Click OK to add the created case to the bottom of the Design Cases dialog box. The fields of the new case are empty and need to be configured. For details about the configuration rules, see Field Description of the Operator Test Case Definition File.
• Copying and adding an operator input

On the operator test case definition page, click \(\text{Input[xx]}\) on the right of \(\text{Input[xx]}\) to copy \(\text{Input[xx]}\) as a new \(\text{Input[xx]}\). Alternatively, click \(+\) to add an operator input. Modify the parameters based on the site requirements.

![Operator Test Case Definition Page]

• Deleting an operator input

In the operator test case definition dialog box, click \(\text{Input[xx]}\) on the right of \(\text{Input[xx]}\) to delete \(\text{Input[xx]}\).

![Operator Test Case Definition Dialog Box]

• Deleting a test case

In the operator test case definition dialog box, click \(\text{case}\) on the right of a case to delete the case.
- Running one or more test cases
  In the operator test case definition dialog box, select one or more cases and click Run.

- Modifying and running ST source code
  a. Generating source code
     After the ST is successful, the ST source code is generated in ./testcases/st/out/OpName/src.
  b. Modifying source code
You can modify the source code as required to implement custom requirements and functions.

c. Running source code

Right-click `.testcases/st/out/<operator name>/` and choose Run St Source from the shortcut menu to run the modified source code.

⚠️ **CAUTION**

If you run the ST from another entry, the modified source code will be overwritten. Therefore, back up the source code in advance.

### Field Description of the Operator Test Case Definition File

**Table 8-30** Operator test case definition file (.json)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test_xxx_xx</td>
<td>(Required) A string. Test case name.</td>
</tr>
<tr>
<td>Input[xx]</td>
<td>(Required) Operator input.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTICE</strong></td>
</tr>
<tr>
<td></td>
<td>The number of parameter values in each input of an operator must be the same. Otherwise, test case generation fails.</td>
</tr>
<tr>
<td></td>
<td>For example, if the Input[01] supports two formats, the Input[02] also supports two formats.</td>
</tr>
<tr>
<td></td>
<td>Similarly, the parameter values of type, shape, data_distribute, and value_range in all Input[xx] must be the same.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| Format | (Required)  
A string or a 1D array of strings. Input format. Must not be empty. Value range:  
- **NCHW**  
- **NHWC**  
- **ND**: any format  
- **NC1HWC0**: the 5D format defined by Huawei. **C0** is closely related to the micro-architecture, and the value is equal to a size of a cube unit, for example, 16. **C1** divides the C dimension by **C0**, that is, \( C1 = C/C0 \). When the division is not exact, the last data segment is padded to **C0**.  
- **FRACTAL_Z**: format of the convolution weight.  
- **FRACTAL_NZ**: fractal format defined by Huawei. The data format of the output matrix is **NW1H1H0W0** for Cube Unit computation. The matrix is divided into \((H1 \times W1)\) fractals in column-major order, which looks like an N-shape layout. Each fractal consists of \((H0 \times W0)\) elements in row-major order, resembling a Z-shaped layout. Thus the **NW1H1H0W0** data format is referred to as an **Nz** format. \((H0 \times W0)\) indicates the size of a fractal, as shown in the following figure.  

![Fractal Matrix Size](image)  

- **RESERVED**: reserved. If this value is used, **type** needs to be set to **UNDEFINED**, indicating that the
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>corresponding operator input is optional.</td>
</tr>
<tr>
<td><strong>OriginFormat</strong></td>
<td>(Optional) Original tensor format. If this parameter is not carried, the original tensor format is retained.</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>(Required) A string or a 1D array of strings. Input data type.</td>
</tr>
<tr>
<td></td>
<td>• bool</td>
</tr>
<tr>
<td></td>
<td>• int8</td>
</tr>
<tr>
<td></td>
<td>• uint8</td>
</tr>
<tr>
<td></td>
<td>• int16</td>
</tr>
<tr>
<td></td>
<td>• uint16</td>
</tr>
<tr>
<td></td>
<td>• int32</td>
</tr>
<tr>
<td></td>
<td>• int64</td>
</tr>
<tr>
<td></td>
<td>• uint32</td>
</tr>
<tr>
<td></td>
<td>• uint64</td>
</tr>
<tr>
<td></td>
<td>• float16</td>
</tr>
<tr>
<td></td>
<td>• float</td>
</tr>
<tr>
<td></td>
<td>• UNDEFINED: applies to an optional input.</td>
</tr>
<tr>
<td><strong>Shape</strong></td>
<td>(Required) An int or a 1D or 2D array of ints. Input shape.</td>
</tr>
<tr>
<td><strong>OriginShape</strong></td>
<td>(Optional) Original tensor shape. If this field is not carried, the original tensor shape is retained.</td>
</tr>
<tr>
<td><strong>ValueRange</strong></td>
<td>(Required) An int or float, or a 1D or 2D array of ints or floats. Value range. Must not be left empty. Formatted as [min_value, max_value], with min_value &lt;= max_value.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>-</td>
<td>DataDistribute</td>
</tr>
<tr>
<td></td>
<td>(Required) A string or a 1D array of strings. Data distribution mode for generating test data.</td>
</tr>
<tr>
<td></td>
<td>● uniform: returns random values that are evenly distributed.</td>
</tr>
<tr>
<td></td>
<td>● normal: returns random values of the normal distribution (Gaussian distribution).</td>
</tr>
<tr>
<td></td>
<td>● beta: returns random values of Beta distribution.</td>
</tr>
<tr>
<td></td>
<td>● laplace: returns random values of Laplace distribution.</td>
</tr>
<tr>
<td></td>
<td>● triangular: returns random values of the triangular distribution.</td>
</tr>
<tr>
<td></td>
<td>● relu: returns random values that are evenly distributed and activated by the ReLU function.</td>
</tr>
<tr>
<td></td>
<td>● sigmoid: returns random values that are evenly distributed and activated by the sigmoid function.</td>
</tr>
<tr>
<td></td>
<td>● softmax: returns random values that are evenly distributed and activated by softmax function.</td>
</tr>
<tr>
<td></td>
<td>● tanh: returns random values that are evenly distributed and activated by tanh function.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output[xx]</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Required)</td>
<td>Operator output.</td>
</tr>
<tr>
<td>NOTICE</td>
<td>The number of values in <strong>Output</strong> must be the same as that in <strong>Input</strong>. Otherwise, test case generation fails.</td>
</tr>
<tr>
<td></td>
<td>For example, if <strong>Input</strong> supports two formats, the <strong>Output</strong> also supports two formats.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Format</td>
<td>(Required) A string or a 1D array of strings. Output format. Must not be empty. Value range:</td>
</tr>
<tr>
<td></td>
<td>- <strong>NCHW</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>NHWC</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>ND</strong>: any format</td>
</tr>
<tr>
<td></td>
<td>- <strong>NC1HWC0</strong>: the 5D format defined by Huawei. C0 is closely related to the micro-architecture, and the value is equal to a size of a cube unit, for example, <strong>16</strong>. C1 divides the C dimension by C0, that is, C1 = C/C0. When the division is not exact, the last data segment is padded to C0.</td>
</tr>
<tr>
<td></td>
<td>- <strong>FRACTAL_Z</strong>: format of the convolution weight.</td>
</tr>
<tr>
<td></td>
<td>- <strong>FRACTAL_NZ</strong>: fractal format defined by Huawei. The data format of the output matrix is <strong>NW1H1H0W0</strong> for Cube Unit computation. The matrix is divided into (H1 x W1) fractals in column-major order, which looks like an N-shape layout. Each fractal consists of (H0 x W0) elements in row-major order, resembling a Z-shaped layout. Thus the NW1H1H0W0 data format is referred to as an Nz format. (H0 x W0) indicates the size of a fractal, as shown in the following figure.</td>
</tr>
</tbody>
</table>

![Fractal Matrix Size](image)

<p>| OriginFormat | (Optional) Original tensor format. If this parameter is not carried, the original tensor format is retained. |</p>
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Type    | (Required)  
A string or a 1D array of strings. Output data type.  
- bool  
- int8  
- uint8  
- int16  
- uint16  
- int32  
- int64  
- uint32  
- uint64  
- float16  
- float |
| Shape   | (Required)  
An int or a 1D or 2D array of ints. Output shape. |
| OriginShape | (Optional)  
Original tensor shape. If this field is not carried, the original tensor shape is retained. |
| Attr    | (Optional)  |
| Name    | Required if attr is configured.  
A string. Attribute name. Must not be empty. |
| Type    | Required if attr is configured.  
Attribute data type.  
A string. Output data type.  
- bool  
- int  
- float  
- string  
- list_bool  
- list_int  
- list_float  
- list_string  
- list_list_int |
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Value         | Required if `attr` is configured. Must not be `null`. A string. Attribute value. Varies according to `type`.
|               | ● bool: true/false                                                          |
|               | ● int: 10                                                                  |
|               | ● float: 1.0                                                                |
|               | ● string: "NCHW"                                                            |
|               | ● list_bool: [false, true]                                                  |
|               | ● list_int: [1, 224, 224, 3]                                                |
|               | ● list_float: [1.0, 0.0]                                                     |
|               | ● list_string: ["str1", "str2"]                                           |
|               | ● list_list_int: [[1, 3, 5, 7], [2, 4, 6, 8]]                              |

### Expected Result Verification
- **Script Path**: Path of the test function for generating expected operator result.
- **Script Function**: Name of the test function for generating expected operator result.

#### NOTE
- When `inputx.paramType` in the operator information library definition file (.ini) is set to `optional`, set format of `inputx` to `UNDEFINED` or `RESERVED`, and `type` to `UNDEFINED` in the generated operator test cases.
- In the operator information library definition file (.ini), if a tensor implementation uses a Huawei-developed format and is different from the original format, you need to manually enter the original format and shape in `OriginFormat` and `OriginShape` to convert the format and shape into those compatible with the offline model.
  - `OriginFormat` is an array specifying the formats supported by the original operator. The array must have the same length as the `Format` array.
  - The value of `OriginShape` must correspond to the value of `Shape` and match the values of `Format` and `OriginFormat`.

## 8.9 Other Functions and Operations
### 8.9.1 Operator Project Management
8.9.1.1 Importing an Operator Project

This section describes how to import a sample project. MindStudio provides a complete set of operator implementation code samples, and allows you to deploy and verify operators and perform operator network verification without modifying the code.

**Step 1** Navigate to the dialog box for project importing.
- If it is your first login to MindStudio, click **Open or Import** in the **Welcome to MindStudio** window.
- If it is not your first login to MindStudio, choose **File > Open...** on the menu bar.

**Step 2** In the **Open File or Project** window, select the folder of the sample project.

The TBE sample project is stored in **MindStudio/samples/operator_demo_projects/** in the MindStudio installation directory.

**Step 3** Click **OK** to import the TBE sample project.

If the message shown in **Figure 8-33** is displayed when you open the project, the ADK version has not been configured in the sample project configuration file. Click **OK** to change the ADK version number to the actual one, as shown in **Figure 8-34**.

**Figure 8-33** Project opening error message

![Figure 8-33 Project opening error message](image)

**Figure 8-34** Editing operator properties

![Figure 8-34 Editing operator properties](image)

----End
8.9.1.2 Editing Project Properties

After an operator project is created, you can edit the project properties, including the ADK version and product form, so that you can run the operator in different forms of products.

Prerequisites

An operator project has been created. For details about how to create an operator project, see 8 Custom Operator Development.

Procedure

**Step 1** Right-click a created operator project and choose Edit Operators Property from the shortcut menu. The Edit Operators Property dialog box is displayed, as shown in Figure 8-35.

![Figure 8-35 Editing operator properties](image)

Table 8-31 describes the parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Project name, which is determined during project creation and is unconfigurable</td>
</tr>
<tr>
<td>Description</td>
<td>Project description (customizable)</td>
</tr>
<tr>
<td>ADK Version</td>
<td>Current version number of the ADK</td>
</tr>
<tr>
<td>Project Location</td>
<td>Project path, which is determined during project creation and is unconfigurable</td>
</tr>
</tbody>
</table>

**Step 2** Modify ADK Version as required and click OK. Proceed project development.

---End
8.9.1.3 Switching to the Operator Logical View

You can switch from the operator project view to the operator logical view. In the logical view, files in the operator project directory are displayed by operator. The actual structure and locations of the files remain unchanged.

In the operator logic view, you can develop, verify, add, and delete operators.

In the Project window of the operator project, click and select Operators from the drop-down list box, as shown in Figure 8-36.

**Figure 8-36 Switching portal**

After the switch is successful, the operator logical view is displayed, as shown in Figure 8-37.

**Figure 8-37 Viewing the operator logical view**
This view displays the operator deliverables by operator, including the operator information definition file, operator implementation file, operator prototype definition file, operator plug-in implementation file, operator fusion information definition file, and operator test cases.

8.9.1.4 Adding or Deleting an Operator

This section describes how to add an operator to a project and delete an existing operator.

Adding an Operator in the Project Window

In the Project window, right-click operator project and choose New > Operator from the shortcut menu. In the displayed dialog box, set the operator information, as shown in Figure 8-38. In this way, multiple operators can be implemented in the current operator project. If the operator type already exists, a message is displayed, asking you whether to overwrite the operator.

![Figure 8-38 Adding an operator](image)

Adding an Operator in the Operator Logical View

Switch to the logical view by referring to 8.9.1.3 Switching to the Operator Logical View. Right-click the operator project, and choose New > Operator from the shortcut menu. In the displayed dialog box, add operator information, as shown in Figure 8-39. If the operator type already exists, a message is displayed, asking you whether to overwrite the operator.
Deleting an Operator in the Operator Logical View

Switch to the operator logical view by referring to 8.9.1.3 Switching to the Operator Logical View, right-click the operator to be deleted, and choose Delete Operator from the shortcut menu, as shown in Figure 8-40.

Figure 8-40 Deleting an operator in the operator logical view

8.9.2 Generating a .cce File

You can learn about the data movement process and buffer splitting implemented at the bottom layer to optimize the operator implementation by analyzing the .cce file of the operator.

In normal cases, the operator binary file (.o) and operator description file (.json) are generated upon operator execution. To also generate a .cce file, you need to set save_temp_cce_file to True in the `{Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/atc/python/site-packages/te/platform/cce_build.py` file before the operator is executed.
After the operator is executed, a .cce file is generated in the same directory of the operator binary file (.o) and operator description file (.json).

## 8.9.3 TIK Function Debugging

### Prerequisites

The implementation code of the TIK operator has been developed.

### Procedure

This section uses the simple_add operator as an example to describe how to debug the TIK function.

#### Step 1

Compile the simple_add operator debugging code file (a .py file).

A complete code example is as follows (the file is named `simple_add.py` in this example).

```python
import numpy as np
from te import tik
from te.platform.cce_conf import te_set_version

def simple_add():
    tik_instance = tik.Tik()
    kernel_name = "tik_vec_add_128_float32"
    dst_ub = tik_instance.Tensor("float32", [128], tik.scope_ubuf, "dst_ub")
    dst_gm = tik_instance.Tensor("float32", (128,), tik.scope_gm, "dst_gm")
    src0_gm = tik_instance.Tensor("float32", (128,), tik.scope_gm, "src0_gm")
    src0_ub = tik_instance.Tensor("float32", (128,), tik.scope_ubuf, "src0_ub")
    src1_gm = tik_instance.Tensor("float32", (128,), tik.scope_gm, "src1_gm")
    src1_ub = tik_instance.Tensor("float32", (128,), tik.scope_ubuf, "src1_ub")

    tik_instance.data_move(src0_ub, src0_gm, 0, 1, 16, 0, 0)
    tik_instance.data_move(src1_ub, src1_gm, 0, 1, 16, 0, 0)
    tik_instance.vec_add(64, dst_ub, src0_ub, src1_ub, 2, 8, 8, 8)
    tik_instance.data_move(dst_gm, dst_ub, 0, 1, 16, 0, 0)
    tik_instance.BuildCCE(kernel_name, [src0_gm, src1_gm], [dst_gm])
    return tik_instance

if __name__ == '__main__':
    te_set_version("Ascend310")  # Replace the parameter as needed. For details, see the parameter
description of `te_set_version`.  
tik_instance = simple_add()  
data_x = np.ones((128,)).astype("float32")
data_y = np.ones((128,)).astype("float32")
feed_dict = {"src0_gm": data_x, "src1_gm": data_y}
model_data, = tik_instance.tikdb.start_debug(feed_dict=feed_dict, interactive=True)
print(model_data)
```

#### Step 2

Right-click the `simple_add.py` file and choose RUN from the shortcut menu, or click the ⏯️ icon on the page to debug the TIK operator.

For details, see "Operator Development Workflow > Operator Code Implementation (TIK Mode) > Debugging" in the *TBE Custom Operator Development Guide*.

----End
8.9.4 Custom Operator Samples

MindStudio provides custom operator samples. To open a sample project, perform the following steps:

**Step 1** Navigate to the dialog box for project importing.
- If it is your first login to MindStudio, click **Open or Import** in the **Welcome to MindStudio** window.
- If it is not your first login to MindStudio, choose **File > Open...** on the menu bar.

**Step 2** In the **Open File or Project** window, select the folder of the sample project.

The sample project is stored in **MindStudio/samples/operator_demo_projects/cann/** under the MindStudio installation directory.

The following figure shows the CANN directory structure. Select the demo project to be opened. For details about the operator template list, see **Table 8-33**.

```
├── AICPU
│   └── Tensorflow
├── DSL
│   ├── Mindspore
│   │   └── Tensorflow
│   ├── PyTorch
│   └── Tensorflow
└── TIK
    ├── Mindspore
    │   └── Tensorflow
    └── PyTorch
```

**Step 3** Click **OK** to import the sample project.

In the **Import Support Configuration** dialog box, set parameters and click **Create**.

**Table 8-32** Import Support Configuration dialog box

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>Project type.</td>
</tr>
<tr>
<td></td>
<td>• Ascend Operator</td>
</tr>
<tr>
<td></td>
<td>• Ascend App</td>
</tr>
<tr>
<td></td>
<td>• Ascend Training</td>
</tr>
<tr>
<td></td>
<td>Select <strong>Ascend Operator</strong> from the drop-down list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Framework</strong></th>
<th>Original framework of the operator.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• MindSpore</td>
</tr>
<tr>
<td></td>
<td>• PyTorch</td>
</tr>
<tr>
<td></td>
<td>• TensorFlow</td>
</tr>
<tr>
<td></td>
<td>Set this parameter based on the framework of the opened sample project.</td>
</tr>
</tbody>
</table>
**NOTE**

- When the ADK package is replaced, the `cann` directory in the operator sample project is overwritten. If the ADK package is modified, back up the ADK package before replacing it.
- The custom operator template code is used only for reference during operator development. Product-based development using the custom operator template is not supported.

### Table 8-33 Custom operator template list

<table>
<thead>
<tr>
<th>Development Method</th>
<th>Framework</th>
<th>Operator</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AICPU</td>
<td>TensorFlow</td>
<td>Less</td>
<td>Determines whether element 1 is less than element 2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TopK</td>
<td>Obtains the values and indexes of the first $K$ largest elements.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cast</td>
<td>Converts the format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SparseToDense</td>
<td>Converts a sparse matrix into a dense matrix.</td>
</tr>
<tr>
<td>DSL</td>
<td>PyTorch</td>
<td>LpNorm</td>
<td>Normalizes batch operations.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MaskedFill</td>
<td>Replaces a value with a mask.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SigmoidCrossEntropy-WithLogitsGrad</td>
<td>Calculates the loss</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SigmoidCrossEntropy-WithLogitsGradV2</td>
<td>Calculates the gradient.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mul</td>
<td>Multiplies elements.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reduceAll</td>
<td>Performs the AND operation on a specified dimension and removes the dimension based on conditions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>batchNorm</td>
<td>Normalizes batches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>max_pool</td>
<td>Performs max pooling.</td>
</tr>
<tr>
<td>MindSpore</td>
<td>TensorFlow</td>
<td>Add3</td>
<td>Adds all elements.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CorrectionMul</td>
<td>Uses the correction factor to process the weight.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Square</td>
<td>Calculates the square root of an element.</td>
</tr>
<tr>
<td>Development Method</td>
<td>Framework</td>
<td>Operator</td>
<td>Functions</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------</td>
<td>----------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TIK</td>
<td>TensorFlow</td>
<td>ScatterNdAdd</td>
<td>Applies the sparse algorithm to a single value or slice in the input data to obtain the output data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DecodeBboxV2</td>
<td>Decodes the bounding box.</td>
</tr>
<tr>
<td>PyTorch</td>
<td></td>
<td>Ptlou</td>
<td>Calculates the IoU.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SortV2</td>
<td>Performs sorting.</td>
</tr>
<tr>
<td>MindSpore</td>
<td></td>
<td>BatchMatmul</td>
<td>Multiplies tensor slices in batches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MatrixCombine</td>
<td>Combines matrices.</td>
</tr>
</tbody>
</table>
9 Model Accuracy Analyzer

9.1 Introduction and Restrictions

Overview

During model conversion, the model is optimized, including operator elimination, operator fusion, and operator split. As a result, the computation result of Huawei proprietary operators may be different from that of third-party operators (for example, from Caffe, TensorFlow and ONNX). The Model Accuracy Analyzer is designed to compare the computation results for developers to quickly resolve the operator accuracy issues.

The Model Accuracy Analyzer compares the computation result of Huawei proprietary operators with that of the standard operators from Caffe, TensorFlow and ONNX to locate the error operators. Currently, the following comparison is provided: cosine similarity, maximum absolute error, accumulated relative error, Euclidean relative distance, Kullback-Leibler divergence (KLD), and standard deviation.

Reference Documents

Table 9-1 Reference documents

<table>
<thead>
<tr>
<th>Product</th>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlas 200 DK</td>
<td>AMCT Instructions (Caffe)</td>
</tr>
<tr>
<td>Others</td>
<td>AMCT Instructions (Caffe) (Inference)</td>
</tr>
</tbody>
</table>
Advantages of the method are as follows:

1. **High Accuracy**: The method provides high accuracy in capturing the processing logic of operators.
2. **Precision**: The method is precise in identifying and analyzing operator logic.
3. **Efficiency**: The method is efficient in processing large datasets.
4. **Versatility**: The method can be applied to various types of models and datasets.

**Limitations**:

- The method may struggle with complex or highly concurrent processing logic.
- It requires significant computational resources, especially for large-scale models.

In conclusion, the method offers a powerful tool for analyzing operator logic in AI models, with significant potential for improving model accuracy and efficiency.
Restrictions

The following formats are supported:

- NCHW
- NHWC
- ND
- NC1HW0
- FRACTAL_Z
- HWCN

The following data types are supported:

- FLOAT
- FLOAT16
- DT_INT8
- DT_UINT8
- DT_INT16
- DT_UINT16
- DT_INT32
- DT_INT64
- DT_UINT32
- DT_UINT64
- DT_BOOL
- DT_DOUBLE

Environment Preparation

The following describes how to use the Model Accuracy Analyzer with the assumption that the tool is installed by HwHiAiUser. You can replace it as required.

Python 3.7.5 should be installed in advance.

9.2 Data Preparation

9.2.1 Dump File Naming Conventions

The current version supports multiple comparison approaches. Keep the following naming conventions in mind when creating dump files and .npy files.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Naming Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dump data of the non-quantized Caffe model</td>
<td>{op_name}.{output_index}.{timestamp}.pb</td>
</tr>
</tbody>
</table>
### Data Type | Naming Format
--- | ---
Dump data of the quantized Caffe model | `{op_name}.{output_index}.{timestamp}.quant`  
Dump data of the non-quantized offline model running on the Ascend AI Processor | `{op_type}.{op_name}.{task_id}.{timestamp}`  
Dump data of the quantized offline model running on the Ascend AI Processor | `{op_type}.{op_name}.{task_id}.{timestamp}`  
Dump data of the non-quantized TensorFlow model | `{op_name}.{output_index}.{timestamp}.pb`  
.npy file (Caffe/TensorFlow/ONNX) | `{op_name}.{output_index}.{timestamp}.npy`  

**NOTE**
- The names of `op_type` and `op_name` must comply with the `A-Za-z0-9-_` regular expression rules.
- The dump data supports the comparison of data types with ultra-long operator names. However, if the dump data contains only the data types with ultra-long operator names, the comparison is not supported.
- If the operator name on the network is too long, the dump data file cannot be named in the format described in the preceding table, and the file name changes to a hash value consisting of 0 to 9. As a result, the tool cannot determine the comparison scenario. When all data file names become hash values, the tool does not support the comparison.
- The timestamp of the dump data generated on an Ascend AI Processor must comply with the `[0-9]{1,255}` regular expression, and that of the dump data generated on a non-Ascend AI Processor device must comply with the `[0-9]{16}` regular expression.
- The values of `output_index` and `task_id` consist of digits 0-9.

### 9.2.2 Inference Scenarios

#### 9.2.2.1 Preparing Dump Data of an Offline Model

This section describes how to dump data of an offline model in MindStudio. Before dump, you should have set up the development environment and developed, built, and run your application project, which means that you have an executable application project.

To generate dump data, perform the following steps:

**Step 1** Open MindStudio and choose **Ascend > Dump Configuration** from the menu bar. The **Select Offline Model** dialog box is displayed, as shown in Figure 9-1.
You can also open the project file, right-click your .om file, and choose **Dump Configuration**.

**Figure 9-1** Selecting an offline model

![Offline model file (.om)](image)

**Step 2** Select an .om file and click **Open**. The network topology of the model is displayed. Perform **Dump Configuration**, as shown in **Figure 9-2**.

**NOTE**

- To dump multiple models, perform **Dump Configuration** for each .om file separately.
- If similar models are used, they might have the same **Model Name**. As a result, there is some confusion between the dump configurations in the acl.json file.

  For example, if the values of YOLOv2 and YOLOv3 in **Model Name** are the same, and the dump configuration of YOLOv2 has been recorded in the acl.json file, then the dump configuration of YOLOv2 applies to YOLOv3.

  To ensure each model has a unique **Model Name**, change the **name** field in the Caffe prototxt file and re-convert the model. You can also choose **None** during **Dump Configuration** to clear the previous configuration, and start **Dump Configuration** all over.
Set the dump parameters of the .om model on the upper right.

- **Dump Option**: dump range
  - **ALL**: dumps all operators.
    - **NOTE**
      Operators that do not have output (for example, StreamActive, NetOutput, Send, Recv, and const) do not generate dump data.
    - **Several**: dumps specified operators.
      Right-click the operator to be dumped, and choose **Enable Dump**, turning the operator orange.
    - **NOTE**
      Use Ascend Tensor Compiler (ATC) to parse the .om file for obtaining the mapping between operators (for example, newly-added, one-to-many, many-to-one, many-to-many, and L1 batch). Mapped operators should have the same **Enable Dump** setting. Dump cannot be enabled for const operators.
      When only certain operators are to be dumped, since the Data operators are not executed on the AI CPU or AI Core, you need to select all the downstream operators of the Data operator.
      For TBE and AI CPU operators that do not output results (for example, StreamActive, Send, Recv, and const), dump data is not generated. For operators that are not executed on the AI CPU or AI Core (for example, concatD) after build, dump data cannot be generated.
  - **None**: disables dump.

- **Dump Mode**: data dump mode
  - **Input**: dumps the operator input.
  - **Output**: dumps the operator output.
  - **All**: dumps both the operator input and output.

- **Dump Path**: path for storing dump files. Defaults to `{project_path}/dump`. If the parameter is set to a different path, ensure that the MindStudio installation user has the read and write permissions on the path.
- **AclConfig File**: ACL configuration file, which stores the dump configuration information of the operator during the dump operation. Generally, the path is `{project_path}/src/acl.json`. You can search for the actual path in the `{project_path}/.project` file. The format is shown in Figure 9-3 or Figure 9-4. To dump all operators of a model, the layer field is not included.

Figure 9-3 Dumping all operators

```json
{
  "dump":{
    "dump_list": [
      {
        "model_name":"ResNet-50"
      },
      "dump_mode":"input",
      "dump_path":"/home/user/AscendProjects/MyApp/dump"
    ]
  }
}
```

Figure 9-4 Dumping several operators

```json
{
  "dump":{
    "dump_list": [
      {
        "model_name":"ResNet-50",
        "layer": [
          "pool1",
          "res2a_branch2ares2a_branch2a_relu",
          "conv1conv1_relu"
        ],
        "dump_mode":"input",
        "dump_path":"/home/user/AscendProjects/MyApp/dump"
      }
    ]
  }
}
```

**Step 3** After **Dump Configuration**, click **Build** and **Run** in MindStudio, to build and run the application project.

**NOTE**

Check the project file code to ensure the .om file with dump configured is used. For example, check `aclmdlLoadFromFile()` or `aclmdlLoadFromFileWithMem()`.

After the project is run, find the dump data in the `{project_path}/dump` path. The path and format are described as follows:

```
time/device_id/model_name/model_id/data_index/dump_file
```

- **time**: dump time, formatted as YYYYMMDDhhmmss
9.2.2.2 Preparing .npy Data of a Caffe Model

This version does not support the generation of .npy files of a Caffe model. You need to install the Caffe environment and prepare the source data NumPy files (.npy) in advance. This section provides only a Caffe .npy file example that meets the accuracy comparison requirements.

Note
- Details about how to prepare the .npy file of a Caffe model are not described herein. You can prepare it by yourself.
- To use a dump file in binary format for comparison, convert the .npy file to a dump file by referring to 9.4.3 How Do I Convert Between Dump File and NumPy File?

Prepare the source .npy file as follows:

- Save the file content in NumPy format.
- Name the file in op_name.output_index.timestamp.npy format. Ensure that the output_index field is contained in the .npy file name and the output_index value of the generated dump data is 0. This is because by default, accuracy comparison starts from the first data whose output_index is 0. Otherwise, no comparison result is returned.
- To ensure that the .npy file is correctly named, remove in-place on the Caffe model file to generate a new .prototxt model file for .npy file generation. For example, if there are four fused operators A, B, C, and D whose in-place is not removed, the output data dumping result is that of operator D, while the file name is prefixed with operator A. For quantization scenarios, install AMCT in the environment before running the command to remove in-place. For details about the installation, see AMCT Instructions (Caffe) in Table 9-1.

Go to the {Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/toolkit/tools/operator_cmp/compare directory and run a command to remove in-place. The following is a command example:

```
python3.7.5 inplace_layer_process.pyc -i /home/HwHiAiUser/resnet50.prototxt
```

After the command is executed, the new_resnet50.prototxt file with in-place removed is generated in the /home/HwHiAiUser directory.

- For quantization scenarios, to ensure accuracy, data preprocessing during Caffe model inference must be the same as that during Caffe model compression.
MindStudio
User Guide

9 Model Accuracy Analyzer

To generate an .npy file that meets the accuracy comparison requirements, add
similar code as follows after the inference is complete.
#read prototxt file
net_param = caffe_pb2.NetParameter()
with open(self.model_file_path, 'rb') as model_file:
google.protobuf.text_format.Parse(model_file.read(), net_param)
# save data to numpy file
for layer in net_param.layer:
name = layer.name.replace("/", "_").replace(".", "_")
index = 0
for top in layer.top:
data = net.blobs[top].data[...]
file_name = name + "." + str(index) + "." + str(
round(time.time() * 1000000)) + ".npy"
output_dump_path = os.path.join(self.output_path, file_name)
np.save(output_dump_path, data)
os.chmod(output_dump_path, FILE_PERMISSION_FLAG)
print('The dump data of "' + layer.name
+ '" has been saved to "' + output_dump_path + '".')
index += 1

After the preceding code is added, run the application project of the Caffe model
to generate an .npy file that meets the requirements.

9.2.2.3 Preparing .npy Data of a TensorFlow Model
This version does not support the generation of .npy files of a TensorFlow model.
You need to install the TensorFlow environment and prepare .npy file in advance.
This section provides only an example of the TensorFlow .npy file for reference.
NOTE
To use a dump file in binary format for comparison, convert the .npy file to a dump file by
referring to 9.4.3 How Do I Convert Between Dump File and NumPy File?

Before generating .npy files of a TensorFlow model, a complete, executable,
standard TensorFlow model application project is required. You can use the
TensorFlow debugger (tfdbg) to generate .npy files. The major steps are as
follows:
Step 1 Modify the TensorFlow application project script to add the following code to the
debugging configuration option:
In Estimator mode:
from tensorflow.python import debug as tf_debug
training_hooks = [train_helper.PrefillStagingAreaHook(),
tf_debug.LocalCLIDebugHook()]
In Session.run mode:
from tensorflow.python import debug as tf_debug
sess = tf_debug.LocalCLIDebugWrapperSession(sess, ui_type="readline")
●

Issue 01 (2021-06-24)

In Estimator mode, add the tfdbg hook, as shown in Figure 9-5.

Copyright © Huawei Technologies Co., Ltd.

430


In **Session.run** mode, set the tfdbg wrapper before run, as shown in **Figure 9-6**.

**Figure 9-6** Session.run mode

```python
from tensorflow.python import debug as tf_debug

tf.Session()

tf_debug.LocalCLIDebugWrapperSession(sess, ui_type="readline")
```

**Step 2** Run the script.

In the interactive debugger command line, enter **run** to run the script.

**Step 3** Collect .npy files.

After the script is executed, you can run the **lt** command to query the stored tensors, run the **pt** command to view the tensor content, and save it as a file in NumPy format.

The tfdbg dumps only one tensor at a time. To dump all tensors, perform the following steps:

1. Run **lt > tensor_name** to temporarily store all tensor names to a file.
2. Exit the tfdbg command line, enter the Linux command line, and run the following command to generate commands to run in tfdbg:

\[
timestamp=$\{\$(date +%s%N)/1000\} ; cat tensor_name | awk '{print "pt", $4,$4}' | awk '{gsub("/", "_", $3);gsub(":\", ".", $3);print($1,$2,"-n 0 -w "$3":""$timestamp"."npy")}'
\]

**NOTE**

The .npy file generated in the example complies with the naming rules for accuracy comparison. `tensor_name` indicates the name of the file corresponding to the customized tensor list. The value of `timestamp` must comply with the `[0-9]{1,255}` regular expression. You can also run the command in the new window without exiting the tfdbg command line.

3. Go back to the tfdbg command line, run the script, and run the command generated in the previous step for saving all .npy files.

By default, .npy files are stored using `numpy.save()`. Slashes (/) and colons (:) are replaced by underscores (_).

**NOTE**

If the command cannot be pasted on the CLI, run the `mouse off` command in the tfdbg command line to disable the mouse mode before pasting again.

4. Check whether names of the generated .npy files comply with the naming rules, as shown in Figure 9-7.

**NOTE**

- An .npy file is named in the format of `{op_name}.{output_index}.{timestamp}.npy`, where the `op_name` field must comply with the `A-Za-z0-9_-` regular expression, the `timestamp` field must comply with the `[0-9]{1,255}` regular expression, and `output_index` is a digit ranging from 0 to 9.
- If the name of an .npy file exceeds 255 characters due to the long operator name, comparison of this operator is not supported.
- The name of some .npy files may not meet the naming requirements due to the tfdbg or operating environment. You can manually rename the files based on the naming rules. If there are a large number of .npy files that do not meet the requirements, generate .npy files again by referring to 9.4.5 How Do I Handle Exceptions in the Generated .npy File Names in Batches?

**Figure 9-7** Viewing the .npy files

```text
truediv_91.0_0123456789012345.npy
truediv_92.0_0123456789012345.npy
truediv_93.0_0123456789012345.npy
truediv_94.0_0123456789012345.npy
truediv_95.0_0123456789012345.npy
truediv_96.0_0123456789012345.npy
truediv_97.0_0123456789012345.npy
truediv_98.0_0123456789012345.npy
truediv_99.0_0123456789012345.npy
```
9.2.2.4 Preparing .npy Data of an ONNX Model

This version does not support the generation of .npy files of an ONNX model. You need to install the ONNX environment and prepare the source data NumPy files (.npy) in advance. This section provides only an ONNX .npy file example that meets the accuracy comparison requirements.

Preparing the .npy File

- Save the file content in NumPy format.
- Name the file in `op.name.output_index.timestamp.npy` format. Ensure that the `output_index` field is contained in the .npy file name and the `output_index` value of the generated dump data is 0. This is because by default, accuracy comparison starts from the first data whose `output_index` is 0. Otherwise, no comparison result is returned.
- Ensure that each operator node has a name. Otherwise, the file name cannot be generated correctly. The operator node without a name must be named first.
- In this example, each node has only one output. For multi-output operators, you need to configure the `output_index` obtaining logic in the file name generation implementation.

Code Example

To generate an .npy file that meets the accuracy comparison requirements, add similar code as follows after the inference is complete.

```python
import onnx
import onnxruntime
import numpy as np
import time
from skl2onnx.helpers.onnx_helper import enumerate_model_node_outputs
from skl2onnx.helpers.onnx_helper import select_model_inputs_outputs
from skl2onnx.helpers.onnx_helper import save_onnx_model

# Modify the model to add an output node.
model_onnx = onnx.load("/resnet50.onnx")
output = []
for out in enumerate_model_node_outputs(model_onnx):
    output.append(out)
num_onnx = select_model_inputs_outputs(model_onnx, outputs=output)
save_onnx_model(num_onnx, "resnet50_dump.onnx")

# Perform inference to obtain an output. In this example, a random input is used.
input_data = np.random.random((1,3,224,224)).astype(np.float32)
input_data.tofile("test_data.bin")
sess = onnxruntime.InferenceSession("resnet50_dump.onnx")
input_name = sess.get_inputs()[0].name
output_name = [node.name for node in sess.get_outputs()]
res = sess.run(output_name, {input_name: input_data})

# Obtain the output name and ensure that the operator nodes are all named properly.
node_name = [node.name for node in model_onnx.graph.node]

# Save data. Configure a loop for `output_index` obtaining for nodes having multiple outputs.
for idx, data in enumerate(res):
    file_name = node_name[idx] + "," + str(0) + "," + str(round(time.time() * 1000000)) + ".npy"
    output_dump_path = "/onnx_dump/" + file_name
    np.save(output_dump_path, data.astype(np.float16))
```
9.2.3 Training Scenarios

9.2.3.1 Preparing .npy Data of a TensorFlow Network on GPUs

Prerequisites

- Before generating the dump data or .npy files of a trained TensorFlow network, a complete, executable, standard TensorFlow model training project is required.
- Regardless of whether the Estimator or Session.run mode is used, disable all random functions in the script, including but not limited to shuffle operations on datasets, random initialization of parameters, and implicit random initialization of some operators (such as the dense operator). Ensure that all parameters in the script are not initialized randomly.

**NOTE**

Accuracy comparison between the dump data of a model running on the Ascend AI Processor and **Ground Truth** (dump data or .npy files of a model running on GPU or CPU) is supported. You only need to prepare the .npy files. To use the dump data of a model running on GPU or CPU for comparison, convert an .npy file into a dump file by referring to [9.4.3 How Do I Convert Between Dump File and NumPy File?](#)

Preparing .npy Files

You can use the TensorFlow debugger (tfdbg) to generate .npy files. The major steps are as follows:

**Step 1** Add the debugging configuration option to the TensorFlow training project script.

- If the **Estimator** mode is used, add the hook of tfdbg as follows:
  - a. Add `from tensorflow.python import debug as tf_debug` to import the debug module.
  - b. Add the code `training_hooks=[tf_debug.LocalCLIDebugHook()]` when the EstimatorSpec object instance is generated, that is, when the network structure is constructed.
If the `Session.run` mode is used, set the tfdebug decorator before running as follows:

a. Add `from tensorflow.python import debug as tf_debug` to import the debug module.

b. After the session is initialized, add `sess = tf_debug.LocalCLIDebugWrapperSession(sess, ui_type="readline")`.

**Step 2** Execute the training script. After the training job is stopped, enter `run` in the command line. The training proceeds to the next step.

For more details, see help.

tfdbg> run
Step 3 Collect .npy files.

After the command is executed and returns the training result of the first step, you can run the `lt` command to query the stored tensors, run the `pt` command to view the tensor content, and save it as a file in NumPy format. The tfdbg dumps only one tensor at a time. To dump all tensors, perform the following steps:

1. Run the `lt > gpu_dump` command to temporarily store all tensor names to the `gpu_dump` file. The command output is as follows:

   Wrote output to tensor_name

2. Exit the tfdbg command line. In the Linux command line, run the following command in the directory where `gpu_dump` is stored to generate commands to run in tfdbg:

   ```
   timestamp=$[$(date +%s%N)/1000] ; cat gpu_dump | awk '{print "pt",$4,$4}' | awk '{gsub("/", "_", $3);gsub(":\", ",", $3);print($1,$2,-n 0 -w "$3";"$timestamp"".npy")}'
   ```

3. Copy all generated commands starting with `pt` and paste them to the tfdbg command line. Run the commands to save all .npy files. The files are saved to the directory where the training script is stored.

   By default, .npy files are stored using `numpy.save()`. Slashes (/) and colons (:) are replaced by underscores (_).

   **NOTE**

   If the command cannot be pasted on the CLI, run the `mouse off` command in the tfdbg command line to disable the mouse mode before pasting again.

4. Check whether names of the generated .npy files comply with the naming rules, as shown in Figure 9-10.

   **NOTE**

   - An .npy file is named in the format of `{op_name}.{output_index}.{timestamp}.npy`, where the `op_name` field must comply with the A-Za-z0-9_ regular expression, the `timestamp` field must comply with the [0-9]{1,255} regular expression, and `output_index` is a digit ranging from 0 to 9.
   - If the name of an .npy file exceeds 255 characters due to the long operator name, comparison of this operator is not supported.
   - The name of some .npy files may not meet the naming requirements due to the tfdbg or operating environment. You can manually rename the files based on the naming rules. If there are a large number of .npy files that do not meet the requirements, generate .npy files again by referring to 9.4.5 How Do I Handle Exceptions in the Generated .npy File Names in Batches?

Figure 9-10 Viewing the .npy files

```text
truediv_91.0.0123456789012345.npy
truediv_92.0.0123456789012345.npy
truediv_93.0.0123456789012345.npy
truediv_94.0.0123456789012345.npy
truediv_95.0.0123456789012345.npy
truediv_96.0.0123456789012345.npy
truediv_97.0.0123456789012345.npy
truediv_98.0.0123456789012345.npy
truediv_99.0.0123456789012345.npy
```
9.2.3.2 Preparing Dump Data and Computational Graphs of a Model on the Ascend AI Processor

Prerequisites

Before dumping data of a trained network after migration, ensure that the model is developed, built, and executed, and the training project is executable.

NOTICE

- If the training network contains random factors, remove them before dumping.
- Ensure that your code is the same as the code trained on the GPUs in terms of the network structure, operator, optimizer, and parameter initialization policy. Otherwise, the comparison is meaningless.
- Do not perform training and validation at the same time in a training script. That is, do not put training and validation in the same script. Otherwise, two groups of dump data will be generated and you cannot distinguish between them.
- Currently, only AI CPU and AI Core operators can be dumped. Operators such as Huawei Collective Communication Library (HCCL) operators cannot be dumped.

Dump Parameter Configuration

**Step 1** To enable the training script to dump computational graphs, introduce the OS to the package reference area in the training script and set the `DUMP_GE_GRAPH` parameter before building a model. In this way, during the training process, the computational graph file is saved in the directory where the training script is located.

```python
import os
...
def main():
    ...
    os.environ['DUMP_GE_GRAPH'] = '2'
```

**Step 2** Modify the script to enable the dump function. Add the lines in bold in the corresponding positions of the script.

- In Estimator mode, collect dump data using `dump_config` in `NPURunConfig`. Before `NPURunConfig` is created, instantiate a `DumpConfig` class for dump configuration, including the dump path, iterations to dump, and the dump mode (operator inputs or outputs). For details about each field in the constructor function of the `DumpConfig` class, see the Network Porting and Training Guide.

```python
from npu_bridge.estimator.npu.npu_config import DumpConfig

# dump_path: dump path. Create the specified path in advance in the training environment (either in a container or on the host). The running user configured during installation must have the read and write permissions on this path.
# enable_dump: dump enable.
# dump_step: iterations to dump.
# dump_mode: dump mode, selected from input, output, and all
dump_config = DumpConfig(enable_dump=True, dump_path="/home/HwHiAiUser/output",
            dump_step="0\[5\[10\]", dump_mode="all")

config = NPURunConfig(
    dump_config=dump_config,
    session_config=session_config
)
```
In `Session.run` mode, set the dump parameters by setting the session configuration items `enable_dump`, `dump_path`, `dump_step`, and `dump_mode`.

```python
config = tf.ConfigProto()

custom_op = config.graph_options.rewrite_options.custom_optimizers.add()
custom_op.name = "NpuOptimizer"
custom_op.parameter_map["use_off_line"]=True

# enable_dump: dump enable.
custom_op.parameter_map["enable_dump"].b = True
# dump_path: dump path. Create the specified path in advance in the training environment (either in a container or on the host). The running user configured during installation must have the read and write permissions on this path.
custom_op.parameter_map["dump_path"]=tf.compat.as_bytes("/home/HwHiAiUser/output")
# dump_step: step whose result you want to dump. Multiple steps are separated by vertical bars (|). Multiple consecutive steps can be expressed in the format such as 5-10.
custom_op.parameter_map["dump_step"]=tf.compat.as_bytes("0|5|10")
# dump_mode: dump mode, selected from input, output, and all.
custom_op.parameter_map["dump_mode"]=tf.compat.as_bytes("all")
config.graph_options.rewrite_options.remapping=RewriterConfig.OFF

with tf.Session(config=config) as sess:
    print(sess.run(cost))
```

Table 9-4 Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enable_dump</code></td>
<td>Data dump enable.</td>
</tr>
<tr>
<td></td>
<td>- <strong>True</strong>: enabled. The dump file path is read from <code>dump_path</code>.</td>
</tr>
<tr>
<td></td>
<td>- <strong>False</strong> (default): disabled.</td>
</tr>
<tr>
<td><code>dump_path</code></td>
<td>Dump path. Required if <code>enable_dump</code> is set to <strong>True</strong>.</td>
</tr>
<tr>
<td></td>
<td>The specified path must be created in advance in the environment (either in</td>
</tr>
<tr>
<td></td>
<td>a container or on the host) where training is performed. The running user</td>
</tr>
<tr>
<td></td>
<td>configured during installation must have the read and write permissions on</td>
</tr>
<tr>
<td></td>
<td>this path. The path can be an absolute path or a relative path relative to</td>
</tr>
<tr>
<td></td>
<td>the path where the command is executed.</td>
</tr>
<tr>
<td></td>
<td>- An absolute path starts with a slash (/), for example, <code>/home/HwHiAiUser/output</code>.</td>
</tr>
<tr>
<td></td>
<td>- A relative path starts with a directory name, for example, <code>output</code>.</td>
</tr>
<tr>
<td><code>dump_step</code></td>
<td>Iterations to dump. Defaults to <strong>None</strong>, indicating that all iterations are</td>
</tr>
<tr>
<td></td>
<td>dumped. Separate multiple iterations using vertical bars (</td>
</tr>
<tr>
<td></td>
<td><code>[0]5[10]</code>. You can also use hyphens (-) to specify the iteration range, for</td>
</tr>
<tr>
<td></td>
<td>example, <code>0[3-5]10</code>.</td>
</tr>
</tbody>
</table>
### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| dump_mode | Dump mode. The values are as follows:  
- **input**: dumps only operator inputs.  
- **output** (default): dumps only operator outputs.  
- **all**: dumps both operator inputs and outputs. |

#### End

### Performing Training to Generate Dump Data

**Step 1** Run the training script to generate the dump data file and computational graph file.

- Computational graph file: The file whose name starts with `ge` is the computational graph file generated when `DUMP_GE_GRAPH` is set to `2`. The file is stored in the directory where the training script is stored.

- Dump data file: The dump data file is generated in the directory specified by `dump_path`, that is, the `{dump_path}/(time)/(deviceid)/(model_name)/{model_id}/(data_index)` directory. For example, if `dump_path` is set to `/home/HwHiAiUser/output`, the dump data file is stored in the `/home/HwHiAiUser/output/20200808163566/0/ge_default_20200808163719_121/11/0` directory.

#### Table 9-5 Path format of a dump file

<table>
<thead>
<tr>
<th>Path Key</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump_path</td>
<td>Dump path set in <strong>Step 2</strong>. (If a relative path is set, the corresponding absolute path applies.)</td>
<td>--</td>
</tr>
<tr>
<td>time</td>
<td>Dump time.</td>
<td>Format: YYYYMMDDHHMMSS</td>
</tr>
<tr>
<td>deviceid</td>
<td>Device ID.</td>
<td>--</td>
</tr>
<tr>
<td>model_name</td>
<td>Subnetwork name.</td>
<td>If the <code>model_name</code> directory contains more than one folder, dump data in the folder with the same name as the computational graph is used. Periods (.), forward slashes (/), backslashes (), and spaces in <code>model_name</code> are replaced with underscores (_).</td>
</tr>
<tr>
<td>model_id</td>
<td>Subnetwork ID.</td>
<td>--</td>
</tr>
<tr>
<td>Path Key</td>
<td>Description</td>
<td>Remarks</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------------</td>
<td>--------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>data_index</td>
<td>Iterations to dump.</td>
<td>If <code>dump_step</code> is specified, <code>data_index</code> equals to <code>dump_step</code>. If it is not specified, <code>data_index</code> starts at 0 and is incremented by 1 with each dump.</td>
</tr>
<tr>
<td>dump_file</td>
<td>Format: <code>{op_type}.{op_name}. {taskid}.{stream_id}. {timestamp}</code> If the length of a file name formatted as required exceeds the OS file name length limit (generally 255 characters), the dump file is renamed a string of random digits. For details about the mapping, see the <code>mapping.csv</code> file in the same directory.</td>
<td>Periods (.), forward slashes (/), backslashes (), and spaces in <code>op_type</code> or <code>op_name</code> are replaced with underscores (_).</td>
</tr>
</tbody>
</table>

**NOTE**

- Dump data is generated in each iteration. A large training dataset generates a large volume of dump data (about dozens of GB or even more). You are advised to control the number of iterations to one.
- In the multi-device training scenario where more than one Ascend AI Processor is used, since the processes are not started at the same time as defined in the training script, multiple timestamp directories are generated when data is dumped.
- When the command is executed in a Docker, the generated data is stored in the Docker.

**Step 2** Select a computational graph file.

**NOTE**

There are a large number of dump graph files whose names start with `ge`, and multiple folders may exist at the `model_name` layer in the dump data file. In fact, you only need to find the computational graph file and the folder whose `model_name` is the name of the computational graph. You can use either of the following methods to quickly find the required file:

- **Method 1:** Search for the keyword `Iterator` in all dump files whose names end with `_Build.txt`. Record the name of the computational graph file, which will be used for accuracy comparison.
  ```bash
grep Iterator *_Build.txt
```

```
ascend@ubuntu:/Downloads/dump/npu$ grep Iterator *_Build.txt
name: "IteratorV2"
input: "IteratorV2:1"
input: "IteratorV2:0"
```
As shown in the preceding figure, the `ge_proto_00292_Build.txt` file is the desired computational graph file.

- Method 2: Save the TensorFlow model as a PB file, view the model, select the name of a computing operator as the keyword, and find the computational graph file that contains the keyword. The value of the `name` field in the computational graph is used as the name of the computational graph.

**Step 3** Select the dump data file.

1. Open the computational graph file found in **Step 2** and record the value of the `name` field in the first graph. In the following example, record the value `ge_default_20201209083353_71`.

```
graph {
  name: "ge_default_20201209083353_71"
  op {
    name: "atomic_addr_clean0_71"
    type: "AtomicAddrClean"
    attr {
      key: "fe_imply_type"
      value {
        i: 6
      }
    }
  }
```

2. Go to the directory for storing the dump file named after the timestamp. The following folders exist in the directory:

3. Find the folder whose name is the recorded value, for example, `ge_default_20201209083353_71`. The files in the folder are the required dump data files.

----End
9.3 Vector Comparison

9.3.1 Comparison Data Description

Before vector comparison, prepare the data by referring to Table 9-6.

**NOTE**

Select files for My Output as follows:

- Offline model file: Select this file for comparing the generated dump data of a model running on the Ascend AI Processor with those of Ground Truth.
- Quantization fusion file: required if to compare quantized and non-quantized dump data

<table>
<thead>
<tr>
<th>No.</th>
<th>Data to Be Compared (My Output)</th>
<th>Standard Data (Ground Truth)</th>
<th>Model File/Quantization Fusion File</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dump data of the non-quantized offline model running on the Ascend AI Processor</td>
<td>.npy file of the non-quantized Caffe model</td>
<td>Non-quantized offline model file (.om)</td>
</tr>
<tr>
<td>2</td>
<td>Dump data of the quantized offline model running on the Ascend AI Processor</td>
<td>.npy file of the non-quantized Caffe model</td>
<td>Quantized offline model file (.om) Quantization fusion file (.json) after model compression</td>
</tr>
<tr>
<td>3</td>
<td>.npy file of the quantized Caffe model</td>
<td>.npy file of the non-quantized Caffe model</td>
<td>Quantization fusion file (.json) after model compression</td>
</tr>
<tr>
<td>4</td>
<td>Dump data of the quantized offline model running on the Ascend AI Processor</td>
<td>.npy file of the quantized Caffe model</td>
<td>Quantized offline model file (.om)</td>
</tr>
<tr>
<td>5</td>
<td>Dump data of the non-quantized offline model running on the Ascend AI Processor</td>
<td>.npy file of the non-quantized TensorFlow model</td>
<td>Non-quantized offline model file (.om)</td>
</tr>
<tr>
<td>6</td>
<td>Dump data of the non-quantized offline model running on the Ascend AI Processor</td>
<td>.npy file of the non-quantized ONNX model</td>
<td>Non-quantized offline model file (.om)</td>
</tr>
<tr>
<td>No</td>
<td>Data to Be Compared (My Output)</td>
<td>Standard Data (Ground Truth)</td>
<td>Model File/Quantization Fusion File</td>
</tr>
<tr>
<td>----</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>7</td>
<td>Dump data of the non-quantized offline model running on the the Ascend AI Processor</td>
<td>Dump data of the non-quantized offline model running on the the Ascend AI Processor</td>
<td>None</td>
</tr>
<tr>
<td>8</td>
<td>Dump data of the quantized offline model running on the the Ascend AI Processor</td>
<td>Dump data of the quantized offline model running on the the Ascend AI Processor</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Training scenario</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Dump data of the model running on the the Ascend AI Processor</td>
<td>.npy file of the TensorFlow network</td>
<td>Computational graph file (*.txt)</td>
</tr>
</tbody>
</table>
9.3.2 Comparison Procedure

Procedure

- The data to be compared should be obtained from the counterpart models. Otherwise, the comparison results of only the counterpart operators are displayed.
- For the Fast R-CNN network, the comparison result is subject to the accuracy of the FSRDetectionOutput operators. It is normal that the ProposalID operator and its downstream operators offer low accuracy.
- During graph build, if some operators of the graph are fused, the outputs of these operators can no longer be found in the built model. As a result, the comparison of these operators is unavailable.
- During graph build, if the structure of a graph is modified (such as strided slice, L1 fusion, and L2 fusion), the comparison of the inputs or outputs of the operators is unavailable.
- When the counterpart operators require different shapes (for example, the offline model operator requires a reduced shape), or format conversion is not supported, the comparison of these operators is unavailable.
- If Data Pre-Processing is switched on (for example, input of the data operator is set to YUV in AIPP), input format of the data operators may be different from that of the original model, leading to an unreliable comparison result.
- If the corresponding fusion rule is enabled, a quantized operator will be fused with its upstream operator. As a result, the comparison result of the operator output is unreliable.
- In a quantized model, the comparison of quantized operators is only available after being dequantized. Comparison of non-quantized operators in a quantized model is available. For example, in a quantized model, the comparison of the output of the AscendQuant operator is not available.
- The path can contain only letters (a–z and A–Z), digits (0–9), and the following special characters: periods (.), forward slashes (/), backslashes (\), colons (:), underscores (_), and hyphens (-).
- For an optimal experience, the following hardware configuration is recommended: 8-core CPU at 2.6 GHz with 16 GB memory.

Perform the following steps to conduct vector comparison:

**Step 1** In MindStudio, choose Ascend > Model Accuracy Analyzer from the main menu. The comparison dialog box is displayed.

**Step 2** Click next to Dump Path in My Output to set the My Output path and select the folder of the .npy or dump data, as shown in Figure 9-11. In the training scenario, select the dump result directory found in Step 3.

**Figure 9-11 Setting the dump path of My Output**
Step 3 Click next to Dump Path in Ground Truth to set the Ground Truth path and select the folder of the .npy or dump data, as shown in Figure 9-12.

Figure 9-12 Setting the dump path of Ground Truth

Step 4 In the Compare Rule Config area, select the Model File and Quant Fusion File, as shown in Figure 9-13.

Figure 9-13 Setting Compare Rule Config

NOTE
The situation shown in Figure 9-13 is used as an example. The configuration items in Compare Rule Config vary with the comparison scenario. Configure this area as prompted.
- Model File: Select an offline model file.
- Model File/Fusion Rule File: In inference scenarios, select an offline model file. In training scenarios, select a computational graph file found in Step 2.
- Quant Fusion File: Select a quantization fusion file.

Step 5 (Optional) Set algorithm thresholds.

Figure 9-14 Setting algorithm thresholds

- **CosineSimilarity**: defaults to 0.99, indicating that CosineSimilarity results less than 0.99 are highlighted in red while those greater than or equal to 0.99 are displayed in green. The value range is [0.00, 1.00].
- **MaxAbsoluteError**: defaults to 5, indicating that the largest five MaxAbsoluteError results are highlighted in red (could be more if some operators have equal MaxAbsoluteError results) while the rest MaxAbsoluteError results are displayed in green. The value range is [1, 5].
- **RelativeEuclideanDistance**: defaults to 0.10, indicating that RelativeEuclideanDistance results greater than 0.10 are highlighted in red while those less than or equal to 0.10 are displayed in green. The value range is [0.00, 1.00].

NOTE
You can enter a number and press Enter for the setting to take effect.
If you change a threshold value after the comparison results are displayed, click the column header whose threshold is modified for the modification to take effect.
Step 6  Click **Compare**.

For details about the comparison results, see **Comparison Results**.

----End

**Comparison Results**

The vector comparison results are described as follows.

**Figure 9-15 Vector comparison results**

Only operators with results are displayed by default. You can click a column header to sort the records.

- **LeftOp**: operator name of the **My Output** data
- **RightOp**: operator name of the **Ground Truth** data
- **TensorIndex**: input ID and output ID of the **My Output** operator
- **CosineSimilarity**: result of the cosine similarity comparison. The value range is \([-1, +1]\). A value closer to 1 indicates higher similarity. A value closer to \(-1\) indicates greater difference.
- **Max Absolute Error**: result of the maximum absolute error comparison. A value closer to 0 indicates higher similarity. Otherwise, it indicates greater difference.
- **Accumulated Relative Error**: result of the accumulated relative error comparison. A value closer to 0 indicates higher similarity. Otherwise, it indicates greater difference.
- **Relative Euclidean Distance**: result of the Euclidean relative distance comparison. A value closer to 0 indicates higher similarity. Otherwise, it indicates greater difference.
- **Kullback-Leibler Divergence**: result of the KLD comparison. The value ranges from 0 to infinity. The smaller the KLD, the closer the approximate distribution is to the true distribution.
- **Standard Deviation**: result of the standard deviation comparison. The value ranges from 0 to infinity. The smaller the standard deviation is, the smaller the dispersion is, and the closer the value is to the average value. The mean value and standard deviation of the dump data are displayed in the format of (mean value;standard deviation). The first set of data is the result of **My Output**, and the second set is the result of **Ground Truth**.
- **Compare Fail Reason**: reason why accuracy comparison fails.
- If the results of cosine similarity and KLD are \textbf{NaN}, and the results of other
algorithms exist, at least one data on the left or the right is 0. If the result of KLD is \textbf{inf}, one data on the right is 0. If \textbf{NaN} is displayed, the dump data contains \textbf{NaN}.

\textbf{NOTE}
- Place the mouse pointer on the table header to view the parameter description.
- Click \textbf{More Help...} to go to the parameter description window. To use this function, you need to use a browser. If no browser is available, install one.

\textbf{Comparison Result Details}

Double-click a specific operator. The comparison result details are displayed, as shown in \textbf{Figure 9-16}.

\textbf{NOTE}
You can click \textbf{ to quickly locate and select values that are marked in red in the \textit{Absolute Error} column.}

When the detailed comparison result is displayed, the \texttt{\{op name\}_{output/input}\_{index}\_{file index}.csv} result file is generated in the comparison result path in the home directory of the MindStudio installation user, for example, \texttt{~/.mindstudio/compare_tool/result/detail_result/pool1_output_0_0.csv}. If the .csv file cannot be opened in the Windows operating system, the possible cause is that the file name is too long. Rename the file and try again.
The following describes the parameters of comparison result details:

- **Absolute Error**: absolute error. The value range is [0.000000, 0.006815], where, 0.004212 is the golden point of an absolute error. If an absolute error is greater than or equal to 0.004212, the value is marked in red. You can enter an integer or a floating-point number with a maximum of six decimal places.

- **Relative Error**: relative error. The value range is [0.000000, 263.258065], where, 162.693484 is the golden point of relative error. If a relative error is greater than or equal to 162.693484, the value is marked in red. You can enter an integer or a floating-point number with a maximum of six decimal places.

- **N C H W**: data format

- **Left**: dump value of the *My Output* operator

- **Right**: dump value of the *Ground Truth* operator

- **Absolute Error**: absolute error. The value is the difference between the dump value of the *My Output* operator and that of the *Ground Truth* operator.

- **Absolute Error**: relative error. The value is obtained by dividing the Absolute Error value by the dump value of the *Ground Truth* operator. If the dump value of the *Ground Truth* operator is 0, a hyphen (-) is displayed.
Operator Mapping

After the accuracy comparison is complete, you can click **Show Model** to display the operators in the model graphs, as shown in Figure 9-17.

**Figure 9-17 Viewing model graphs**

In the **Model Compare** dialog box displayed on the right, set the model file.

- In the **My Model** area, click **Input Model** and select the **My Output** model.
- In the **Standard Model** area, click **Input Model** and select the **Ground Truth** model.

After the settings are complete, the topologies of the two models are automatically drawn in MindStudio, as shown in Figure 9-18.

- The model colors are the same as the cosine similarity algorithm column.
- Select a row of accuracy analysis results. The **LeftOp** and **RightOp** are highlighted in the model graph on the right. Select an operator in the graph on the right. The corresponding row of comparison results is highlighted. The **LeftOp** and **RightOp** are also highlighted.
9.3.3 Subnetwork Export and Comparison

Overview

The offline model inference result on the Ascend AI Processor may be different from that of the counterpart original model. The Model Accuracy Analyzer locates the accuracy drop causes at layers that are not customized (which constitute a subnetwork) so that you can compare the inference results of these layers. To simplify the workflow, you can leverage the Caffe or TensorFlow subnetwork export capability provided by MindStudio to easily export the subnetwork. Then, send the exported subnetwork to Huawei engineers to further locate the fault. Figure 9-19 describes the export workflow.
1. Convert the original model into an offline model adapted to the Ascend AI Processor using ATC by referring to 4 Model Conversion.

2. Start inference with the offline model.

3. If the inference result is inconsistent with that of the original model, use the Model Accuracy Analyzer to locate the problem layers, and export a subnetwork consisting of the problem layers of the original model by referring to Export Procedure. Repeat the subnetwork export and accuracy comparison workload to narrow down the problem layers.

4. Send the exported subnetwork to Huawei engineers for fault locating.
   - If the fault is rectified, convert the model again and perform model inference.
Export Portals

The following two portals are provided to export a subnetwork. This section uses the Model Accuracy Analyzer window as an example.

- In the Model Accuracy Analyzer window: After setting Dump Path and Model File, click Dump Network to go to the subnetwork export window. In this case, Dump Data Path and Offline Model File are automatically filled in. You can also click on the right to set them manually.

- In MindStudio: Choose Ascend > Dump Network to go to the subnetwork export window. In this case, Dump Data Path and Offline Model File are not automatically filled in. You need to click on the right to set them manually.

Export Procedure

Step 1 Perform Step 1 to Step 4 by referring to 9.3.2 Comparison Procedure.

Step 2 After setting Model File and Dump Path, click Dump Network in the Model Accuracy Analyzer window. The subnetwork export window is displayed as shown in Figure 9-20.

![Exporting a subnetwork](image)

Table 9-7 Parameter and button description

<table>
<thead>
<tr>
<th>Parameter and Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dump Data Path</td>
<td>Dump path set in Dump Path in the My Output area in the Model Accuracy Analyzer window. This parameter is automatically filled when the Dump Network dialog box is opened. You can also click on the right to select a dump file manually.</td>
</tr>
<tr>
<td>Offline Model File</td>
<td>Offline model file set in Model File in the Model Accuracy Analyzer window. This parameter is automatically filled when the Dump Network dialog box is opened. You can also click on the right to select an offline model file manually.</td>
</tr>
</tbody>
</table>
### Parameter and Button

<table>
<thead>
<tr>
<th>Parameter and Button</th>
<th>Description</th>
</tr>
</thead>
</table>
| Deploy Model         | ● Original model file. Click ➔ on the right to select the file on the MindStudio installation server.  
                        ● For a model file that is compressed by using AMCT, select the quantized model file that can be deployed on the Ascend AI Processor. |
| Deploy Weight        | ● Weight file of the original model. If the model file and weight file are stored in the same directory on the MindStudio installation server, this parameter is automatically set after selecting the model file. Otherwise, click ➔ on the right to select it manually.  
                        ● For a weight file that is compressed by using AMCT, select the quantized weight file that can be deployed on the Ascend AI Processor.  
                        **NOTE** This parameter is available only when the original model is a Caffe model. Other frameworks do not require `.caffemodel` weight files. |
| Dump Output Path     | Subnetwork export path. Click ➔ on the right to select an export directory. If the directory does not exist, create one. |
| Show Result          | Displays the parsing result of the original model in a chart on the right. |
| Dump                 | Exports a subnetwork. |

**Step 3** After setting the parameters, click **Show Result** to parse the original model file. If **Model Name** of the original model file after conversion is different from that of the original Caffe model file (.prototxt) or TensorFlow model file (.pb), the dialog box shown in Figure 9-21 is displayed during parsing.

- Click **Yes** to continue the parsing and obtain the parsing results, as shown in Figure 9-22.
- Click **No** to return to the subnetwork export window.

**Figure 9-21** Inconsistent Model Names

![Figure 9-21: Inconsistent Model Names](image)

**Figure 9-22** shows that the original model is successfully parsed.
Figure 9-22 Parsing results

Where,

- The nodes in red are operators with cosine similarity values less than the algorithm threshold of vector comparison.
- For a node that consists of multiple operators, select the node, and the node is outlined green and all operators in the node are displayed in the central area of Figure 9-22.
  - **View the operator details.**
    In the central area of Figure 9-22: Click \( \wedge \) to expand the details. Click \( \vee \) to view the details of a specific metric. Click \( \wedge \) to collapse the details.
  - **View the operator output shape.**
    Figure 9-22 also displays the output format (for example, NCHW) and shape (for example, 1,3,224,224) of each operator on the connection lines.
  - **Search for an operator.**
    In Figure 9-22, click **Op Search**. The search area is displayed below. All operators on the network are listed in the search area. Enter an operator name in the search text box. The matched operators are displayed. Select an operator, and the operator is outlined green in the network topology on the left. Details about the operator are displayed in the central area, including the operator name and the inputs and outputs.
  - **Find operator details.**
    Select a layer and click \( \wedge \). All operators at this layer are displayed below. Click \( \square \) or press **Ctrl+F** to find a specific operator metric in the displayed **Find** dialog box, as shown in Figure 9-23.
- **Case sensitive**: If selected, the search operation follows the exact letter case of the entered keyword.
- **Wrap Search**: If selected, the search operation is continued from the start of the list when the end of the list is reached, until all operators are covered.
- **Backward**: If selected, the search operation is from current position to the start of the list.

In **Figure 9-23**, enter the keyword to search for in the text box, select required search options, and click **Find**. The matched information in the list will be highlighted. If no match is found, the message **Value not found** is displayed. Click **Close** to exit the search.

**Figure 9-23** Finding operator details

---

**Step 4** To create a subnetwork for further locating, right-click the start node (input) and choose **Start Layer** from the shortcut menu, turning the node yellow, and then right-click the end node (output) and choose **End Layer** from the shortcut menu, turning the node purple. Green boxes appear around the nodes between the **Start Layer** and **End Layer**, indicating that they are selected, as shown in **Figure 9-22**.

**NOTE**

If an invalid start node or end node are selected, the node is outlined in red, and an error message is displayed. In this case, the **Dump** icon is dimmed and dump export fails. You can reselect the start node and end node as prompted.

- With the input and output nodes selected, if you want to modify the selection, right-click a new node and select the corresponding attribute as required. The color of the new node changes accordingly, and the previously selected node is restored to the original color.
- If your subnetwork consists of only one node, right-click the node and choose **Start Layer** and **End Layer** from the shortcut menu, and the node turns yellow and then purple. In the dialog box shown in **Figure 9-24**, click **OK** to export a single-node subnetwork by referring to **Step 5**.

**Figure 9-24** A node as both the start layer and end layer

---

**Step 5** Click **Dump** to export the subnetwork.
MindStudio searches for the subnetwork from the original network model in the background. The export progress is displayed in the lower right corner. You can click the progress bar to view the detailed progress.

- The message shown in **Figure 9-25** indicates an export success.

**Figure 9-25** Export success message

- If the input shape for a layer cannot be found during subnetwork export, the message shown in **Figure 9-26** is displayed. In this case, you need to view the original model file and enter the shape manually. Separate the dimension values by colons (:). As shown in **Figure 9-26**, the shape of the input operator pool1 of the res2a_branch1 operator fails to be parsed. In this case, view the original model file to obtain the output shape (1:64:56:56, for example) and enter 1:64:56:56 in the text box.

**Figure 9-26** Input shape for the layer not found

**Step 6** After the export is successful:

1. In the directory set in **Dump Output Path** on the MindStudio installation server, find the folder formatted as dump_result_xxxx_xx_xx_xx_xx_xx, where xxxxx_xx_xx_xx_xx_xx is a timestamp. In the folder:
   - **dump.prototxt**: model file of the Caffe subnetwork
   - **dump.pb**: model file of the TensorFlow subnetwork
   - **dump.caffemodel**: weight file of the Caffe subnetwork
   - **dump_data**: dump data of all operators of the subnetwork
   
   This folder also contains the input data of the **Start Layer**, which is used as the input data for subnetwork inference.

   You can perform further accuracy comparison using the exported subnetwork or directly send it to Huawei engineers for fault locating. If the problem is solved, go to the next step.

2. Find the export log messages in the **Output** window in MindStudio. The following gives some example log messages.

   ```
   // Define the input information of the subnetwork.
   2020-05-22 16:27:57 Add Layer Into Prototxt:
   name: "pool1_input_0"
   type: "Input"
   top: "conv1"
   input_param {
     shape {
       ...
   
   ```
Step 7  Export the subnetwork and perform inference with the subnetwork.

1. Convert the .prototxt file of the subnetwork exported in Step 6.1 into an .om file by referring to 4 Model Conversion.

   If the exported subnetwork contains no input data node, turn the AIPP switch off during model conversion.

2. Convert the input data of the Start Layer into the binary format, which is used as the input of the offline model during inference.

   a. Search for the input data of the Start Layer.

      Search for the file that is named the same as the Input File in the dump_data directory on the MindStudio installation server.

   b. Convert the input data in Protobuf format into the NumPy format.

      Log in to the MindStudio installation server, switch to the {Ascend-CANN-Toolkit installation path}/ascend-toolkit/{version}/{arch}-linux/toolkit/tools/operator_cmp/compare directory, and run the following command for format conversion:

      ```python
      python3.7.5 shape_conversion.pyc -i<absolute path of the input data in Protobuf format> -o<path of the .npy file>
      ```

      You can replace the paths as required. After conversion, you can find the input data (.npy) in the specified path.

   c. Convert the input data in NumPy format into the binary format.

      Replace the data path in the following script with the actual path and save the script as a .py file (for example, subnetworkdataprocess.py).

      ```python
      import numpy as np

      # Convert the file in NumPy format into the binary format.
      conv1_relu_0 = np.load("/home/hisisoc/dumpdate/subNetworkData/xxx.npy")  # absolute path of the input data in NumPy format
      conv1_relu_0 = conv1_relu_0.astype(np.float16)
      conv1_relu_0.tofile("/home/hisisoc/dumpdate/subNetworkData/xxx.bin")  # storage path of data in binary format
      ```
Upload the `subnetworkdataprocess.py` script to any directory on the MindStudio installation server. Run the following command to convert the `.npy` file into the binary format:

```
python3.7.5 subnetworkdataprocess.py
```

In the binary file storage path, you can find the converted `.bin` file.

3. Perform inference in the original environment using the subnetwork, and perform inference in the HiSilicon SoC board environment using the offline model and input data generated in Step 7.1 and Step 7.2. Compare the inference results. If the difference persists, perform accuracy comparison using the subnetwork generated in Step 6.1 and the offline model generated in Step 7.1 by referring to 9 Model Accuracy Analyzer.

4. Repeat the subnetwork export and subnetwork accuracy comparison, minimize the subnetwork, and send the exported data to Huawei engineers for fault locating.

--- End

9.4 FAQs

9.4.1 What Do I Do If an Error Message "The data in dump file have zero" Is Displayed During Vector Comparison?

**Symptom**

During vector comparison, an error message is displayed, as shown in Figure 9-27.

**Figure 9-27** Error message: The data in dump file have zero

**Possible Cause**

Dump data in Output contains 0. As a result, computation cannot be executed.

**Solution**

Delete the dump file displayed in Output before performing accuracy comparison.
9.4.2 What Do I Do If an Error Message "ModuleNotFoundError: No module named 'google'" Is Displayed During Vector Comparison?

Symptom

During vector comparison, an error message is displayed, as shown in Figure 9-28.

Figure 9-28 Error message: Protobuf is not installed

The following message is displayed in the output column:

```
```

Possible Cause

Protobuf is not installed in the OS. As a result, an error occurs when the dump data of the Caffe model is parsed.

Solution

Run the following command to install Protobuf before accuracy comparison.

Log in to the MindStudio installation server as the MindStudio installation user, and run `pip3.7.5 install protobuf --user`.

9.4.3 How Do I Convert Between Dump File and NumPy File?

Overview

The `dump_data_conversion.pyc` script is provided to implement the conversion between dump files and NumPy files. The command syntax is as follows:

```
python3.7.5 dump_data_conversion.pyc -type TYPE -target TARGET -i INPUT_PATH -o OUTPUT_PATH
```

- `-type`: (required) data type, selected from:
  - `quant`: quantized Caffe model data
  - `tf`: non-quantized TensorFlow model data
  - `caffe`: non-quantized Caffe model data
- offline: offline model data

- -target: (required) target format, either numpy or dump
  - numpy: converts a dump file into a NumPy file.
  - dump: converts a NumPy file into a dump file.

- -i: (required) data folder/file path
  - Converting NumPy files into dump files:
    To pass a folder path to i, ensure that names of the files in the folder are in op_name.output_index.timestamp.npy format.
    To pass a file path to i, ensure that the name of the file is in op_name.output_index.timestamp.npy format. Only one file can be set at a time.
    The value of op_name must comply with the A-Za-z0-9_- regular expression rule. The value of output_index consists of digits 0-9. The value of timestamp must comply with the [0-9]{1,255} regular expression.
  - Converting dump files into NumPy files:
    To pass a folder path to i, ensure that names of the files in the folder comply with the naming conventions described in 9.2.1 Dump File Naming Conventions. To pass a file path to i, ensure that the name of the file complies with the naming conventions described in 9.2.1 Dump File Naming Conventions.

- -o: (required) conversion output path

**NOTE**

The dump_data_conversion.pyc script is stored in {Ascend-CANN-Toolkit installation path}/ascend-toolkit/[version]/[arch]-linux/toolkit/tools/operator_cmp/compare.

The execution of this script depends on the NumPy, Protobuf, and SciPy libraries. You need to run the pip install numpy protobuf scipy --user command to install them. If they have been installed as part of the MindStudio installation, skip this step.

To use this script for conversion, ensure that the host has at least 15 GB disk space. If the size of a single dump file to be converted exceeds 441 MB, you are advised to use a host with larger disk space.

**Converting the Original Caffe .npy File into a Dump Data File**

Upload the original Caffe .npy file to the MindStudio server and execute the dump_data_conversion.pyc script. A dump file in binary format is generated.

The following gives an example command:

```bash
python3.7.5 dump_data_conversion.pyc -type caffe -target dump -i /home/HwHiAiUser/caffenpyfile -o /home/HwHiAiUser/caffedump
```

**NOTE**

- /home/HwHiAiUser/caffenpyfile: path of the original .npy file
- /home/HwHiAiUser/caffedump: path of the target dump file
Converting the Original TensorFlow .npy File into a Dump Data File

Upload the original TensorFlow .npy file to the MindStudio server and execute the `dump_data_conversion.pyc` script. A dump file in binary format is generated.

The following gives an example command:

```
python3.7 dump_data_conversion.pyc -type tf -target dump -i /home/HwHiAiUser/tfnpyfile -o /home/HwHiAiUser/tfdump
```

**NOTE**
- `/home/HwHiAiUser/tfnpyfile`: path of the original .npy file
- `/home/HwHiAiUser/tfdump`: path of the target dump file

### 9.4.4 How Do I View a Dump File?

Dump files cannot be viewed with a text tool. Therefore, you need to convert your dump file into a NumPy file and save the NumPy file as a text file using `numpy.savetxt`.

**Step 1** Log in to the development environment as the installation user.

**Step 2** Go to the `/home/HwHiAiUser/Ascend/ascend-toolkit/latest/toolkit/tools/operator_cmp/compare` directory.

**Step 3** Run the `dump_data_conversion.pyc` script to convert the dump file into a NumPy file. The following is an example:

```
python3.7.5 dump_data_conversion.pyc -target numpy -type offline -i /home/HwHiAiUser/dump -o /home/HwHiAiUser/dumptonumpy
```

**NOTE**
For details about the parameters of the `dump_data_conversion.pyc` script, see 9.4.3 How Do I Convert Between Dump File and NumPy File?

**Step 4** Use Python to save the NumPy data into a text file.

```
>>> import numpy as np
>>> a = np.load("/home/HwHiAiUser/dumptonumpy/Pooling.pool1.1147.1589195081588018.output.0.npy")
>>> b = a.flatten()
>>> np.savetxt("/home/HwHiAiUser/dumptonumpy/Pooling.pool1.1147.1589195081588018.output.0.txt", b)
```

The dimension information and Dtype no longer exist in the .txt file. For details, visit the NumPy website.

---End
9.4.5 How Do I Handle Exceptions in the Generated .npy File Names in Batches?

When generating dump data of a TensorFlow model, the names of some .npy files may be truncated due to the tfdbg or operating environment. As a result, the .npy file names may not meet the naming requirements, and dump file conversion may fail. In this case, perform the following steps to re-generate the .npy files:

**NOTE**
- The script names and paths in this document are used as examples. Replace them as required.
- After batch processing, if the dump file of an operator exists but the comparison result is NaN, check whether the `{op_name}` field of the dump file name of the operator is the same as the TensorFlow operator name. If not, manually change `{op_name}` to the TensorFlow operator name. If a slash (/) exists, replace it with an underscore (_).

**Step 1** Execute the TensorFlow project.
In the interactive debugger command line, enter `run` to run the script.

**Step 2** Run `lt > tensor_name` to temporarily store all tensor names to a file.

**Step 3** Create an executable script, for example, `pt_cmd.sh`, to obtain the `tensor_index` corresponding to `tensor_name` in the `tensor_name` file.

The script content is as follows.

```bash
#!/bin/bash
timestamp=$(date +%s%N)/1000
index=1
while read -r line
do
tensor_index=`echo $line | awk '{print $4}'`
echo "pt "$tensor_index" -n 0 -w "$((index++))".timestamp.npy" >> $2
done < $1
```

Grant the execute permission on the `pt_cmd.sh` script and execute the script.

```bash
chmod +x pt_cmd.sh
bash pt_cmd.sh tensor_name tensor_name.txt
```

**Step 4** Go back to the tfdbg command line, run the script, and paste and execute the content in the `tensor_name.txt` file generated in the previous step to save all .npy files.

**Step 5** Move the generated .npy files to a new folder, for example, `npy_dir`.

**Step 6** Create an executable script, for example, `index_to_tensorname.sh`, and run the script to change the .npy file names in batches.

The script content is as follows.

```bash
#!/bin/bash
timestamp=$(date +%s%N)/1000
while read -r line
do
tensor_index=`echo $line | awk '{print $2}'`
real_file=`echo $line | awk '{print $6}'`
changed1_tensor_index=${tensor_index////_}
changed2_tensor_index=${changed1_tensor_index//:/.}
ext
```

```bash
# MindStudio User Guide
9 Model Accuracy Analyzer
Issue 01 (2021-06-24) Copyright © Huawei Technologies Co., Ltd. 462
```
```bash
echo $2/$real_file $2/$changed2_tensor_index".$timestamp".npy
if [ -r $2/$real_file ]
    then
        mv $2/$real_file $2/$changed2_tensor_index".$timestamp".npy
    fi
done < $1
```

Grant the execute permission on the `index_to_tensorname.sh` script and execute the script.

```bash
chmod +x index_to_tensorname.sh
bash index_to_tensorname.sh tensor_name.txt npy_dir
```

----End
10 Profiling

10.1 Overview

Profiling collects host and device profile data, offers comprehensive timelines for interactive analysis, and displays detailed host and device profiling analysis reports. It helps identify performance optimization opportunities in your AI applications by accurate location of bottlenecks in algorithm performance and system resource utilization. The tool profiles and visualizes resource usage on the host and device. Specifically, host-side system profiling focuses on the CPU, memory, disk, and network while device-side system profiling focuses on hardware and software performance data of an application project.

Profiling collects, analyzes, and summarizes the hardware and software performance data of application projects. The general process is as follows:

1. Start the Profiling tool.
   Enable Profiling in the configuration window with the premise that the application project can be properly executed.

2. Profiling collects profile data.
   MindStudio builds the current project to generate an executable file and copies the file to the device. Then MindStudio delivers an instruction to Profiling that profiles the device and host. When the instruction is complete, copy the generated data file to MindStudio.

3. MindStudio queries and analyzes profile data.
   After data collection, MindStudio queries the data with the Profiling API and stores the data in JSON format to the project directory~/profiling path.
4. MindStudio displays profiling results.

MindStudio processes the JSON file to display profiling results in frontend. To view the results graphically, you can right-click the project file name and choose View Profiling Result from the shortcut menu.

10.2 Restrictions

- It is not allowed to initiate two profiling tasks with the same result directory from the MindStudio installation server.
- Dump and Profiling should not be both enabled. To enable Profiling, disable dump. Dump could affect the system performance, resulting in inaccurate profile data collected by Profiling. For details about dump, see 9.2.2.1 Preparing Dump Data of an Offline Model.
- Profiling is used to collect host data in the development or testing environment. Do not use it in the production environment. Otherwise, privilege escalation with sudo may occur.
- Profiling cannot collect host data in the Ascend RC scenario.
- Currently, Profiling applies to the inference scenario only.
- When profile data is displayed, if you choose Ascend > System Profiler > Open on the MindStudio menu bar, click in the upper left corner of the profiling report page, or directly open a historical report on the profiling report page, you are advised not to select the profiling report of an earlier version as profile data of the current version is incompatible with that of the earlier version.

10.3 Before You Start

Develop a project file by referring to 7 Application Development. Build and run the project in MindStudio to ensure that the application project runs properly.

10.4 Profile Data Collection

To start Profiling, perform the following steps:

Step 1  Open a built project. See Figure 10-1.
Figure 10-1 Opening a built project

- MyApp2
  - /AscendProjects/MyApp2
  - arm_mdc_app

- MyApp12
  - /AscendProjects/MyApp12
  - tbe_operator_sample

- MyApp1
  - /AscendProjects/MyApp1

Step 2 On the menu bar, choose Ascend > System Profiler > New Project. The Profiling configuration window is displayed.

Step 3 Under Project Properties, set Project Name and Project Location. See Figure 10-2. Click Next.
Step 4 The **Executable Properties** page is displayed. You can select **Remote Run** or **Local Run**. See Figure 10-3 and Figure 10-4.
Figure 10-3 Remote Run

Run Mode
- Remote Run
- Local Run

SSH Connection:

App Project Path:

Executable File:

Command Arguments:

Environment Variables:

Remote Toolkit Path:
Figure 10-4 Local Run

Table 10-1 Executable Properties parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| Run Mode           | ● Remote Run  
                     ● Local Run  
                     This parameter is unavailable in Windows. |
| SSH Connection     | SSH connection, available when the Remote Run mode is selected. This parameter is mandatory. For details, see 12.1 SSH Connection Management. 
                     Select the IP address of the remote server that runs the application from the drop-down list box. If the IP address is not added, click + to add it. For details, see 12.1.3 Adding a New SSH Connection. |
<p>| App Project Path   | Application project path, available when you select the Remote Run mode. This parameter is mandatory. |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executable File</td>
<td>Path for storing the executable file of the application project. This parameter is mandatory.</td>
</tr>
<tr>
<td>Command Arguments</td>
<td>Application execution parameters. Configure this as required and separate arguments with spaces. By default, this parameter is left empty.</td>
</tr>
<tr>
<td>Environment Variables</td>
<td>Environment variable configuration. Enter environment variables in the text box or click to add more by clicking in the displayed dialog box. This parameter is optional.</td>
</tr>
<tr>
<td>Remote Toolkit Path</td>
<td>Installation path of the toolkit software package in the remote operating environment, available when you select the Remote Run mode. This parameter is mandatory. For example, set this parameter to /home/HwHiAiUser/Ascend/ascend-toolkit/latest/toolkit.</td>
</tr>
</tbody>
</table>

**Step 5** The Profiling Options page is displayed. You can configure Task-based or Sample-based in AI Core Profiling. See Figure 10-5 and Figure 10-6.
Figure 10-5 Task-based scenario

- Project Properties
- Executable Properties
- Profiling Options

Default

- AI Core Profiling
  - Task-based
  - Sample-based
  - Pipeline Utilization
  - UB/L1/L2/Main Memory Bandwidth

API Trace
- ACL API
- Runtime API
- OS Runtime API
- No Graph Engine (GE)

Device System Sampling
- DDR
  - Frequency (Hz): 50

Host System Sampling
- CPU
- Memory
- Disk
- Network
Figure 10-6 Sample-based scenario

Table 10-2 Profiling Options parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI Core Profiling</td>
<td>AI Core profiling switch. It collects profile data task by task.</td>
</tr>
<tr>
<td>Task-based</td>
<td></td>
</tr>
<tr>
<td>Pipeline Utilization</td>
<td></td>
</tr>
<tr>
<td>UB/L1/L2/Main Memory Bandwidth</td>
<td>ratio of time taken by the compute units to that of MTEs.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pipeline Utilization: ratio of time taken by the compute units to that of MTEs.</td>
</tr>
<tr>
<td></td>
<td>UB/L1/L2/Main Memory Bandwidth: memory read/write rate.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Sample-based</td>
<td>AI Core profiling switch. It collects profile data at a fixed interval (AI Core-Sampling Interval).</td>
</tr>
<tr>
<td></td>
<td>- <strong>Pipeline Utilization</strong> and <strong>UB/L1/L2/Main Memory Bandwidth</strong> are defined previously.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Frequency(hz)</strong>: sampling frequency (Hz). The default value is <strong>100</strong>.</td>
</tr>
<tr>
<td>API Trace</td>
<td><strong>ACL API</strong></td>
</tr>
<tr>
<td></td>
<td>AscendCL profiling switch. It traces AscendCL API calls. This option is selected by default and cannot be deselected.</td>
</tr>
<tr>
<td></td>
<td><strong>Runtime API</strong></td>
</tr>
<tr>
<td></td>
<td>Runtime profiling switch. It traces Runtime API calls. This option is selected by default and cannot be deselected.</td>
</tr>
<tr>
<td>OS Runtime API</td>
<td>Sets the function library API and Pthreads API during system running. This parameter is optional and is selected by default.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong></td>
</tr>
<tr>
<td></td>
<td>To collect OS Runtime API call data, you need to install the third-party open-source tools perf and ltrace. Using ltrace to collect OS Runtime API call data may cause high CPU usage. In addition, using this tool is related to the application's pthread locking and unlocking, which may affect the application process running speed.</td>
</tr>
<tr>
<td></td>
<td><strong>Graph Engine (GE)</strong></td>
</tr>
<tr>
<td></td>
<td>Graph Engine profiling switch. It traces the scheduling information of Graph Engine. This parameter is optional.</td>
</tr>
<tr>
<td>Device System Sampling</td>
<td><strong>DDR</strong></td>
</tr>
<tr>
<td></td>
<td>Sets the DDR sampling frequency (Hz). This parameter is optional and is selected by default.</td>
</tr>
<tr>
<td></td>
<td>You can change the sampling frequency. The default value is <strong>50 Hz</strong>.</td>
</tr>
<tr>
<td>Host System Sampling</td>
<td><strong>CPU</strong></td>
</tr>
<tr>
<td></td>
<td>Samples the host CPU usage. This parameter is optional and is selected by default.</td>
</tr>
<tr>
<td></td>
<td><strong>Memory</strong></td>
</tr>
<tr>
<td></td>
<td>Samples the host memory usage. This parameter is optional.</td>
</tr>
<tr>
<td></td>
<td><strong>Disk</strong></td>
</tr>
<tr>
<td></td>
<td>Samples the host disk usage. This parameter is optional.</td>
</tr>
<tr>
<td></td>
<td><strong>Network</strong></td>
</tr>
<tr>
<td></td>
<td>Samples the host network usage. This parameter is optional.</td>
</tr>
</tbody>
</table>

**Step 6** After the preceding configurations are complete, click **Start** in the lower right corner of the window to start Profiling.
The profiling results will be automatically displayed at the bottom of the MindStudio window after the execution is complete.

---End

10.5 Profiling Project Management

Project Explorer displays your projects and the generated profiling reports.

Creating a Profiling Project

Step 1  Open a built project.

Step 2  Choose Ascend > System Profiler > New Project on the menu bar or click in the upper left corner of the profiling report page.

Step 3  Finish your project creation. For details, see Step 3 to Step 6.

---End

Opening a Profiling Report Not Displayed

Step 1  Open a built project.

Step 2  Choose Ascend > System Profiler > Open on the menu bar or click in the upper left corner of the profiling report page to open the profiling report that is not displayed.

---End

Opening a Report

NOTE

After Profiling is successfully executed, the profiling report is automatically opened and the profiling results are displayed as well.

Step 1  Right-click the corresponding report file in the navigation tree on the left of the report page. A shortcut menu is displayed.

Step 2  Click Open Report.

---End

Save as

Step 1  Right-click the corresponding report file in the navigation tree on the left of the report page. A shortcut menu is displayed.

Step 2  Click Save As.

Step 3  Select a destination path and click Save.

---End
Deleting a Report

**Step 1** Right-click the corresponding report file in the navigation tree on the left of the report page. A shortcut menu is displayed.

**Step 2** Click **Delete Report**.

**Step 3** Click **Yes** in the displayed dialog box.

----End

Refreshing the Navigation Bar

Click ( ) in the upper left corner of the report page. This operation will refresh the navigation bar and all expanded projects will be collapsed.

10.6 Profile Data Display

After Profiling is successfully executed, the profiling results are automatically displayed. See **Figure 10-7**.

**Figure 10-7** Profiling result view

Profiling supports two AI Core profiling methods: task-based and sample-based. Therefore, the results displayed on each tab page of Profiling are different.

**Table 10-3** AI Core profiling results in different views

<table>
<thead>
<tr>
<th>View</th>
<th>Task-Based</th>
<th>Sample-Based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary View (AI Core Utilization)</td>
<td>Not available</td>
<td>Available</td>
</tr>
<tr>
<td>Timeline View</td>
<td>Available</td>
<td>Available</td>
</tr>
</tbody>
</table>
10.6.1 Merge Reports

After the `msprof` command is executed to collect profile data, you can use the Merge Reports function to merge the collected data and display it on MindStudio. To merge reports, perform the following steps:

**Step 1** On the menu bar, choose **Ascend > System Profiler > Merge Reports**. The **Merge Reports** configuration window is displayed.

**Step 2** Configure the input directory, output directory, and output JSON file name. See **Figure 10-8**.

![Figure 10-8 Configuring Merge Reports](image)

**Table 10-4 Merge Reports parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input directory</td>
<td>Data source path. This is the output path specified by the <code>msprof</code> command.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>Output directory</td>
<td>Output path.</td>
</tr>
<tr>
<td>Output Json name</td>
<td>Name of an output JSON file.</td>
</tr>
</tbody>
</table>

**NOTE**

- For details about how to use the `msprof` command, see the CANN Development Auxiliary Tool Guide.
- If the `msprof` command is run in a multi-device environment, how many devices there are, how many JOBxxx files will be generated in the data output path.
- If the `msprof` command is run for multiple times in the same path, multiple JOBxxx files are generated in the data output path. Currently, only the upper-layer file of the specified JOBxxx file can be generated as a JSON file for display in Merge Reports. Therefore, ensure that the `msprof` command is run only once.

---End

## 10.6.2 TimeLine View

### 10.6.2.1 Timeline Color Configuration

On the MindStudio menu bar, choose Ascend > System Profiler > Timeline Color, or click ![Project Explorer](image) in the upper left corner of Project Explorer. Figure 10-9 shows the timeline color configuration window.

**Figure 10-9** Timeline color configuration

Timeline color is customizable based on the API execution time.

The default configuration is shown in Figure 10-9. The API execution time greater than or equal to 5.00% and less than 10.00% is highlighted in yellow while that greater than or equal to 10.00% is highlighted in red.

**NOTE**

To apply your new color settings, close and open the report again.
10.6.2.2 TimeLine View

TimeLine View consists of the navigation pane on the left, graphical pane on the right, and data pane at the bottom. The navigation pane displays the name of each timeline and the dependencies between timelines. The graphical pane corresponds to the navigation pane and graphically displays the timelines. The data pane displays the data collected by Profiling in tables, including Event View, Statistics, and AI Core Metrics.

NOTE

- Start Time and End Time indicate the time range of the collected performance data.
- Current Time indicates the start time of the time segment where the cursor is located.
- Move the cursor to a sampling point to view the detailed analyses.
- To see the sequence of a timeline in the Event View window, right-click a timeline label in the left navigation pane and choose Show in Event View from the shortcut menu.
- In TimeLine View, you can check APIs and operations used.
- Multiple OS Runtime APIs within a thread executed simultaneously are displayed in different lines.
- Multiple AI Core tasks within a stream executed simultaneously are displayed in different lines.
- To zoom in or out the timeline of interest, select a time point, and hold down Ctrl as scrolling the mouse wheel upward or downward. You can also click or in the upper right corner of the window. Click in the upper right corner of the window to restore the original view.
- When selecting a time point in a timeline, you can drag the cursor leftward or rightward to select the duration. The time segment to the left of Current Time(us) is displayed as a negative value while the time segment to the right of Current Time(us) is displayed as a positive value.

After profiling is complete, the running status of the entire inference process is displayed in a sequence diagram. The overall running status is displayed based on the called processes, such as OS Runtime, ACL API, Runtime API, GE, AI Core task, and System task.

- OS Runtime displays the time sequence information when each thread calls OS Runtime APIs.
- ACL API displays the time taken by models, operators, and Runtime APIs. If a certain metric is not collected, the profiling result will be not available.
- Runtime API displays the time sequence information about Runtime APIs called by each thread.
- GE displays the time taken by the model to input, infer, and output data.
- AI Core task displays the the time sequence information about about AI Core tasks within each stream.
- System task displays the time sequence information about system tasks within each stream.

10.6.2.3 Event View

To see the sequence of a timeline in the Event View window in the data pane at the bottom, right-click a timeline label in the left navigation pane of the TimeLine
View window and choose Show in Event View from the shortcut menu as shown in Figure 10-10.

Figure 10-10 Show in Event View

You can view:

- Process > Thread > Os Runtime
- Process > Thread > ACL API (model/runtime)
- Process > Thread > Runtime API
- NPU 0> Stream > AI Core task
### ACL API (model/runtime)

**Figure 10-11** ACL API (model/runtime)

![Figure showing ACL API (model/runtime)]

**Table 10-5** Parameter description (model/runtime)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>ACL API ID</td>
</tr>
<tr>
<td>Name</td>
<td>ACL API name</td>
</tr>
<tr>
<td>Start Time(us)</td>
<td>Time (μs) when the ACL API starts to run</td>
</tr>
<tr>
<td>End Time(us)</td>
<td>Time (μs) when the ACL API stops running</td>
</tr>
<tr>
<td>Duration(us)</td>
<td>ACL API running duration (μs)</td>
</tr>
<tr>
<td>Process ID</td>
<td>Process ID of the ACL API</td>
</tr>
<tr>
<td>Thread ID</td>
<td>Thread ID of the ACL API</td>
</tr>
</tbody>
</table>

### Runtime API

**Figure 10-12** Runtime API

![Figure showing Runtime API]
### Table 10-6 Parameter description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Runtime API ID</td>
</tr>
<tr>
<td>Name</td>
<td>Runtime API name</td>
</tr>
<tr>
<td>Start Time(us)</td>
<td>Time (μs) when the Runtime API starts to run</td>
</tr>
<tr>
<td>End Time(us)</td>
<td>Time (μs) when the Runtime API stops running</td>
</tr>
<tr>
<td>Duration(us)</td>
<td>Runtime API running duration (μs)</td>
</tr>
<tr>
<td>Process ID</td>
<td>Process ID of the Runtime API</td>
</tr>
<tr>
<td>Thread ID</td>
<td>Thread ID of the Runtime API</td>
</tr>
</tbody>
</table>

### Os Runtime

#### Figure 10-13 Os Runtime

![Os Runtime](image)

### Table 10-7 Parameter description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>OS Runtime ID</td>
</tr>
<tr>
<td>Name</td>
<td>OS Runtime name</td>
</tr>
<tr>
<td>Start Time(us)</td>
<td>Time (μs) when the OS Runtime starts to run</td>
</tr>
<tr>
<td>End Time(us)</td>
<td>Time (μs) when the OS Runtime stop running</td>
</tr>
<tr>
<td>Duration(us)</td>
<td>OS Runtime running duration (μs)</td>
</tr>
<tr>
<td>Process ID</td>
<td>Process ID of OS Runtime</td>
</tr>
<tr>
<td>Thread ID</td>
<td>Thread ID of OS Runtime</td>
</tr>
</tbody>
</table>
AI Core task

**Figure 10-14 AI Core task**

![Figure 10-14 AI Core task]

**Table 10-8 Parameter description**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>AI Core task ID</td>
</tr>
<tr>
<td>Device ID</td>
<td>ID of the device where the AI Core task is running</td>
</tr>
<tr>
<td>Op Name</td>
<td>Operator name of the AI Core task</td>
</tr>
<tr>
<td>Start Time(us)</td>
<td>Time (μs) when the AI Core task starts to run</td>
</tr>
<tr>
<td>End Time(us)</td>
<td>Time (μs) when the AI Core task stops running</td>
</tr>
<tr>
<td>Duration(us)</td>
<td>AI Core task duration (μs)</td>
</tr>
<tr>
<td>Status</td>
<td>Running status of the AI Core task</td>
</tr>
<tr>
<td>Task Type</td>
<td>AI Core task type</td>
</tr>
<tr>
<td>Stream ID</td>
<td>Stream ID of the AI Core task</td>
</tr>
<tr>
<td>Task ID</td>
<td>AI Core task ID</td>
</tr>
</tbody>
</table>

10.6.2.4 Statistics View

Click **Statistics** in the data pane at the bottom and select a data type from the drop-down list in the upper left corner to view the statistics of **Figure 10-15**, **Figure 10-16**, **Figure 10-17**, **Figure 10-18**, and **Figure 10-19**.

**Figure 10-15 ACL API**

![Figure 10-15 ACL API]
NOTE

The parameters in Table 10-9 and Table 10-10 are arranged in the sequence that they appear.

Table 10-9 Parameter description (ACL API, OS Runtime API, Runtime API, and OPs)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process ID</td>
<td>Process ID of the API</td>
</tr>
<tr>
<td>Parameter</td>
<td>Parameter description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Thread ID</td>
<td>Thread ID of the API</td>
</tr>
<tr>
<td>Type (ACL API only)</td>
<td>Type of the ACL API</td>
</tr>
<tr>
<td>Name</td>
<td>API name</td>
</tr>
<tr>
<td>Time(%)</td>
<td>Percentage of time taken by the API</td>
</tr>
<tr>
<td>Time(us)</td>
<td>Time (μs) taken by the API</td>
</tr>
<tr>
<td></td>
<td>You can rearrange the information in ascending or descending order by clicking the triangle icon next to the field name.</td>
</tr>
<tr>
<td>Count</td>
<td>API calls</td>
</tr>
<tr>
<td>Avg(us)</td>
<td>Average duration (μs) of a single API call</td>
</tr>
<tr>
<td>Max(us)</td>
<td>Maximum duration (μs) of an API call</td>
</tr>
<tr>
<td>Min(us)</td>
<td>Minimum duration (μs) of an API call</td>
</tr>
<tr>
<td>Calls (Runtime API only)</td>
<td>Runtime API calls</td>
</tr>
<tr>
<td>Op Name (OPs only)</td>
<td>OPs operator name</td>
</tr>
</tbody>
</table>

Table 10-10 Parameter description (Op Info)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Name</td>
<td>Model name</td>
</tr>
<tr>
<td>Task ID</td>
<td>Task ID</td>
</tr>
<tr>
<td>Stream ID</td>
<td>Stream ID</td>
</tr>
<tr>
<td>Op Name</td>
<td>Operator name</td>
</tr>
<tr>
<td>Op Type</td>
<td>Operator type</td>
</tr>
<tr>
<td>Task Start Time</td>
<td>Time when the task starts to run</td>
</tr>
<tr>
<td>Task Duration(us)</td>
<td>Task running duration (us)</td>
</tr>
<tr>
<td>Task Wait Time(us)</td>
<td>Interval between tasks (μs)</td>
</tr>
<tr>
<td>Block Dim</td>
<td>Index of the Core that executes the task</td>
</tr>
<tr>
<td>Input Shapes</td>
<td>Input shapes</td>
</tr>
<tr>
<td>Input Data Types</td>
<td>Input data types</td>
</tr>
<tr>
<td>Input Formats</td>
<td>Input formats</td>
</tr>
</tbody>
</table>
### 10.6.2.5 AI Core Metrics View

Click **AI Core Metrics** in the data pane at the bottom to view AI Core metrics.

#### Figure 10-20 AI Core Metrics

![Figure 10-20 AI Core Metrics](image)

#### Table 10-11 AI Core Metrics

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task ID</td>
<td>Task ID</td>
</tr>
<tr>
<td>Stream ID</td>
<td>Stream ID</td>
</tr>
<tr>
<td>Op Name</td>
<td>Operator name</td>
</tr>
<tr>
<td>OP Type</td>
<td>Operator type</td>
</tr>
<tr>
<td>Task Start Time</td>
<td>Task start time</td>
</tr>
<tr>
<td>Task Duration(us)</td>
<td>Task running duration (μs)</td>
</tr>
<tr>
<td>Task Wait Time(us)</td>
<td>Task waiting time (μs)</td>
</tr>
<tr>
<td>Aicore Time(us)</td>
<td>AI Core running duration (μs)</td>
</tr>
<tr>
<td>Total Cycles</td>
<td>Number of cycles taken to execute all task instructions</td>
</tr>
<tr>
<td>Vec Time(us)</td>
<td>Time (μs) taken to execute Vector instructions</td>
</tr>
<tr>
<td>Vec Ratio</td>
<td>Percentage of cycles taken to execute Vector instructions</td>
</tr>
<tr>
<td>Mac Time(us)</td>
<td>Time (μs) taken to execute Cube instructions</td>
</tr>
</tbody>
</table>
### Field & Description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mac Ratio</td>
<td>Percentage of cycles taken to execute Cube instructions</td>
</tr>
<tr>
<td>Scalar Time(us)</td>
<td>Time (μs) taken to execute Scalar instructions</td>
</tr>
<tr>
<td>Scalar Ratio</td>
<td>Percentage of cycles taken to execute Scalar instructions</td>
</tr>
<tr>
<td>Mte1 Time(us)</td>
<td>Time (μs) taken to execute MTE1 instructions (L1-to-L0A/L0B movement)</td>
</tr>
<tr>
<td>Mte1 Ratio</td>
<td>Percentage of cycles taken to execute MTE1 instructions (L1-to-L0A/L0B movement)</td>
</tr>
<tr>
<td>Mte2 Time(us)</td>
<td>Time (μs) taken to execute MTE2 instructions (DDR-to-AI Core movement)</td>
</tr>
<tr>
<td>Mte2 Ratio</td>
<td>Percentage of cycles taken to execute MTE2 instructions (DDR-to-AI Core movement)</td>
</tr>
<tr>
<td>Mte3 Time(us)</td>
<td>Time (μs) taken to execute MTE3 instructions (AI Core-to-DDR movement)</td>
</tr>
<tr>
<td>Mte3 Ratio</td>
<td>Percentage of cycles taken to execute MTE3 instructions (AI Core-to-DDR movement)</td>
</tr>
<tr>
<td>Icache Miss Rate</td>
<td>I-Cache miss rate. The smaller the value, the higher the performance.</td>
</tr>
<tr>
<td>Memory Bound</td>
<td>Metric used to identify whether a memory bottleneck exists during the operator computation performed by the AI Core. The value is calculated as follows: ( \frac{\text{Mte2 Ratio}}{\text{max(Mac Ratio, Vec Ratio)}} ). If the value is less than 1, no memory bottleneck exists. If the value is greater than 1, a memory bottleneck exists. The greater the value is, the severer the bottleneck is.</td>
</tr>
</tbody>
</table>

### 10.6.3 Analysis Summary

In Analysis Summary, select a device ID to view the detailed analyses on the device.
**Figure 10-21** Analysis Summary view

### Profiling Info

**Table 10-12** Collecting information

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result Size</td>
<td>Size of a result file</td>
</tr>
<tr>
<td>Profiling Elapsed Time</td>
<td>Duration of information collection</td>
</tr>
</tbody>
</table>

### Host System Info

**Table 10-13** Parameters

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Operating System</td>
<td>Host OS information</td>
</tr>
<tr>
<td>Host Computer Name</td>
<td>Host computer name</td>
</tr>
</tbody>
</table>

### Host CPU Info

**Table 10-14** Parameters

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU ID</td>
<td>CPU ID</td>
</tr>
</tbody>
</table>
### Device Info

**Table 10-15 Parameters**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI CPU Number</td>
<td>Number of AI CPUs</td>
</tr>
<tr>
<td>AI Core Number</td>
<td>Number of AI Cores</td>
</tr>
<tr>
<td>Control CPU Number</td>
<td>Number of Control CPUs</td>
</tr>
<tr>
<td>Control CPU Type</td>
<td>Ctrl CPU type</td>
</tr>
<tr>
<td>TS CPU Number</td>
<td>Number of TS CPUs</td>
</tr>
</tbody>
</table>

### DDR

**Table 10-16 Parameters**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric</td>
<td>Bandwidth (MB/s)</td>
</tr>
<tr>
<td>Read(MB/s)</td>
<td>Read bandwidth (MB/s)</td>
</tr>
<tr>
<td>Write(MB/s)</td>
<td>Write bandwidth (MB/s)</td>
</tr>
</tbody>
</table>

### AI Core Utilization

The AI Core utilization is displayed in a line chart (displayed only when the sample-based mode is selected).
11 Al Core Error Analyzer

11.1 Overview
When you encounter Al Core errors in the inference or training process, you can use the Al Core Error Analyzer to collect necessary information for quickly locating the Al Core errors.

NOTE
- This tool cannot be deployed in a container.
- This tool can be used only for local analysis. That is, the environment where this tool is deployed must be the same as the environment where logs are stored.
- In an inference scenario, the root user is used to run the Al Core Error Analyzer by default. If you want to run the tool as a common user, modify the directory attributes as the root user. For details, see “Appendixes > msnpureport Tool Instructions” in the Log Reference (Inference). Then assign the common user the permission to use the msnpureport tool.
11.2 Troubleshooting Workflow

Inference Scenario

Figure 11-1 Troubleshooting workflow in inference scenario

Step 1  Encounter an AI Core error in the inference process.

Step 2  Configure the AI Core Error Analyzer parameters including \texttt{op\_debug\_level} and \texttt{debug\_dir}, for locating the code line numbers of error operators.

Step 3  Convert the model again to generate an instruction mapping file.

Step 4  Analyze the error using the AI Core Error Analyzer.

-----End
Training Scenario

Figure 11-2 Troubleshooting workflow in training scenario

1. **Step 1** Encounter an AI Core error in the model training process.
2. **Step 2** Configure the AI Core Error Analyzer parameters, including:
   - `op_debug_level` for locating the code line numbers of error operators.
   - `enable_exception_dump` for dumping error operators.
3. **Step 3** Run the model training script again to generate an instruction mapping file and a dump file of each error operator.
4. **Step 4** Analyze the error using the AI Core Error Analyzer.

---End

11.3 Preparing Data

Training Scenario

1. If an AI Core error occurs during training, perform the following steps to configure the `op_debug_level` and `enable_exception_dump` parameters:
   - In `Estimator` mode, set `op_debug_level` and `enable_exception_dump` as follows.
from npu_bridge.estimator.npu.npu_config import NPURunConfig
from npu_bridge.estimator.npu.npu_config import DumpConfig

session_config=tf.ConfigProto()

config = NPURunConfig(
    op_debug_level = 2,  //Enable operator debug.
    session_config=session_config,
    enable_exception_dump=1  //Dump the inputs and outputs of the error operator to the script execution directory. Dynamic-shape operators cannot be dumped.
)

- In `sess.run` mode, set `op_debug_level` and `enable_exception_dump` as follows.

import tensorflow as tf
from npu_bridge.estimator import npu_ops
from tensorflow.core.protobuf.rewriter_config_pb2 import RewriterConfig

config = tf.ConfigProto()
custom_op = config.graph_options.rewrite_options.custom_optimizers.add()
custom_op.name = "NpuOptimizer"
custom_op.parameter_map["use_off_line"].b = True
custom_op.parameter_map["enable_exception_dump"].i = 1  //Dump the inputs and outputs of the error operator to the script execution directory. Dynamic-shape operators cannot be dumped.
custom_op.parameter_map["op_debug_level"].i = 2  //Enable operator debug.
config.graph_options.rewrite_options.remapping = RewriterConfig.OFF  # Disable remapping.

with tf.Session(config=config) as sess:
    print(sess.run(cost))

2. After the retrain is complete, an instruction mapping file and an error operator dump file are generated in the training execution directory.

**Inference Scenario**

Include the `op_debug_level` and `debug_dir` options in the ATC command, as follows:

**Step 1** Open the Model Converter dialog box.

**Step 2** Expand Advanced Options.

**Step 3** Turn the Additional Arguments switch on, and add `op_debug_level` (for example, `--op_debug_level=2`) and `debug_dir` (debug output path, which is under the project directory).
**Value Range of op_debug_level**

**Table 11-1 Value range of op_debug_level**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disables operator debug.</td>
</tr>
<tr>
<td>1</td>
<td>Enables operator debug and generates a TBE instruction mapping file. In this case, an operator CCE file (<em>.cce) and a Python-CCE mapping file (</em>.loc.json), and operator .o and .json files are generated in the <strong>kernel_meta</strong> folder in the training script execution directory. You can locate the AI Core error by using the line numbers in the CCE code and TBE code of the error operator.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>2</td>
<td>Enables operator debug and generates a TBE instruction mapping file. In this case, an operator CCE file (<em>.cce) and a Python-CCE mapping file (</em>.loc.json), and operator .o and .json files are generated in the kernel_meta folder in the training script execution directory, and the build optimization is disabled by enabling the CCE compiler -O0-g. You can locate the AI Core error by using the line numbers in the CCE code and TBE code of the error operator.</td>
</tr>
</tbody>
</table>

11.4 Analyzing AI Core Errors

Starting AI Core Error Analyzer

**Step 1** Choose Ascend > AI Core Error Analyzer.

**Step 2** Set the AI Core Error Analyzer parameters.

**Compile Path**: In the inference scenario, set this parameter to the model conversion debug output path (see **Step 3** in section "Preparing Data" for details). In the training scenario, set this parameter to the script execution path. It stores the outputs and .pbtxt files generated after operator build. Generally, it is the parent path of the kernel_meta file (for example, ~/model_convert).

**Figure 11-4 Parameter settings**

![Compile Path: csendProjects/MyApp/complie_path_infer](image)

**Step 3** Click **Analyze**. The analysis result will be output to the **Output** window.

**NOTE**

If the error message "TOOLCHAIN_HOME is empty" is displayed, configure the TOOLCHAIN_HOME environment variable by following the $HOME/ascend-toolkit/set_env.sh script during the toolkit installation in the CANN 5.0.1 Software Installation Guide (ascend-deployer).

----End

Viewing Analysis Result

The outputs of the AI Core Error Analyzer are generated to the IDE directory formatted as info_timeline (for example, info_20200903114406). The generated result file varies according to the actual situation. The following example is for reference only.

- Inference example:
- Ascend EP:
  ```
  aicore_error
  ├── aicerr_out
  │    └── info_xxxxx
  │         ├── aicerror_xxxxx  //Directory of AI Core Error Analyzer outputs
  │         │    ├── info.txt  //AI Core Error Analyzer report
  │         │    └── te_transdata_xxxxx.o
  │         │         └── te_transdata_xxxxx.o.txt  //Decompiration file
  │         └── collection  //Directory of error operator files
  │             └── compil
  │                 └── kernel_meta
  │                     └── CCE file
  │                 └── JSON file
  │                     └── loc.json file
  │                 └── o file
  │                     └── ge_proto_xxxxx_Build.txt
  │                 └── dump  // Directory of dump files
  │                     └── log  // Directory of host logs
  │                 └── xxxx  // Timestamp
  │                     └── error.log  //ERROR-level log file
  │                         └── README.txt
  │   └── npu_report
  │       └── xxxx  // Timestamp directory
  │           └── hisi_logs  // Black Box logs
  │                 └── message  // Device OS logs
  │                         └── slog
  │                                 └── stackcore
  ```

- Ascend RC:
  ```
  aicore_error
  ├── aicerr_out
  │    └── info_xxxxx
  │         ├── aicerror_xxxxx  //Directory of AI Core Error Analyzer outputs
  │         │    ├── info.txt  // AI Core Error Analyzer report
  │         │    └── te_transdata_xxxxx.o
  │         │         └── te_transdata_xxxxx.o.txt  // Decompliation file
  │         └── collection  // Directory of error operator files
  │             └── compil
  │                 └── kernel_meta
  │                     └── CCE file
  │                 └── JSON file
  │                     └── loc.json file
  │                 └── o file
  │                     └── ge_proto_xxxxx_BUILD.txt
  │                 └── dump  // Directory of dump files
  │                     └── log  // Directory of host logs
  │                 └── xxxx  // Timestamp
  │                     └── error.log  // ERROR-level log file
  │                         └── README.txt
  │   └── npu_report
  │       └── xxxx  // Timestamp directory
  │           └── hisi_logs  // Black Box logs
  │                 └── message  // Device OS logs
  │                         └── slog
  │                                 └── stackcore
  ```

- Training example:
  ```
  aicore_error
  ├── aicerr_out
  │    └── info_xxxxx
  │         └── collection  // Directory of error operator files
  │             └── log  // Directory of host logs
  │                     └── xxxx  // Process ID
  │             └── xxxx  // Timestamp
  │                 └── hisi_logs  // Black Box logs
  │                     └── slog
  │   └── npu_report
  │       └── xxxx  // Timestamp directory
  │           └── hisi_logs  // Black Box logs
  │                 └── message  // Device OS logs
  │                         └── slog
  │                                 └── stackcore
  ```
11.5 Locating AI Core Errors

You can analyze and locate AI Core errors based on the logs in the Run window.

***************Root cause conclusion***************
# Gives the root cause if the error matches known error patterns.

***********************1. Basic information***********************
# Gives the basic information about the device occurred with the AI Core error.
#kernel name: operator kernel name
#op address: address of the operator code in the DDR
#args address: address of the operator arguments in the DDR
error time   : 2020-08-26-11:24:07
device id    : 0
core id      : 0
task id      : 60
stream id    : 517
node name    : trans_TransData_167
kernel name  : te_transdata_16b6e15e2a5cc7f70_33e5fb7ae8478ddb
op address   : 0x101000120000
args address : 0X101000053000

***********************2. AICERROR code***********************
# Gives the AI Core error code and description.
code  : 0x10
CCU_ERR_INFO: 0xb166486200070074

***********************3. Instructions************************
# Gives the error instructions.
start   pc   : 0x101000120000
current pc   : 0x1010001201e0
Error occured most likely at line: 1d0

/{IDE path}/aicerror_xxxx/te_transdata_16b6e15e2a5cc7f70_33e5fb7ae8478ddb.o.txt:1d0
/{IDE path}/collection/compile/kernel_meta/te_transdata_16b6e15e2a5cc7f70_33e5fb7ae8478ddb.cce:32    //
CCE code line number of the error operator
/{Python script path}/nz_2_nd.py:4486 //Python code line number of the error operator
related instructions (error occurred before the mark *):

1bc: <not available>
1c0: <not available>
1c4: <not available>
1c8: <not available>
1cc: <not available>
1d0: <not available>
1d4: <not available>
1d8: <not available>
1dc: <not available>
  1e0: <not available>

For complete instructions, please see  /{IDE path}/aicerror_xxxx/
te_transdata_16b6e15e2a5cc7f70_33e5fb7ae8478ddb.o.txt

***********************4. Input and output of node***********************
# Gives the input and output information.
# The input and output addresses are parsed from the IMAS log of GE and the size is parsed from the build graph.
# In the case of memory zero copy, the new address (new addr) can also be parsed from the log.
# If the address is not within the range of the RTS allocation log, an overflow flag is added.
# If the device memory data is collected, NaN and INF verification will also be performed. The collected data is accurate only when the device is suspended.
# If the detected input count and output count are inconsistent with those defined in the kernel function, a WARNING is returned. There is a high probability of misplacement between the
arguments provided by GE and those processed by the operator.
input[0]  addr: 0x100801126600  size: 32288
output[0]  addr: 0x100801157c00  size: 2048

***********************5. Op in graph*************************
# Gives information of the error operator.
# The operator information is taken from the build graph for viewing convenience.

***********************6. Dump info***************************
# Dump file of the error operator.
# This information is available only in the training scenario.
12 Operation Guidelines

12.1 SSH Connection Management

12.1.1 Restrictions

For security considerations, you are advised to perform the following security hardening operations on the server connected in SSH mode:

- The SHA-256 or SHA-512 algorithm is recommended for encrypting the passwords of Linux system users.
- In the configuration file of the SSH server, harden the following configuration options:
  - **MACs**: selects the message digest algorithm (or algorithms separated by commas) for data verification in SSH-2. The `hmac-sha2-512` and `hmac-sha2-256` algorithms are recommended.
  - **Ciphers**: selects the encryption algorithm (or algorithms separated by commas) of SSH-2. The `aes128-ctr`, `aes192-ctr`, and `aes256-ctr` algorithms are recommended.
  - **PermitRootLogin**: allows root login. You are advised not to allow root login.
12.1.2 Overview

Portals

You can open the SSH connection configuration window in any of the following ways:

- On the menu bar, choose Ascend > SSH Configurations.
- On the menu bar, choose File > Settings... > Tools > SSH Configurations.
- In the welcome window, choose Configure > Settings.

![Figure 12-1 SSH configuration window](image)

**NOTE**

- For the first two methods (with an active project), **For current project** and **Visible only for this project** check boxes are displayed in the upper part of the configuration window.
- If **Visible only for this project** is selected, the configured SSH connection is visible only to this project.
- If not selected, the configured SSH connection is visible to all projects.
- The aliases of the SSH connections are displayed on the navigation pane on the left of the configuration window. If no alias is set, each SSH connection is displayed as `<username>@<host>:<port>`. You can view the detailed parameter settings on the right by selecting and clicking a connection.
Parameters and Icons

Table 12-1 Parameters and icons

<table>
<thead>
<tr>
<th>Parameter/Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Destination IP address of a connection.</td>
</tr>
<tr>
<td>Port</td>
<td>Port number of the destination IP address of the connection.</td>
</tr>
<tr>
<td>User name</td>
<td>Username for logging in to the target host.</td>
</tr>
<tr>
<td>Authentication type</td>
<td>Authentication method. Two methods are available:</td>
</tr>
<tr>
<td></td>
<td>● <strong>PASSWORD</strong>: Enter a password for authentication.</td>
</tr>
<tr>
<td></td>
<td>● <strong>KEY_PAIR</strong>: Use an SSH key for authentication.</td>
</tr>
<tr>
<td></td>
<td>Prepare the key in advance by yourself.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong></td>
</tr>
<tr>
<td></td>
<td>In the EulerOS 2.8 AArch architecture or Kylin OS V10 SP1 AArch architecture, when <strong>KEY_PAIR</strong> is used for identity authentication, you need to run the <code>ssh-keygen -m PEM</code> or <code>ssh-keygen -t rsa -m PEM</code> command to apply for an SSH key.</td>
</tr>
<tr>
<td>Password</td>
<td>Available when <strong>Authentication type</strong> is set to <strong>PASSWORD</strong>. Enter the correct password. You can select <strong>Save password</strong> to save the password.</td>
</tr>
<tr>
<td>Private key file</td>
<td>Available when <strong>Authentication type</strong> is set to <strong>KEY_PAIR</strong>. Select a local private key file.</td>
</tr>
<tr>
<td>Passphrase</td>
<td>Available when <strong>Authentication type</strong> is set to <strong>KEY_PAIR</strong>. Enter the correct passphrase. You can select <strong>Save passphrase</strong> to save the passphrase.</td>
</tr>
<tr>
<td>Test Connection</td>
<td>Checks whether the connection is successful.</td>
</tr>
<tr>
<td>SSH tunnel</td>
<td>Mapping between the local port and the remote port. The format is <code>&lt;local port&gt;:&lt;remote port&gt;</code>.</td>
</tr>
<tr>
<td>Remote work path</td>
<td>Remote work path.</td>
</tr>
<tr>
<td>Adds</td>
<td>Adds a new SSH connection.</td>
</tr>
<tr>
<td>Deletes</td>
<td>Deletes an SSH connection. Select an SSH connection to be deleted and click this icon.</td>
</tr>
<tr>
<td>Save</td>
<td>Save as icon. Select an SSH connection to be saved and click this icon.</td>
</tr>
<tr>
<td>Modifies</td>
<td>Modifies the alias of an SSH connection. Select the SSH connection to be modified and click this icon.</td>
</tr>
<tr>
<td>OK</td>
<td>Saves the SSH configuration. Click this icon to save and apply the current SSH connection configuration, and close the configuration window.</td>
</tr>
</tbody>
</table>
### 12.1.3 Adding a New SSH Connection

**Step 1** Open the SSH connection configuration window by referring to Portals.

**Step 2** Click + on the navigation pane.

**Step 3** Set parameters by referring to Parameters and Icons and click OK or Apply.

--- End

### 12.1.4 Deleting an SSH Connection

**Step 1** Open the SSH connection configuration window by referring to Portals.

**Step 2** Select the SSH connection to be deleted, and click − on the navigation pane.

**Step 3** Click OK or Apply.

--- End

### 12.1.5 Modifying an SSH Connection

**Step 1** Open the SSH connection configuration window by referring to Portals.

**Step 2** Select the SSH connection to be modified, and click ⤵️ on the navigation pane to modify the connection name.

**Step 3** Modify parameters by referring to Parameters and Icons and click OK or Apply.

--- End

### 12.1.6 Copying an SSH Connection

**Step 1** Open the SSH connection configuration window by referring to Portals.

**Step 2** Select the SSH connection to be copied, and click ⌃️ on the navigation pane.

**Step 3** Enter a new SSH name, configure the parameters by referring to Parameters and Icons, and then click OK or Apply.

--- End

### 12.1.7 Encrypting the SSH Password

**Step 1** Open the SSH connection configuration window by referring to Portals.
Step 2  Enter the set password, select Save password, and click Apply. The Choose password for new keyring window is displayed.

Figure 12-2 Choose password for new keyring

- **Password**: Enter the password to be encrypted.
- **Confirm**: Enter the password again.

Step 3  Determine whether to save the password encrypted.
- Cancel the encryption: Click Cancel shown in Figure 12-2.
- Encrypted:
  a. Set Password and Confirm in Figure 12-2.
  b. Click Continue to encrypt the password.
- Unencrypted:
  a. Ignore Password and Confirm, and click Continue in Figure 12-2. The following dialog box is displayed.

Figure 12-3 Store passwords unencrypted

b. Click Continue. The password is saved unencrypted.

Step 4  Click OK.

----End

12.1.8 Changing the SSH Password Saving Mode

Step 1  Open the SSH connection configuration window by referring to Portals.

Step 2  You can open the Save passwords window in either of the following ways:
- On the navigation pane, choose Appearance & Behavior > System Settings > Passwords.
- Enter Passwords in the search box.
Step 3  Select a password saving mode.

- In native Keychain
- In KeePass
- Do not save, forget passwords after restart

**NOTE**
- For details about the password saving modes, see "Passwords" in IntelliJ IDEA overview.
- If the In native Keychain mode cannot be selected due to the installation failure of the GNOME Keyring tool, try the In KeePass mode.

Step 4  Click OK or Apply.

---End

12.2 Project Management

12.2.1 Opening/Closing/Removing a Project

Opening a Project

Open an existing project in either of the following ways:

- Choose File > Open... from the main menu and select a project. The Open File or Project dialog box is displayed, as shown in Figure 12-5.
Select a project and click **OK**.

- In **Welcome to MindStudio** window, click **Open or Import**. In the displayed dialog box, select the path of the project. The **Open File or Project** dialog box is displayed, as shown in **Figure 12-5**. Select a project and click **OK**.

If the dialog box shown in **Figure 12-6** is displayed, the ADK version of the existing project does not match that of MindStudio. Click **OK**. In the displayed dialog box, select the current **ADK Version** and click **OK**.

**Figure 12-6** Project opening error message

---

**Opening a Recent Project**

A recently closed project can be opened in either of the following ways:

1. Choose **File > Open Recent** from the main menu, and choose the project to be opened, for example the project shown in **Figure 12-7**.
Figure 12-7 Open Project

The **Open Project** dialog box shown in **Figure 12-8** is displayed.

Figure 12-8 Open Project dialog box

- Clicking **This Window** opens the project in the current window.
- Clicking **New Window** opens the project in a new window.

2. Choose **File > Open Recent > Manage Projects...** from the menu bar. Select the project to be opened. The project is opened in a new window.

**Closing a Project**

Choose **File > Close Project** to close the currently active project. If the last project is closed, the welcome window shown in **Figure 12-9** is displayed. You can open the closed project, open a different project, or create a project (by right-clicking the blank area and choosing **Create new project** from the shortcut menu).

Figure 12-9 Welcome to MindStudio

**Removing a Project**

You can remove a closed project in either of the following ways:
In the window shown in Figure 12-9, select the project to be removed and click on the right or right-click the project and choose Remove Selected from Welcome Screen from the shortcut menu.

Choose File > Open Recent Projects >Manage Projects.... Select the project to be removed and click on the right.

Figure 12-10 Removing a project

---

12.2.2 Creating/Deleting a File

Creating a File

In the Projects window, right-click a project and choose New from the shortcut menu or choose File > New. In the displayed submenu, select the type of the file to be created, or click File and enter the file name (without the file name extension) in the displayed dialog box. For example, after you click File, the dialog box shown in Figure 12-11 is displayed.

Figure 12-11 Entering a file name

Click OK. Enter the file name in the dialog box shown in Figure 12-11. If the file type has not been specified, the dialog box shown in Figure 12-12 is displayed.
Selecting a file type

The file ‘test’ is not associated with any file type. Please define the association:

File pattern: test

Open matching files in MindStudio:
- Archive
- AspectJ (Syntax Highlighting Only)
- Buildout Config
- C Code
- C or C++ Header
- C#
- C++ Code
- C++ Header
- C/C++
- CCE Code

Open matching files in associated application

Select a file type and click OK.

Deleting a File

Right-click a file. In the shortcut menu displayed, click Delete.... Then perform operations as prompted.

12.3 Run Configuration

12.3.1 Editing a Run Configuration

Step 1 On the MindStudio home page, click Edit Configurations... in the toolbar, as shown in Figure 12-13.

Step 2 In the navigation bar on the left, select a run configuration to be modified and modify the configuration on the right.

Step 3 Click Apply to save the configuration. Click OK to close the dialog box.
Step 4  Click the icon shown in Figure 12-14 to run a project.

Figure 12-14 Running the project

----End

12.3.2 Pausing and Resuming

To stop the execution of UT/ST test cases, click either shown in Figure 12-15 to view the printed runtime logs.

Figure 12-15 Pausing icon

To resume the execution of the UT/ST test cases, click .

12.4 Device Manager

This topic describes how to manage devices with the Device Manager tool in MindStudio.

Device Manager displays the details of each connected device, facilitating device management.

- In MindStudio, choose Ascend > Device Manager from the menu bar. The Device Manager dialog box is displayed.
- On the shortcut toolbar of MindStudio, click the icon. The Device Manager dialog box is displayed, as shown in Figure 12-16.
Figure 12-16 Device Manager dialog box

Table 12-2 describes the parameters and icons in the Device Manager dialog box.

Table 12-2 Parameter description

<table>
<thead>
<tr>
<th>Parameter/Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>Device alias. Using aliases can help manage devices expediently when multiple devices are connected. The default format is <code>&lt;username&gt;@&lt;ip&gt;:&lt;port&gt;</code>.</td>
</tr>
<tr>
<td>Arch</td>
<td>Device OS architecture.</td>
</tr>
<tr>
<td>Connectivity</td>
<td>Status of the connection between MindStudio and the device: Connected, Disconnected</td>
</tr>
<tr>
<td></td>
<td>Edits a device in the displayed dialog box by referring to 12.1 SSH Connection Management.</td>
</tr>
<tr>
<td></td>
<td>Refreshes a device.</td>
</tr>
</tbody>
</table>

12.5 ADK Manager

12.5.1 Overview

ADK Manager allows you to switch between and update available Ascend-CANN-Toolkits without reinstalling MindStudio. Before using this function, ensure that MindStudio and an Ascend-CANN-Toolkit have been installed.

Open the ADK Manager dialog box in any of the following ways:

- On the menu bar of the MindStudio project page, choose File > Settings... > Appearance&Behavior > System Settings > ADK. The ADK Manager dialog box is displayed.
On the menu bar of the MindStudio project page, choose **Ascend > ADK Manager**. The **ADK Manager** dialog box is displayed.

On the shortcut toolbar of MindStudio, click the following icon. The **ADK Manager** dialog box is displayed.

**Figure 12-17** Opening ADK Manager by clicking the shortcut icon

In the lower right corner of the **Welcome to MindStudio** window, choose **Configure > ADK Manager**. The **ADK Manager** dialog box is displayed.

**Figure 12-18** shows the **ADK Manager** dialog box.

**Table 12-3** describes the parameters and icons in the **ADK Manager** dialog box.

<table>
<thead>
<tr>
<th>Parameter/Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADK Location</td>
<td>ADK installation path. Defaults to $HOME/Ascend/ascend-toolkit/{software version}. You can click <img src="image" alt="switch between ADKs" /> on the right to switch between the ADKs.</td>
</tr>
<tr>
<td>Component</td>
<td>Component name</td>
</tr>
<tr>
<td>Version</td>
<td>ADK version number and software package version number</td>
</tr>
<tr>
<td>Host OS Arch</td>
<td>Host OS and architecture</td>
</tr>
</tbody>
</table>
### 12.5.2 Updating ADK

This section describes how to update an ADK without reinstalling MindStudio.

**Step 1** Open ADK Manager by referring to 12.5.1 Overview and click Install ADK on the right. The dialog box for installing the ADK is displayed.

**Step 2** Install an ADK by referring to 2.4.4 Installing MindStudio. After the installation is complete, the ADK information of the new version is generated in the installation path.

**Step 3** After MindStudio automatically restarts, the ADK update is complete.

----End

### 12.5.3 Switching/Activating ADK

#### Switching Between ADKs

**Step 1** Open the ADK Manager by referring to 12.5.1 Overview and click on the right of ADK location. The dialog box of selecting the Toolkit path is displayed, as shown in Figure 12-19.
Step 2  Select the new Toolkit and click OK. Confirm the change, as shown in Figure 12-20.

Figure 12-20 Confirming the change

- Click OK. After MindStudio automatically reboots, the new ADK is activated. If a project has been opened before ADK switching, the dialog boxes shown in Figure 12-21 and Figure 12-22 are displayed, asking you to confirm the ADK version.

Figure 12-21 Message indicating that the ADK is not installed

ADK or Target is not installed for the project.
**Figure 12-22** Confirming the ADK version

- Click **Cancel**. The new ADK is not activated and is unavailable. To activate the ADK, click ![image](image_url) and perform operations as prompted.

---End

**Activating an ADK**

**CAUTION**

You can perform this operation only when the ![image](image_url) icon is available, that is, an installed but not activated ADK exists in the system.

Open the **ADK Manager** dialog box by referring to **12.5.1 Overview** and click ![image](image_url) corresponding to the ADK to activate the ADK.

**12.5.4 Configuring the Cross-Compilation Environment**

In the **ADK Manager** dialog box, click the **Cross Compiler** tab. The compilation environment to be set for the activated ADK is displayed, as shown in **Figure 12-23**. If **Status** is **Not installed**, manually set up the compilation environment by referring to **2.4.5 Configuring the Compilation Environment**.

**NOTE**

- The version of the Ascend-CANN-Toolkit on the MindStudio installation server must be the same as that of the CANN software package in the operating environment.
- The Ascend-CANN-Toolkit must be installed on the MindStudio installation server, with the corresponding CANN software package in the operating environment.
After the configuration is complete, restart MindStudio and open the **ADK Manager** dialog box again. You will find that **Status** has changed to **Installed**.

### 12.6 Log Management

#### 12.6.1 Before You Start

**Reference Documents**

<table>
<thead>
<tr>
<th>Product</th>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlas 200 DK</td>
<td><em>Log Reference</em></td>
</tr>
<tr>
<td>Others</td>
<td>• <em>Log Reference (Inference)</em></td>
</tr>
<tr>
<td></td>
<td>• <em>Log Reference (Training)</em></td>
</tr>
</tbody>
</table>
Tool Overview

MindStudio provides a system-wide log collection and analysis solution for the Ascend AI Processor, improving the efficiency of locating algorithm problems at runtime. MindStudio also provides a unified log format and a graphical user interface (GUI) for visualized analysis of cross-platform logs and runtime diagnosis, improving the ease of use of the log analysis system.

By using the Log tool, you can perform the following operations:

- Log management
- Log level setting

Click the Log tab at the bottom of the MindStudio window. The Log window is displayed, as shown in Figure 12-24.

Figure 12-24 Log window

- **NOTE**
  - localhost is always on top by default and is used to view runtime logs of the local system.
  - To use the Log tool to open folders and files in Linux, the Nautilus and Gedit software are required. Run the following command as the MindStudio installation user to install them:
    - Ubuntu 18.04-x86/AArch OS: `sudo apt-get install nautilus gedit`
    - EulerOS2.8-AArch, Centos7.6-x86/AArch, and Kylin OS V10 SP1-AArch: `sudo yum install nautilus gedit`
  - By default, log files are opened using the Gedit software. The configuration command is `xdg-mime default org.gnome.gedit.desktop text/x-log`.

12.6.2 Log Management

On the System Log tab page, perform the following steps to view system runtime logs.

- **NOTE**
  - System logs cannot be deleted in the MindStudio GUI. To manage logs, perform the following operations:
    - RC scenario: Log in to the device as the root user and manage logs in the `/var/log/npu/slog` directory.
    - EP scenario: Log in to the system as the SSH user and manage logs in the `/home/ascend/log/` directory.
  - Before viewing system logs, you need to add a device.
Step 1  In the Log window in MindStudio, click the System Log tab page. In the device list, click a device to expand the log list of the device.

Figure 12-25 Log list (RC mode)

```
% Localhost
% device-0
  % device-0_2020111807342123.
  % device-0_2020110815385024.
% device-os:0
% device-os:1
  % device-os:1_202012151050250.
  % device-os:1_202012151946000.
  % device-os:1_202012161012191.
  % device-os:1_202012161080800.
  % device-os:1_202011132234411.
  % device-os:1_202011132226411.
% device-os:1_20201109175170.
```

Figure 12-26 Log list (EP mode)

```
% Localhost
% 192.168.10.2
% plog
  % plog:34123_202102120205255331
```

Step 2  Double-click a log file in the log list. The log content is displayed.

**NOTE**

Only the most recent 5000 log messages are displayed, and the maximum size of each log file is 1 MB.

Figure 12-27 System Log tab page (RC mode)
Step 3 On the **System Log** tab page, click the following icons as required:

- ![icon](image) : searches for log messages by keyword.
- ![icon](image) : clears the displayed log messages.
- ![icon](image) **Open Log File** : opens a log file. All the log contents of the log file are displayed.
- ![icon](image) **Open Log Folder** : opens the local path where the log file is downloaded on the device in the IDE.

---

### 12.6.3 Log Level Setting

You can set the log levels of the system in the IDE.

Step 1 In the **Log** window in MindStudio, click the **System Log** tab page. In the device list, right-click a device and choose **Set Log Level** from the shortcut menu. See **Figure 12-29**.

![Figure 12-29 Choosing Set Log Level from the shortcut menu](image)

Step 2 In the **Set Log Level** dialog box displayed, set the log level. See **Figure 12-30**.

**NOTE**

- For details about log levels, see *Log Reference* described in *Table 12-4*.
- After the log level is changed, if the Ascend AI Processor is restarted in the PCIe scenario, the log level on the device side is restored to the default level.
Step 3  Set the log level and click OK.

---End

12.7 Shortcuts

This section describes common shortcuts of the IntelliJ IDEA. For more information, see related official documents.

<table>
<thead>
<tr>
<th>Shortcut</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alt + Left</td>
<td>(Required) Goes to previous editor tab.</td>
</tr>
<tr>
<td>Alt + Right</td>
<td>(Required) Goes to next editor tab.</td>
</tr>
<tr>
<td>Alt + Up</td>
<td>(Required) Goes to previous method.</td>
</tr>
<tr>
<td>Alt + Down</td>
<td>(Required) Goes to next method.</td>
</tr>
<tr>
<td>Ctrl + =</td>
<td>Expands the code.</td>
</tr>
<tr>
<td>Ctrl + -</td>
<td>Folds the code.</td>
</tr>
<tr>
<td>Ctrl + /</td>
<td>(Required) Comments/Uncomments with line comment according to the file type.</td>
</tr>
<tr>
<td>Ctrl + F</td>
<td>(Required) Finds in file.</td>
</tr>
<tr>
<td>Ctrl + R</td>
<td>(Required) Replace</td>
</tr>
<tr>
<td>Ctrl + Z</td>
<td>(Required) Undo</td>
</tr>
<tr>
<td>Ctrl + Y</td>
<td>(Required) Deletes line at caret.</td>
</tr>
<tr>
<td>Ctrl + X</td>
<td>Cuts current line or selected block.</td>
</tr>
<tr>
<td>Ctrl + C</td>
<td>Copies current line or selected block.</td>
</tr>
<tr>
<td>Ctrl + D</td>
<td>(Required) Duplicates current line or selected block.</td>
</tr>
<tr>
<td>Ctrl + W</td>
<td>(Required) Extends selection. You can increase the scope of selection according to specific code constructs.</td>
</tr>
<tr>
<td>Shortcut</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Ctrl + E</td>
<td>(Required) Displays recent files</td>
</tr>
<tr>
<td>Ctrl + N</td>
<td>(Required) Quickly opens any class.</td>
</tr>
<tr>
<td>Ctrl + G</td>
<td>Goes to a line.</td>
</tr>
<tr>
<td>Ctrl + Q</td>
<td>Displays context info.</td>
</tr>
<tr>
<td>Ctrl + Alt + L</td>
<td>(Required) Reformats code, applying to the current file and the entire package directory.</td>
</tr>
<tr>
<td>Ctrl + Alt + O</td>
<td>(Required) Optimizes imports, applying to the current file and the entire package directory.</td>
</tr>
<tr>
<td>Ctrl + Shift + F</td>
<td>(Required) Finds in path.</td>
</tr>
<tr>
<td>Ctrl + Shift + R</td>
<td>(Required) Replaces in path.</td>
</tr>
</tbody>
</table>

### 12.8 Common Native Functions

This section describes the common functions of IntelliJ IDEA. For more information, see related official documents.

#### Collapsing a Directory

As shown in Figure 12-31, deselecting **Compact Middle Packages** indicates that the directory is not collapsed, and vice versa.

**Figure 12-31** Directory collapse

![Directory collapse](image)

**Figure 12-32** compares a collapsed directory with a directory that is not collapsed.
Figure 12-32 Directory comparison
13 MindStudio Management (Linux)

13.1 Querying MindStudio Version

After the installation is complete, you can check whether the MindStudio version is as expected in either of the following ways:

- After MindStudio is installed, choose Get Help > About in the lower right corner of the Welcome to MindStudio window, as shown in Figure 13-1.

Figure 13-1 Querying the version
Go to the MindStudio project page and choose Help > About on the menu bar.

NOTE
The version in Figure 13-1 is an example only.

13.2 Launching MindStudio

To launch MindStudio, perform the following operations:

As the MindStudio installation user, go to the MindStudio/bin directory generated after package extraction and run the following command:

```
./MindStudio.sh
```

Run the following command to query the MindStudio process:

```
ps -ef|grep idea
```

13.3 Exiting MindStudio

To exit MindStudio, click in the upper right corner in MindStudio.

13.4 Updating MindStudio

Manually Update

You are advised to update MindStudio to the latest version to experience the latest features and enhancements. If MindStudio is not installed yet, install it by referring to 2.4.4 Installing MindStudio.

MindStudio is green software. To update MindStudio, uninstall it by referring to 13.5 Uninstalling MindStudio, and then install the latest version by referring to 2.4.4 Installing MindStudio.

Update by msInstaller

For details, see the "Upgrade" section in CANN Software Installation Guide (msInstaller).

Updating MindStudio Online in the IDE

Step 1 On the MindStudio home page, choose Help > Check for Updates... on the menu bar. An update message is displayed in the lower right corner of the page, as shown in Figure 13-2.

Figure 13-2 Update message
If the message "No plugin updates available" is displayed in the lower right corner of the page, MindStudio is of the latest version.

**Step 2** Click **Update...** in the displayed information.

**Step 3** Select whether to update immediately, as shown in **Figure 13-3**.
- **Ignore This Update**: Ignore the update and exit the update page.
- **Remind Me Later**: Remind you to update MindStudio later.
- **Update and Restart**: Update and restart MindStudio.

**Figure 13-3** Update page

![Update page](image)

**Step 4** Click **Update and Restart** to update and restart MindStudio.

The system starts to download the patch file, as shown in **Figure 13-4**.

**Figure 13-4** Downloading the patch file

![Downloading the patch file](image)

- **Cancel**: Cancel the patch file download and MindStudio update.
- **Background**: Download the patch file in the background.

**Step 5** After the patch file is downloaded, update MindStudio, as shown in **Figure 13-5**.
During the update, you will be prompted to modify some executable files and configuration information of MindStudio due to version inconsistency. Figure 13-6 shows an example of the prompt message.

- **Cancel**: Cancel the MindStudio update and wait for MindStudio of the current version to automatically restart.
- **Proceed**: Proceed to update MindStudio. After the update is complete, MindStudio of the new version automatically restarts.

13.5 Uninstalling MindStudio

If you do not need MindStudio and Ascend-CANN-Toolkit anymore, uninstall them as follows. Generally, you do not want to uninstall the Toolkit only, because other MindStudio functions may be unavailable. If you want to upgrade Ascend-CANN-Toolkit, install the latest version of the Ascend-CANN-Toolkit by referring to 12.5 ADK Manager.

Manually Uninstalling MindStudio

Remove the `MindStudio` directory from the `$HOME` directory in Linux as the MindStudio installation user. (The directory is generated after the extraction of the MindStudio tar.gz package.)

The deletion command is as follows:
rm -rf MindStudio

- Uninstalling MindStudio does not delete data in the following directories:
  - $HOME/.cache/Huawei/MindStudioMS-{version}: stores runtime logs generated after MindStudio is installed and launched.
  - $HOME/.config/Huawei/MindStudioMS-{version}: stores user configurations.
  - $HOME/.mindstudio: stores the MindStudio property file mindstudio.properties (recording the Toolkit installation path) and records of operations performed by users on the privacy statement window.
  - AscendProjects: projects created by the user

- To restore the initial MindStudio configuration, run the following commands as the MindStudio installation user in the $HOME directory to delete the user configurations:
  rm -rf .cache/Huawei/MindStudioMS-x.x/
  rm -rf .config/Huawei/MindStudioMS-x.x/
  rm -rf .mindstudio/

Manually Uninstalling Ascend-CANN-Toolkit

For details, see the "Uninstallation" section in CANN Software Installation Guide (ascend-deployer).

Uninstalling MindStudio Using mslInstaller

For details, see the "Uninstallation" section in CANN Software Installation Guide (msInstaller).

13.6 FAQs

13.6.1 What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When "pip3 install" Is Run?

Symptom

During the installation of dependencies, a message is displayed indicating that the network connection fails when the command of pip3 install xxx is run, and Could not find a version that satisfies the requirement xxx is displayed. Run the apt-get update command to check whether the sources are available. The displayed message is as follows.

Figure 13-7 Message displayed upon pip3 install
Possible Cause

The pip source is not configured.

Solution

Configure the pip source as follows:

**Step 1** Run the following command as the MindStudio installation user:

```
cd ~/.pip
```

If a message indicating that the directory does not exist is displayed, create one:

```
mkdir ~/.pip
cd ~/.pip
```

Create a `pip.conf` file in the `.pip` directory:

```
touch pip.conf
```

**Step 2** Edit the `pip.conf` file.

Run the `vi pip.conf` command to open the `pip.conf` file and edit the file as follows.

```
[install]
#Configure the trusted host as required.
trusted-host=repo.huaweicloud.com

[global]
#Configure the sources as required.
index-url=https://repo.huaweicloud.com/repository/pypi/simple
```

**Step 3** Run the `.wq!` command to save the file and exit.

-----End
14 MindStudio Management (Windows)

Uninstalling MindStudio

You can uninstall MindStudio as follows: If you want to upgrade Ascend-CANN-Toolkit, install the latest version of the Ascend-CANN-Toolkit by referring to 12.5 ADK Manager.

Step 1 Go to the C:\Users\Individual user directory and delete the .mindstudio folder.

Step 2 Uninstall the MindStudio installation package.

- Uninstalling MindStudio does not delete data in the following directories:
  - Runtime logs generated after MindStudio is installed and launched
  - User configuration data

- If you want to restore the initial configuration of MindStudio, delete the MindStudioMS-{version} folder in C:\Users\Individual user\AppData\Roaming\Huawei.

----End

Updating MindStudio Online in the IDE

Step 1 On the MindStudio home page, choose Help > Check for Updates... on the menu bar. An update message is displayed in the lower right corner of the page, as shown in Figure 14-1.

Figure 14-1 Update message

NOTE

If the message "No plugin updates available" is displayed in the lower right corner of the page, MindStudio is of the latest version.
Step 2 Click **Update**... in the displayed information.

Step 3 Select whether to update immediately, as shown in **Figure 14-2**.
- **Ignore This Update**: Ignore the update and exit the update page.
- **Remind Me Later**: Remind you to update MindStudio later.
- **Update and Restart**: Update and restart MindStudio.

**Figure 14-2** Update page

---

Step 4 Click **Update and Restart** to update and restart MindStudio.

The system starts to download the patch file, as shown in **Figure 14-3**.

**Figure 14-3** Downloading the patch file

- **Cancel**: Cancel the patch file download and MindStudio update.
- **Background**: Download the patch file in the background.

Step 5 After the patch file is downloaded, update MindStudio, as shown in **Figure 14-4**.
During the update, you will be prompted to modify some executable files and configuration information of MindStudio due to version inconsistency. Figure 14-5 shows an example of the prompt message.

- **Cancel**: Cancel the MindStudio update and wait for MindStudio of the current version to automatically restart.
- **Proceed**: Proceed to update MindStudio. After the update is complete, MindStudio of the new version automatically restarts.
15 Appendixes

15.1 MindStudio Package Overview
15.2 Installing Dependencies
15.3 Operator Specifications
15.4 Feature Configuration
15.5 How Do I Determine the Video Stream Format Standard When I Perform CSC on a Model Using AIPP?
15.6 Configuring a System Network Proxy
15.7 Security Hardening Suggestions
15.8 FAQs

15.1 MindStudio Package Overview

Table 15-1 describes the files extracted from the MindStudio tar.gz package.
Table 15-1 Description of the files extracted from the tar.gz package

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| MindStudio_{version}_linux.tar.gz | MindStudio  
\- **bin**: MindStudio execution directory, storing dependent binary files  
\- **build.txt**: build information of the tar.gz package  
\- **config**: MindStudio configuration file  
\- **Install-Linux.tar.txt**: MindStudio installation description  
\- **jbr**: library and file dependencies of a 64-bit OS  
\- **lib**: Java library dependencies  
\- **license**: third-party license dependencies  
\- **LICENSE.txt**: Apache licence description  
\- **NOTICE.txt**: considerations  
\- **plugins**: plug-in and Java library dependencies of MindStudio  
\- **product-info.json**: file storing the MindStudio version and executable file path  
\- **redist**: Java annotations  
\- **samples**: operator project samples and model files of typical network models  
\- **tools**: tools | MindStudio installatio |

### 15.2 Installing Dependencies

#### 15.2.1 Ubuntu 18.04-x86_64

**⚠️ NOTE**

- If you install the following dependencies as the root user, delete `sudo` or `--user` from the commands.
- If you install the following dependencies as a non-root user, run the `su - username` command to switch to the MindStudio installation user.

**Step 1** Check whether the required software is installed.

1. Check whether the Python dependencies and GCC software are installed.

   Check whether software including GCC, g++, Make, and Python is installed:
If the following information is displayed, the software has been installed:

```
g++ (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0
gcc (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0
GNU Make 4.1
cmake version 3.10.2
UnZip 6.00 of 20 April 2009, by Debian. Original by Info-ZIP.
zlib1g:amd64   1:1.2.11.dfsg-0ubuntu2 amd64        compression library - runtime
zlib1g-dev:amd64 1:1.2.8.dfsg-2ubuntu4.1 amd64        compression library - development
libbz2-dev:amd64 1.0.6-8ubuntu0.2 amd64        high-quality block-sorting file compressor library - development
libsqlite3-dev:amd64 3.11.0-1ubuntu1.2 amd64        SQLite 3 development files
openssl        1.1.1-1ubuntu2.1~18.04.5 amd64        Secure Sockets Layer toolkit - cryptographic utility
libssl-dev:amd64 1.0.2g-1ubuntu4.15 amd64        Secure Sockets Layer toolkit - development files
libxslt1-dev:amd64 1.1.28-2.1ubuntu3.3 amd64        XSLT 1.0 processing library - development kit
libffi-dev:amd64 3.2.1-4 amd64        Foreign Function Interface library (development files)
libncurses5-dev:amd64 6.1-1ubuntu1.18.04 amd64        developer's libraries for ncurses
libncursesw5-dev:amd64 6.1-1ubuntu1.18.04 amd64        developer's libraries for ncurses
```

Otherwise, run the following command to install the software. You can modify the following command to install only some of them as required.

```
sudo apt-get install -y gcc g++ make cmake unzip zlib1g zlib1g-dev libbz2-dev libsqlite3-dev openssl libssl-dev libxslt1-dev libffi-dev libncurses5-dev libncursesw5-dev gnome-keyring pcoutils net-tools
```

2. Check whether the Python development environment is installed.

The ATC installation depends on the Python 3 environment. Run the `python3.7.5 --version` and `pip3.7.5 --version` commands to check whether Python 3 is installed. If the following messages are displayed, Python 3 is installed.

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

Otherwise, use the following procedure to install Python 3.7.5:

a. Run the `wget` command to download the source code package of Python 3.7.5 to any directory of the MindStudio installation server:

```
wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
```

b. Go to the download directory and decompress the source code package:

```
tar -zxvf Python-3.7.5.tgz
```

c. Go to the decompressed folder, create an installation directory, and run the configuration, build, and installation commands.

```
cd Python-3.7.5
sudo mkdir /usr/local/python3.7.5
./configure --prefix=/usr/local/python3.7.5
--enable-loadable-sqlite-extensions --enable-shared
make
sudo make install
```
The --prefix parameter specifies the Python installation path. You can change it based on the site requirements. The --enable-shared parameter is used to compile the libpython3.7m.so.1.0 dynamic library. The --enable-loadable-sqlite-extensions parameter is used to load libsqlite3-dev.

This document uses --prefix=/usr/local/python3.7.5 as an example. After the configuration, build, and installation commands are executed, the installation package is output to the /usr/local/python3.7.5 directory, and the libpython3.7m.so.1.0 dynamic library is output to the /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 directory.

d. Run the following command to copy the libpython3.7m.so.1.0 dynamic library:
   
i. Copy the built file libpython3.7m.so.1.0 to /usr/lib64:
      
      

      

      When the following information is displayed, enter y to overwrite the libpython3.7m.so.1.0 file provided by the system.

      cp: overwrite 'libpython3.7m.so.1.0'?

      If the /usr/lib64 directory does not exist in the environment, copy the file to the /usr/lib directory.

      sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib

      Replace the path of the libpython3.7m.so.1.0 file as required.

   
e. Set the soft links:
   
   

   

   sql
   

   f. Set the Python 3.7.5 environment variables.
   
i. Run the vi ~/.bashrc command in any directory as the installation user to open the .bashrc file and append the following line to the file:

   # Set the Python 3.7.5 library path.
   export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
   # If multiple Python 3 versions exist in the user environment, specify Python 3.7.5.
   export PATH=/usr/local/python3.7.5/bin:$PATH

   ii. Run the :wq! command to save the file and exit.
   
   iii. Run the source ~/.bashrc command for the modification to take effect immediately.

   g. After the installation is complete, run the following commands to check the installed version. If the required version information is displayed, the installation is successful.

   python3.7.5 --version
   pip3.7.5 --version
   python3.7 --version
   pip3.7 --version

3. Check that the library files that MindStudio depends on have been installed.

   xterm is required for MindStudio to display the dependent library files. Run the dpkg -I xterm| grep xterm| grep ii command to check whether the library files have been installed. If the following information is displayed, the library files have been installed.

   xterm 322-Tubuntu1 amd64 X terminal emulator

   Otherwise, run the following command.

   sudo apt-get install -y xterm
4. Check whether the software for displaying HTML reports is installed for MindStudio.

Use the `firefox --version` and `dpkg -l xdg-utils| grep xdg-utils| grep ii` commands to check whether dependencies Firefox and xdg-utils for displaying HTML reports are installed. If the following information is displayed, the dependencies have been installed.

```
Mozilla Firefox 71.0
xdg-utils 1.1.1-1ubuntu1.16.04.3 all desktop integration utilities from freedesktop.org
```

Otherwise, run the following command to install the software. You can modify the following command to install only some of them as required.

```
sudo apt-get install -y firefox xdg-utils
```

5. Check whether the font software is installed.

Viewing the online help in MindStudio requires the font software. Run the following commands to check whether the font software is installed:

```
dpkg -l fonts-droid-fallback| grep fonts-droid-fallback| grep ii
dpkg -l fonts-wqy-zenhei| grep fonts-wqy-zenhei| grep ii
dpkg -l fonts-wqy-microhei| grep fonts-wqy-microhei| grep ii
dpkg -l fonts-aphic-ukai| grep fonts-aphic-ukai| grep ii
dpkg -l fonts-aphic-uming| grep fonts-aphic-uming| grep ii
```

If the following information is displayed, the software has been installed:

```
fonts-droid-fallback 1:6.0.1r16-1 all handheld device font with extensive style and language support (fallback)
fonts-wqy-zenhei 0.9.45-6ubuntu1 all "WenQuanYi Zen Hei" A Hei-Ti Style (sans-serif) Chinese font
fonts-wqy-microhei 0.2.0-beta-2 all Sans-serif style CJK font derived from Droid
fonts-aphic-ukai 0.2.20080216.2-4ubuntu2 all "AR PL UKai" Chinese Unicode TrueType font collection Kaithi style
fonts-aphic-uming 0.2.20080216.2-7ubuntu2 all "AR PL UMing" Chinese Unicode TrueType font collection Mingti style
```

Otherwise, run the following command to install the software. You can modify the following command to install only some of them as required.

```
sudo apt-get install -y fonts-droid-fallback fonts-wqy-zenhei fonts-wqy-microhei fonts-aphic-ukai fonts-aphic-uming
```

**Step 2** Install the Python 3 dependencies.

TBE depends on NumPy and decorator. TIK depends on SymPy, CFFI, and GNU readline. The uTST tool depends on the coverage software. Model Accuracy Analyzer depends on protobuf and SciPy. Profiling depends on protobuf, grpcio, grpcio-tools, and requests. If you want to use the operator ST cases to obtain the operator shape and more from the model, you need to install TensorFlow. Otherwise, you do not need to install TensorFlow.

Before the installation, run the `pip3.7.5 list` command to check whether dependencies have been installed. If not, run the following command to install them. You can modify the following command to install only some of them as required.

```
pip3.7.5 install attrs --user
pip3.7.5 install psutil --user
pip3.7.5 install decorator --user
pip3.7.5 install numpy --user
pip3.7.5 install protobuf==3.11.3 --user
pip3.7.5 install scipy --user
pip3.7.5 install sympy --user
pip3.7.5 install grpc --user
pip3.7.5 install coverage --user
pip3.7.5 install pylint --user
pip3.7.5 install matplotlib --user
```
pip3.7.5 install grpcio --user
pip3.7.5 install grpcio-tools --user
pip3.7.5 install requests --user
pip3.7.5 install xlrd==1.2.0 --user
pip3.7.5 install absl-py --user
pip3.7.5 install pandas --user
pip3.7.5 install tensorflow==1.15.0 --user
pip3.7.5 install pytest --user

During the command execution, if the network connection fails and the message "Could not find a version that satisfies the requirement xxx" is displayed, fix the error by referring to the What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When "pip3 install" Is Run?

**Step 3** Install the Java Development Kit (JDK).

Before installing the JDK, run the following command to check whether Java exists in the system:

```
which java
```

If the related path is returned, Java exists. Otherwise, run the following command to install the JDK:

```
sudo apt-get install -y openjdk-11-jdk
```

1. The installation of MindStudio depends on the `JAVA_HOME` environment variable. Run the following commands for variable setting:

   a. Run the following command in any directory to open the `.bashrc` file:
      ```
      vi ~/.bashrc
      ```
   
   b. Append the following lines to the file:
      ```
      export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
      export PATH=$JAVA_HOME/bin:$PATH
      ```

   **NOTE**

   `JAVA_HOME` indicates the JDK installation path. If the JDK has been installed, modify `JAVA_HOME` as required. If the JDK has been installed according to the preceding steps, skip this modification.

   c. Save the file and exit:
      ```
      :wq!
      ```

   d. Make the configuration take effect:
      ```
      source ~/.bashrc
      ```

   e. Set the environment variable:
      ```
      echo $JAVA_HOME
      ```

      The command output is as follows:
      ```
      /usr/lib/jvm/java-11-openjdk-amd64
      ```

   f. Check the JDK installation.
      ```
      which jconsole
      ```

      If the command output is as follows, the JDK is successfully installed.
      Otherwise, the JDK installation fails.
      ```
      /usr/lib/jvm/java-11-openjdk-amd64/bin/jconsole
      ```

**Step 4** Go back to 2.4.4 Installing MindStudio.

-----End
15.2.2 Ubuntu 18.04-aarch64

- If you install the following dependencies as the root user, delete sudo or --user from the commands.
- If you install the following dependencies as a non-root user, run the su - username command to switch to the MindStudio installation user.

**Step 1** Check whether the required software is installed.

1. Check whether the Python dependencies and GCC software are installed.

Check whether software including GCC, g++, and Make are installed:

```bash
gcc --version
g++ --version
make --version
cmake --version
unzip --version
dpkg -l zlib1g| grep zlib1g| grep ii
dpkg -l libbz2-2-dev| grep libbz2-2-dev| grep ii
dpkg -l libsqlite3-dev| grep libsqlite3-dev| grep ii
dpkg -l openssl| grep openssl| grep ii
dpkg -l libssl-dev| grep libssl-dev| grep ii
dpkg -l libffi-dev| grep libffi-dev| grep ii
dpkg -l libncurses5-dev| grep libncurses5-dev| grep ii
dpkg -l gnome-keyring| grep gnome-keyring| grep ii
dpkg -l pugixml| grep pugixml| grep ii
dpkg -l net-tools| grep net-tools| grep ii
dpkg -l libblas-dev| grep libblas-dev| grep ii
dpkg -l gfortran| grep gfortran| grep ii
dpkg -l libblas3| grep libblas3| grep ii
dpkg -l libopenblas-dev| grep libopenblas-dev| grep ii
dpkg -l build-essential | grep build-essential | grep ii
dpkg -l libgdbm-dev | grep libgdbm-dev | grep ii
dpkg -l libncursesw5-dev | grep libncursesw5-dev | grep ii
dpkg -l gdb | grep gdb | grep ii
dpkg -l liblapack-dev | grep liblapack-dev | grep ii
dpkg -l m4 | grep m4 | grep ii
dpkg -l openjfx | grep openjfx | grep ii
dpkg -l libqhull-dev | grep libqhull-dev | grep ii
```

If the following information is displayed, the software has been installed:

```
gcc (Ubuntu/Linaro 7.5.0-3ubuntu1-18.04) 7.5.0
g++ (Ubuntu/Linaro 7.5.0-3ubuntu1-18.04) 7.5.0
GNU Make 4.1
cmake version 3.10.2
UnZip 6.00 of 20 April 2009, by Debian. Original by Info-ZIP.
ii zlib1g-arm64 1.2.11.dfsg-0ubuntu2 arm64 compression library - runtime
ii zlib1g-dev-arm64 1.2.11.dfsg-0ubuntu2 arm64 compression library - development
ii libbz2-2-dev-arm64 1.0.6-8.1ubuntu0.2 arm64 high-quality block-sorting file compressor library - development
ii libsqlite3-dev-arm64 3.22.0-1ubuntu0.4 arm64 SQLite 3 development files
ii openssl 1.1.1-Tubuntu2.1-18.04.7 arm64 Secure Sockets Layer toolkit - cryptographic utility
ii libssl-dev-arm64 1.1.1-Tubuntu2.1-18.04.7 arm64 Secure Sockets Layer toolkit - development files
dpkg-query: no packages found matching libxslt1-dev
ii libffi-dev-arm64 3.2.1-8 arm64 Foreign Function Interface library (development files)
ii libncurses5-dev-arm64 6.1-1ubuntu1.18.04 arm64 developer's libraries for ncurses
gnome-keyring 3.28.0.2-Tubuntu1.18.04.1 arm64 GNOME keyring services (daemon and tools)
```
dpkg-query: no packages found matching pcutils
dpkg-query: no packages found matching net-tools
ii  gfortran       4:7.4.0-1ubuntu2.3 arm64        GNU Fortran 95 compiler
ii  libopenblas-dev:arm64 0.2.20+ds-4 arm64 Optimized BLAS (linear algebra) library (development files)
ii  build-essential 12.4ubuntu1 arm64        Informational list of build-essential packages
ii  libgdbm-dev:arm64 1.14.1-6 arm64 GNU dbm database routines (development files)
ii  libncursesw5-dev:arm64 6.1-1ubuntu1.18.04 arm64 developer's libraries for cursesw
ii  pkg-config 0.29.1-0ubuntu2 arm64 manage compile and link flags for libraries
ii  libglib2.0-dev:arm64 2.56.4-0ubuntu1.18.04.6 arm64 Development files for the Glib library
ii  libusb-1.0-0-dev:arm64 0.1.10-2 arm64 deprecated library for D-Bus IPC (development files)
ii  gdb 8.1.1-1ubuntu1 arm64 GNU Debugger
ii  liblapack-dev:arm64 3.7.1-4ubuntu1 arm64 Library of linear algebra routines 3 - static version
ii  m4 1.4.18-1 arm64 macro processing language
ii  openjfx 11.0.2+1-1~18.04.2 arm64 JavaFX/OpenFX - Rich client application platform for Java
ii  libfreetype6-dev:arm64 2.8.1-2ubuntu2.1 arm64 FreeType 2 font engine, development files
ii  libghc-8.8.1-dev:arm64 0.15.2-4 arm64 calculate convex hulls and related structures (development files)
ii  openssh-server 1:7.6p1-4ubuntu0.3 arm64 secure shell (SSH) server, for secure access from remote machines

Otherwise, run the following command to install the software. You can modify the following command to install only some of them as required.
```
sudo apt-get install -y gcc g++ make cmake unzip zlib1g zlib1g-dev libbz2-dev libsqlite3-dev openssl libffi-dev libncurses5-dev gnome-keyring pcutils net-tools libblas-dev gfortran libblas3 libopenblas-dev build-essential libgdbm-dev libncursesw5-dev pkg-config libglib2.0-dev libdbus-glib-1-dev gdb liblapack-dev m4 openjfx libfreetype6-dev libqhull-dev openssh-server
```

2. Check whether the Python development environment is installed.

The ATC installation depends on the Python 3 environment. Run the `python3.7.5 --version`, `python3.7 --version`, `pip3.7.5 --version`, and `pip3 --version` commands to check whether Python 3 is installed. If the following messages are displayed, Python 3 is installed.

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

Otherwise, use the following procedure to install Python 3.7.5:

a. Run the `wget` command to download the source code package of Python 3.7.5 to any directory of the MindStudio installation server:
```
wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
```

b. Go to the download directory and decompress the source code package:
```
tar -zxvf Python-3.7.5.tgz
```

c. Go to the decompressed folder, create an installation directory, and run the configuration, build, and installation commands.
```
cd Python-3.7.5
sudo mkdir /usr/local/python3.7.5
./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --enable-shared
make
sudo make install
```

The `--prefix` parameter specifies the Python installation path. You can change it based on the site requirements. The `--enable-shared` parameter is used to compile the `libpython3.7m.so.1.0` dynamic library. The `--enable-loadable-sqlite-extensions` parameter is used to load libsqlite3-dev.

This document uses `--prefix=/usr/local/python3.7.5` as an example. After the configuration, build, and installation commands are executed, the installation package is output to the `/usr/local/python3.7.5` directory, and the `libpython3.7m.so.1.0` dynamic library is output to the `/usr/local/python3.7.5/lib/libpython3.7m.so.1.0` directory.
d. Run the following command to copy the `libpython3.7m.so.1.0` dynamic library:

i. Copy the built file `libpython3.7m.so.1.0` to `/usr/lib64`:

```
sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
```

When the following information is displayed, enter `y` to overwrite the `libpython3.7m.so.1.0` file provided by the system.

```
cp: overwrite 'libpython3.7m.so.1.0'?
```

If the `/usr/lib64` directory does not exist in the environment, copy the file to the `/usr/lib` directory.

```
sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
```

Replace the path of the `libpython3.7m.so.1.0` file as required.

e. Set the soft links:

```
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/local/python3.7.5/bin/python3.7.5
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/local/python3.7.5/bin/pip3.7.5
```

f. Set the Python 3.7.5 environment variables.

i. Run the `vi ~/.bashrc` command in any directory as the installation user to open the `.bashrc` file and append the following line to the file:

```
# Set the Python 3.7.5 library path.
export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
# If multiple Python 3 versions exist in the user environment, specify Python 3.7.5.
export PATH=/usr/local/python3.7.5/bin:$PATH
```

ii. Run the `:wq!` command to save the file and exit.

iii. Run the `source ~/.bashrc` command for the modification to take effect immediately.

g. After the installation is complete, run the following commands to check the installed version. If the required version information is displayed, the installation is successful.

```
python3.7.5 --version
pip3.7.5 --version
python3.7 --version
pip3.7 --version
```

3. Check that the library files that MindStudio depends on have been installed.

xterm is required for MindStudio to display the dependent library files. Run the `dpkg -l xterm| grep xterm| grep ii` command to check whether the library files have been installed. If the following information is displayed, the library files have been installed.

```
xterm          322-1ubuntu1 amd64        X terminal emulator
```

Otherwise, run the following command.

```
sudo apt-get install -y xterm
```

4. Check whether the software for displaying HTML reports is installed for MindStudio.

Use the `firefox --version` and `dpkg -l xdg-utils| grep xdg-utils| grep ii` commands to check whether dependencies Firefox and xdg-utils for displaying HTML reports are installed. If the following information is displayed, the dependencies have been installed.

```
Mozilla Firefox 71.0
xdg-utils      1.1.1-1ubuntu1.16.04.3 all desktop integration utilities from freedesktop.org
```

Otherwise, run the following command to install the software. You can modify the following command to install only some of them as required.

```
sudo apt-get install -y firefox xdg-utils
```
5. Check whether the font software is installed.

Viewing the online help in MindStudio requires the font software. Run the following commands to check whether the font software is installed:

```
dpkg -l fonts-droid-fallback grep fonts-droid-fallback grep ii
dpkg -l fonts-wqy-zenhei grep fonts-wqy-zenhei grep ii
dpkg -l fonts-wqy-microhei grep fonts-wqy-microhei grep ii
dpkg -l fonts-arphic-ukai grep fonts-arphic-ukai grep ii
dpkg -l fonts-arphic-uming grep fonts-arphic-uming grep ii
```

If the following information is displayed, the software has been installed:

- `fonts-droid-fallback` 1:6.0.1r16-1 all          handheld device font with extensive style and language support (fallback)
- `fonts-wqy-zenhei` 0.9.45-6ubuntu1 all          "WenQuanYi Zen Hei" A Hei-Ti Style (sans-serif) Chinese font
- `fonts-wqy-microhei` 0.2.0-beta-2 all          Sans-serif style CJK font derived from Droid font
- `fonts-arphic-ukai` 0.2.20080216.2-4ubuntu2 all          "AR PL UKai" Chinese Unicode TrueType font collection Kaiti style
- `fonts-arphic-uming` 0.2.20080216.2-7ubuntu2 all          "AR PL UMing" Chinese Unicode TrueType font collection Mingti style

Otherwise, run the following command to install the software. You can modify the following command to install only some of them as required.

```
sudo apt-get install -y fonts-droid-fallback fonts-wqy-zenhei fonts-wqy-microhei fonts-arphic-ukai fonts-arphic-uming
```

### Step 2 Install the Python 3 dependencies.

TBE depends on NumPy and decorator. TIK depends on SymPy, CFFI, and GNU readline. The uTST tool depends on the coverage software. Model Accuracy Analyzer depends on protobuf and SciPy. Profiling depends on protobuf, grpcio, grpcio-tools, and requests. If you want to use the operator ST cases to obtain the operator shape and more from the model, you need to install TensorFlow. Otherwise, you do not need to install TensorFlow.

Before the installation, run the `pip3.7.5 list` command to check whether dependencies have been installed. If not, run the following command to install them. You can modify the following command to install only some of them as required.

```
pip3.7.5 install attrs --user
pip3.7.5 install psutil --user
pip3.7.5 install decorator --user
pip3.7.5 install numpy --user
pip3.7.5 install protobuf==3.11.3 --user
pip3.7.5 install scipy --user
pip3.7.5 install sympy --user
pip3.7.5 install cffi --user
pip3.7.5 install gnureadline --user
pip3.7.5 install coverage --user
pip3.7.5 install pylint --user
pip3.7.5 install grpcio --user
pip3.7.5 install grpcio-tools --user
pip3.7.5 install requests --user
pip3.7.5 install xlrd==1.2.0 --user
pip3.7.5 install absl-py --user
pip3.7.5 install pandas --user
pip3.7.5 install pytest --user
```

During the command execution, if the network connection fails and the message "Could not find a version that satisfies the requirement xxx" is displayed, fix the error by referring to What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When "pip3 install" Is Run?

### Step 3 Build and install matplotlib.
1. Run the `wget` command to download the source code package of matplotlib-3.3.3 to any directory of the MindStudio installation server:

```
wget https://files.pythonhosted.org/packages/7b/b3/7c48f648b0f3f39d4385e0169d11b68218b838e185047f7f613bcfc57947/matplotlib-3.3.3.tar.gz
```

2. Go to the download directory and decompress the source code package:

```
tar -zxvf matplotlib-3.3.3.tar.gz
```

3. Go to the decompressed folder and run the following commands:

```
cd matplotlib-3.3.3
echo "[libs]" >> ./setup.cfg
echo "system_freetype=true" >> ./setup.cfg
echo "system_qhull=true" >> ./setup.cfg
python3.7.5 -m pip install .
```

**Step 4** Install the Java Development Kit (JDK).

1. Before installing the JDK, run the following command to check whether Java exists in the system:

```
which java
```

If the related path is returned, Java exists. Otherwise, run the following command to install the JDK:

```
sudo apt-get install -y openjdk-11-jdk
```

2. The installation of MindStudio depends on the `JAVA_HOME` environment variable. Run the following commands for variable setting:

   a. Run the following command in any directory to open the `.bashrc` file:

   ```
   vi ~/.bashrc
   ```

   b. Append the following lines to the file:

   ```
   export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-arm64
   export PATH=$JAVA_HOME/bin:$PATH
   ```

   **NOTE**
   
   `JAVA_HOME` indicates the JDK installation path. If the JDK has been installed, modify `JAVA_HOME` as required. If the JDK has been installed according to the preceding steps, skip this modification.

   c. Save the file and exit:

   ```
   :wq!
   ```

   d. Make the configuration take effect:

   ```
   source ~/.bashrc
   ```

   e. Set the environment variable:

   ```
   echo $JAVA_HOME
   ```

   The command output is as follows:

   ```
   /usr/lib/jvm/java-11-openjdk-arm64
   ```

   f. Check the JDK installation.

   ```
   which jconsole
   ```

   If the command output is as follows, the JDK is successfully installed. Otherwise, the JDK installation fails.

   ```
   /usr/lib/jvm/java-11-openjdk-arm64/bin/jconsole
   ```

**Step 5** (Optional) Install TensorFlow 1.15.0. For details, see *Installing TensorFlow 1.15.0*.

**Step 6** Go back to *2.4.4 Installing MindStudio*.

-----End
Installing TensorFlow 1.15.0

TensorFlow 1.15.0 is necessary to develop and verify operators and develop training services.

Preparations

In the AArch64 architecture, TensorFlow depends on h5py, and h5py depends on HDF5. Therefore, you need to compile and install HDF5 first. Otherwise, an error is reported when you use pip to install h5py. Perform the following operations as the root user:

1. Compile and install HDF5.
   
   a. Run the `wget` command to download the source code package of HDF5 to any directory of the installation environment. The command is as follows:
      ```bash
      wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.5/src/hdf5-1.10.5.tar.gz --no-check-certificate
      ``
   
   b. Go to the download directory and decompress the source code package:
      ```bash
      tar -zxvf hdf5-1.10.5.tar.gz
      ```
   
   c. Go to the new folder and run the following configuration, build, and installation commands:
      ```bash
      cd hdf5-1.10.5/
      ./configure --prefix=/usr/include/hdf5
      make
      make install
      ```
   
   2. Configure environment variables and create a soft link to the dynamic link library (DLL).
      
      a. Set the environment variables:
      ```bash
      export CPATH="/usr/include/hdf5/include/:/usr/include/hdf5/lib/"
      ```
      
      b. Run the following commands as the root user to create a soft link to the DLL. Add `sudo` before the following commands as a non-root user:
      ```bash
      ln -s /usr/include/hdf5/lib/libhdf5.so /usr/lib/libhdf5.so
      ln -s /usr/include/hdf5/lib/libhdf5_hl.so /usr/lib/libhdf5_hl.so
      ```
   
   3. Install h5py.
      
      a. Run the following command as the root user to install the h5py dependency:
      ```bash
      pip3.7 install Cython
      ```
      
      b. Run the following command to install h5py:
      ```bash
      pip3.7 install h5py==2.8.0
      ```
   
   4. Install grpcio.
      
      Run the following command to install grpcio as the root user:
      ```bash
      pip3.7 install grpcio==1.32.0
      ```
   
   5. Install NumPy.
      
      Run the following command to install NumPy as the root user:
      ```bash
      pip3.7 install numpy
      ```

Procedure

1. Download the compiled TensorFlow 1.15 software package:
   ```bash
   tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
   ```

2. Install the compiled TensorFlow.
Go to the directory where the installation package is stored and run the following command as the root user to install TensorFlow 1.15:

```bash
pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
```

Run the following command as a non-root user:

```bash
pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl --user
```

### 15.2.3 EulerOS 2.8-aarch64

--- NOTE ---

- If you install the following dependencies as the root user, delete `sudo` or `--user` from the commands.
- If you install the following dependencies as a non-root user, run the `su - username` command to switch to the MindStudio installation user.

#### Step 1 Check whether the required software is installed.

1. Check whether the Python dependencies and GCC software are installed.

Check whether software including GCC, g++, and Make are installed:

```bash
gcc --version
g++ --version
make --version
cmake --version
rpm -qa | grep automake
rpm -qa | grep kernel-devel
rpm -qa | grep zlib-devel
rpm -qa | grep bzip2-devel
rpm -qa | grep sqlite-devel
rpm -qa | grep openssl-devel
rpm -qa | grep libssl-devel
rpm -qa | grep libffi-devel
rpm -qa | grep gnome-keyring
rpm -qa | grep glib2
rpm -qa | grep dbus-glib-devel
rpm -qa | grep xterm
rpm -qa | grep openssh-clients
rpm -qa | grep openssh-server
rpm -qa | grep ncurses-devel
rpm -qa | grep readline-devel
rpm -qa | grep gdb-gdbserve
rpm -qa | grep atlas
rpm -qa | grep atlas-devel
rpm -qa | grep lapack
rpm -qa | grep lapack-devel
rpm -qa | grep gcc-gfortran
rpm -qa | grep initscripts
rpm -qa | grep freetype-devel
rpm -qa | grep libpng-devel
rpm -qa | grep xdg-utils
```

If the following information is displayed, the software has been installed:

- gcc (GCC) 7.3.0
- g++ (GCC) 7.3.0
- GNU Make 4.2.1
- cmake version 3.12.1
- automake-1.16.1-5.eulerosv2r8.noarch
- kernel-devel-4.19.36-vhulk1907.1.0.h821.eulerosv2r8.aarch64
- zlib-devel-1.2.11-14.h1.eulerosv2r8.aarch64
- bzip2-devel-1.0.6-29.h2.eulerosv2r8.aarch64
- sqlite-devel-3.34.0-2.h19.eulerosv2r8.aarch64
- openssl-devel-1.1.1-3.h11.eulerosv2r8.aarch64
- libssl-devel-1.1.32-3.h6.eulerosv2r8.aarch64
- libffi-devel-3.1-18.h3.eulerosv2r8.aarch64
- gnome-keyring-3.28.2-2.eulerosv2r8.aarch64
- glib2-devel-2.58.1-1.h4.eulerosv2r8.aarch64
Otherwise, run the following command to install the software. You can modify the following command to install only some of them as required.

```bash
```

2. Check whether the Python development environment is installed.

The ATC installation depends on the Python 3 environment. Run the `python3.7.5 --version` and `pip3.7.5 --version` commands to check whether Python 3 is installed. If the following messages are displayed, Python 3 is installed.

```
Python 3.7.5
pip 20.3.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

Otherwise, use the following procedure to install Python 3.7.5:

a. Run the `wget` command to download the source code package of Python 3.7.5 to any directory of the MindStudio installation server:

```bash
wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
```

b. Go to the download directory and decompress the source code package:

```bash
tar -zxvf Python-3.7.5.tgz
```

c. Go to the decompressed folder, create an installation directory, and run the configuration, build, and installation commands:

```bash
cd Python-3.7.5
sudo mkdir /usr/local/python3.7.5
./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --enable-shared
make
sudo make install
```

The `--prefix` parameter specifies the Python installation path. You can change it based on the site requirements. The `--enable-shared` parameter is used to compile the `libpython3.7m.so.1.0` dynamic library. The `--enable-loadable-sqlite-extensions` parameter is used to load sqlite-devel.

This document uses `--prefix=/usr/local/python3.7.5` as an example. After the configuration, build, and installation commands are executed, the installation package is output to the `/usr/local/python3.7.5` directory, and the `libpython3.7m.so.1.0` dynamic library is output to the `/usr/local/python3.7.5/lib/libpython3.7m.so.1.0` directory.

d. Copy the `libpython3.7m.so.1.0` dynamic library.
i. The yum tool strongly depends on the `libpython3.7m.so.1.0` file provided by the system. Therefore, you need to back up the `libpython3.7m.so.1.0` file. Run the following commands:

```bash
cd /usr/lib64
cp libpython3.7m.so.1.0 libpython3.7m.so.1.0.bak
```

If `/usr/lib64` does not exist in the environment, use the actual path.

ii. Copy the compiled file `libpython3.7m.so.1.0` to `/usr/lib64`:

```bash
sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
```

When the following information is displayed, enter `y` to overwrite the `libpython3.7m.so.1.0` file provided by the system.

```bash
cp: overwrite 'libpython3.7m.so.1.0'?
```

If the `/usr/lib64` directory does not exist in the environment, copy the file to the `/usr/lib` directory.

```bash
sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
```

Replace the path of the `libpython3.7m.so.1.0` file as required.

iii. If you need to use the yum tool, run the following command to restore the `libpython3.7m.so.1.0.bak` backup.

```bash
mv libpython3.7m.so.1.0.bak libpython3.7m.so.1.0
```

e. Set the soft links:

```bash
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
sudo ln -s /usr/local/python3.7.5/bin/python3.7.5 /usr/bin/python3.7.5
sudo ln -s /usr/local/python3.7.5/bin/pip3.7.5 /usr/bin/pip3.7.5
```

If a message indicating that the links exist is displayed during the command execution, run the following commands to delete the existing links and run the preceding commands again:

```bash
sudo rm -rf /usr/bin/python3.7
sudo rm -rf /usr/bin/pip3.7
sudo rm -rf /usr/bin/python3.7.5
sudo rm -rf /usr/bin/pip3.7.5
```

f. Set the Python 3.7.5 environment variables.

i. Run the `vi ~/.bashrc` command in any directory as the installation user to open the `.bashrc` file and append the following line to the file:

```bash
# Set the Python 3.7.5 library path.
export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
# If multiple Python 3 versions exist in the user environment, specify Python 3.7.5.
# export PATH=/usr/local/python3.7.5/bin:$PATH
```

ii. Run the `:wq!` command to save the file and exit.

iii. Run the `source ~/.bashrc` command for the modification to take effect immediately.

g. After the installation is complete, run the following commands to check the installed version. If the required version information is displayed, the installation is successful.

```bash
python3.7.5 --version
pip3.7.5 --version
python3.7 --version
pip3.7 --version
```

**Step 2** Install the Python 3 dependencies.

TBE depends on NumPy and decorator. TIK depends on SymPy, CFFI, and GNU readline. The uTST tool depends on the coverage software. Model Accuracy Analyzer depends on protobuf and SciPy. Profiling depends on protobuf, grpcio,
grpcio-tools, and requests. If you want to use the operator ST cases to obtain the operator shape and more from the model, you need to install TensorFlow. Otherwise, you do not need to install TensorFlow.

Before the installation, run the `pip3.7.5 list` command to check whether dependencies have been installed. If not, run the following command to install them. You can modify the following command to install only some of them as required.

```
pip3.7.5 install attrs --user
pip3.7.5 install psutil --user
pip3.7.5 install decorator --user
pip3.7.5 install numpy --user
pip3.7.5 install protobuf==3.11.3 --user
pip3.7.5 install scipy --user
pip3.7.5 install sympy --user
pip3.7.5 install cffi --user
pip3.7.5 install gnureadline --user
pip3.7.5 install coverage --user
pip3.7.5 install pylint --user
pip3.7.5 install grpcio --user
pip3.7.5 install grpcio-tools --user
pip3.7.5 install requests --user
pip3.7.5 install xlrd==1.2.0 --user
```

During the command execution, if the network connection fails and the message "Could not find a version that satisfies the requirement xxx" is displayed, fix the error by referring to [What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When "pip3 install" Is Run?](#)

**Step 3** Compile and install Matplotlib as the `root` user. Common installation users can directly use Matplotlib.

1. Run the `wget` command to download the source code package of matplotlib-3.3.3 to any directory of the MindStudio installation server:
```
wget https://files.pythonhosted.org/packages/7b/b3/7c4f8648bf83f39d4385e01690d1b68218b838e185047f7f613b1cfc57947/matplotlib-3.3.3.tar.gz
```
2. Go to the download directory and decompress the source code package:
```
tar -zxvf matplotlib-3.3.3.tar.gz
```
3. Go to the decompressed directory and run the following commands:
```
cd matplotlib-3.3.3
python3.7.5 setup.py build
python3.7.5 setup.py install
```

**NOTE**

If the `python3.7.5 setup.py build` command fails, perform the following steps:

1. Check whether the `build` directory exists. If the directory does not exist, run the `mkdir build` command to create the directory.
2. Run the `cd build` command to open the `build` directory.
4. Run the `tar -zxvf freetype-2.6.1.tar.gz` command to decompress the installation package.
5. Perform Step 3.3 again.

**Step 4** Install jbr to run MindStudio because Open JDK 11 cannot be installed on Euler using Yum.
1. Log in to https://jetbrains.bintray.com/intellij-jbr/ to download the `jbr-11_0_9-linux-aarch64-b944.49.tar.gz` package.

2. Install jbr.
   - (Solution 1) Decompress the package to a directory, for example, `/home/xxx/jbr`. Set the JAVA_HOME environment variable to `/home/xxx/jbr`.
   - (Solution 2) Decompress the jbr package to the root directory of MindStudio.

**Step 5** (Optional) Install TensorFlow 1.15.0. For details, see Installing TensorFlow 1.15.0.

**Step 6** Go back to 2.4.4 Installing MindStudio.

----End

### Installing TensorFlow 1.15.0

TensorFlow 1.15.0 is necessary to develop and verify operators and develop training services.

**Preparations**

In the AArch64 architecture, TensorFlow depends on h5py, and h5py depends on HDF5. Therefore, you need to compile and install HDF5 first. Otherwise, an error is reported when you use pip to install h5py. Perform the following operations as the root user:

1. Compile and install HDF5.
   a. Run the `wget` command to download the source code package of HDF5 to any directory of the installation environment. The command is as follows:

```plaintext
wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.5/src/
hdf5-1.10.5.tar.gz --no-check-certificate
```

   b. Go to the download directory and decompress the source code package:

```plaintext
tar -zxvf hdf5-1.10.5.tar.gz
```

   c. Go to the new folder and run the following configuration, build, and installation commands:

```plaintext
cd hdf5-1.10.5/
./configure --prefix=/usr/include/hdf5
make
make install
```

2. Configure environment variables and create a soft link to the dynamic link library (DLL).
   a. Set the environment variables:

```plaintext
export CPATH="/usr/include/hdf5/include:/usr/include/hdf5/lib/"
```

   b. Run the following commands as the root user to create a soft link to the DLL. Add `sudo` before the following commands as a non-root user:

```plaintext
ln -s /usr/include/hdf5/lib/libhdf5.so /usr/lib/libhdf5.so
ln -s /usr/include/hdf5/lib/libhdf5_hl.so /usr/lib/libhdf5_hl.so
```

3. Install h5py.
   a. Run the following command as the root user to install the h5py dependency:

```plaintext
pip3.7 install Cython
```
b. Run the following command to install h5py:
   ```bash
   pip3.7 install h5py==2.8.0
   ```

4. Install grpcio.
   Run the following command to install grpcio as the **root** user:
   ```bash
   pip3.7 install grpcio==1.32.0
   ```

5. Install NumPy.
   Run the following command to install NumPy as the **root** user:
   ```bash
   pip3.7 install numpy
   ```

**Procedure**

1. Download the compiled TensorFlow 1.15 software package:
   ```
   tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
   ```

2. Install the compiled TensorFlow.
   Go to the directory where the installation package is stored and run the following command as the **root** user to install TensorFlow 1.15:
   ```bash
   pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
   ```
   Run the following command as a non-root user:
   ```bash
   pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl --user
   ```

### 15.2.4 CentOS7.6-x86_64

**NOTE**

- If you install the following dependencies as the **root** user, delete `sudo` or `--user` from the commands.
- If you install the following dependencies as a non-root user, run the `su - username` command to switch to the MindStudio installation user.

**Step 1** Check whether the required dependent software is installed.

1. Check whether the Python dependencies and GCC software are installed.
   Run the following commands to check whether software including GCC, g++, and Make are installed:
   ```bash
   gcc --version
   g++ --version
   make --version
   cmake --version
   rpm -qa | grep automake
   rpm -qa | grep kernel-devel
   rpm -qa | grep zlib-devel
   rpm -qa | grep bzip2
   rpm -qa | grep bzip2-devel
   rpm -qa | grep sqlite-devel
   rpm -qa | grep openssl
   rpm -qa | grep openssl-devel
   rpm -qa | grep libxml2-devel
   rpm -qa | grep libffi-devel
   rpm -qa | grep gnome-keyring
   rpm -qa | grep glib2
   rpm -qa | grep dbus-glib-devel
   rpm -qa | grep xterm
   rpm -qa | grep openssh-clients
   rpm -qa | grep openssh-server
   rpm -qa | grep ncurses-devel
   rpm -qa | grep readline-devel
   rpm -qa | grep gdb-gdbserver
   rpm -qa | grep atlas
   ```
If the following information is displayed, the software has been installed:

gcc (GCC) 7.3.0
+ (GCC) 7.3.0
GNU Make 3.82
cmake version 3.6.2
automake-1.13.4-3.el7.noarch
kernel-devel-3.10.0-1160.11.1.el7.x86_64
zlib-devel-1.2.7-18.el7.x86_64
bz2-1.0.6-13.el7.x86_64
bz2-1.0.6-13.el7.x86_64
bz2-1.0.6-13.el7.x86_64
bz2-1.0.6-13.el7.x86_64
sqlite-devel-3.7.17-8.el7_7.1.x86_64
openssl-devel-1.0.2k-21.el7.x86_64
openssl-devel-1.0.2k-21.el7.x86_64
openssl-devel-1.0.2k-21.el7.x86_64
openssl-devel-1.0.2k-21.el7.x86_64
openssl-devel-1.0.2k-21.el7.x86_64
libxml2-devel-1.1.28-6.el7.x86_64
libffi-devel-3.0.13-19.el7.x86_64
gnome-keyring-3.28.2-1.el7.x86_64
glib2-devel-2.56.1-8.el7.x86_64
glib2-2.56.1-8.el7.x86_64
dbus-glib-devel-0.100-7.el7.x86_64
xterm-295-3.el7.x86_64
openssh-clients-7.4p1-21.el7.x86_64
openssh-server-7.4p1-21.el7.x86_64
ncurses-devel-5.9-14.20130511.el7_4.x86_64
readline-devel-6.2-11.el7.x86_64
gdb-gdbserver-7.6.1-8.el7.x86_64
gdbserver-7.6.1-8.el7.x86_64
gcc-gfortran-4.8.5-44.el7.x86_64
initscripts-noarch-9.49.53-1.el7_9.1.x86_64
freetype-devel-2.8-14.el7_9.1.x86_64
libpng-devel-1.5.13-8.el7.x86_64
xdg-utils-1.1.0-0.17.20120809git.el7.noarch
firefox-78.6.1-1.el7.centos.x86_64
unzip-6.0-21.el7.x86_64
pcre2-8.38-12.el7.x86_64
pcre2-8.38-12.el7.x86_64
net-tools-2.0-0.25.20131004git.el7.x86_64
blas-devel-3.4.2-8.el7.x86_64
blas-devel-3.4.2-8.el7.x86_64
blas-devel-3.4.2-8.el7.x86_64
openssl-askpass-7.4p1-21.el7.x86_64
texinfo-5.1-5.el7.x86_64

Otherwise, run the following command to install the software. You can modify the following command to install only some of them as required.
2. Check whether the Python development environment is installed.

The ATC installation depends on the Python 3 environment. Run the `python3.7.5 --version` and `pip3.7.5 --version` commands to check whether Python 3 is installed. If the following messages are displayed, Python 3 is installed.

Python 3.7.5

```
pip 20.3.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

Otherwise, install Python 3.7.5 as follows:

a. Run the `wget` command to download the source code package of Python 3.7.5 to any directory of the MindStudio installation server:

```
wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
```

b. Go to the download directory and decompress the source code package:

```
tar -zxvf Python-3.7.5.tgz
```

c. Go to the decompressed folder, create an installation directory, and run the configuration, build, and installation commands.

```
cd Python-3.7.5
sudo mkdir /usr/local/python3.7.5
./configure --prefix=/usr/local/python3.7.5
--enable-loadable-sqlite-extensions --enable-shared
make
```

```
sudo make install
```

The `--prefix` option specifies the Python installation path. You can change it as required. The `--enable-shared` parameter is used to compile the `libpython3.7m.so.1.0` dynamic library. The `--enable-loadable-sqlite-extensions` parameter is used to load sqlite-devel.

This document uses `--prefix=/usr/local/python3.7.5` as an example. After the configuration, build, and installation commands are executed, the installation package is output to the `/usr/local/python3.7.5` directory, and the `libpython3.7m.so.1.0` dynamic library is output to the `/usr/local/python3.7.5/lib/libpython3.7m.so.1.0` directory.

d. Run the following command to copy the `libpython3.7m.so.1.0` dynamic library:

i. Copy the built file `libpython3.7m.so.1.0` to `/usr/lib64`:

```
sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
```

When the following information is displayed, enter `y` to overwrite the `libpython3.7m.so.1.0` file provided by the system.

```
cp: overwrite 'libpython3.7m.so.1.0'?
```

If the `/usr/lib64` directory does not exist in the environment, copy the file to the `/usr/lib` directory.

```
sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
```

Replace the path of the `libpython3.7m.so.1.0` file as required.
If this step fails, the `yum` command is unavailable. Run the following commands to rectify the fault:

1. `sed -i 's/#!\usr/bin/python/#!\usr/bin/python2.7/g' /usr/bin/yum`
2. `sed -i 's/#!\usr/bin/python/#!\usr/bin/python2.7/g' /usr/libexec/urlgrabber-ext-down`
3. `sed -i 's/#!\usr/bin/python/#!\usr/bin/python2.7/g' /usr/bin/yum-config-manager`

- Set the soft links:
  ```
  sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
  sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
  sudo ln -s /usr/local/python3.7.5/bin/python3.7.5 /usr/bin/python3.7.5
  sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
  ```

  If a message indicating that the links exist is displayed during the command execution, run the following commands to delete the existing links and run the preceding commands again:
  ```
  sudo rm -rf /usr/bin/python3.7
  sudo rm -rf /usr/bin/pip3.7
  sudo rm -rf /usr/bin/python3.7.5
  sudo rm -rf /usr/bin/pip3.7.5
  ```

- Set the Python 3.7.5 environment variables.
  ```
  sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3
  sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3
  ```

  Run the `vi ~/.bashrc` command in any directory as the installation user to open the `.bashrc` file and append the following line to the file:
  ```
  # Set the Python 3.7.5 library path.
  export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
  # If multiple Python 3 versions exist in the user environment, specify Python 3.7.5.
  export PATH=/usr/local/python3.7.5/bin:$PATH
  ```

  Run the `:wq!` command to save the file and exit.

  Run the `source ~/.bashrc` command for the modification to take effect immediately.

- After the installation is complete, run the following commands to check the installed version. If the required version information is displayed, the installation is successful.
  ```
  python3.7.5 --version
  pip3.7.5 --version
  python3.7 --version
  pip3.7 --version
  ```

### Step 2 Install the Python 3 dependencies.

TBE depends on NumPy and decorator. TIK depends on SymPy, CFFI, and GNU readline. The uTST tool depends on the coverage software. Model Accuracy Analyzer depends on protobuf and SciPy. Profiling depends on protobuf, grpcio, grpcio-tools, and requests. If you want to use the operator ST cases to obtain the operator shape and more from the model, you need to install TensorFlow. Otherwise, you do not need to install TensorFlow.

Before the installation, run the `pip3.7.5 list` command to check whether dependencies have been installed. If not, run the following command to install them. You can modify the following command to install only some of them as required.

```bash
pip3.7.5 install --upgrade pip --user
pip3.7.5 install attrs --user
```
<table>
<thead>
<tr>
<th>Command</th>
<th>Installer</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>pip3.7.5 install</td>
<td>psutil</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>decorator</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>numpy</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>protobuf==3.11.3</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>scipy</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>gnureadline</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>coverage</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>pylint</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>grpcio</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>grpcio-tools</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>requests</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>xlrd==1.2.0</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>absl-py</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>pytest</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>wheel</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>pyyaml</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>pathlib2</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>sympy==1.4.0</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>pandas==1.1.5</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>pillow==8.0.1</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>cffi==1.12.3</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>pyparsing&gt;=2.0.3</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>matplotlib</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>xlwt</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>xlutils</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>Cython</td>
<td>--user</td>
</tr>
<tr>
<td>pip3.7.5 install</td>
<td>tensorflow==1.15.0</td>
<td>--user</td>
</tr>
</tbody>
</table>

During the command execution, if the network connection fails and the message "Could not find a version that satisfies the requirement xxx" is displayed, fix the error by referring to **What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When "pip3 install" Is Run?**

### Step 3 Install the JDK.

1. **Before installing the JDK, run the following command to check whether Java exists in the system:**

   ```bash
   which java
   ```

   If the related path is returned, Java exists. Otherwise, run the following command to install the JDK:

   ```bash
   sudo yum install -y java-11-openjdk
   ```

2. **The installation of MindStudio depends on the JAVA_HOME environment variable. Run the following commands for variable setting:**

   a. **Run the following command in any directory to open the .bashrc file:**

   ```bash
   vi ~/.bashrc
   ```

   b. **Append the following lines to the file:**

   ```bash
   export JAVA_HOME=/usr/lib/jvm/java-11-openjdk
   export PATH=$JAVA_HOME/bin:$PATH
   ```

   **NOTE**

   JAVA_HOME indicates the JDK installation path. If the JDK has been installed, modify JAVA_HOME as required. If the JDK has been installed according to the preceding steps, skip this modification.

   c. **Save the file and exit:**

   ```bash
   :wq!
   ```

   d. **Make the configuration take effect:**

   ```bash
   source ~/.bashrc
   ```

   e. **Set the environment variable:**

   ```bash
   echo $JAVA_HOME
   ```

   The command output is as follows:
f. Check the JDK installation.

Check the JDK installation.

```bash
which jconsole
```

If the command output is as follows, the JDK is successfully installed.

Otherwise, the JDK installation fails.

```bash
/usr/lib/jvm/java-11-openjdk-arm64/bin/jconsole
```

**Step 4** Go back to 2.4.4 Installing MindStudio.

-----End

### 15.2.5 CentOS7.6-AArch64

**NOTE**

- If you install the following dependencies as the root user, delete sudo or --user from the commands.
- If you install the following dependencies as a non-root user, run the su - username command to switch to the MindStudio installation user.

**Step 1** Check whether the required software is installed.

1. Check whether the Python dependencies and GCC software are installed.

Run the following commands to check whether software including GCC, g++, and Make are installed:

```bash
gcc --version
g++ --version
make --version
cmake --version
rpm -qa | grep automake
rpm -qa | grep kernel-devel
rpm -qa | grep zlib-devel
rpm -qa | grep bzip2
rpm -qa | grep bzip2-devel
rpm -qa | grep sqlite-devel
rpm -qa | grep openssl
rpm -qa | grep openssl-devel
rpm -qa | grep libxslt-devel
rpm -qa | grep libfp-devel
rpm -qa | grep gnome-keyring
rpm -qa | grep glib2
rpm -qa | grep dbus-glib-devel
rpm -qa | grep xterm
rpm -qa | grep openssh-clients
rpm -qa | grep openssh-server
rpm -qa | grep ncurses-devel
rpm -qa | grep readline-devel
rpm -qa | grep gdb-gdbserver
rpm -qa | grep atlas
rpm -qa | grep atlas-devel
rpm -qa | grep lapack
rpm -qa | grep lapack-devel
rpm -qa | grep gcc-gfortran
rpm -qa | grep initscripts
rpm -qa | grep freetype-devel
rpm -qa | grep libpng-devel
rpm -qa | grep xdg-utils
rpm -qa | grep firefox
rpm -qa | grep unzip
rpm -qa | grep pcutils
rpm -qa | grep net-tools
rpm -qa | grep blas
rpm -qa | grep blas-devel
rpm -qa | grep openssh-askpass
rpm -qa | grep texinfo
```
If the following information is displayed, the software has been installed:

gcc (GCC) 7.3.0

g++ (GCC) 7.3.0

GNU Make 3.82

cmake version 3.6.2

automake-1.13.4-3.el7.noarch

kernel-devel-4.18.0-193.28.1.el7.aarch64

zlib-devel-1.2.7-18.el7.aarch64

bzip2p-1.0.6-13.el7.aarch64

bzip2-1.0.6-13.el7.aarch64

bzip2-devel-1.0.6-13.el7.aarch64

sqlite-devel-3.7.17-8.el7.7.1.aarch64

openssl-devel-1.0.2k-21.el7_9.aarch64

openssl-1.0.2k-21.el7_9.aarch64

openssl-libs-1.0.2k-21.el7_9.aarch64

openssl-devel-1.0.2k-21.el7_9.aarch64

libxslt-devel-1.1.28-6.el7.aarch64

libffi-devel-3.0.13-19.el7.aarch64

gnome-keyring-3.28.2-1.el7.aarch64

glib2-devel-2.56.1-8.el7.aarch64

glib2-2.56.1-8.el7.aarch64

dbus-glib-devel-0.100-7.el7.aarch64

term-295-3.el7.aarch64

openssl-clients-7.4p1-21.el7.aarch64

openssl-server-7.4p1-21.el7.aarch64

ncurses-devel-5.9-14.20130511.el7_4.aarch64

readline-devel-6.2-11.el7.aarch64

gdb-gdbserver-7.6.1-120.el7.aarch64

atlas-devel-3.10.1-12.el7.aarch64

atlas-3.10.1-12.el7.aarch64

lapack-devel-3.4.2-8.el7.aarch64

lapack-3.4.2-8.el7.aarch64

lapack-devel-3.4.2-8.el7.aarch64

gcc-gfortran-4.8.5-44.el7.aarch64

initscripts-9.49.53-1.el7_9.1.aarch64

libgcc-2.8-14.el7_9.1.aarch64

libgcc-2.8-14.el7_9.1.aarch64

libgfortran-4.8.5-44.el7.aarch64

libming-1.5.13-8.el7.aarch64

libdjpeg-dev-1.4.0b-8.el7.aarch64

libjpeg-8d-8.el7.aarch64

libpng-devel-1.5.13-8.el7.aarch64

libprotobuf-2.6.1-1.el7.centos.aarch64

unzip-6.0-21.el7.aarch64

pcutils-libts-3.5.1-3.1.el7.aarch64

pcutils-3.5.1-3.1.el7.aarch64

net-tools-2.0-0.25.20131004git.el7.aarch64

blas-devel-3.4.2-8.el7.aarch64

blas-3.4.2-8.el7.aarch64

blas-devel-3.4.2-8.el7.aarch64

openssl-askpass-7.4p1-21.el7.aarch64

texinfo-5.1.5-5.el7.aarch64

Otherwise, run the following command to install the software. You can modify the following command to install only some of them as required.

```
```

2. Check whether the Python development environment is installed.

The ATC installation depends on the Python 3 environment. Run the `python3.7.5 --version` and `pip3.7.5 --version` commands to check whether Python 3 is installed. If the following messages are displayed, Python 3 is installed.

```
Python 3.7.5
pip 20.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

Otherwise, install Python 3.7.5 as follows:
a. Run the `wget` command to download the source code package of Python 3.7.5 to any directory of the MindStudio installation server:
```
wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
```

b. Go to the download directory and decompress the source code package:
```
tar -zxvf Python-3.7.5.tgz
```

c. Go to the decompressed folder, create an installation directory, and run the configuration, build, and installation commands.
```
cd Python-3.7.5
sudo mkdir /usr/local/python3.7.5
./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --enable-shared
make
sudo make install
```
The `--prefix` option specifies the Python installation path. You can change it as required. The `--enable-shared` parameter is used to compile the `libpython3.7m.so.1.0` dynamic library. The `--enable-loadable-sqlite-extensions` parameter is used to load sqlite-devel.

This document uses `--prefix=/usr/local/python3.7.5` as an example. After the configuration, build, and installation commands are executed, the installation package is output to the `/usr/local/python3.7.5` directory, and the `libpython3.7m.so.1.0` dynamic library is output to the `/usr/local/python3.7.5/lib/libpython3.7m.so.1.0` directory.

d. Run the following command to copy the `libpython3.7m.so.1.0` dynamic library:
```
i. Copy the built file `libpython3.7m.so.1.0` to `/usr/lib64`:
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
   When the following information is displayed, enter `y` to overwrite the `libpython3.7m.so.1.0` file provided by the system.
   
   cp: overwrite `libpython3.7m.so.1.0`?
   
   If the `/usr/lib64` directory does not exist in the environment, copy the file to the `/usr/lib` directory.
   
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
   Replace the path of the `libpython3.7m.so.1.0` file as required.
```

![NOTE]

If this step fails, the `yum` command is unavailable. Run the following commands to rectify the fault:
```
1. sed -i 's/#!/usr/bin/python/#!/usr/bin/python2.7/g' /usr/bin/yum
2. sed -i 's/#! /usr/bin/python/#!/usr/bin/python2.7/g' /usr/libexec/urlgrabber-ext-down
3. sed -i 's/#!/usr/bin/python/#!/usr/bin/python2.7/g' /usr/bin/yum-config-manager
```

e. Set the soft links:
```
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
```

If a message indicating that the links exist is displayed during the command execution, run the following commands to delete the existing links and run the preceding commands again:
```
sudo rm -rf /usr/bin/python3.7
sudo rm -rf /usr/bin/pip3.7
```
f. Set the Python 3.7.5 environment variables.
   i. Run the `vi ~/.bashrc` command in any directory as the installation user to open the `.bashrc` file and append the following line to the file:
      ```
      # Set the Python 3.7.5 library path.
      export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
      # If multiple Python 3 versions exist in the user environment, specify Python 3.7.5.
      export PATH=/usr/local/python3.7.5/bin:$PATH
      ```
   ii. Run the `.wq!` command to save the file and exit.
   iii. Run the `source ~/.bashrc` command for the modification to take effect immediately.

g. After the installation is complete, run the following commands to check the installed version. If the required version information is displayed, the installation is successful.
   ```
   python3.7.5 --version
   pip3.7.5 --version
   python3.7 --version
   pip3.7 --version
   ```

Step 2 Install the Python 3 dependencies.

TBE depends on NumPy and decorator. TIK depends on SymPy, CFFI, and GNU readline. The uTST tool depends on the coverage software. Model Accuracy Analyzer depends on protobuf and SciPy. Profiling depends on protobuf, grpcio, grpcio-tools, and requests. If you want to use the operator ST cases to obtain the operator shape and more from the model, you need to install TensorFlow. Otherwise, you do not need to install TensorFlow.

Before the installation, run the `pip3.7.5 list` command to check whether dependencies have been installed. If not, run the following command to install them. You can modify the following command to install only some of them as required.

   ```
   pip3.7.5 install --upgrade pip --user
   pip3.7.5 install attrs --user
   pip3.7.5 install psutil --user
   pip3.7.5 install decorator --user
   pip3.7.5 install numpy --user
   pip3.7.5 install protobuf==3.11.3 --user
   pip3.7.5 install scipy --user
   pip3.7.5 install gnureadline --user
   pip3.7.5 install coverage --user
   pip3.7.5 install pyLint --user
   pip3.7.5 install grpcio --user
   pip3.7.5 install grpcio-tools --user
   pip3.7.5 install requests --user
   pip3.7.5 install xld==1.2.0 --user
   pip3.7.5 install absl-py --user
   pip3.7.5 install pytest --user
   pip3.7.5 install wheel --user
   pip3.7.5 install pyyaml --user
   pip3.7.5 install pathlib2 --user
   pip3.7.5 install sympy==1.4.0 --user
   pip3.7.5 install pandas==1.1.5 --user
   pip3.7.5 install pillow==8.0.1 --user
   pip3.7.5 install cffi==1.12.3 --user
   pip3.7.5 install pypparsing==2.0.3 --user
   pip3.7.5 install xlwt --user
   pip3.7.5 install xlutils --user
   pip3.7.5 install Cython --user
   ```
During the command execution, if the network connection fails and the message "Could not find a version that satisfies the requirement xxx" is displayed, fix the error by referring to What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When "pip3 install" Is Run?

**Step 3** Install Matplotlib in the user directory.

```
pip3.7.5 install matplotlib -i https://mirrors.aliyun.com/pypi/simple/ --user
```

**Step 4** Install the JDK.

1. Before installing the JDK, run the following command to check whether Java exists in the system:

   ```
   which java
   ```

   If the related path is returned, Java exists. Otherwise, run the following command to install the JDK:

   ```
   sudo yum install -y java-11-openjdk
   ```

2. The installation of MindStudio depends on the JAVA_HOME environment variable. Run the following commands for variable setting:

   a. Run the following command in any directory to open the .bashrc file:

   ```
   vi ~/.bashrc
   ```

   b. Append the following lines to the file:

   ```
   export JAVA_HOME=/usr/lib/jvm/java-11-openjdk
   export PATH=$JAVA_HOME/bin:$PATH
   ```

   **NOTE**

   JAVA_HOME indicates the JDK installation path. If the JDK has been installed, modify JAVA_HOME as required. If the JDK has been installed according to the preceding steps, skip this modification.

   c. Save the file and exit:

   ```
   :wq!
   ```

   d. Make the configuration take effect:

   ```
   source ~/.bashrc
   ```

   e. Set the environment variable:

   ```
   echo $JAVA_HOME
   ```

   The command output is as follows:

   ```
   /usr/lib/jvm/java-11-openjdk
   ```

   f. Check the JDK installation.

   ```
   which jconsole
   ```

   If the command output is as follows, the JDK is successfully installed. Otherwise, the JDK installation fails.

   ```
   /usr/lib/jvm/java-11-openjdk-arm64/bin/jconsole
   ```

**Step 5** (Optional) Install TensorFlow 1.15.0. For details, see Installing TensorFlow 1.15.0.

**Step 6** Go back to 2.4.4 Installing MindStudio.

----End

**Installing TensorFlow 1.15.0**

TensorFlow 1.15.0 is necessary to develop and verify operators and develop training services.

**Preparations**
In the AArch64 architecture, TensorFlow depends on h5py, and h5py depends on HDF5. Therefore, you need to compile and install HDF5 first. Otherwise, an error is reported when you use pip to install h5py. Perform the following operations as the root user:

1. Compile and install HDF5.
   a. Run the **wget** command to download the source code package of HDF5 to any directory of the installation environment. The command is as follows:
      
      ```bash
      wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.5/src/
      hdf5-1.10.5.tar.gz --no-check-certificate
      ```
   
   b. Go to the download directory and decompress the source code package:
      
      ```bash
      tar -zxvf hdf5-1.10.5.tar.gz
      ```
   
   c. Go to the new folder and run the following configuration, build, and installation commands:
      
      ```bash
      cd hdf5-1.10.5/
      ./configure --prefix=/usr/include/hdf5
      make
      make install
      ```

2. Configure environment variables and create a soft link to the dynamic link library (DLL).
   a. Set the environment variables:
      
      ```bash
      export CPATH="/usr/include/hdf5/include/:/usr/include/hdf5/lib/"
      ```
   
   b. Run the following commands as the root user to create a soft link to the DLL. Add **sudo** before the following commands as a non-root user:
      
      ```bash
      ln -s /usr/include/hdf5/lib/libhdf5.so /usr/lib/libhdf5.so
      ln -s /usr/include/hdf5/lib/libhdf5_hl.so /usr/lib/libhdf5_hl.so
      ```

3. Install h5py.
   a. Run the following command as the root user to install the h5py dependency:
      
      ```bash
      pip3.7 install Cython
      ```
   
   b. Run the following command to install h5py:
      
      ```bash
      pip3.7 install h5py==2.8.0
      ```

4. Install grpcio.
   Run the following command to install grpcio as the root user:
   
   ```bash
   pip3.7 install grpcio==1.32.0
   ```

5. Install NumPy.
   Run the following command to install NumPy as the root user:
   
   ```bash
   pip3.7 install numpy
   ```

**Procedure**

1. Download the compiled TensorFlow 1.15 software package:
   
   ```bash
   tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
   ```

2. Install the compiled TensorFlow.
   Go to the directory where the installation package is stored and run the following command as the root user to install TensorFlow 1.15:
   
   ```bash
   pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
   ```
   
   Run the following command as a non-root user:
   
   ```bash
   pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl --user
   ```
15.2.6 Kylin OS V10 SP1-aarch64

**NOTE**

- If you install the following dependencies as the root user, delete `sudo` or `--user` from the commands.
- If you install the following dependencies as a non-root user, run the `su - username` command to switch to the MindStudio installation user.

**Step 1** Check whether the required dependent software is installed.

1. Check whether the Python dependencies and GCC software are installed. Run the following commands to check whether software including GCC, g++, and Make are installed:

```
gcc --version
g++ --version
make --version
cmake --version
rpm -qa | grep automake
rpm -qa | grep kernel-devel
rpm -qa | grep zlib-devel
rpm -qa | grep bzip2
rpm -qa | grep bzip2-devel
rpm -qa | grep sqlite-devel
rpm -qa | grep openssl
rpm -qa | grep openssl-devel
rpm -qa | grep libxml2-devel
rpm -qa | grep libffi-devel
rpm -qa | grep gnome-keyring
rpm -qa | grep glib2
rpm -qa | grep dbus-glib-devel
rpm -qa | grep xterm
rpm -qa | grep ncurses-devel
rpm -qa | grep readline-devel
rpm -qa | grep gdb-gdbserver
rpm -qa | grep atlas
rpm -qa | grep atlas-devel
rpm -qa | grep lapack
rpm -qa | grep lapack-devel
rpm -qa | grep gcc
rpm -qa | grep gcc-gfortran
rpm -qa | grep initscripts
rpm -qa | grep freetype-devel
rpm -qa | grep libpng-devel
rpm -qa | grep xorg-x11-utils
rpm -qa | grep firefox
rpm -qa | grep unzip
rpm -qa | grep pcre
rpm -qa | grep net-tools
rpm -qa | grep blas
rpm -qa | grep blas-devel
rpm -qa | grep texinfo

If the following information is displayed, the software has been installed:
```
gcc (GCC) 7.3.0
g++ (GCC) 7.3.0
GNU Make 4.2.1
cmake version 3.12.1
automake-1.16.1-6.ky10.noarch
kernel-devel-4.19.90-17.ky10.aarch64
zlib-devel-1.2.11-17.1.ky10.aarch64
bzip2-1.0.8-3.ky10.aarch64
bzip2-devel-1.0.8-3.ky10.aarch64
sqlite-devel-3.24.0-9.ky10.aarch64
openssl-1.1.1a-9.ky10.aarch64
openssl-devel-1.1.1a-9.ky10.aarch64
openssl-1.1.1a-9.ky10.aarch64
```

Copyright © Huawei Technologies Co., Ltd.

Issue 01 (2021-06-24)
OpenSSL is used to create a software package distribution file.
This document uses `--prefix=/usr/local/python3.7.5` as an example. After the configuration, build, and installation commands are executed, the installation package is output to the `/usr/local/python3.7.5` directory, and the `libpython3.7m.so.1.0` dynamic library is output to the `/usr/local/python3.7.5/lib/libpython3.7m.so.1.0` directory.

d. Run the following command to copy the `libpython3.7m.so.1.0` dynamic library:

   i. Copy the built file `libpython3.7m.so.1.0` to `/usr/lib64`:

```
sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
```

   When the following information is displayed, enter `y` to overwrite the `libpython3.7m.so.1.0` file provided by the system.

   ```
cp: overwrite 'libpython3.7m.so.1.0'?
```

   If the `/usr/lib64` directory does not exist in the environment, copy the file to the `/usr/lib` directory.

```
sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
```

   Replace the path of the `libpython3.7m.so.1.0` file as required.

   **NOTE**

   If this step fails, the `yum` command is unavailable. Run the following commands to rectify the fault:

   1. ```
   sed -i 's/#!/usr/bin/python/#!/usr/bin/python2.7/g' /usr/bin/yum
   ```

   2. ```
   sed -i 's/#! /usr/bin/python/#! /usr/bin/python2.7/g' /usr/libexec/urlgrabber-ext-down
   ```

   3. ```
   sed -i 's/#!/usr/bin/python/#!/usr/bin/python2.7/g' /usr/bin/yum-config-manager
   ```

  e. Set the soft links:

```
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
```

   If a message indicating that the links exist is displayed during the command execution, run the following commands to delete the existing links and run the preceding commands again:

```
sudo rm -rf /usr/bin/python3.7
sudo rm -rf /usr/bin/pip3.7
sudo rm -rf /usr/bin/python3.7.5
sudo rm -rf /usr/bin/pip3.7.5
```

f. Set the Python 3.7.5 environment variables.

   i. Run the `vi ~/.bashrc` command in any directory as the installation user to open the `.bashrc` file and append the following line to the file:

```
# Set the Python 3.7.5 library path.
export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
# If multiple Python 3 versions exist in the user environment, specify Python 3.7.5.
export PATH=/usr/local/python3.7.5/bin:$PATH
```

   ii. Run the `.wq!` command to save the file and exit.

   iii. Run the `source ~/.bashrc` command for the modification to take effect immediately.

   g. After the installation is complete, run the following commands to check the installed version. If the required version information is displayed, the installation is successful.
Step 2 Install the Python 3 dependencies.

TBE depends on NumPy and decorator. TIK depends on SymPy, CFFI, and GNU readline. The uTST tool depends on the coverage software. Model Accuracy Analyzer depends on protobuf and SciPy. Profiling depends on protobuf, grpcio, grpcio-tools, and requests. If you want to use the operator ST cases to obtain the operator shape and more from the model, you need to install TensorFlow. Otherwise, you do not need to install TensorFlow.

Before the installation, run the `pip3.7.5 list` command to check whether dependencies have been installed. If not, run the following command to install them. You can modify the following command to install only some of them as required.

```
pip3.7.5 install --upgrade pip --user
pip3.7.5 install attrs --user
pip3.7.5 install psutil --user
pip3.7.5 install decorator --user
pip3.7.5 install numpy --user
pip3.7.5 install protobuf==3.11.3 --user
pip3.7.5 install scipy --user
pip3.7.5 install gnureadline --user
pip3.7.5 install coverage --user
pip3.7.5 install pylint --user
pip3.7.5 install grpcio --user
pip3.7.5 install grpcio-tools --user
pip3.7.5 install requests --user
pip3.7.5 install xlrd==1.2.0 --user
pip3.7.5 install absl-py --user
pip3.7.5 install pytest --user
pip3.7.5 install wheel --user
pip3.7.5 install pyyaml --user
pip3.7.5 install pathlib2 --user
pip3.7.5 install sympy==1.4.0 --user
pip3.7.5 install pandas==1.1.5 --user
pip3.7.5 install pillow==8.0.1 --user
pip3.7.5 install cffi==1.12.3 --user
pip3.7.5 install pyparsing>=2.0.3 --user
pip3.7.5 install xlwt --user
pip3.7.5 install xlutils --user
pip3.7.5 install Cython --user
```

During the command execution, if the network connection fails and the message "Could not find a version that satisfies the requirement xxx" is displayed, fix the error by referring to What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When "pip3 install" Is Run?

Step 3 Install Matplotlib in the user directory.

```
pip3.7.5 install matplotlib -i https://mirrors.aliyun.com/pypi/simple/ --user
```

Step 4 Install the JDK.

1. Before installing the JDK, run the following command to check whether Java exists in the system:

   ```
   which java
   ```

   If the related path is returned, Java exists. Otherwise, run the following command to install the JDK:

   ```
   sudo yum install -y java-11-openjdk
   ```

2. The installation of MindStudio depends on the JAVA_HOME environment variable. Run the following commands for variable setting:
a. Run the following command in any directory to open the .bashrc file:
   
   vi ~/.bashrc

b. Append the following lines to the file:
   
   export JAVA_HOME=/usr/lib/jvm/java-11-openjdk
   export PATH=$JAVA_HOME/bin:$PATH

   **NOTE**
   
   JAVA_HOME indicates the JDK installation path. If the JDK has been installed, modify JAVA_HOME as required. If the JDK has been installed according to the preceding steps, skip this modification.

c. Save the file and exit:
   
   :wq!

d. Make the configuration take effect:
   
   source ~/.bashrc

e. Set the environment variable:
   
   echo $JAVA_HOME
   
   The command output is as follows:
   
   /usr/lib/jvm/java-11-openjdk

f. Check the JDK installation.
   
   which jconsole
   
   If the command output is as follows, the JDK is successfully installed. Otherwise, the JDK installation fails.
   
   /usr/lib/jvm/java-11-openjdk-arm64/bin/jconsole

**Step 5** (Optional) Install TensorFlow 1.15.0. For details, see [Installing TensorFlow 1.15.0](#).

**Step 6** Go back to [2.4.4 Installing MindStudio](#).

---End

### Installing TensorFlow 1.15.0

TensorFlow 1.15.0 is necessary to develop and verify operators and develop training services.

**Preparations**

In the AArch64 architecture, TensorFlow depends on h5py, and h5py depends on HDF5. Therefore, you need to compile and install HDF5 first. Otherwise, an error is reported when you use pip to install h5py. Perform the following operations as the root user:

1. Compile and install HDF5.
   
   a. Run the wget command to download the source code package of HDF5 to any directory of the installation environment. The command is as follows:
      
      wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.5/src/
      
      b. Go to the download directory and decompress the source code package:
         
         tar -zxvf hdf5-1.10.5.tar.gz
         
      c. Go to the new folder and run the following configuration, build, and installation commands:
         
         cd hdf5-1.10.5/
         
         configure --prefix=/usr/include/hdf5
         
         make
         
         make install
2. Configure environment variables and create a soft link to the dynamic link library (DLL).
   a. Set the environment variables:
      ```bash
      export CPATH="/usr/include/hdf5/include/:/usr/include/hdf5/lib/
      ```
   b. Run the following commands as the **root** user to create a soft link to the DLL.
      Add **sudo** before the following commands as a non-root user:
      ```bash
      ln -s /usr/include/hdf5/lib/libhdf5.so /usr/lib/libhdf5.so
      ln -s /usr/include/hdf5/lib/libhdf5_hl.so /usr/lib/libhdf5_hl.so
      ```

3. Install h5py.
   a. Run the following command as the **root** user to install the h5py dependency:
      ```bash
      pip3.7 install Cython
      ```
   b. Run the following command to install h5py:
      ```bash
      pip3.7 install h5py==2.8.0
      ```

4. Install grpcio.
   Run the following command to install grpcio as the **root** user:
   ```bash
   pip3.7 install grpcio==1.32.0
   ```

5. Install NumPy.
   Run the following command to install NumPy as the **root** user:
   ```bash
   pip3.7 install numpy
   ```

**Procedure**

1. Download the compiled TensorFlow 1.15 software package:
   ```bash
   tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
   ```

2. Install the compiled TensorFlow.
   Go to the directory where the installation package is stored and run the following command as the **root** user to install TensorFlow 1.15:
   ```bash
   pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
   ```
   Run the following command as a non-root user:
   ```bash
   pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl --user
   ```

### 15.2.7 Installing Dependencies (NeoKylin OS 7.6-AArch64)

**NOTE**
- If you install the following dependencies as the **root** user, delete **sudo** or **--user** from the commands.
- If you install the following dependencies as a non-root user, run the `su - username` command to switch to the MindStudio installation user.

**Step 1** Check whether the required dependent software is installed.

1. Check whether the Python dependencies and GCC software are installed.
   Run the following commands to check whether software including GCC, g++, and Make are installed:
   ```bash
   gcc --version
   g++ --version
   make --version
   cmake --version
   rpm -qa | grep automake
   rpm -qa | grep kernel-devel
   rpm -qa | grep zlib-devel
   ```
If the following information is displayed, the software has been installed:

gcc (GCC) 4.8.5 20150623 (NeoKylin 4.8.5-39)
g++ (GCC) 4.8.5 20150623 (NeoKylin 4.8.5-39)
GNU Make 3.82

cmake version 2.8.12.2
automake-1.13.4-3.el7.noarch
kernel-devel-4.14.0-115.26.1.el7a.02.aarch64
zlib-devel-1.2.7-18.el7.aarch64
bzip2-devel-1.0.6-13.el7.aarch64
bzip2-1.0.6-13.el7.aarch64
bzip2-libs-1.0.6-13.el7.aarch64
sqlite-devel-3.7.17-8.el7.71.aarch64
open ssl-devel-1.0.2k-21.el7.9.71.01.aarch64
openssl-1.0.2k-21.el7.9.71.01.aarch64
openssl-1.0.2k-21.el7.9.71.01.aarch64
libxml2-devel-1.1.28-5.el7.aarch64
libffi-devel-3.0.13-18.el7.aarch64
gnome-keyring-3.28.2-1.el7.aarch64
libg 2-2.56.1-4.el7.6.aarch64
glib2-devel-2.56.1-4.el7.6.aarch64
dbus-glib-devel-0.100-7.el7.aarch64
xterm-295-3.el7.aarch64
ncurses-devel-6.1-13.ft.aarch64
readline-devel-6.2-10.el7.aarch64
gdb-gdbserver-7.6.1-114.el7.aarch64
atlas-devel-3.10.1-12.el7.aarch64
atlas-3.10.1-12.el7.aarch64
atlas-devel-3.10.1-12.el7.aarch64
lapack-devel-3.4.2-8.el7.aarch64
lapack-3.4.2-8.el7.aarch64
lapack-devel-3.4.2-8.el7.aarch64
gcc-gfortran-4.8.5-39.el7.ns7.01.aarch64
initscripts-9.49.46-1.el7.aarch64
freetype-devel-2.8-12.el7.aarch64
libpng-devel-1.5.13-7.el7_2.aarch64
firefox-52.7.2-1.el7.4.ns7.02.aarch64
unzip-6.0-19.el7.aarch64

rpm -qa | grep bzip2
rpm -qa | grep bzip2-devel
rpm -qa | grep sqlite-devel
rpm -qa | grep openssl
rpm -qa | grep openssl-devel
rpm -qa | grep libxslt-devel
rpm -qa | grep glib2
rpm -qa | grep dbus-glib-devel
rpm -qa | grep xterm
rpm -qa | grep ncurses-devel
rpm -qa | grep readline-devel
rpm -qa | grep gdb-gdbserve
rpm -qa | grep atlas
rpm -qa | grep atlas-devel
rpm -qa | grep lapack
rpm -qa | grep lapack-devel
rpm -qa | grep gcc-gfortran
rpm -qa | grep initscripts
rpm -qa | grep freetype-devel
rpm -qa | grep libpng-devel
rpm -qa | grep xdg-utils
rpm -qa | grep firefox
rpm -qa | grep unzip
rpm -qa | grep p7-utils
rpm -qa | grep readline
rpm -qa | grep gdb-gdbserver
rpm -qa | grep atlas
rpm -qa | grep lapack
rpm -qa | grep lapack-devel
rpm -qa | grep gcc-gfortran
rpm -qa | grep initscripts
rpm -qa | grep freetype-devel
rpm -qa | grep libpng-devel
rpm -qa | grep xdg-utils
rpm -qa | grep firefox
rpm -qa | grep unzip
rpm -qa | grep p7-utils
rpm -qa | grep readline
rpm -qa | grep gdb-gdbserver
rpm -qa | grep atlas
rpm -qa | grep lapack
rpm -qa | grep lapack-devel
rpm -qa | grep gcc-gfortran
rpm -qa | grep initscripts
rpm -qa | grep freetype-devel
rpm -qa | grep libpng-devel
rpm -qa | grep xdg-utils
rpm -qa | grep firefox
rpm -qa | grep unzip
rpm -qa | grep p7-utils
rpm -qa | grep readline
rpm -qa | grep gdb-gdbserver
rpm -qa | grep atlas
rpm -qa | grep lapack

Otherwise, run the following command to install the software. You can change the following command to install only some of them as required.

```bash
sudo yum install -y gcc gcc-c++ make cmake automake kernel-devel zlib-devel bzip2-devel sqlite-devel openssl-devel libxslt-devel libffi-devel
```

```bash
```

![NOTE](image)

The operator UT and the installation of some dependencies depend on GCC 7.3.0. Therefore, upgrade the GCC to 7.3.0 to ensure a successful UT and dependency installation.

2. Check whether the Python development environment is installed.

The ATC installation depends on the Python 3 environment. Run the `python3.7.5 --version` and `pip3.7.5 --version` commands to check whether Python 3 is installed. If the following messages are displayed, Python 3 is installed.

```
Python 3.7.5
pip 20.3.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

Otherwise, install Python 3.7.5 as follows:

a. Run the `wget` command to download the source code package of Python 3.7.5 to any directory of the MindStudio installation server:

```bash
wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
```

b. Go to the download directory and decompress the source code package:

```bash
tar -zxvf Python-3.7.5.tgz
```

c. Go to the decompressed folder, create an installation directory, and run the configuration, build, and installation commands.

```bash
cd Python-3.7.5
sudo mkdir /usr/local/python3.7.5
./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --enable-shared
make
sudo make install
```

The `--prefix` option specifies the Python installation path. You can change it as required. The `--enable-shared` parameter is used to compile the `libpython3.7m.so.1.0` dynamic library. The `--enable-loadable-sqlite-extensions` parameter is used to load sqlite-devel.

This document uses `--prefix=/usr/local/python3.7.5` as an example. After the configuration, build, and installation commands are executed, the installation package is output to the `/usr/local/python3.7.5` directory, and the `libpython3.7m.so.1.0` dynamic library is output to the `/usr/local/python3.7.5/lib/libpython3.7m.so.1.0` directory.
d. Run the following command to copy the `libpython3.7m.so.1.0` dynamic library:

i. Copy the built file `libpython3.7m.so.1.0` to `/usr/lib64`:

```
sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
```

When the following information is displayed, enter `y` to overwrite the `libpython3.7m.so.1.0` file provided by the system.

```
cp: overwrite 'libpython3.7m.so.1.0'?
```

If the `/usr/lib64` directory does not exist in the environment, copy the file to the `/usr/lib` directory.

```
sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
```

Replace the path of the `libpython3.7m.so.1.0` file as required.

**NOTE**

If this step fails, the `yum` command is unavailable. Run the following commands to rectify the fault:

1. `sed -i 's/#!/usr/bin/python/#!/usr/bin/python2.7/g' /usr/bin/yum`
2. `sed -i 's/#! /usr/bin/python/#!/usr/bin/python2.7/g' /usr/libexec/urlgrabber-ext-down`
3. `sed -i 's/#!/usr/bin/python/#!/usr/bin/python2.7/g' /usr/bin/yum-config-manager`

e. Set the soft links:

```
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
```

If a message indicating that the links exist is displayed during the command execution, run the following commands to delete the existing links and run the preceding commands again:

```
sudo rm -rf /usr/bin/python3.7
sudo rm -rf /usr/bin/pip3.7
sudo rm -rf /usr/bin/python3.7.5
sudo rm -rf /usr/bin/pip3.7.5
```

f. Set the Python 3.7.5 environment variables.

i. Run the `vi ~/.bashrc` command in any directory as the installation user to open the `.bashrc` file and append the following line to the file:

```
# Set the Python 3.7.5 library path.
export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
# If multiple Python 3 versions exist in the user environment, specify Python 3.7.5.
export PATH=/usr/local/python3.7.5/bin:$PATH
```

ii. Run the `:wq!` command to save the file and exit.

iii. Run the `source ~/.bashrc` command for the modification to take effect immediately.

g. After the installation is complete, run the following commands to check the installed version. If the required version information is displayed, the installation is successful.

```
python3.7.5 --version
pip3.7.5 --version
python3.7 --version
pip3.7 --version
```

**Step 2** Install the Python 3 dependencies.
TBE depends on NumPy and decorator. TIK depends on SymPy, CFFI, and GNU readline. The uTST tool depends on the coverage software. Model Accuracy Analyzer depends on protobuf and SciPy. Profiling depends on protobuf, grpcio, grpcio-tools, and requests. If you want to use the operator ST cases to obtain the operator shape and more from the model, you need to install TensorFlow. Otherwise, you do not need to install TensorFlow.

Before the installation, run the `pip3.7.5 list` command to check whether dependencies have been installed. If not, run the following command to install them. You can modify the following command to install only some of them as required.

```
pip3.7.5 install --upgrade pip --user
pip3.7.5 install attrs --user
pip3.7.5 install psutil --user
pip3.7.5 install decorator --user
pip3.7.5 install numpy --user
pip3.7.5 install protobuf==3.11.3 --user
pip3.7.5 install scipy --user
pip3.7.5 install gnureadline --user
pip3.7.5 install coverage --user
pip3.7.5 install pylint --user
pip3.7.5 install grpcio --user
pip3.7.5 install grpcio-tools --user
pip3.7.5 install requests --user
pip3.7.5 install xlrd==1.2.0 --user
pip3.7.5 install absl-py --user
pip3.7.5 install pytest --user
pip3.7.5 install wheel --user
pip3.7.5 install pyyaml --user
pip3.7.5 install pathlib2 --user
pip3.7.5 install sympy==1.4.0 --user
pip3.7.5 install pandas==1.1.5 --user
pip3.7.5 install pillow==8.0.1 --user
pip3.7.5 install cffi==1.12.3 --user
pip3.7.5 install pyparsing>=2.0.3 --user
pip3.7.5 install xlwt --user
pip3.7.5 install xlutils --user
pip3.7.5 install Cython --user
```

During the command execution, if the network connection fails and the message "Could not find a version that satisfies the requirement xxx" is displayed, fix the error by referring to What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When "pip3 install" Is Run?

**Step 3** Install Matplotlib in the user directory.

1. Go to the user's home directory and run the `mkdir .pip` command to create a .pip directory.
2. Run the `cd .pip` command to open the .pip directory.
3. Run the `vi pip.conf` command to edit the configuration file and add the following content. Then, save the modifications and exit.

   ```
   [global]
   index-url=http://mirrors.aliyun.com/pypi/simple/
   trusted-host=mirrors.aliyun.com
   ```

4. Run the `pip3.7.5 install matplotlib --user` command to install Matplotlib.

**Step 4** Install the JDK.

1. Before installing the JDK, run the following command to check whether Java exists in the system:

   ```
   which java
   ```
If the related path is returned, Java exists. Otherwise, run the following command to install the JDK:

```
sudo yum install -y java-11-openjdk
```

2. The installation of MindStudio depends on the `JAVA_HOME` environment variable. Run the following commands for variable setting:

   a. Run the following command in any directory to open the `.bashrc` file:

   ```bash
   vi ~/.bashrc
   ```

   b. Append the following lines to the file:

   ```
   export JAVA_HOME=/usr/lib/jvm/java-11-openjdk
   export PATH=$JAVA_HOME/bin:$PATH
   ```

   **NOTE**

   `JAVA_HOME` indicates the JDK installation path. If the JDK has been installed, modify `JAVA_HOME` as required. If the JDK has been installed according to the preceding steps, skip this modification.

   c. Save the file and exit:

   ```bash
   :wq!
   ```

   d. Make the configuration take effect:

   ```bash
   source ~/.bashrc
   ```

   e. Set the environment variable:

   ```bash
   echo $JAVA_HOME
   ```

   The command output is as follows:

   ```bash
   /usr/lib/jvm/java-11-openjdk
   ```

   f. Check the JDK installation.

   ```bash
   which jconsole
   ```

   If the command output is as follows, the JDK is successfully installed. Otherwise, the JDK installation fails.

   ```bash
   /usr/lib/jvm/java-11-openjdk-arm64/bin/jconsole
   ```

**Step 5** (Optional) Installing TensorFlow 1.15.0. For details, see Installing TensorFlow 1.15.0.

**Step 6** Go back to 2.4.4 Installing MindStudio.

----End

**Installing TensorFlow 1.15.0**

TensorFlow 1.15.0 is necessary to develop and verify operators and develop training services.

**Preparations**

In the AArch64 architecture, TensorFlow depends on h5py, and h5py depends on HDF5. Therefore, you need to compile and install HDF5 first. Otherwise, an error is reported when you use pip to install h5py. Perform the following operations as the root user:

1. Compile and install HDF5.

   a. Run the `wget` command to download the source code package of HDF5 to any directory of the installation environment. The command is as follows:

   ```bash
   wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.5/src/
   hdf5-1.10.5.tar.gz --no-check-certificate
   ```
b. Go to the download directory and decompress the source code package:
   tar -zxvf hdf5-1.10.5.tar.gz

c. Go to the new folder and run the following configuration, build, and installation commands:
   cd hdf5-1.10.5/
   ./configure --prefix=/usr/include/hdf5
   make
   make install

2. Configure environment variables and create a soft link to the dynamic link library (DLL).
   a. Set the environment variables:
      export CPATH="/usr/include/hdf5/include/:/usr/include/hdf5/lib/"
   b. Run the following commands as the root user to create a soft link to the DLL. Add sudo before the following commands as a non-root user:
      ln -s /usr/include/hdf5/lib/libhdf5.so /usr/lib/libhdf5.so
      ln -s /usr/include/hdf5/lib/libhdf5_hl.so /usr/lib/libhdf5_hl.so

3. Install h5py.
   a. Run the following command as the root user to install the h5py dependency:
      pip3.7 install Cython
   b. Run the following command to install h5py:
      pip3.7 install h5py==2.8.0

4. Install grpcio.
   Run the following command to install grpcio as the root user:
   pip3.7 install grpcio==1.32.0

5. Install NumPy.
   Run the following command to install NumPy as the root user:
   pip3.7 install numpy

Procedure

1. Download the compiled TensorFlow 1.15 software package:
   tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl

2. Install the compiled TensorFlow.
   Go to the directory where the installation package is stored and run the following command as the root user to install TensorFlow 1.15:
   pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl

   Run the following command as a non-root user:
   pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl --user

15.2.8 Installing Dependencies (Windows 10)

Installing the JDK 11 Dependency

Step 1 Download the Java installation package to a local Windows PC.

Step 2 In the Windows 10 OS, choose Control Panel > System and Security > System > Advanced system settings. See Figure 15-1.
Figure 15-1 Advance system settings

Step 3 In the System Properties dialog box displayed, click Environment Variables. See Figure 15-2.

Figure 15-2 Environment variables

Step 4 In the Environment Variables > User variables dialog box, click New. In the displayed New User Variable dialog box, enter the new environment variable and click OK. See Figure 15-3.

- Variable name: JAVA_HOME
- Variable value: path for installing the Java package
Figure 15-3 New User Variable

Step 5 In the Environment Variables > User variables dialog box, select the Path variable and click Edit. See Figure 15-4.

Figure 15-4 Editing the Path environment variable

Step 6 In the Edit environment variable dialog box, click New, enter the path of the bin directory in the Java installation package, and click OK. See Figure 15-5.
Step 7 Click **OK**.

Step 8 Open the CLI and run the **java -version** command to check the Java version.

---End

**Installing the Python 3.7.5 Dependency**

Install this dependency if you want to use the image input function during operator creation and sample project conversion.

**Step 1** Download the Python 3.7.5 installation package from the Python official website to a local Windows OS.

**Step 2** Install Python 3.7.5.

**Step 3** In the Windows 10 OS, choose **Control Panel > System and Security > System > Advanced system settings**. See **Figure 15-6**.
Step 4 In the **System Properties** dialog box displayed, click **Environment Variables**. See **Figure 15-7**.

**Figure 15-7** Environment variables

Step 5 In the **Environment Variables > User variables** dialog box, select the **Path** variable and click **Edit**. See **Figure 15-8**.
Step 6  In the Edit environment variable dialog box, click New, enter the paths of the Python 3.7.5 installation package directory and script directory, for example, D:\python and D:\python\Scripts, and click OK. See Figure 15-9.

Step 7  Click OK.
Step 8  Open the CLI and enter the `python -V` command to check whether the Python version is 3.7.5.

Step 9  Install the Python 3 dependencies.

```
pip install xlrd==1.2.0
pip install absl-py
pip install numpy
```

If the following information is displayed, the installation is successful:

```
Successfully installed xlrd-1.2.0
Successfully installed absl-py-0.12.0 six-1.15.0
Successfully installed numpy-1.20.1
```

----End

Installing MinGW Dependency

Step 1  Download the MinGW installation package to a local Windows OS.

Step 2  Decompress the MinGW installation package to a custom directory.

Step 3  In the Windows 10 OS, choose Control Panel > System and Security > System > Advanced system settings. See Figure 15-10.

Figure 15-10 Advance system settings

![Control Panel Home](image)

- Device Manager
- Remote settings
- System protection
- Advanced system settings

View basic information about your computer

Windows edition

Windows 10 Pro

© 2019 Microsoft Corporation. All rights reserved.

Step 4  In the System Properties dialog box displayed, click Environment Variables. See Figure 15-11.
Step 5  In the Environment Variables > System variables dialog box, select the Path variable and click Edit. See Figure 15-12.

Figure 15-12 Editing the Path environment variable

Step 6  In the Edit environment variable dialog box, click New, enter the path of the bin directory generated after the MinGW installation package is decompressed, for example, D:\mingw64\bin, and click OK. See Figure 15-13.
Step 7  Open the CLI and enter the `gcc -v` command.

If message "gcc version x.x.x (x86_64-posix-sjlj-rev0, Built by MinGW-W64 project)" is displayed, the installation is successful.

----End

(Optional) Installing the Windows ACLlib Package

To compile and run AscendCL applications on a local PC running the Windows OS, install this package and the corresponding dependencies. For details, see sections "ACLlib Installation and Uninstallation" and "Development Environment Setup" in the CANN V100R020C20 Windows User Guide.

15.3 Operator Specifications

15.3.1 Caffe Operator Specifications

This section applies only to networks defined by Caffe native IRs. For details about the operator specifications defined in Ascend IRs, see Operator Lists.
General restrictions:

1. Currently, operators with commit ID 9b891540183ddc834a02b2bd81b31afae71b2153 under the Caffe-master branch are supported.
2. Unless otherwise specified, the input and output: w <= 4096, h <= 4096.

For the full list of supported Caffe operators, see Caffe Operator Specification List.

15.3.2 TensorFlow Operator Specifications

This section applies only to networks defined by the TensorFlow native IRs. Only TensorFlow 1.15 is supported. For details about the operator specifications defined in Ascend IRs, see Operator ListsOperator ListsOperator Lists.

For the full list of supported TensorFlow operators, see TensorFlow Operator Specification List.

15.3.3 ONNX Operator Specifications

This chapter applies only to networks defined by ONNX native IRs. For details about the operator specifications defined in Ascend IRs, see Operator ListsOperator ListsOperator Lists.

Currently, ONNX 1.8.0 Opset v9 to v13 under ONNX Runtime 1.6.0 are supported.

For the full list of supported ONNX operators, see ONNX Operator Specification List.

15.4 Feature Configuration

15.4.1 Custom Caffe Network Modification

15.4.1.1 Overview

This chapter is limited to Caffe network modification.

Operators supported by the the Ascend AI Processor are classified as follows.

- Standard operators: standard Caffe operators, such as Convolution.
- Extended operators: open-source but non-standard Caffe operators, including:
  - Operators extended based on the Caffe framework, such as ROIPooling in Faster R-CNN and Normalize in SSD.
  - Operators extended based on other deep learning frameworks, such as PassThrough in YOLOv2.

Networks such as Faster R-CNN and SSD include some operator structures not defined in the Caffe framework, such as ROIPooling, Normalize, PSROI Pooling, and Upsample. To support these networks, extend the Caffe networks for the the Ascend AI Processor to reduce the programing workload of operator customization and postprocessing. If these extended operators are used in Caffe networks, you need to modify or add the definition of the extension layer in the .prototxt file prior to model conversion.
This chapter provides the rundown of the extended operators supported by the Ascend AI Processor and the instructions of modifying the .prototxt file.

15.4.1.2 Custom Operator List

<table>
<thead>
<tr>
<th>Category</th>
<th>Operator Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation operators</td>
<td>Reverse</td>
<td>Reverses the dimensions of a tensor.</td>
</tr>
<tr>
<td></td>
<td>ROIPooling</td>
<td>Performs region-of-interest pooling in Faster R-CNN, which is mainly used for an object detection task.</td>
</tr>
<tr>
<td></td>
<td>PSROIPooling</td>
<td>Performs position-sensitive region-of-interest pooling in R-FCN, which is mainly used for an object detection task.</td>
</tr>
<tr>
<td></td>
<td>Upsample</td>
<td>Performs upsampling using pooling mask. Used in the YOLO network.</td>
</tr>
<tr>
<td></td>
<td>Normalize</td>
<td>Normalizes the input tensor along the channel dimension using an L2 norm.</td>
</tr>
<tr>
<td></td>
<td>Reorg</td>
<td>Rearranges blocks of spatial data into depth, or vice versa, in Darknet.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Implemented as a PassThrough operator as defined in the operator specifications.</strong></td>
</tr>
<tr>
<td>Proposal</td>
<td></td>
<td>Filters bounding boxes (BBoxes) and outputs only those with the highest prediction confidence based on the foreground output of rpn_cls_prob and BBox regression output of rpn_bbox_pred in Faster R-CNN.</td>
</tr>
<tr>
<td>ROIAlign</td>
<td></td>
<td>A regional feature aggregation method that solves the problem of misalignment caused by two quantifications in ROIPooling operation.</td>
</tr>
<tr>
<td>Category</td>
<td>Operator Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ShuffleChannel</strong></td>
<td></td>
<td>Permutes data in the channel dimension of the input.</td>
</tr>
<tr>
<td><strong>YOLO (Yolo/Detection/Region)</strong></td>
<td></td>
<td>Replacement to Yolo, Detection, and Region operators to generate coordinates, confidence scores, and category probability of the anchor boxes on the feature map output by the convolutional network.</td>
</tr>
<tr>
<td><strong>PriorBox</strong></td>
<td></td>
<td>Generates prior boxes based on the input parameters in the SSD network.</td>
</tr>
<tr>
<td><strong>SpatialTransformer</strong></td>
<td></td>
<td>Performs affine transformation.</td>
</tr>
<tr>
<td>Postprocessing operators</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>YoloV3DetectionOutput</td>
<td>Generates coordinates, confidence scores, and category probability of the anchor boxes on the feature map output by the convolutional network for the postprocessing of YOLOv3.</td>
</tr>
<tr>
<td></td>
<td>YoloV2DetectionOutput</td>
<td>Generates coordinates, confidence scores, and category probability of the anchor boxes on the feature map output by the convolutional network for the postprocessing of YOLOv2.</td>
</tr>
<tr>
<td></td>
<td>SSDDetectionOutput</td>
<td>Integrates the BBoxes, BBox offsets, and scores, and outputs object predictions of SSD.</td>
</tr>
<tr>
<td></td>
<td>FSRDetectionOutput</td>
<td>Classifies the results, and outputs the final number, coordinates, category probability, and category indexes of BBoxes of Faster R-CNN.</td>
</tr>
</tbody>
</table>
15.4.1.3 Custom Operator Description

Custom operators can be implemented in deep learning frameworks including Caffe, TensorFlow, and MindSpore. The first type of operators is customized from the Caffe framework, such as the ROIPooling, Normalize, PSROIPolling, and Upsample layers. In other words, these operators have been defined in Caffe's prototxt files. For operators customized from frameworks other than Caffe, it is also necessary to give corresponding definitions in prototxt format.

Reverse

Reverses the dimensions of a tensor. For example, reverse from [1, 2, 3] to [3, 2, 1].

Define the operator as follows.

1. Add ReverseParameter to LayerParameter.
   ```
   message LayerParameter {
     ...
     optional ReverseParameter reverse_param = 157;
     ...
   }
   ```

2. Define the data types and attributes of ReverseParameter.
   ```
   message ReverseParameter {
     repeated int32 axis = 1;
   }
   ```

ROIPooling

The major hurdle for going from image classification to object detection is fixed size input requirement to the network because of the existing fully connected (FC) layers. In object detection, different proposals have different shapes. Therefore, it is necessary to convert all the proposals to a fixed shape as required by FC layers.

Region of Interest pooling (ROIPooling) is used for utilizing a single feature map for all the generated proposals in a single pass. ROIPooling solves the problem of fixed image size requirement for object detection network.

You need to extend the caffe.proto file and define ROIPoolingParameter as follows.

- **spatial_scale**: multiplicative spatial scale factor to translate ROI coordinates from their input scale to the scale used when pooling
- **pooled_h** and **pooled_w**: height and width of the ROI output feature map

1. Add ROIPoolingParameter to LayerParameter.
   ```
   message LayerParameter {
     ...
     optional ROIPoolingParameter roi_pooling_param = 161;
     ...
   }
   ```

2. Define the data types and attributes of ROIPoolingParameter.
   ```
   message ROIPoolingParameter {
     required int32 pooled_h = 1;
     required int32 pooled_w = 2;
     optional float spatial_scale = 3 [default=0.0625];
     optional float spatial_scale_h = 4;
     optional float spatial_scale_w = 5;
   }
   ```
Example .prototxt definition of ROIPooling:

```protobuf
layer {
  name: "roi_pooling"
  type: "ROIPooling"
  bottom: "res4f"
  bottom: "rois"
  bottom: "actual_rois_num"
  top: "roi_pool"
  roipooling_param {
    pooled_h: 14
    pooled_w: 14
    spatial_scale: 0.0625
    spatial_scale_h: 0.0625
    spatial_scale_w: 0.0625
  }
}
```

**PSROIPooling**

Position Sensitive ROI Pooling (PSROIPooling) works in similar way to ROIPooling. However, unlike ROIPooling, the feature map output from PSROIPooling is obtained from different feature map channels, and average pooling (instead of max-pooling) is performed on each divided bin.

PSROIPooling divides the ROI into $k \times k$ bins and outputs a $k \times k$ feature map. The number of output channels for pooling is the same as the number of input channels.

You need to extend the `caffe.proto` file and define `PSROIPoolingParameter` as follows.

- **spatial_scale**: multiplicative spatial scale factor to translate ROI coordinates from their input scale to the scale used when pooling
- **output_dim**: number of output channels
- **group_size**: number of groups to encode position-sensitive score maps, that is, $k$

1. Add `PSROIPoolingParameter` to `LayerParameter`.

```protobuf
message LayerParameter {
  ...
  optional PSROIPoolingParameter psroi_pooling_param = 207;
  ...
}
```

2. Define the data types and attributes of `PSROIPoolingParameter`.

```protobuf
message PSROIPoolingParameter {
  required float spatial_scale = 1;
  required int32 output_dim = 2; // output channel number
  required int32 group_size = 3; // number of groups to encode position-sensitive score maps
}
```

Example .prototxt definition of PSROIPooling:

```protobuf
layer {
  name: "psroipooling"
  type: "PSROIPooling"
  bottom: "some_input"
  bottom: "some_input"
  top: "some_output"
  psroi_pooling_param {
    spatial_scale: 0.0625
    output_dim: 21
    group_size: 7
  }
}
```
### Upsample

The Upsample layer is the reverse of the Pooling layer. Each decoder upsamples the activations generated by the corresponding encoder.

You need to extend the `caffe.proto` file and define `UpsampleParameter` as follows. The `stride` parameter is the upsampling factor, for example, 2.

1. Add `UpsampleParameter` to `LayerParameter`.
   ```protobuf
   message LayerParameter {
     ...
     optional UpsampleParameter upsample_param = 160;
     ...
   }
   ```

2. Define the data types and attributes of `UpsampleParameter`.
   ```protobuf
   message UpsampleParameter{
     optional float scale = 1[default = 1];
     optional int32 stride = 2[default = 2];
     optional int32 stride_h = 3[default = 2];
     optional int32 stride_w = 4[default=2];
   }
   ```

Example .prototxt definition of Upsample:

```plaintext
layer {
  name: "layer86-upsample"
  type: "Upsample"
  bottom: "some_input"
  top: "some_output"
  upsample_param {
    scale: 1
    stride: 2
  }
}
```

### Normalize

The Normalize layer is a normalization layer in the SSD network, and is mainly used to normalize elements in a space or a channel to the range [0, 1]. The Normalize layer is to output a tensor of a same size for a c * h * w three-dimensional tensor. In the formula, Normalize is calculated based on the square root of the sum of squares in the channel direction for each element. The formula is as follows.

\[
x'_j = \frac{x_j}{\sqrt{\sum_{i=1}^{c} x_i^2}}
\]

where, the cumulative vector of the square sum in the denominator part is the sum of the channel vectors that share the same height and width, as the orange part shown in **Figure 15-14**.
After the preceding normalization calculation, the Normalize layer scales each feature map using separate scale factors.

You need to extend the caffe.proto file and define NormalizeParameter as follows.

- **across_spatial**: a bool. If True, normalizes every channel to 1 x c x h x w. If False, normalizes every pixel to 1 x c x 1 x 1.
- **channels_shared**: a bool. If True, the scale parameters are shared across channels. Defaults to True.
- **eps**: (optional) a small number to avoid division by zero while normalizing. Defaults to 1e-10.

The mathematical formulation of Normalize is as follows.

\[
X'_j = \frac{X_j}{\left(\sum_{i=1}^{n} X_i^2 + \text{eps}\right)^{\frac{1}{2}}} \times \text{scale}_j
\]

Define the operator as follows.

1. Add NormalizeParameter to LayerParameter.

   ```protobuf
   message LayerParameter {
     ...
     optional NormalizeParameter norm_param = 206;  // Initial value of scale. Default is 1.0 for all
     ...
   }
   ``

2. Define the data types and attributes of NormalizeParameter.

   ```protobuf
define NormalizeParameter {
    optional bool across_spatial = 1 [default = true];  // Whether or not scale parameters are shared across channels.
    optional FillerParameter scale_filler = 2;  // Epsilon for not dividing by zero while normalizing variance
    optional bool channel_shared = 3 [default = true];
    optional float eps = 4 [default = 1e-10];
   }
   ```

Example .prototxt definition of Normalize:

```protobuf
layer {
  name: "normalize_layer"
  type: "Normalize"
}
```
Reorg

The Reorg operator is implemented as a PassThrough operator in the Ascend AI Processor, which rearranges blocks of spatial data into depth, or vice versa.

The PassThrough layer is not implemented using the Caffe framework. Therefore, there is no standard definition for this layer. The PassThrough layer concatenates the higher resolution features with lower one by stacking adjacent features into different channels instead of spatial locations.

Define the operator as follows.

1. Add ReorgParameter to LayerParameter.

   ```
   message LayerParameter {
      ... 
      optional ReorgParameter reorg_param = 155; 
      ... 
   }
   ```

2. Define the data types and attributes of ReorgParameter.

   ```
   message ReorgParameter {
      optional uint32 stride = 2 [default = 2];
      optional bool reverse = 1 [default = false];
   }
   ```

Example .prototxt definition of Proposal:

```
layer {
   bottom: "some_input"
   top: "some_output"
   name: "reorg"
   type: "Reorg"
   reorg_param {
      stride: 2
   }
}
```
Define the operator as follows.

1. Add **ProposalParameter** to **LayerParameter**.

   ```
   message LayerParameter {
       ...
       optional ProposalParameter proposal_param = 201;
       ...
   }
   ```

2. Define the data types and attributes of **ProposalParameter**.

   ```
   message ProposalParameter {
       optional float feat_stride = 1 [default = 16];
       optional float base_size = 2 [default = 16];
       optional float min_size = 3 [default = 16];
       repeated float ratio = 4;
       repeated float scale = 5;
       optional int32 pre_nms_topn = 6 [default = 3000];
       optional int32 post_nms_topn = 7 [default = 304];
       optional float iou_threshold = 8 [default = 0.7];
       optional bool output_actual_rois_num = 9 [default = false];
   }
   ```

Example .prototxt definition of Proposal:

```
layer {
    name: "faster_rcnn_proposal"
    type: "Proposal"  //Operator type
    bottom: "rpn_cls_prob_reshape"
    bottom: "rpn_bbox_pred"
    bottom: "im_info"
    top: "rois"
    top: "actual_rois_num"  // Added operator output
    proposal_param {
        feat_stride: 16
        base_size: 16
        min_size: 16
        pre_nms_topn: 3000
        post_nms_topn: 304
        iou_threshold: 0.7
        output_actual_rois_num: true
    }
}
```

**ROIAlign**

ROIAlign is a regional feature aggregation method proposed by Mask-RCNN, which solves the problem of misalignment caused by two quantifications in ROIPooling operation.
The size of the feature map after pooling is \( \text{pooled}_w \times \text{pooled}_h \). Each ROI is divided into \( \text{sampling}_\text{ratio} \times \text{sampling}_\text{ratio} \) grids of the same size. The grid points are the sampling points. As shown in Figure 15-16, the dashed line indicates the feature map, and the solid line indicates the ROI, which is divided into \( 2 \times 2 \) cells. Assuming that the number of sampling points is 4, it means that four grids are equally divided, each of which takes its center point position. The pixel of the center point position (denoted by four arrows in Figure 15-16) is calculated by bilinear interpolation method. Finally, average the four pixel values as the ROIAlign result.

**Figure 15-16 ROIAlign diagram**

Define the operator as follows.

1. Add **ROIAlignParameter** to **LayerParameter**.

```protobuf
define the data types and attributes of **ROIAlignParameter**.
message ROIAlignParameter {
  // Pad, kernel size, and stride are all given as a single value for equal
  // dimensions in height and width or as Y, X pairs.
  optional uint32 pooled_h = 1 [default = 0]; // The pooled output height
  optional uint32 pooled_w = 2 [default = 0]; // The pooled output width
  // Multiplicative spatial scale factor to translate ROI coords from their
  // input scale to the scale used when pooling
  optional float spatial_scale = 3 [default = 1];
  optional int32 sampling_ratio = 4 [default = -1];
  optional int32 roi_end_mode = 5 [default = 0];
}
```

You can customize the .prototxt file based on the preceding data types and attributes.

**ShuffleChannel**

ShuffleChannel permutes data in the channel dimension of the input.

For example, if \( \text{channel} = 4 \) and \( \text{group} = 2 \), ShuffleChannel transposes channel[1] and channel[2].
Define the operator as follows.

1. Add `ShuffleChannelParameter` to `LayerParameter`.

   ```proto
   message LayerParameter {
      ...
      optional ShuffleChannelParameter shuffle_channel_param = 159;
      ...
   }
   ``

2. Define the data types and attributes of `ShuffleChannelParameter`.

   ```proto
   message ShuffleChannelParameter {
      optional uint32 group = 1 [default = 1]; // The number of group
   }
   ``

Example .prototxt definition of ShuffleChannel:

```proto
layer {
   name: "layer_shuffle"
   type: "ShuffleChannel"
   bottom: "some_input"
   top: "some_output"
   shuffle_channel_param {
      group: 3
   }
}
```

YOLO

The YOLO operator is introduced to the YOLOv2 network and is applied only on the YOLOv2 and YOLOv3 networks. It performs sigmoid and softmax operations on input.

- In YOLOv2, there are four scenarios based on the `background` and `softmax` parameters:
  
  a. background = false, softmax = true:
     sigmoid is performed on (x, y) in (x, y, h, w), sigmoid is performed on b, and softmax is performed on classes.
  
  b. background = false, softmax = false:
     sigmoid is performed on (x, y) in (x, y, h, w), sigmoid is performed on b, and sigmoid is performed on classes.
  
  c. background = true, softmax = false:
     sigmoid is performed on (x, y) in (x, y, h, w), b is ignored, and sigmoid is performed on classes.
  
  d. background = true, softmax = true:
     sigmoid is performed on (x, y) in (x, y, h, w), and softmax is performed on b and classes.

- In YOLOv3, there is only one scenario: sigmoid is performed on (x, y) in (x, y, h, w), sigmoid is performed on b, and sigmoid is performed on classes.

The input data format is `Tensor(n, coords+background+classes,l,h,l,w)`, where `n` indicates the number of anchor boxes and `coords` indicates x, y, w, and h.

Define the operator as follows.

1. Add `YoloParameter` to `LayerParameter`.

   ```proto
   message LayerParameter {
      ...
      optional YoloParameter yolo_param = 199;
   }
   ```
2. **Define the data types and attributes of **YoloParameter**.  
   ```
   message YoloParameter {
     optional int32 boxes = 1 [default = 3];
     optional int32 coords = 2 [default = 4];
     optional int32 classes = 3 [default = 80];
     optional string yolo_version = 4 [default = "V3"];
     optional bool softmax = 5 [default = false];
     optional bool background = 6 [default = false];
     optional bool softmaxtree = 7 [default = false];
   }
   ```

   **Example .prototxt definition of YOLO:**
   ```
   layer {
     bottom: "layer82-conv"
     top: "yolo1_coords"
     top: "yolo1_obj"
     top: "yolo1_classes"
     name: "yolo1"
     type: "Yolo"
     yolo_param {
       boxes: 3
       coords: 4
       classes: 80
       yolo_version: "V3"
       softmax: true
       background: false
     }
   }
   ```

**PriorBox**

The prior box is generated according to the arguments.

The following uses conv7_2_mbox_priorbox as an example. The definition is as follows.

```
layer{
  name: "conv7_2_mbox_priorbox"
  type: "PriorBox"
  bottom: "conv7_2"
  bottom: "data"
  top: "conv7_2_mbox_priorbox"
  prior_box_param{
    min_size: 162.0
    max_size: 213.0
    aspect_ratio: 2
    aspect_ratio: 3
    flip: true
    clip: false
    variance: 0.1
    variance: 0.1
    variance: 0.2
    variance: 0.2
    img_size: 300
    step: 64
    offset: 0.5
  }
}
```

1. A prior box is generated when the width and height are both **min_size**.
2. If **max_size** is available, \( \sqrt{\text{min_size} \times \text{max_size}} \) is used to determine the width and height of generated boxes (\( \text{max_size} > \text{min_size} \)).
3. The prior box is generated based on the aspect ratios (1/2 and 1/3 according to the definition).

Therefore, num_priors_ = min_sizes + aspect_ratios * min_size + max_size

Define the operator as follows.

1. **Add PriorBoxParameter to LayerParameter.**

   ```
   message LayerParameter {
     ...
     optional PriorBoxParameter prior_box_param = 203;
     ...
   }
   ```

2. **Define the data types and attributes of PriorBoxParameter.**

   ```
   message PriorBoxParameter {
     // Encode/decode type.
     enum CodeType {
       CORNER = 1;
       CENTER_SIZE = 2;
       CORNER_SIZE = 3;
     }
     // Minimum box size (in pixels). Required!
     repeated float min_size = 1;
     // Maximum box size (in pixels). Required!
     repeated float max_size = 2;
     // Various of aspect ratios. Duplicate ratios will be ignored.
     // If none is provided, we use default ratio 1.
     repeated float aspect_ratio = 3;
     // If true, will flip each aspect ratio.
     // For example, if there is aspect ratio "r",
     // we will generate aspect ratio "1.0/r" as well.
     optional bool flip = 4 [default = true];
     // If true, will clip the prior so that it is within [0, 1]
     optional bool clip = 5 [default = false];
     // Variance for adjusting the prior bboxes.
     repeated float variance = 6;
     // By default, we calculate img_height, img_width, step_x, step_y based on
     // bottom[0] (feat) and bottom[1] (img). Unless these values are explicitly
     // provided.
     // Explicitly provide the img_size.
     optional uint32 img_size = 7;
     // Either img_size or img_h/img_w should be specified; not both.
     optional uint32 img_h = 8;
     optional uint32 img_w = 9;
     // Explicitly provide the step size.
     optional float step = 10;
     // Either step or step_h/step_w should be specified; not both.
     optional float step_h = 11;
     optional float step_w = 12;
     // Offset to the top left corner of each cell.
     optional float offset = 13 [default = 0.5];
   }
   ```

**Example .prototxt definition of PriorBox:**

```plaintext
layer {
  name: "layer_priorbox"
  type: "PriorBox"
  bottom: "some_input"
  bottom: "some_input"
  top: "some_output"
  prior_box_param {
    min_size: 30.0
    max_size: 60.0
    aspect_ratio: 2
    flip: true
  }
}
```
Spatial Transformer

This operator performs affine transformation in the computation process. If you need only one set of parameters of affine transformation, define them in the .prototxt file and use them for multiple batches. Alternatively, use dynamic parameters as the second input of the operator layer. In this way, parameters for each batch are different.

The procedure is as follows.

1. Convert the output coordinates into values in the range of $[-1, 1]$ by using the following formulas.
   \[
   x' = x \times \frac{1.0}{\text{output}_h} \times 2 - 1 \\
   y' = y \times \frac{1.0}{\text{output}_w} \times 2 - 1
   \]
   The corresponding code is as follows.
   ```cpp
   Dtype* data = output_grid.mutable_cpu_data();
   for (int i = 0; i < output_H_ * output_W_; ++i) {
       data[3 * i] = (i / output_W_) * 1.0 / output_H_ * 2 - 1;
       data[3 * i + 1] = (i % output_W_) * 1.0 / output_W_ * 2 - 1;
       data[3 * i + 2] = 1;
   }
   ```

2. Perform affine transformation to convert the output coordinates into input coordinates. In the following formula, $s$ indicates input coordinates, and $t$ indicates output coordinates.
   \[
   \begin{pmatrix}
   x_i^s \\
   y_i^s \\
   \end{pmatrix}
   = \begin{bmatrix}
   \theta_{11} & \theta_{12} & \theta_{13} \\
   \theta_{21} & \theta_{22} & \theta_{23} \\
   \end{bmatrix}
   \begin{pmatrix}
   x_i^t \\
   y_i^t \\
   1
   \end{pmatrix}
   \]
   The corresponding code is as follows.
   ```cpp
   caffe_cpu_gemm<Dtype>(CblasNoTrans, CblasTrans, output_H_ * output_W_, 2, 3, (Dtype)1.,
   output_grid_data, full_theta_data + 6 * i, (Dtype)0., coordinates);
   ```

3. Obtain the value of a specific position based on the input coordinates and assign the value to the corresponding output position.
   The output coordinates are converted in step 1. Therefore, you need to convert the input coordinates in the same way. The following is a code example.
   ```cpp
   Dtype x = (px + 1) / 2 * H;
   Dtype y = (py + 1) / 2 * W;
   if (debug) std::cout << prefix << "(x, y) = (" << x << ", " << y << ")" << std::endl;
   for (int m = floor(x); m <= ceil(x); ++m)
   ```
for(int n = floor(y); n <= ceil(y); ++n) {
    if(debug) std::cout<<prefix<<"(m, n) = ("<<m<<", "<<n<<")"<<std::endl;
    if(m >= 0 && m < H && n >= 0 && n < W) {
        res += (1 - abs(x - m)) * (1 - abs(y - n) * pic[m * W + n]);
        if(debug) std::cout<<prefix<<"pic["<<m<<" * W + "<<n<<"]="<<std::endl;
    }
}

Define the operator as follows.

1. **Add SpatialTransformParameter to LayerParameter.**
   message LayerParameter {
   ...
   optional SpatialTransformParameter spatial_transform_param = 153;
   ...
   }

2. **Define the SpatialTransformParameter class and attribute parameters.**
   message SpatialTransformParameter {
   optional uint32 output_h = 1 [default = 0];
   optional uint32 output_w = 2 [default = 0];
   optional float border_value = 3 [default = 0];
   repeated float affine_transform = 4;
   enum Engine {
      DEFAULT = 0;
      CAFFE = 1;
      CUDNN = 2;
   }
   optional Engine engine = 15 [default = DEFAULT];
   }

Example .prototxt definition of SpatialTransform:

layer {
    name: "st_1"
    type: "SpatialTransformer"
    bottom: "data"
    bottom: "theta"
    top: "transformed"
    st_param {
        to_compute_dU: false
        theta_1_1: -0.129
        theta_1_2: 0.626
        theta_2_1: 0.344
        theta_2_2: 0.157
    }
}

### 15.4.1.4 Samples

This section provides instructions for modifying frequently-used networks.

#### 15.4.1.4.1 Modifying Faster R-CNN Prototxt

**NOTE**

All code samples in this section cannot be directly copied to your network model. You need to adjust the parameters to suit your use case. For example, the **bottom** and **top** parameters must match those in the corresponding network model, and the sequence of the bottom and top parameters is fixed.

The following uses the Faster R-CNN ResNet-34 model as an example.

1. **Modify the Proposal operator.**

   According to **15.3.1 Caffe Operator Specifications**, the operator has three inputs and two outputs. Modify the **type** argument to that defined in the
caffe.proto file and add the actual_rois_num output node. Add attribute description by referring to the attribute definition in the caffe.proto file. Figure 15-17 shows the .prototxt file before and after modification for adapting to the Ascend AI Processor.

Figure 15-17 Prototxt file before and after modification

A code example is as follows.

layer {
  name: "faster_rcnn_proposal"
  type: "Proposal"  //Operator type

  bottom: "rpn_cls_prob_reshape"
  bottom: "rpn_bbox_pred"
  bottom: "im_info"
  top: "rois"
  top: "actual_rois_num"  // Added operator output

  proposal_param {
    feat_stride: 16
    base_size: 16
    min_size: 16
    pre_nms_topn: 3000
    post_nms_topn: 304
    iou_threshold: 0.7
    output_actual_rois_num: true
  }
}

For details about parameter description, see 15.3.1 Caffe Operator Specifications.

2. Add an FSRDetectionOutput operator to the output layer to output the final detection result.

If your network is Faster R-CNN, add a postprocessing layer FSRDetectionOutput to the end of the original .prototxt file by referring to 15.4.1.2 Custom Operator List. The FSRDetectionOutput operator has five inputs and two outputs as described in 15.3.1 Caffe Operator Specifications. Define the data types and the attributes of the operator accordingly.

A code example is as follows.

layer {
  name: "FSRDetectionOutput_1"
  type: "FSRDetectionOutput"

  bottom: "rois"
  bottom: "bbox_pred"
  bottom: "cls_prob"
  bottom: "im_info"
  bottom: "actual_rois_num"
}
For details about parameter description, see 15.3.1 Caffe Operator Specifications.

15.4.1.4.2 Modifying YOLOv3 Prototxt

NOTE

All code samples in this section cannot be directly copied to your network model. You need to adjust the parameters to suit your use case. For example, the bottom and top parameters must match those in the corresponding network model, and the sequence of the bottom and top parameters is fixed.

1. Modify the upsample_param attribute of the Upsample operator.
   Change scale:2 in the .prototxt file of the original operator to scale:1 stride:2 by referring to 15.3.1 Caffe Operator Specifications.
   Figure 15-18 shows the .prototxt file before and after modification for adapting to the Ascend AI Processor.

   Figure 15-18 Prototxt file before and after modification

   For details about parameter description, see 15.3.1 Caffe Operator Specifications.

2. Add three YOLO operators.
   The YOLO and DetectionOutput operators complete the postprocessing logic of the feature detection network. According to the original operator .prototxt file, three YOLO operators should be added before adding the YoloV3DetectionOutput operator.
   According to 15.3.1 Caffe Operator Specifications, a Yolo operator has one input and three outputs. The code examples of the Yolo operators are provided.

   - Code example of operator 1
     ```
     layer {
       bottom: "layer82-conv"
       top: "yolo1_coords"
       top: "yolo1_obj"
       top: "yolo1_classes"
       name: "yolo1"
       type: "Yolo"
       yolo_param {
         boxes: 3
         coords: 4
         classes: 80
       }
     }
     ```
For details about parameter description, see 15.3.1 Caffe Operator Specifications.

3. Add a YoloV3DetectionOutput operator to the output layer.

On the YOLOv3 network, add a postprocessing layer YoloV3DetectionOutput to the end of the original .prototxt file by referring to 15.4.1.2 Custom Operator List. The YoloV3DetectionOutput operator has ten inputs and two outputs as described in 15.3.1 Caffe Operator Specifications.
obj_threshold: 0.5
score_threshold: 0.5
iou_threshold: 0.45
pre_nms_topn: 512
post_nms_topn: 1024
biases_high: 10
biases_high: 13
biases_high: 16
biases_high: 30
biases_high: 33
biases_high: 23
biases_mid: 30
biases_mid: 61
biases_mid: 62
biases_mid: 45
biases_mid: 59
biases_mid: 119
biases_low: 116
biases_low: 90
biases_low: 156
biases_low: 198
biases_low: 373
biases_low: 326

For details about parameter description, see 15.3.1 Caffe Operator Specifications.

4. Add the input.

The YoloV3DetectionOutput operator has the img_info input. Add img_info to model inputs. Figure 15-19 shows the .prototxt file before and after modification for adapting to the the Ascend AI Processor.

Figure 15-19 Prototxt file before and after modification

The following is a code example. img_info has shape [batch, 4], where 4 is formatted [netH, netW, scaleH, scaleW]. netH and netW are H and W of the network model input, and scaleH and scaleW are H and W of the original image.

input: "img_info"
input_shape {
  dim: 1
  dim: 4
}

15.4.1.4.3 Modifying YOLOv2 Prototxt

NOTE

All code samples in this section cannot be directly copied to your network model. You need to adjust the parameters to suit your use case. For example, the bottom and top parameters must match those in the corresponding network model, and the sequence of the bottom and top parameters is fixed.
1. Modify the Region operator.

The YOLO and DetectionOutput operators complete the postprocessing logic of the feature detection network. Before adding the YoloV2DetectionOutput operator, replace the Region operator with a YOLO operator.

A YOLO operator has one input and three outputs according to **15.3.1 Caffe Operator Specifications**. Figure 15-20 shows the .prototxt file before and after modification for adapting to the the Ascend AI Processor.

**Figure 15-20** Prototxt file before and after modification

![Prototxt file before and after modification](image)

A code example is as follows.

```protobuf
define your code here
```

For details about parameter description, see **15.3.1 Caffe Operator Specifications**.

2. Add a YoloV2DetectionOutput operator to the output layer.

On the YOLOv2 network, add a postprocessing layer YoloV2DetectionOutput to the end of the original .prototxt file by referring to **15.4.1.2 Custom Operator List**. The YoloV2DetectionOutput operator has four inputs and two outputs as described in **15.3.1 Caffe Operator Specifications**.

```protobuf
define your code here
```
post_nms_topn: 1024
biases: 0.572730
biases: 0.677385
biases: 1.874460
biases: 2.062530
biases: 3.338430
biases: 5.474340
biases: 7.882820
biases: 9.770520
biases: 9.168280
}

For details about parameter description, see 15.3.1 Caffe Operator Specifications.

3. Add the input.

The YoloV2DetectionOutput operator has the img_info input. Add img_info to model inputs. Figure 15-21 shows the .prototxt file before and after modification for adapting to the the Ascend AI Processor.

**Figure 15-21** Prototxt file before and after modification

The following is a code example. img_info has shape `[batch, 4]`, where 4 is formatted `[netH, netW, scaleH, scaleW]`. netH and netW are H and W of the network model input, and scaleH and scaleW are H and W of the original image.

```plaintext
input: "img_info"
input_shape {
  dim: 1
  dim: 4
}
```

15.4.1.4.4 Modifying SSD Prototxt

**NOTE**

All code samples in this section cannot be directly copied to your network model. You need to adjust the parameters to suit your use case. For example, the bottom and top parameters must match those in the corresponding network model, and the sequence of the bottom and top parameters is fixed.

If your network is SSD, add a postprocessing layer SSDDetectionOutput to the end of the original .prototxt file by referring to 15.4.1.2 Custom Operator List.

For details, see the caffe.proto file in `$install_path/atc/include/proto`. Add the declaration of the custom layer to the LayerParameter message. (The following custom layer has been declared in caffe.proto and you do not need to add it again.)

```plaintext
message LayerParameter {
  ...
```
According to the `caffe.proto` file, the operator type and attributes are defined as follows.

```protobuf
textproto
message SSDDetectionOutputParameter {
  optional int32 num_classes = 1 [default = 2];
  optional bool share_location = 2 [default = true];
  optional int32 background_label_id = 3 [default = 0];
  optional float iou_threshold = 4 [default = 0.45];
  optional int32 top_k = 5 [default = 400];
  optional float eta = 6 [default = 1.0];
  optional bool variance_encoded_in_target = 7 [default = false];
  optional int32 code_type = 8 [default = 2];
  optional int32 keep_top_k = 9 [default = 200];
  optional float confidence_threshold = 10 [default = 0.01];
}
```

As described in 15.3.1 Caffe Operator Specifications, the SSDDetectionOutput operator has three inputs and two outputs. A code example is provided as follows.

```protobuf
textproto
layer {
  name: "detection_out"
  type: "SSDDetectionOutput"
  bottom: "bbox_delta"
  bottom: "score"
  bottom: "anchors"
  top: "out_boxnum"
  top: "y"
  sssdetectionoutput_param {
    num_classes: 2
    share_location: true
    background_label_id: 0
    iou_threshold: 0.45
    top_k: 400
    eta: 1.0
    variance_encoded_in_target: false
    code_type: 2
    keep_top_k: 200
    confidence_threshold: 0.01
  }
}
```

- In the **bottom** input, `bbox_delta` corresponds to `mbox_loc` in the original Caffe network, `score` corresponds to `mbox_conf_flatten` in the original Caffe network, and `anchors` corresponds to `mbox_priorbox` in the original Caffe network. The value of `num_classes` must be the same as that in the original network.

- In the scenario where the top output has a batch size greater than 1:
  - The output shape of `out_boxnum` is (batchnum, 8). The first element of `batchnum` is the number of actual boxes.
  - The output shape of `y` is (batchnum, len, 8), where `len` is the value of `keep_top_k` aligned to 128. For example, if `batch = 2` and `keep_top_k = 200`, the output shape is (2, 256, 8), the first 256 x 8 data elements are the result of the first batch.

For details about parameter description, see 15.3.1 Caffe Operator Specifications.
### 15.4.1.4.5 Modifying BatchedMatMul Prototxt

**NOTE**

All code samples in this section cannot be directly copied to your network model. You need to adjust the parameters to suit your use case. For example, the `bottom` and `top` parameters must match those in the corresponding network model, and the sequence of the bottom and top parameters is fixed.

The BatchedMatMul operator multiplies the two tensors: \( y = x_1 \times x_2 \). (The number of \( x_1 \) and \( x_2 \) dimensions must be greater than 2 and less than or equal to 8.) To use this operator in a network model, modify its .prototxt file by referring to this section and then convert the model.

For details, see the `caffe.proto` file in `{install_path}/atc/include/proto`. Add the declaration of the custom layer to the `LayerParameter` message. (The following custom layer has been declared in `caffe.proto` and you do not need to add it again.)

```protobuf
google.protobuf.Message LayerParameter {
  ...  
  optional BatchMatMulParameter batch_matmul_param = 235; 
  ...  
}
```

According to the `caffe.proto` file, the operator type and attributes are defined as follows.

```protobuf
google.protobuf.Message BatchMatMulParameter {
  optional bool adj_x1 = 1 [default = false];
  optional bool adj_x2 = 2 [default = false];
}
```

According to [15.3.1 Caffe Operator Specifications](#15.3.1-Caffe-Operator-Specifications), the BatchedMatMul operator has two inputs and one output. An example of the constructed operator code is as follows.

```protobuf
layer {
  name: "batchmatmul"
  type: "BatchedMatMul"
  bottom: "matmul_data_1"
  bottom: "matmul_data_2"
  top: "batchmatmul_1"
  batch_matmul_param {
    adj_x1: false
    adj_x2: true
  }
}
```

For details about parameter description, see [15.3.1 Caffe Operator Specifications](#15.3.1-Caffe-Operator-Specifications).

### 15.4.1.4.6 Modifying SENet Prototxt

**NOTE**

All code samples in this section cannot be directly copied to your network model. You need to adjust the parameters to suit your use case. For example, the `bottom` and `top` parameters must match those in the corresponding network model, and the sequence of the bottom and top parameters is fixed.

The Axpy operator in the network model needs to be modified to the Reshape, Scale, and Eltwise operators. The following figure shows the modification.

---

**Note:**

This document is a part of the MindStudio User Guide, Issue 01 (2021-06-24). Copyright © Huawei Technologies Co., Ltd.
An example of the modified code is as follows.

layer {
    name: "conv3_1_axpy_reshape"
    type: "Reshape"
    bottom: "conv3_1_1x1_up"
    top: "conv3_1_axpy_reshape"
    reshape_param {
        shape {
            dim: 0
            dim: -1
        }
    }
}
layer {
    name: "conv3_1_axpy_scale"
    type: "Scale"
    bottom: "conv3_1_1x1_increase"
    bottom: "conv3_1_axpy_reshape"
    top: "conv3_1_axpy_scale"
    scale_param {
        axis: 0
        bias_term: false
    }
}
layer {
    name: "conv3_1_axpy_eltwise"
    type: "Eltwise"
    bottom: "conv3_1_axpy_scale"
    bottom: "conv3_1_1x1_proj"
    top: "conv3_1"
}

For details about parameter description, see 15.3.1 Caffe Operator Specifications.

15.5 How Do I Determine the Video Stream Format Standard When I Perform CSC on a Model Using AIPP?

Q: How do I determine the video stream format standard when I perform CSC on a model using AIPP?
A: The third-party tool ffprobe is used as an example. You can use other third-party tools.

1. Download the tool and related documents from https://www.ffmpeg.org/ffprobe-all.html#Description.
2. Obtain the video information with the `ffprobe -show_frames filename` option.
   This option displays information about each frame and subtitle contained in the input multimedia stream. The information for each single frame is printed within a dedicated section with name "FRAME" or "SUBTITLE".
3. Determine the video standard based on the result:
   - `color_range`: tv or pc
   - `color_space`: bt709 or bt601
   tv indicates "limited", that is, narrow range. pc indicates "full", that is, wide range.
   For example, `color_range=tv` and `color_space=bt709`. The video stream format standard is NARROW, BT-709.

   NOTE

   If the command is different from the example, refer to the official description of the tool.

15.6 Configuring a System Network Proxy

The following procedure is a general method for configuring a network proxy. It may not be applicable to all network environments. The method of configuring the network proxy depends on the actual network environment.

Prerequisites

- The network cable of the server has been connected and the proxy server can connect to the external network.
- The server is located on an intranet and cannot be directly connected to the external network.

Proxy Configuration

Step 1  Log in to the user environment as the root user.

Step 2  Edit the `/etc/profile` file:
   
   vi /etc/profile

   Append the following lines to the file, save the file, and exit:
   
   ```
   export http_proxy="http://user:password@proxyserverip:port"
   export https_proxy="http://user:password@proxyserverip:port"
   ```

   In the preceding commands, `user` indicates the username on the intranet, `password` indicates the user password, `proxyserverip` indicates the IP address of the proxy server, and `port` indicates the port number.

Step 3  Make the configuration take effect:
   
   source /etc/profile
Step 4  Check the Internet connectivity:

    wget www.google.com

If the HTML file can be downloaded, the server is connected to the external network successfully.

----End

15.7 Security Hardening Suggestions

- Ports 63342 to 63391 are built-in server ports of MindStudio and can be used to open local web services, HTML files, and XML files. To improve security, it is recommended that the port range be used only locally and not open to other devices on the firewall.
- Ports 6942 to 6991 are bound to a single instance of the MindStudio process to prevent multiple IDE processes from being started. These ports are not used for communication. To improve security, it is recommended that the port range be used only locally and not open to other devices on the firewall.
- When an application project is executed remotely, related files are transferred to the remote end. To improve security, you are advised to set the umask value of the remote user to 0027.

15.8 FAQs

15.8.1 What Do I Do If the Copied Content Cannot Be Pasted to the Editor Window Occasionally When MindStudio Is Opened Remotely on Windows?

Symptom

When MindStudio is remotely opened on Windows, the copied content cannot be pasted to the editor window occasionally.

Solution

MobaXterm supports the `copy on select` function. Therefore, MindStudio also supports this function. You can disable the `copy on select` function, as shown in the following figure.
15.8.2 What Do I Do If the Arrow Keys Occasionally Change to Numbers When MobaXterm Starts MindStudio to Edit Code?

**Symptom**

When the MobaXterm starts MindStudio to edit code or enters characters on the MindStudio configuration page, the arrow keys occasionally change to numbers.

**Solution**

Uninstall Sogou Pinyin or switch to another input mode, and then use MobaXterm to start MindStudio remotely.

15.8.3 How Do I Allow Unsigned Access Requests?

After the UT cases are executed, if you need to open the coverage HTML file using a browser, the prompt message shown in Figure 15-22 may be displayed.

**Figure 15-22 Error message**

Choose **File > Settings** from the main menu and modify the parameters by referring to Figure 15-23.
15.8.4 What Do I Do If the GUI Cannot Be Displayed When MindStudio Is Started?

**Symptom**

Error message "Failed to initialize graphics environment" or "Unable to detect graphics environment" is displayed when MindStudio is started. See Figure 15-24 or Figure 15-25.
Figure 15-24 Failure to display the GUI 1

Cause

- If Ubuntu Server is installed, after a user logs in to MobaXterm, the graphical environment has been occupied by the user group to which the user belongs. After the user changes, the graphical environment becomes unavailable.
- If Ubuntu Desktop is installed, the user for logging in to the Ubuntu system is different from that for launching MindStudio.

Solution

- If Ubuntu Server is installed, perform the following operations:
  - Add the new user to the user group of the original user.
  - Log in to MobaXterm as the user who launches MindStudio.
  - Set the DISPLAY environment variable as the user who launches MindStudio.
    
    ```bash
    export DISPLAY=host IP that displays the GUI:0
    ```
  - Open the .bashrc file as the user who launches MindStudio, add "export DISPLAY=host IP that displays the GUI:0" to the file, save the file, and exit. Then, run the `source ~/.bashrc` command.
- If Ubuntu Desktop is installed, perform the following operations:
  - Log in to the Ubuntu system as the user who launches MindStudio.

Figure 15-25 Failure to display the GUI 2
15.8.5 How Do I Export a TensorFlow .pb Model?

If your model is built using Keras, you can try the following method to export a PB model:

- **For TensorFlow 1.15.x:**
  ```python
  import tensorflow as tf
  from tensorflow.python.framework import graph_io
  from tensorflow.python.keras.applications.inception_v3 import InceptionV3

  def freeze_graph(graph, session, output_nodes, output_folder: str):
      ""
      Freeze graph for tf 1.x.x.
      Args:
        graph (tf.Graph): Graph instance.
        session (tf.Session): Session instance.
        output_nodes (list): Output nodes name.
        output_folder (str): Output folder path for frozen model.
      ""
      with graph.as_default():
          graphdef_inf = tf.graph_util.remove_training_nodes(graph.as_graph_def())
          graphdef_frozen = tf.graph_util.convert_variables_to_constants(session, graphdef_inf, output_nodes)
          graph_io.write_graph(graphdef_frozen, output_folder, "frozen_model.pb", as_text=False)
  
  tf.keras.backend.set_learning_phase(0)
  keras_model = InceptionV3()
  session = tf.keras.backend.get_session()
  INPUT_NODES = [ipt.op.name for ipt in keras_model.inputs]
  OUTPUT_NODES = [opt.op.name for opt in keras_model.outputs]
  freeze_graph(session.graph, session, OUTPUT_NODES, "/home/user/xxx")
  
  - **For TensorFlow 2.x.x:**
  ```python
  import tensorflow as tf
  from tensorflow.python.framework.convert_to_constants import convert_variables_to_constants_v2

  def convert_to_frozen_graph(keras_model: tf.python.keras.models.Model, model_name: str, output_folder: str):
      ""
      Export keras model to frozen model.
      Args:
        keras_model (tensorflow.python.keras.models.Model):
        model_name (str): Model name for the file name.
        output_folder (str): Output folder for saving model.
      ""
      full_model = tf.function(lambda x: keras_model(x))
      full_model = full_model.get_concrete_function(
          tf.TensorSpec(keras_model.inputs[0].shape, keras_model.inputs[0].dtype)
      )
      frozen_func = convert_variables_to_constants_v2(full_model)
      print("Model inputs: (frozen_func.inputs)")
      print("Model outputs: (frozen_func.outputs)")
      tf.io.write_graph(graph_or_graph_def=frozen_func.graph, 
                       logdir=output_folder,
                       name=model_name + '.pb')
  ```
15.8.6 "Enter password to unlock" Is Displayed When a User Performs Operations on MindStudio

**Symptom**

When a user performs operations on MindStudio, a dialog box is displayed, asking you to enter the password:

![Password Dialog](image)

**Solution**

Enter the encrypted SSH password and click **Unlock**. For details about how to encrypt the SSH password, see 12.1.7 Encrypting the SSH Password.

15.8.7 How Do I Solve the Problem that Chinese Characters Are Displayed as Garbled Characters and the GUI Is Incompletely Displayed or Is Not Well-Organized?

**Symptom**

- When an app project is created, Chinese characters are displayed as garbled characters in, for example, the README_CN.md file. See **Figure 15-26**.

**Figure 15-26 Garbled Chinese characters**

![Garbled Chinese Characters](image)
- The GUI is incompletely displayed or is not well-organized. For example, the title cannot be completely displayed, as shown in **Figure 15-27**.

**Figure 15-27** Incompletely displayed title

---

**Solution**

Install the Microsoft YaHei font (msyh.ttc).

### 15.8.8 What Can I Do If the Maximum Line Width of the Code Is Inconsistent with the Reference Value?

#### Symptom

In a file under the project directory, the maximum line width of the code is different from the reference value. The default reference value is 120. See **Figure 15-28**.

**Figure 15-28** Inconsistency between the maximum line width of the code and the reference value
Solution

Step 1  On the MindStudio home page, choose File > Settings > Editor > Color Scheme > Color Scheme Font. The font selection page is displayed. See Figure 15-29.

Figure 15-29 Color Scheme Font

Step 2  Select a font that does not match the current font from the Font drop-down list box and choose Apply > OK. Return to the file in the project directory and check whether the maximum line width of the code is the same as the reference value.

- If yes, no further action is required.
- If no, repeat Step 1 to Step 2 until the maximum line width of the code is the same as the reference value.

**NOTE**

If you cannot find a proper font among the equal-width fonts, deselect Show only monospaced fonts and select another font.

End

15.8.9 What Do I Do If the Browser Cannot Be Connected When I Use MindInsight to View Training Project Logs?

Symptom

In MindInsight, click View to view the training project log information. Open the WebUI and view the real-time training information. However, the browser cannot be connected, as shown in Figure 15-30.
**Figure 15-30** Browser connection failure

```
This site can’t be reached
127.0.0.1 refused to connect.
Try:
  • Checking the connection
  • Checking the proxy and the firewall
ERR_CONNECTION_REFUSED
```

**Solution**

**Step 1** Go to the MindStudio background and edit the `sshd_config` file.
```
vi /etc/ssh/sshd_config
```

**Step 2** Change the value of `AllowAgentForwarding` from **no** to **yes**, save the change, and exit.

**Step 3** Restart the sshd service for the modification to take effect.
```
systemctl restart sshd.service
```

**Step 4** Enable MindInsight visualization again and click **View** to view the training project log information.

----End

15.8.10 How Do I Get Help and Provide Suggestions?

If you have any questions or suggestions when using MindStudio, choose **Help > Ascend Developer Zone** on the menu bar of the MindStudio home page to go to the Ascend forum for feedback. See **Figure 15-31**.

**NOTE**

To use the Ascend Developer Zone function, you need to use a browser. If no browser is available, install one.
Figure 15-31 Ascend Developer Zone