

ModelArts

User Guide for Senior AI Engineers (To Be Offline)

Issue 01
Date 2023-09-04



Copyright © Huawei Technologies Co., Ltd. 2023. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: <https://www.huawei.com>

Email: support@huawei.com

Contents

1 Operation Guide.....	1
2 Data Management (Old Version to Be Terminated).....	3
2.1 Introduction to Data Management.....	3
2.2 Creating a Dataset (Old Version).....	8
2.3 Labeling Data.....	21
2.3.1 Image Classification.....	21
2.3.2 Object Detection.....	26
2.3.3 Image Segmentation.....	32
2.3.4 Text Classification.....	38
2.3.5 Named Entity Recognition.....	41
2.3.6 Text Triplet.....	44
2.3.7 Sound Classification.....	48
2.3.8 Speech Labeling.....	51
2.3.9 Speech Paragraph Labeling.....	53
2.3.10 Video Labeling.....	55
2.4 Importing Data.....	57
2.4.1 Import Operation.....	57
2.4.2 Specifications for Importing Data from an OBS Directory.....	61
2.4.3 Specifications for Importing the Manifest File.....	66
2.5 Exporting Data.....	83
2.6 Modifying a Dataset.....	86
2.7 Publishing a Dataset.....	87
2.8 Deleting a Dataset.....	90
2.9 Managing Dataset Versions.....	90
2.10 Auto Labeling.....	92
2.11 Confirming Hard Examples.....	95
2.12 Auto Grouping.....	98
2.13 Data Features.....	100
2.14 Team Labeling.....	107
2.14.1 Introduction to Team Labeling.....	107
2.14.2 Team Management.....	108
2.14.3 Member Management.....	110
2.14.4 Managing Team Labeling Tasks.....	111

2.15 Data Processing.....	117
2.15.1 Introduction to Data Processing.....	117
2.15.2 Creating a Data Processing Task.....	118
2.15.3 Managing and Viewing Data Processing Tasks.....	120
2.15.4 Built-in Operators.....	121
2.15.4.1 Data Validation.....	121
2.15.4.2 Data Cleansing.....	125
2.15.4.3 Data Selection.....	128
2.15.4.4 Data Selection (Hard Examples).....	131
2.15.4.5 Data Augmentation (Data Amplification).....	136
2.15.4.6 Data Augmentation (Image Generation).....	142
3 Training Management (Old Version).....	147
3.1 Introduction to Model Training.....	147
3.2 Frequently-used Frameworks.....	148
3.3 Creating a Training Job.....	152
3.3.1 Introduction to Training Jobs.....	152
3.3.2 Using Existing Algorithms to Train Models.....	152
3.3.3 Using Frequently-used Frameworks to Train Models.....	156
3.3.4 Using Custom Images to Train Models.....	161
3.4 Stopping or Deleting a Job.....	165
3.5 Managing Training Job Versions.....	166
3.6 Viewing Job Details.....	168
3.7 Managing Job Parameters.....	170
3.8 Adding the Evaluation Code.....	171
3.9 Managing Visualization Jobs.....	175
4 Resource Pools (Old Version to Be Terminated).....	179
5 Custom Images.....	186
5.1 Introduction to Custom Images.....	186
5.2 Creating and Uploading a Custom Image.....	188
5.3 Using Custom Images to Train Models (Old Version to Be Terminated).....	188
5.3.1 Specifications for Custom Images Used for Training Jobs.....	188
5.3.2 Creating a Training Job Using a Custom Image (GPU).....	192
5.3.3 Example: Creating a Training Job Using a Custom Image.....	195
6 Permissions Management.....	198
6.1 Creating a User and Granting Permissions.....	198
6.2 Creating a Custom Policy.....	200
7 Audit Logs.....	202
7.1 Key Operations Recorded by CTS.....	202
7.2 Viewing Audit Logs.....	208

A Change History.....209

1 Operation Guide

ModelArts provides online code compiling environments as well as AI development lifecycle that covers data preparation, model training, model management, and service deployment for developers who are familiar with code compilation, debugging, and common AI engines, helping the developers build models efficiently and quickly.

This document describes how to perform AI development on the ModelArts management console. If you use the APIs or SDKs for development, view [ModelArts SDK Reference](#) or [ModelArts API Reference](#).

To view the examples of AI development lifecycle, see [Getting Started](#) and [Best Practices](#).

AI Development Lifecycle

AI development lifecycle provided by ModelArts takes developers' habits into consideration and provides a variety of engines and scenarios for developers to choose. The following describes the entire process from data preparation to service development using ModelArts.

Table 1-1 Process description

Task	Sub Task	Description	Reference
Development	Creating a Notebook Instance	Create a notebook instance as the development environment.	Creating a Notebook Instance
	Compiling Debugging Code	Compile code in an existing notebook to directly build a model.	Introduction to JupyterLab and Common Operations
Training a Model	Selecting an Algorithm	Before creating a training job, you need to select an algorithm. You can subscribe to a preset ModelArts algorithm or use your own algorithm.	Introduction to Algorithm Preparation

Task	Sub Task	Description	Reference
	Creating a Training Job	Create a training job, and use the compiled training script. After training is complete, a model is generated and stored in OBS.	Creating a Training Job
Managing AI Applications	Compile Inference Code and Configuration Files	Following the model package specifications provided by ModelArts, compile inference code and configuration files for your model, and save the inference code and configuration files to the training output location.	Introduction to Model Package Specifications
	Creating an AI Application	Import a trained model to ModelArts to create an AI application, facilitating AI application deployment and publishing.	Creating an AI Application
Deploying AI Applications	Deploying a Model as a Service	Deploy a model as a real-time service or a batch service.	<ul style="list-style-type: none"> • Deploying as a Real-Time Service • Deploying as a Batch Service
	Accessing the Service	If the model is deployed as a real-time service, you can access and use the service. If the model is deployed as a batch service, you can view the prediction result.	<ul style="list-style-type: none"> • Accessing Real-Time Services • Viewing the Batch Service Prediction Result

2 Data Management (Old Version to Be Terminated)

2.1 Introduction to Data Management

NOTE

ModelArts provides both new and old versions of datasets. This section describes the dataset and data management functions of the old version.

Datasets of the old version are to be taken offline. You are advised to use datasets of the new version and related functions.

The new version decouples dataset creation and labeling task creation. Datasets and labeling tasks are separately created.

For the old version, you need to create a labeling task when creating a dataset.

In ModelArts, you can import and label data on the **Data Management** page to prepare for model building. ModelArts uses datasets as the basis for model development or training.




Dataset Types

ModelArts supports datasets of images, audio, text, tables, videos, and other types for the following purposes:

- Images
 - Image classification: identifies a class of objects in images.
 - Object detection: identifies the position and class of each object in an image.
 - Image segmentation: identifies the outline of each object in an image.
- Audio
 - Sound classification: classifies and identifies different sounds.
 - Speech labeling: labels speech content.
 - Speech paragraph labeling: segments and labels speech content.
- Text

- Text classification: assigns labels to text according to its content.
- Named entity recognition: assigns labels to named entities in text, such as time and locations.
- Text triplet: assigns labels to entity segments and entity relationships in the text.
- Tables
 - Table: applies to structured data processing such as tables. The file format can be CSV. Tables cannot be labeled but you can preview up to 100 data records in a table.
- Videos
 - Video labeling: identifies the position and class of each object in a video. Only the MP4 format is supported.
- Others
 - Free format: manages data in any format. Labeling is not available for data of the free format type. The free format type applies to scenarios where labeling is not required or developers customize labeling. Select this format if your data is in multiple formats or your data is not in any of the preceding formats.

Figure 2-1 Example of a dataset in free format

<input type="checkbox"/> File 	Size 	Format 
<input type="checkbox"/> image003_1594105979724.jpeg	90.41 KB	jpeg
<input type="checkbox"/> image005_1594105981161.gif	341.28 KB	gif
<input type="checkbox"/> iris_1592881356613.csv	2.36 KB	csv
<input type="checkbox"/> train_1592881356925.csv	8.26 KB	csv
<input type="checkbox"/> vedio_1594106046159.MP4	430.71 KB	mp4

Size Limit

- A video, text, or audio dataset can have a maximum size of 5 GB.
- For an image dataset for object detection, image classification, or image segmentation, a single image can have a maximum size of 25 MB.
- A manifest file can have a maximum size of 5 GB.
- The maximum size of a line in a text file is 100 KB.
- A data labeling result file can have a maximum size of 100 MB.

Dataset Management Process and Functions

Figure 2-2 Labeling management process

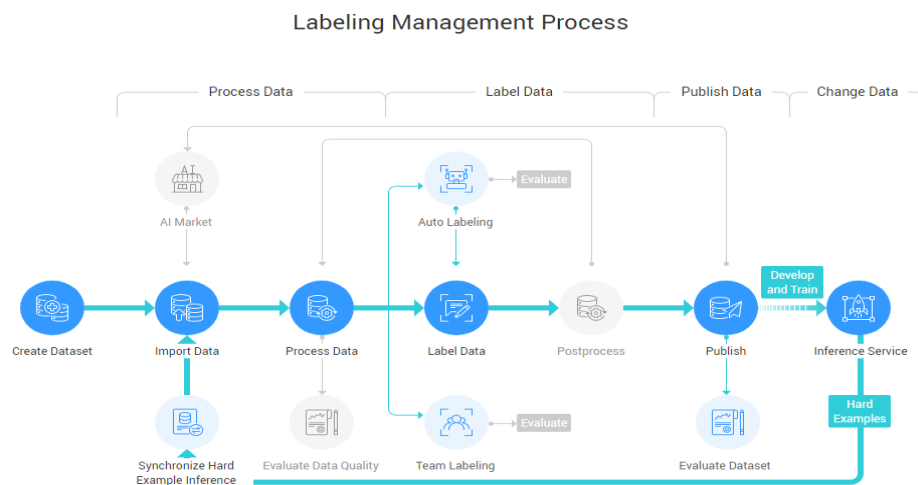


Table 2-1 Function description

Function	Description
Creating a Dataset (Old Version)	Create a dataset.
Image Classification Object Detection Text Classification Named Entity Recognition Text Triplet Sound Classification Speech Labeling Speech Paragraph Labeling Video Labeling	Label data based on the types of datasets. Data labeling is not supported for datasets in free format or table format.
Import Operation	Import data to the dataset.
Exporting Data	Export part of the data as a new dataset or to OBS. Historical tasks can be viewed and managed.
Modifying a Dataset	Modify the basic information about a dataset, such as the dataset name, description, and labels.

Function	Description
Publishing a Dataset	Publish the labeled dataset as a new version for model building.
Managing Dataset Versions	View data version updates.
Auto Labeling	quickly label unlabeled data, reducing labeling time by 70%
Auto Grouping	Enable auto grouping to improve data labeling efficiency.
Data Features	Analyze data features to help you understand data.
Introduction to Team Labeling	Allow multiple users to label the same dataset and enable the dataset creator to manage labeling tasks in a unified manner. Add a team and its members to participate in labeling datasets.
Deleting a Dataset	Delete a dataset to release resources.

Functions Supported by Different Types of Datasets

Different types of datasets provide different functions. For details, see [Table 2-2](#).

Table 2-2 Functions supported by different types of datasets

Dataset Type	Creating a Dataset	Importing Data	Exporting Data	Publishing a Dataset	Modifying a Dataset	Managing Dataset Versions	Auto Labeling	Team Labeling	Auto Grouping	Data Features	Deploying a Model in One Click
Image classification	Supported	Supported	Supported	Supported	Supported	Supported	Supported	Supported	Supported	Supported	Supported
Object detection	Supported	Supported	Supported	Supported	Supported	Supported	Supported	Supported	Supported	Supported	Supported

Dataset Type	Creating a Dataset	Importing Data	Exporting Data	Publishing a Dataset	Modifying a Dataset	Managing Dataset Versions	Auto Labeling	Team Labeling	Auto Grouping	Data Features	Deploying a Model in One Click
Image segmentation	Supported	Supported	Supported	Supported	Supported	Supported	N/A	N/A	Supported	N/A	N/A
Sound classification	Supported	Supported	N/A	Supported	Supported	Supported	N/A	N/A	N/A	N/A	N/A
Speech labeling	Supported	Supported	N/A	Supported	Supported	Supported	N/A	N/A	N/A	N/A	N/A
Speech paragraph labeling	Supported	Supported	N/A	Supported	Supported	Supported	N/A	Supported	N/A	N/A	N/A
Text classification	Supported	Supported	N/A	Supported	Supported	Supported	N/A	Supported	N/A	N/A	N/A
Named entity recognition	Supported	Supported	N/A	Supported	Supported	Supported	N/A	Supported	N/A	N/A	N/A
Text triplet	Supported	Supported	N/A	Supported	Supported	Supported	N/A	Supported	N/A	N/A	N/A

Dataset Type	Creating a Dataset	Importing Data	Exporting Data	Publishing a Dataset	Modifying a Dataset	Managing Dataset Versions	Auto Labeling	Team Labeling	Auto Grouping	Data Features	Deploying a Model in One Click
Table	Supported	Supported	N/A	Supported	Supported	Supported	N/A	N/A	N/A	N/A	N/A
Video	Supported	Supported	N/A	Supported	Supported	Supported	N/A	N/A	N/A	N/A	N/A
Free format	Supported	N/A	Supported	Supported	Supported	Supported	N/A	N/A	N/A	N/A	N/A

2.2 Creating a Dataset (Old Version)

To manage data using ModelArts, you need to create a dataset first. Then you can perform operations on the dataset, such as labeling data, importing data, and publishing the dataset.

NOTE

ModelArts provides both new and old versions of datasets.

The new version decouples dataset creation and labeling task creation. Datasets and labeling tasks are separately created.

For the old version, you need to create a labeling task when creating a dataset.

This section describes how to create a dataset of the old version.

Prerequisites

- Before using the data management function, you need permissions to access OBS. This function cannot be used if you are not authorized to access OBS. Before using the data management function, go to the **Settings** page and complete access authorization using an agency.
- You have created OBS buckets and folders for storing data. In addition, the OBS buckets and ModelArts are in the same region.
- You have uploaded data to be used to OBS.

Procedure

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management** > **Datasets**. The **Datasets** page is displayed.

2. Click **Create Dataset**. On the **Create Dataset** page, create datasets of different types based on the data type and data labeling requirements.
 - a. Set the basic information, the name and description of the dataset.

Figure 2-3 Basic information about a dataset

* Name ✓

Description

0/256

- b. Select a labeling scene and type as required. For details about the types supported by ModelArts, see [Dataset Types](#).

Figure 2-4 Selecting a labeling scene and type

* Labeling Scene Images Audio Text Table Video Other

* Labeling Type

Image classification

Identify if an image contains a class of object.

cat
 dog

Object detection

Identify the position and class of each object in an I...

- c. Set the parameters based on the dataset type. For details, see the parameters of the following dataset types:
 - [Images \(Image Classification, Object Detection, and Image Segmentation\)](#)
 - [Audio \(Sound Classification, Speech Labeling, and Speech Paragraph Labeling\)](#)
 - [Text \(Text Classification, Named Entity Recognition, and Text Triplet\)](#)
 - [Table](#)
 - [Video](#)
 - [Other \(Free Format\)](#)
 - d. Click **Create** in the lower right corner of the page.
 After the dataset is created, the dataset management page is displayed. You can perform the following operations on the dataset: label data, publish dataset versions, manage dataset versions, modify the dataset, import data, and delete the dataset. For details about the operations supported by different types of datasets, see [Functions Supported by Different Types of Datasets](#).

Images (Image Classification, Object Detection, and Image Segmentation)

Figure 2-5 Parameters of datasets for image classification and object detection

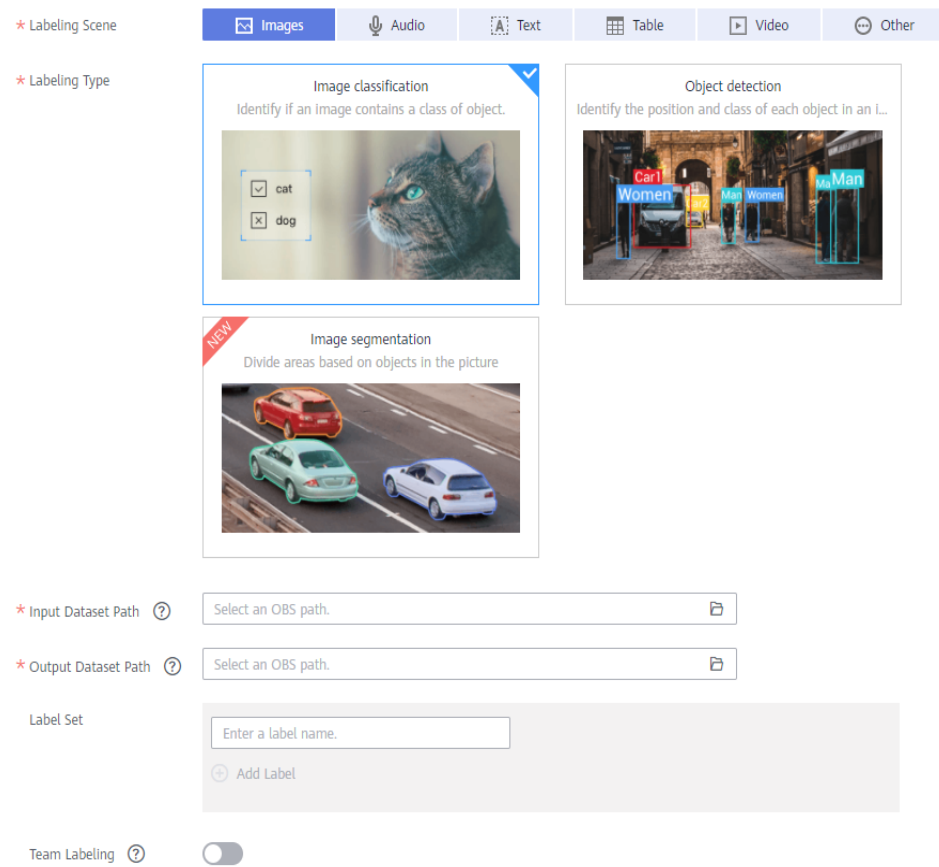


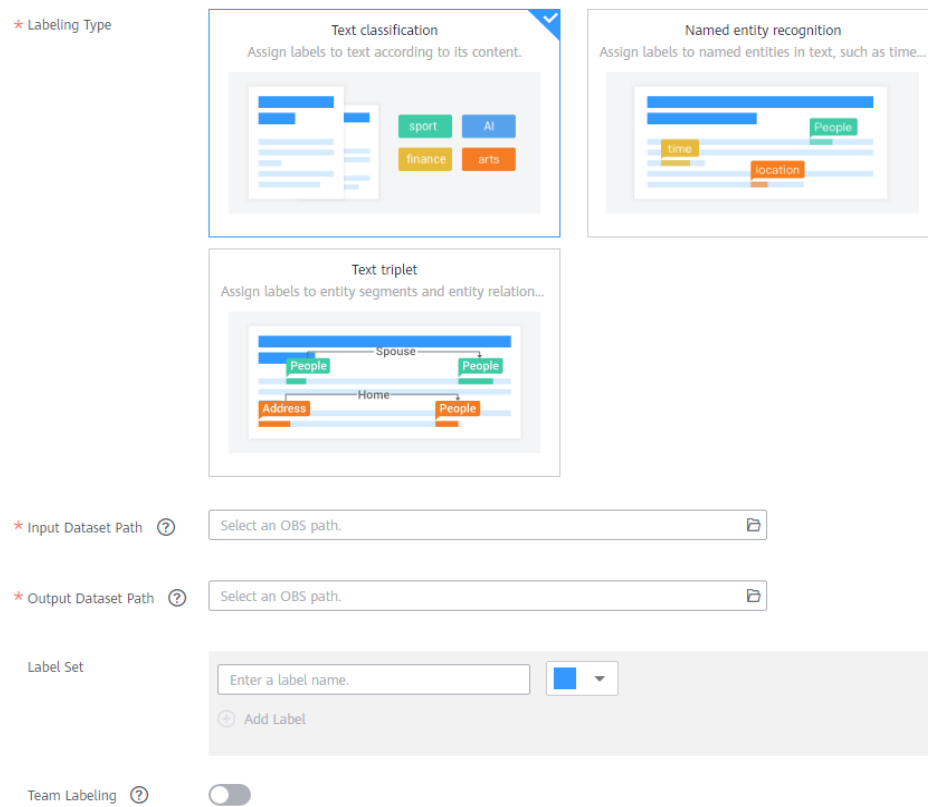
Table 2-3 Dataset parameters

Parameter	Description
Input Dataset Path	Select the OBS path to the input dataset. NOTE When you create a dataset, data in the OBS path will be imported to the dataset. If you modify data in OBS, the data in the dataset will be inconsistent with that on OBS. As a result, some data may be unavailable. If you need to modify data in a dataset, use the Synchronizing Data Sources or Import Operation function.
Output Dataset Path	Select the OBS path to the output dataset. NOTE The output dataset path cannot be the same as the input dataset path or cannot be the subdirectory of the input dataset path. Select an empty directory as the Output Dataset Path .

Parameter	Description
Label Set	<ul style="list-style-type: none">● Label name: Enter a label name. The label name can contain only letters, digits, underscores (_), and hyphens (-). The name contains 1 to 32 characters.● Add Label: Click Add Label to add more labels.● Setting a label color: This function is available only for datasets of the object detection type. Select a color from the color palette on the right of a label, or enter the hexadecimal color code to set the color.● Setting label attributes: For an object detection dataset, you can click the plus sign (+) on the right to add label attributes after setting a label color. Label attributes are used to distinguish different attributes of the objects with the same label. For example, yellow kittens and black kittens have the same label cat and their label attribute is color.
Team Labeling	<p>Enable or disable team labeling. Image segmentation does not support team labeling. Therefore, this parameter is unavailable when you use image segmentation.</p> <p>To enable the team labeling function, you need to enter the name and type of the team labeling task, and select the labeling team and team members. For details about the parameter settings, see Creating Team Labeling Tasks.</p> <p>Before enabling the team labeling function, ensure that you have added a team and members on the Labeling Teams page. If no labeling team is available, click the link on the page to go to the Labeling Teams page, and add your team and members. For details, see Introduction to Team Labeling.</p> <p>After a dataset is created with team labeling enabled, you can view the Team Labeling mark in Labeling Type.</p>

Audio (Sound Classification, Speech Labeling, and Speech Paragraph Labeling)

Figure 2-6 Parameters of datasets for sound classification, speech labeling, and speech paragraph labeling



Parameter	Description
Input Dataset Path	Select the OBS path to the input dataset.
Output Dataset Path	Select the OBS path to the output dataset. NOTE The output dataset path cannot be the same as the input dataset path or cannot be the subdirectory of the input dataset path. Select an empty directory as the Output Dataset Path .
Label Set (Sound Classification)	You need to set labels only for datasets of the sound classification type. <ul style="list-style-type: none"> Label name: Enter a label name. The label name can contain only letters, digits, underscores (_), and hyphens (-). The name contains 1 to 32 characters. Add Label: Click Add Label to add more labels.

Parameter	Description
Label Management (Speech Paragraph Labeling)	<p>Only datasets for speech paragraph labeling support multiple labels.</p> <ul style="list-style-type: none"> • Single Label A single label is used to label a piece of audio that has only one class. <ul style="list-style-type: none"> - Label Name: Enter a label name. The label name can contain contains 1 to 32 characters. Only letters, digits, underscores (_), and hyphens (-) are allowed. - Label Color: Set the label color in the Label Color column. You can select a color from the color palette or enter a hexadecimal color code to set the color. • Multiple Labels Multiple labels are suitable for multi-dimensional labeling. For example, you can label a piece of audio as both noise and speech. For speech, you can label the audio with different speakers. You can click Add Label Class to add multiple label classes. A label class can contain multiple labels. The label class and name can contain contains 1 to 32 characters. Only letters, digits, underscores (_), and hyphens (-) are allowed. <ul style="list-style-type: none"> - Label Class: Set a label class. - Label Name: Enter a label name. - Add Label: Click Add Label to add more labels.
Speech Labeling (Speech Paragraph Labeling)	<p>Only datasets for speech paragraph labeling support speech labeling. By default, speech labeling is disabled. If this function is enabled, you can label speech content.</p>
Team Labeling	<p>Only datasets of speech paragraph labeling support team labeling.</p> <p>After enabling team labeling, you need to set the name and type of the team labeling task, and select the team and team members. For details about the parameter settings, see Creating Team Labeling Tasks.</p> <p>Before enabling the team labeling function, ensure that you have added a team and members on the Labeling Teams page. If no labeling team is available, click the link on the page to go to the Labeling Teams page, and add your team and members. For details, see Introduction to Team Labeling.</p> <p>After a dataset is created with team labeling enabled, you can view the Team Labeling mark in Labeling Type.</p>

Text (Text Classification, Named Entity Recognition, and Text Triplet)

Figure 2-7 Parameters of datasets for text classification, named entity recognition, and text triplet

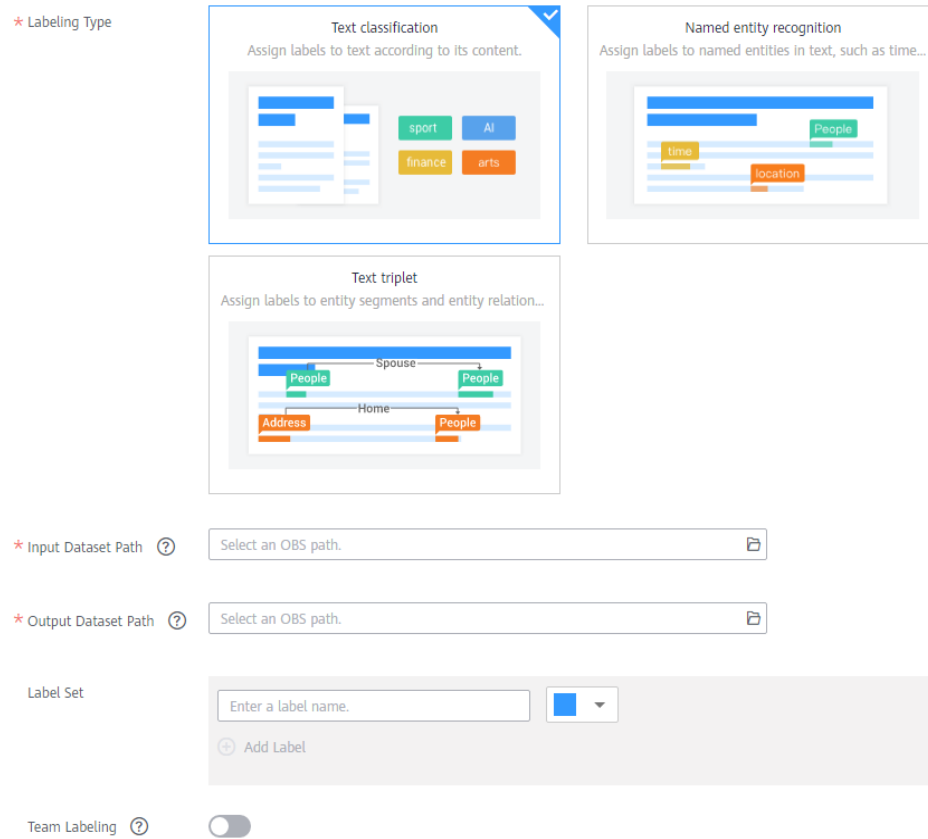
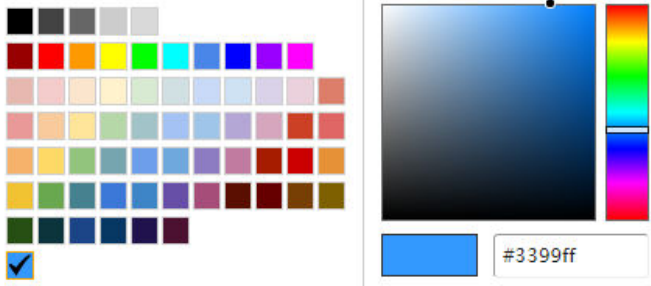
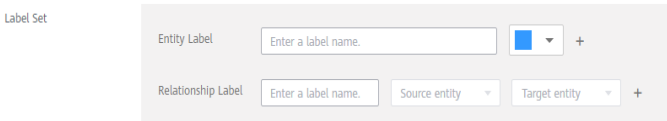


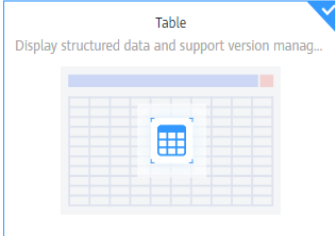
Table 2-4 Dataset parameters



Parameter	Description
Input Dataset Path	Select the OBS path to the input dataset. NOTE Labeled text classification data can be identified only when you import data. When creating a dataset, set an empty OBS directory. After the dataset is created, import the labeled data into it. For details about the format of the data to be imported, see Specifications for Importing Data from an OBS Directory .
Output Dataset Path	Select the OBS path to the output dataset. NOTE The output dataset path cannot be the same as the input dataset path or cannot be the subdirectory of the input dataset path. Select an empty directory as the Output Dataset Path .


Parameter	Description
<p>Label Set (for text classification and named entity recognition)</p>	<ul style="list-style-type: none"> Label name: Enter a label name. The label name can contain only letters, digits, underscores (_), and hyphens (-). The name contains 1 to 32 characters. Add Label: Click Add Label to add more labels. Setting a label color: Select a color from the color palette or enter the hexadecimal color code to set the color. 
<p>Label Set (for text triplet)</p>	<p>For datasets of the text triplet type, you need to set entity labels and relationship labels.</p> <ul style="list-style-type: none"> Entity Label: You need to set the label name and label color. You can click the plus sign (+) on the right of the color area to add multiple labels. Relationship Label: A relationship label is a relationship between two entities. You need to set the source entity and target entity. You need to add at least two entity labels before adding a relationship label. 
<p>Team Labeling</p>	<p>Enable or disable team labeling.</p> <p>To enable the team labeling function, you need to enter the name and type of the team labeling task, and select the labeling team and team members. For details about the parameter settings, see Creating Team Labeling Tasks.</p> <p>Before enabling the team labeling function, ensure that you have added a team and members on the Labeling Teams page. If no labeling team is available, click the link on the page to go to the Labeling Teams page, and add your team and members. For details, see Introduction to Team Labeling.</p> <p>After a dataset is created with team labeling enabled, you can view the Team Labeling mark in Labeling Type.</p>


Table


Figure 2-8 Parameters of datasets of the table type

* Labeling Type  **Table**
Display structured data and support version manag...



* Storage Path  

Import 

Data Source  **OBS** DLI MRS

* File Path 

Contain Table Header

* Schema 
Column Name Type
 Add Schema

NOTE

When using a CSV file, pay attention to the following:

- When the data type is set to **String**, the data in the double quotation marks is regarded as one record by default. Ensure that the double quotation marks in the same row are closed. Otherwise, the data will be too large to display.
- If the number of columns in a row of the CSV file is different from that defined in the schema, the row will be ignored.

Table 2-5 Dataset parameters

Parameter	Description
Storage Path	<p>Select the OBS path for storing table data. The data imported from the data source is stored in this path. The path cannot be the same as or a subdirectory of the file path in the OBS data source.</p> <p>After a table dataset is created, the following four directories are automatically generated in the storage path:</p> <ul style="list-style-type: none"> • annotation: version publishing directory. Each time a version is published, a subdirectory with the same name as the version is generated in this directory. • data: data storage directory. Imported data is stored in this directory. • logs: directory for storing logs • temp: temporary working directory
Import	<p>If you have stored table data on other cloud services, you can enable this function to import data stored on OBS, DLI, or MRS.</p>
Data Source (OBS)	<ul style="list-style-type: none"> • File Path: Browse all OBS buckets of the account and select the directory where the data file to be imported is located. • Contain Table Header: If this parameter is enabled, the imported file contains table headers. In this case, the first row of the imported file is used as the column name. Otherwise, the default column name is added and automatically filled in the schema information. <p>For details about OBS functions, see Object Storage Service Console Operation Guide.</p>
Data Source (DWS)	<ul style="list-style-type: none"> • Cluster Name: The system automatically displays the DWS clusters of the account. You can select a DWS cluster from the drop-down list. • Database Name: Enter the name of the database where the data is located based on the selected DWS cluster. • Table Name: Enter the name of the table where the data is located based on the selected database. • User Name: Enter the username of the DWS cluster administrator. • Password: Enter the password of the DWS cluster administrator. <p>For details about DWS functions, see Data Warehouse Service User Guide.</p> <p>NOTE To import data from DWS, you need to use DLI functions. If you do not have the permission to access DLI, create a DLI agency as prompted.</p>

Parameter	Description
Data Source (DLI)	<ul style="list-style-type: none"> • Queue Name: The system automatically displays the DLI queues of the account. You can select a queue from the drop-down list. • Database Name: All databases are displayed based on the selected queue. Select the required database from the drop-down list. • Table Name: All tables in the selected database are displayed. Select the required table from the drop-down list. <p>For details about DLI functions, see Data Lake Insight User Guide.</p>
Data Source (MRS)	<ul style="list-style-type: none"> • Cluster Name: The system automatically displays the MRS clusters of the account in the list. However, streaming clusters do not support data import. Select the required cluster from the drop-down list. • File Path: Enter the file path based on the selected cluster. The file path is an HDFS path. • Contain Table Header: If this parameter is enabled, the imported file contains table headers. <p>For details about MRS functions, see MapReduce Service User Guide.</p>
Schema	<p>Names and types of table columns, which must be the same as those of the imported data. Set the column name based on the imported data and select the column type. For details about the supported types, see Table 2-6.</p> <p>Click Add Schema to add a new record. When creating a dataset, you must specify a schema. Once created, the schema cannot be modified.</p> <p>When data is imported from OBS, the schema of the CSV file in the file path is automatically obtained. If the schemas of multiple CSV files are inconsistent, an error is reported.</p>

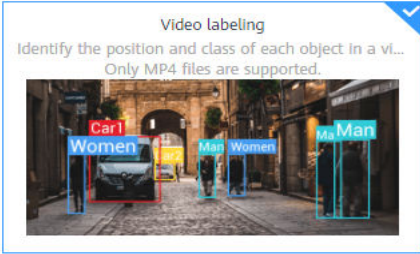
Table 2-6 Migration data types



Type	Description	Storage Space	Range
String	String	-	-
Short	Signed integer	2 bytes	-32768-32767
Int	Signed integer	4 bytes	-2147483648 to 2147483647
Long	Signed integer	8 bytes	-9223372036854775808 to 9223372036854775807



Type	Description	Storage Space	Range
Double	Double-precision floating point	8 bytes	-
Float	Single-precision floating point	4 bytes	-
Byte	Signed integer	1 byte	-128-127
Date	Date type in the format of <i>yyyy-MM-dd</i> , for example, 2014-05-29	-	-
Timestamp	Timestamp that represents date and time. Format: <i>yyyy-MM-dd HH:mm:ss</i>	-	-
Boolean	Boolean	1 byte	TRUE/FALSE

Video


Figure 2-9 Parameters of datasets of the video type

* Labeling Type 

* Input Dataset Path  

* Output Dataset Path  

Label Set




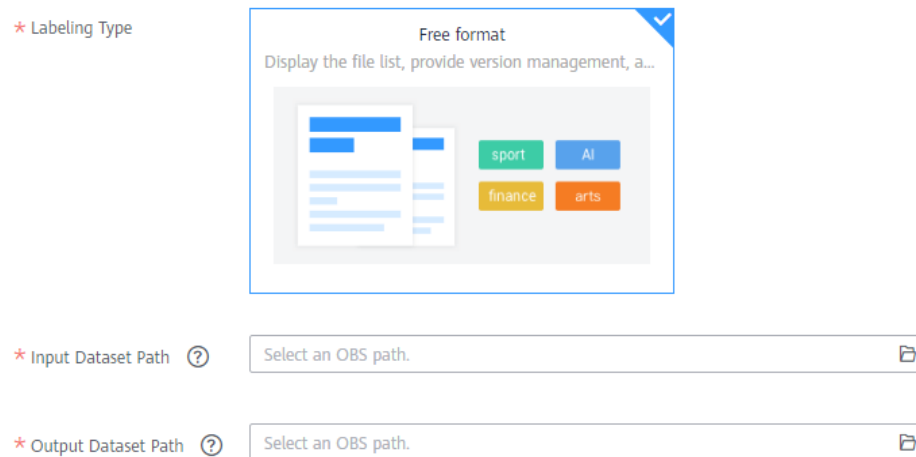
 Add Label

Table 2-7 Dataset parameters

Parameter	Description
Input Dataset Path	Select the OBS path to the input dataset.
Output Dataset Path	Select the OBS path to the output dataset. NOTE The output dataset path cannot be the same as the input dataset path or cannot be the subdirectory of the input dataset path. It is a good practice to select an empty directory for Output Dataset Path .
Label Set	<ul style="list-style-type: none"> • Label name: Enter a label name. The label name can contain only letters, digits, underscores (_), and hyphens (-). The name contains 1 to 32 characters. • Add Label: Click Add Label to add more labels. • Setting a label color: Select a color from the color palette or enter the hexadecimal color code to set the color.

Other (Free Format)

Figure 2-10 Parameters of datasets of the free format type**Table 2-8** Dataset parameters

Parameter	Description
Input Dataset Path	Select the OBS path to the input dataset.
Output Dataset Path	Select the OBS path to the output dataset. NOTE The output dataset path cannot be the same as the input dataset path or cannot be the subdirectory of the input dataset path. It is a good practice to select an empty directory for Output Dataset Path .

2.3 Labeling Data

2.3.1 Image Classification

Model training uses a large number of labeled images. Therefore, before the model training, add labels to the images that are not labeled. You can add labels to images by manual labeling or auto labeling. In addition, you can modify the labels of images, or remove their labels and label the images again.

Before labeling an image in image classification scenarios, pay attention to the following:

- You can add multiple labels to an image.
- A label name can contain a maximum of 32 characters, including letters, digits, hyphens (-), and underscores (_).

Starting Labeling

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. In the dataset list, select the dataset to be labeled based on the labeling type, and click the dataset name to go to the **Dashboard** tab page of the dataset.
By default, the **Dashboard** tab page of the current dataset version is displayed. If you need to label the dataset of another version, click the **Versions** tab and then click **Set to Current Version** in the right pane. For details, see [Managing Dataset Versions](#).
3. On the **Dashboard** page of the dataset, click **Label** in the upper right corner. The dataset details page is displayed. By default, all data of the dataset is displayed on the dataset details page.

Synchronizing Data Sources

ModelArts automatically synchronizes data and labeling information from **Input Dataset Path** to the dataset details page.

- For an image classification dataset, the .txt file with the same name in the same directory as the data source is used as the label of the target image.
- For an object detection dataset or image segmentation dataset, the .xml file with the same name in the same directory is used as the label of the target image.

To quickly obtain the latest data in the OBS bucket, on the **All** or **Unlabeled** tab page of the dataset details page, click **Synchronize Data Source** to add data from OBS to the dataset.

Filtering Data

On the **Dashboard** page of the dataset, click **Label** in the upper right corner. The dataset details page is displayed, showing all data in the dataset by default. On the **All**, **Unlabeled**, or **Labeled** tab page, you can add filter criteria in the filter criteria area to quickly filter the data you want to view.

The following filter criteria are supported. You can set one or more filter criteria.

- **Example Type:** Select **Hard example** or **Non-hard example**.
- **Label:** Select **All** or one or more labels you specified.
- **Sample Creation Time:** Select **Within 1 month**, **Within 1 day**, or **Custom** to customize a time range.
- **File Name** or **Path:** Filter files by file name or file storage path.
- **Labeled By:** Select the name of the user who performs the labeling operation.
- **Sample Attribute:** Select the attribute generated by auto grouping. This filter criterion can be used only after **auto grouping** is enabled.
- **Data Attribute:** This criterion is not available.

Figure 2-11 Filter criteria

Labeling Images (Manually)

The dataset details page displays images on the **All**, **Labeled**, and **Unlabeled** tabs. Images on the **All** tab page are displayed by default. Click an image to preview it. For the images that have been labeled, the label information is displayed at the bottom of the preview page.

1. On the **Unlabeled** tab page, select the images to be labeled.
 - Manual selection: In the image list, click the selection box in the upper left corner of an image to enter the selection mode, indicating that the image is selected. You can select multiple images of the same type and add labels to them together.
 - Batch selection: If all the images on the current page of the image list belong to the same type, you can click **Select Images on Current Page** in the upper right corner to select all the images on the current page.
2. Add labels to the selected images.
 - a. In the label adding area on the right, set a label in the **Label** text box. Click the **Label** text box and select an existing label from the drop-down list. If the existing labels cannot meet the requirements, you can go to the page for **modifying a dataset** and add labels.
 - b. Confirm the **Labels of Selected Image** information and click **OK**. The selected image is automatically moved to the **Labeled** tab page. On the **Unlabeled** and **All** tab pages, the labeling information is updated along with the labeling process, including the added label names and the number of images for each label.

Figure 2-12 Adding labels

Add label Selected 1 image

Label

All Labels 5

Name	Labels
daisy	1
roses	1
dandelion	1
sunflowers	1
tulips	1

Viewing Labeled Images

On the dataset details page, click the **Labeled** tab to view the list of the labeled images. By default, the corresponding labels are displayed under the image thumbnails. You can also select an image and view the label information of the image in the **Labels of Selected Images** area on the right.

Modifying Labeling Information

After labeling data, you can modify labeled data on the **Labeled** tab page.

- **Modifying based on images**



On the dataset details page, click the **Labeled** tab, and select one or more images to be modified from the image list. Modify the image information in the label information area on the right.

Modifying a label: In the **Labels of Selected Images** area, click the edit icon in the **Operation** column, enter the correct label name in the text box, and click the check mark.

Deleting a label: In the **Labels of Selected Images** area, click the delete icon in the **Operation** column to delete the label. This operation deletes only the labels added to the selected image.

Figure 2-13 Modifying a label

Labels of Selected Images

Label	Count	Operation
roses	1	 


- **Modifying based on labels**

On the dataset details page, click the **Labeled** tab. The information about all labels is displayed on the right.

- Modifying a label: Click the edit icon in the **Operation** column. In the dialog box that is displayed, enter the new label name and click **OK**. After the modification, the images that have been added with the label use the new label name.
- Deleting a label: Click the delete icon in the **Operation** column. In the displayed dialog box, select **Delete the label, Delete the label and the images that only have this label, but do not delete source files**, or **Delete the label and the images that only have this label and also delete source files**, and click **OK**.

Figure 2-14 Information about all labels

All Labels 5

Label	Count 
tulips	0
roses	0
sunflowers	0
dandelion	0
daisy	0

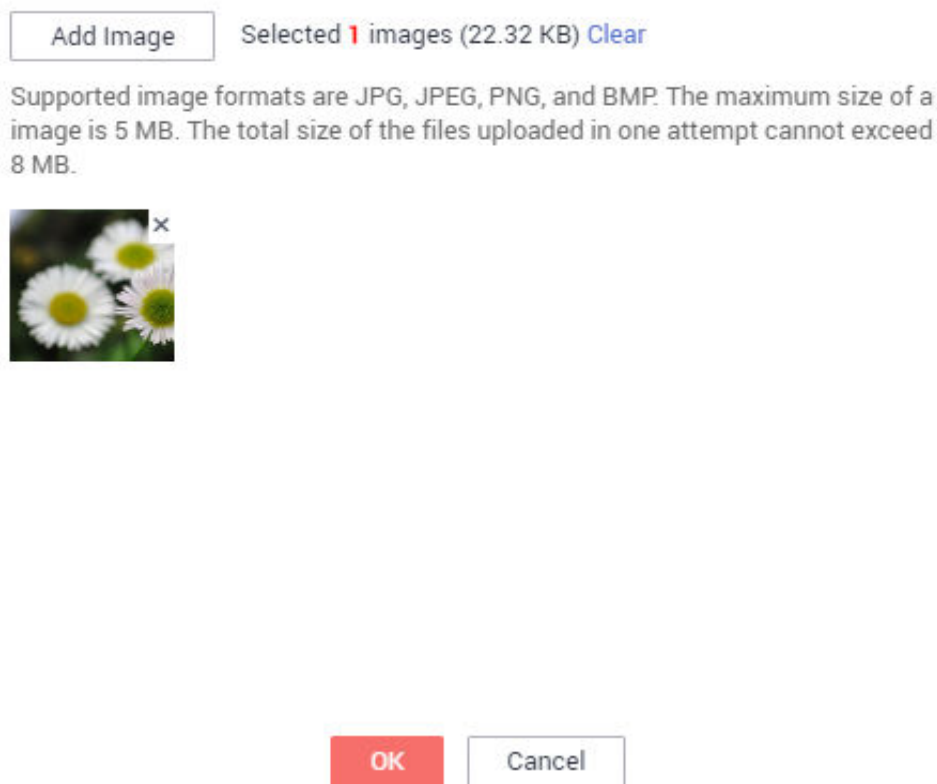
Adding Images

In addition to automatically synchronizing data from **Input Dataset Path**, you can directly add images on ModelArts for data labeling.

1. On the dataset details page, click the **All** or **Unlabeled** tab. Then click **Add**.
2. On the **Add** page that is displayed, click **Add Image**.

Select one or more images to be uploaded in the local environment. Images in JPG, JPEG, PNG, and BMP formats are supported. The size of a single image cannot exceed 5 MB, and the total size of all images uploaded at a time cannot exceed 8 MB.

After the images are selected, their thumbnails and sizes are displayed on the **Add** page.

Figure 2-15 Adding images**Add**

3. On the **Add** page, click **OK**.

The images you have added will be automatically displayed in the image list on the **Unlabeled** tab page. In addition, the images are automatically saved to the OBS directory specified by **Input Dataset Path**.

Deleting Images

You can quickly delete the images you want to discard.

On the **All**, **Unlabeled**, or **Labeled** tab page, select the images to be deleted or click **Select Images on Current Page** to select all images on the page, and click **Delete** in the upper left corner to delete the images. In the displayed dialog box, select or deselect **Delete source files** as required. After confirmation, click **OK** to delete the images.

If a tick is displayed in the upper left corner of an image, the image is selected. If no image is selected on the page, the **Delete** button is unavailable.

NOTE

If you select **Delete source files**, images stored in the OBS directory will be deleted accordingly. This operation may affect other dataset versions or datasets using those files, for example, leading to an error in page display, training, or inference. Deleted data cannot be recovered. Exercise caution when performing this operation.

2.3.2 Object Detection

Training a model uses a large number of labeled images. Therefore, label images before the model training. You can add labels to images by manual labeling or auto labeling. In addition, you can modify the labels of images, or remove their labels and label the images again.

Before labeling an image in object detection scenarios, pay attention to the following:

- All target objects in the image must be labeled.
- Target objects are clear without any blocking and contained within bounding boxes.
- Only the entire object must be contained within a bounding box. The bounding box contains the entire object. The edge of the bounding box cannot intersect the edge outline of the object to be labeled. Ensure that there is no gap between the edge and the object to be labeled to prevent the background from interfering with the model training.

Starting Labeling

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. In the dataset list, select the dataset to be labeled based on the labeling type, and click the dataset name to go to the **Dashboard** tab page of the dataset. By default, the **Dashboard** tab page of the current dataset version is displayed. If you need to label the dataset of another version, click the **Versions** tab and then click **Set to Current Version** in the right pane. For details, see [Managing Dataset Versions](#).
3. On the **Dashboard** page of the dataset, click **Label** in the upper right corner. The dataset details page is displayed. By default, all data of the dataset is displayed on the dataset details page.

Synchronizing Data Sources

ModelArts automatically synchronizes data and labeling information from **Input Dataset Path** to the dataset details page.

- For an image classification dataset, the .txt file with the same name in the same directory as the data source is used as the label of the target image.
- For an object detection dataset or image segmentation dataset, the .xml file with the same name in the same directory is used as the label of the target image.

To quickly obtain the latest data in the OBS bucket, on the **All** or **Unlabeled** tab page of the dataset details page, click **Synchronize Data Source** to add data from OBS to the dataset.

Filtering Data

On the **Dashboard** tab page of the dataset, the summary of the dataset is displayed by default. In the upper right corner of the page, click **Label**. The dataset details page is displayed, showing all data in the dataset by default. On

the **All**, **Unlabeled**, or **Labeled** tab page, you can add filter criteria in the filter criteria area to quickly filter the data you want to view.

The following filter criteria are supported. You can set one or more filter criteria.

- **Example Type:** Select **Hard example** or **Non-hard example**.
- **Label:** Select **All** or one or more labels you specified.
- **Sample Creation Time:** Select **Within 1 month**, **Within 1 day**, or **Custom** to customize a time range.
- **File Name** or **Path:** Filter files by file name or file storage path.
- **Labeled By:** Select the name of the user who performs the labeling operation.
- **Sample Attribute:** Select the attribute generated by auto grouping. This filter criterion can be used only after **auto grouping** is enabled.
- **Data Attribute:** This criterion is not available.

Figure 2-16 Filter criteria

Labeling Images (Manually)







The dataset details page provides the **Labeled** and **Unlabeled** tabs. The **All** tab page is displayed by default.

1. On the **Unlabeled** tab page, click an image. The image labeling page is displayed. For details about how to use common buttons on the **Labeled** tab page, see [Table 2-10](#).
2. In the tool bar, select a proper labeling shape. The default labeling shape is a rectangle. In this example, the rectangle is used for labeling.

NOTE

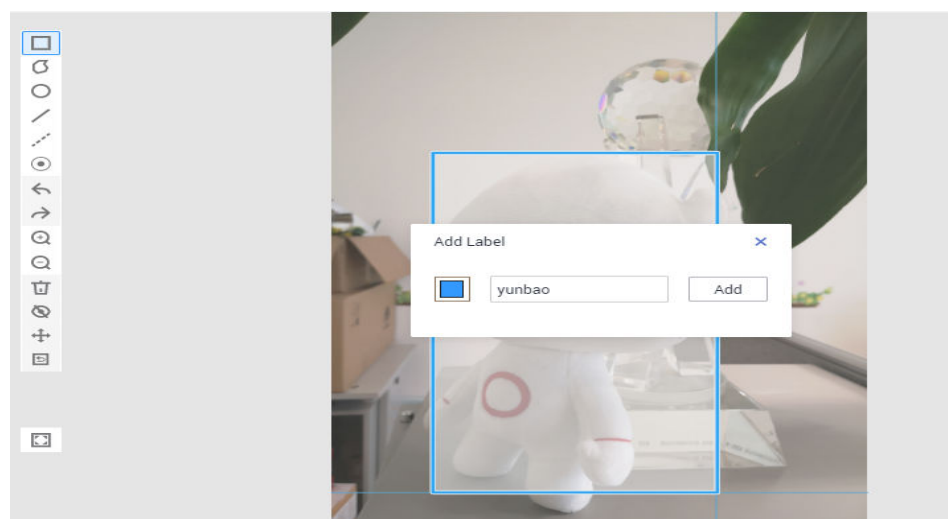
On the left of the page, multiple tools are provided for you to label images. However, you can use only one tool at a time.

Table 2-9 Supported bounding box

Icon	Description
	Rectangle. Click the edge of the upper left corner of the object to be labeled. A rectangle will be displayed. Drag the rectangle to cover the object and click to label the object.
	Polygon. In the area where the object to be labeled is located, click to label a point, move the mouse and click multiple points along the edge of the object, and then click the first point again. All the points form a polygon. Therefore, the object to be labeled is in the bounding box.
	Circle. Click the center point of an object, and move the mouse to draw a circle to cover the object and click to label the object.
	Straight line. Click to specify the start and end points of an object, and move the mouse to draw a straight line to cover the object and click to label the object.
	Dotted line. Click to specify the start and end points of an object, and move the mouse to draw a dotted line to cover the object and click to label the object.
	Point. Click the object in an image to label a point.

3. In the **Add Label** text box, enter a new label name, select the label color, and click **Add**. Alternatively, select an existing label from the drop-down list.










Label all objects in an image. Multiple labels can be added to an image. After labeling an image, click an image that has not been labeled in the image list below to label the new image.

Figure 2-17 Adding an object detection label

4. Click **Back to Data Labeling Preview** in the upper left part of the page to view the labeling information. In the dialog box that is displayed, click **Yes** to save the labeling settings.

The selected image is automatically moved to the **Labeled** tab page. On the **Unlabeled** and **All** tab pages, the labeling information is updated along with the labeling process, including the added label names and the number of images for each label.

Table 2-10 Common icons on the labeling page

Icon	Description
	Cancel the previous operation.
	Redo the previous operation.
	Zoom in an image.
	Zoom out an image.
	Delete all bounding boxes on the current image.
	Display or hide a bounding box. You can perform this operation only on a labeled image.
	Drag a bounding box to another position or drag the edge of the bounding box to resize it.
	Reset. After dragging a bounding box, you can click this button to quickly restore the bounding box to its original shape and position.
	Display the labeled image in full screen.

Viewing Labeled Images

On the dataset details page, click the **Labeled** tab to view the list of the labeled images. You can click an image to view the label information about the image in the **All Labels** area on the right.

Modifying Labeling Information

After labeling data, you can modify labeled data on the **Labeled** tab page.

- **Modifying based on images**

On the dataset details page, click the **Labeled** tab, select the images to be modified, and click the images. The labeling page is displayed. Modify the image information in the label information area on the right.

- **Modifying a label:** In the **Labeling** area, click the edit icon, enter the correct label name in the text box, and click the check mark to complete the modification. Alternatively, click a label. In the image labeling area,

adjust the position and size of the bounding box. After the adjustment is complete, click another label to save the modification.

- Deleting a label: In the **Labeling** area, click the deletion icon to delete a label from the image.

After deleting the label, click **Back to Data Labeling Preview** in the upper left corner of the page to exit the labeling page. In the dialog box that is displayed, save the modification. After all labels of an image are deleted, the image is displayed on the **Unlabeled** tab page.

Figure 2-18 Editing an object detection label

Label	Count	selected images	Operation
yunbao	1		

- **Modifying based on labels**

On the dataset details page, click the **Labeled** tab. The information about all labels is displayed on the right.

- Modifying a label: Click the editing icon in the **Operation** column. In the dialog box that is displayed, enter the new label name, select the new label color, and click **OK**. After the modification, the images that have been added with the label use the new label name.
- Deleting a label: Click the deletion icon in the **Operation** column to delete a label.

Figure 2-19 All labels for object detection

Label	Count	Operation
yunbao	20	

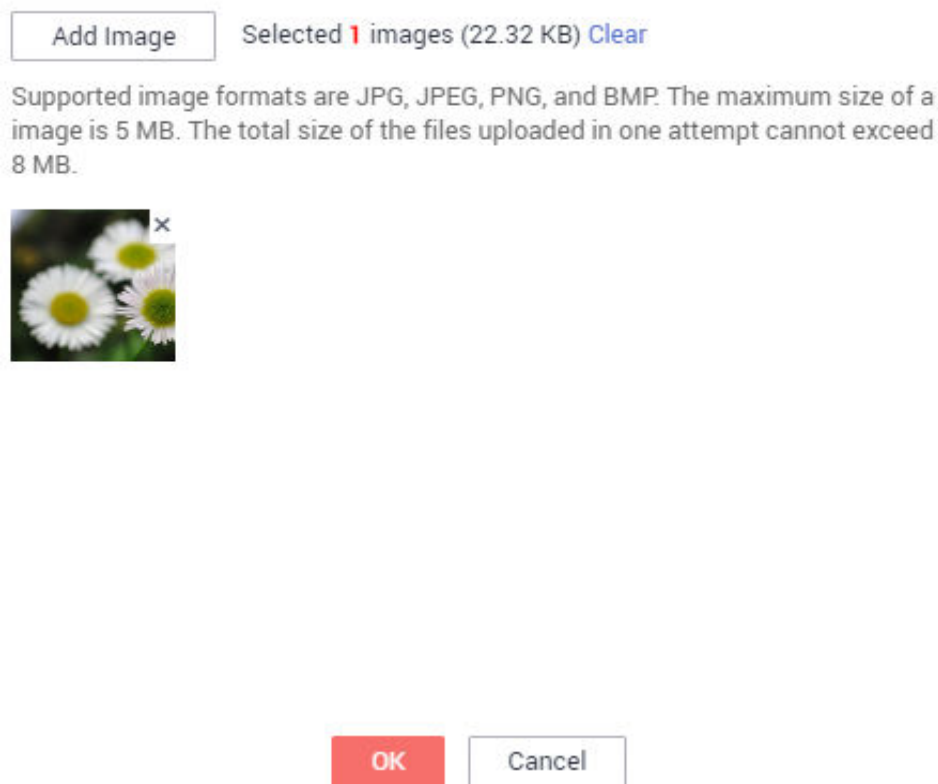
Adding Images

In addition to automatically synchronizing data from **Input Dataset Path**, you can directly add images on ModelArts for data labeling.

1. On the dataset details page, click the **All** or **Unlabeled** tab. Then click **Add**.
2. On the **Add** page that is displayed, click **Add Image**.

Select one or more images to be uploaded in the local environment. Images in JPG, JPEG, PNG, and BMP formats are supported. The size of a single image cannot exceed 5 MB, and the total size of all images uploaded at a time cannot exceed 8 MB.

After the images are selected, their thumbnails and sizes are displayed on the **Add** page.

Figure 2-20 Adding images**Add**

3. On the **Add** page, click **OK**.

The images you have added will be automatically displayed in the image list on the **Unlabeled** tab page. In addition, the images are automatically saved to the OBS directory specified by **Input Dataset Path**.

Deleting Images

You can quickly delete the images you want to discard.

On the **All**, **Unlabeled**, or **Labeled** tab page, select the images to be deleted or click **Select Images on Current Page** to select all images on the page, and click **Delete** in the upper left corner to delete the images. In the displayed dialog box, select or deselect **Delete source files** as required. After confirmation, click **OK** to delete the images.

If a tick is displayed in the upper left corner of an image, the image is selected. If no image is selected on the page, the **Delete** button is unavailable.

NOTE

If you select **Delete source files**, images stored in the OBS directory will be deleted accordingly. This operation may affect other dataset versions or datasets using those files, for example, leading to an error in page display, training, or inference. Deleted data cannot be recovered. Exercise caution when performing this operation.

2.3.3 Image Segmentation

Training a model uses a large number of labeled images. Therefore, label images before the model training. You can label images on the ModelArts management console. Alternatively, modify labels, or delete them and label them again.

Before labeling an image in image segmentation scenarios, understand the following:

- All objects whose contours need to be extracted from the image must be labeled.
- Polygons and points can be used for labeling.
 - In polygon labeling, draw a polygon based on the outline of the target object.
 - In point labeling, label the top, bottom, leftmost, and rightmost points on the object contour. The system will infer the outline of the object based on the labeled points. For images with complex backgrounds, it is a good practice to use polygons for labeling.
- When labeling an image, ensure that the polygons or points are within the image. Otherwise, an error will occur in subsequent operations.

Starting Labeling

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. In the dataset list, select the dataset to be labeled based on the labeling type, and click the dataset name to go to the **Dashboard** tab page of the dataset.
By default, the **Dashboard** tab page of the current dataset version is displayed. If you need to label the dataset of another version, click the **Versions** tab and then click **Set to Current Version** in the right pane. For details, see [Managing Dataset Versions](#).
3. On the **Dashboard** page of the dataset, click **Label** in the upper right corner. The dataset details page is displayed. By default, all data of the dataset is displayed on the dataset details page.

Synchronizing Data Sources

ModelArts automatically synchronizes data and labeling information from **Input Dataset Path** to the dataset details page.

- For an image classification dataset, the .txt file with the same name in the same directory as the data source is used as the label of the target image.
- For an object detection dataset or image segmentation dataset, the .xml file with the same name in the same directory is used as the label of the target image.

To quickly obtain the latest data in the OBS bucket, on the **All** or **Unlabeled** tab page of the dataset details page, click **Synchronize Data Source** to add data from OBS to the dataset.

Filtering Data

On the **Dashboard** tab page of the dataset, the summary of the dataset is displayed by default. In the upper right corner of the page, click **Label**. The dataset details page is displayed, showing all data in the dataset by default. On the **All**, **Unlabeled**, or **Labeled** tab page, you can add filter criteria in the filter criteria area to quickly filter the data you want to view.

The following filter criteria are supported. You can set one or more filter criteria.

- **Example Type:** Select **Hard example** or **Non-hard example**.
- **Label:** Select **All** or one or more labels you specified.
- **Sample Creation Time:** Select **Within 1 month**, **Within 1 day**, or **Custom** to customize a time range.
- **File Name or Path:** Filter files by file name or file storage path.
- **Labeled By:** Select the name of the user who labeled the image.
- **Sample Attribute:** Select the attribute generated by auto grouping. This filter criterion can be used only after **auto grouping** is enabled.
- **Data Attribute:** This criterion is not available.

Figure 2-21 Filter criteria

Filter Criteria	No filter criteria selected.	Hide
Example Type	<input type="radio"/> Hard Example <input type="radio"/> Non-hard example	
Label	<input type="radio"/> All <input type="radio"/> Apple	
Sample Creation Time	<input type="radio"/> Within 1 month <input type="radio"/> Within 1 day <input type="radio"/> Custom	
File Name	<input type="text" value="Enter a keyword and press Enter to create a filter criterion."/> <input type="button" value="x"/>	
Labeled By	<input type="text" value="Select an annotator."/>	
Sample Attribute	<input type="text" value="--Select--"/> No attributes available. Click Auto Grouping and select Start Task to create data management attributes.	
Data Attribute	<input type="text" value="Select an attribute."/> <input type="text" value="Select an attribute ..."/>	

Manually Labeling Images

The dataset details page provides the **Labeled** and **Unlabeled** tabs. The **All** tab page is displayed by default.

1. On the **Unlabeled** tab page, click an image. The system automatically directs you to the page for labeling the image. For details about how to use common buttons on this page, see [Table 2-12](#).
2. Select a labeling method.

On the labeling page, common **labeling methods** and **buttons** are provided in the toolbar. By default, polygon labeling is selected. Use polygon or point labeling as needed.

NOTE

After you select a method to label the first image, the labeling method automatically applies to subsequent images.

Figure 2-22 Toolbar**Table 2-11** Labeling methods












Icon	Description
	Polygon labeling. In the area where the object to be labeled is located, click to label a point, move the mouse and click multiple points along the edge of the object, and then click the first point again. All the points form a polygon. In this way, the object to be labeled is within the bounding box.
	Point labeling. Label the top, bottom, leftmost, and rightmost points on the object contour. The system will infer the outline of the object based on the labeled points.

Table 2-12 Toolbar buttons

Icon	Description
	Cancel the previous operation.
	Redo the previous operation.
	Zoom in an image.
	Zoom out an image.
	Delete all bounding boxes on the current image.
	Display or hide a bounding box. This operation can be performed only on a labeled image.
	Drag a bounding box to another position or drag the edge of the bounding box to resize it.
	Reset a bounding box. After dragging a bounding box, you can click this button to quickly restore the bounding box to its original shape and position.
	Display the labeled image in full screen.

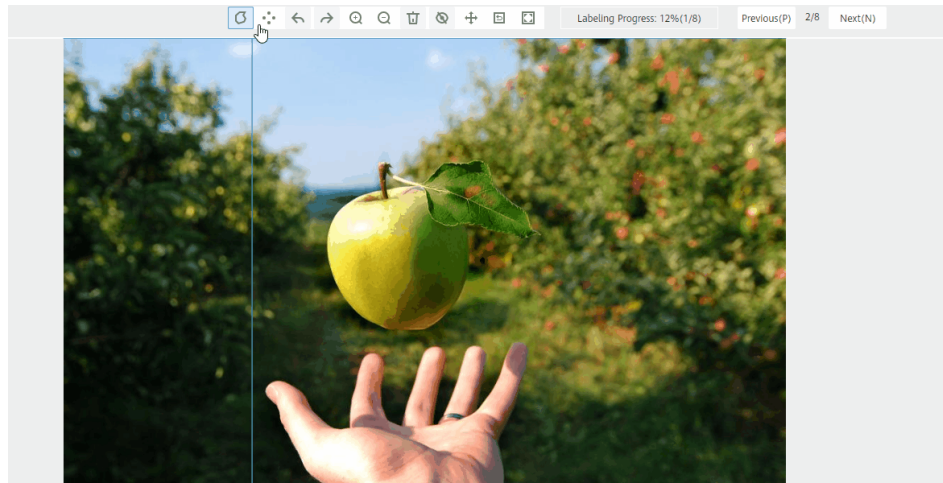
3. Label an object.

This section uses point labeling as an example. Identify an object in an image. Click to label the top, bottom, leftmost, and rightmost points on the object

contour. In the dialog box that is displayed, set the label name and click **Add**. Then, the system automatically infers the object contour.

After labeling an image, click an image that has not been labeled in the image list below to label the new image.

Figure 2-23 Labeling an object outline



4. Click **Back to Data Labeling Preview** in the upper left part of the page to view the labeling information. In the dialog box that is displayed, click **OK** to save the labeling settings.

The selected image is automatically moved to the **Labeled** tab page. On the **Unlabeled** and **All** tab pages, the labeling information is updated along with the labeling process, including the added label names and the number of images for each label.

Viewing Labeled Images

On the dataset details page, click the **Labeled** tab to view the list of labeled images. Click an image to view its labeling information in the **File Labels** area on the right.

Modifying a Label

After labeling an object, you can modify labeled data on the **Labeled** tab page.

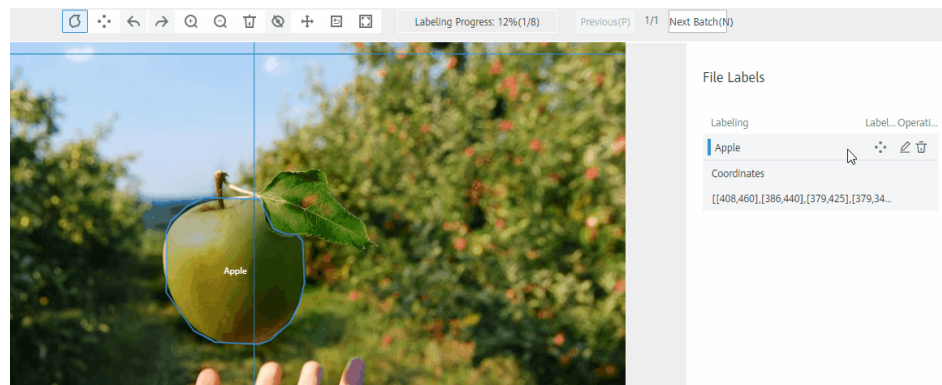
On the dataset details page, click the **Labeled** tab and then the image to be modified. On the labeling page that is displayed, modify the labeling information in the **File Labels** area on the right.

- **Modifying a label:** In the **Labeling** area, click the edit icon, set the target label name or color in the displayed dialog box, and click the save icon to save the modification. Alternatively, click a label to be modified. In the image labeling area, adjust the position and size of the bounding box. After the adjustment is complete, click another label to save the modification.
- **Modifying image labeling information:** In the area for displaying images, click the target bounding box. Then, blue points on the bounding box are displayed. Drag a blue point and adjust the bounding box to the edge of the object.

- Deleting a label: In the **Labeling** area, click the deletion icon to delete a label from the image. After all labels of an image are deleted, the image is displayed on the **Unlabeled** tab page.

After the labeling information is modified, click **Back to Data Labeling Preview** in the upper left part of the page to exit the labeling page. In the dialog box that is displayed, click **OK** to save the modification.

Figure 2-24 Editing image labeling information



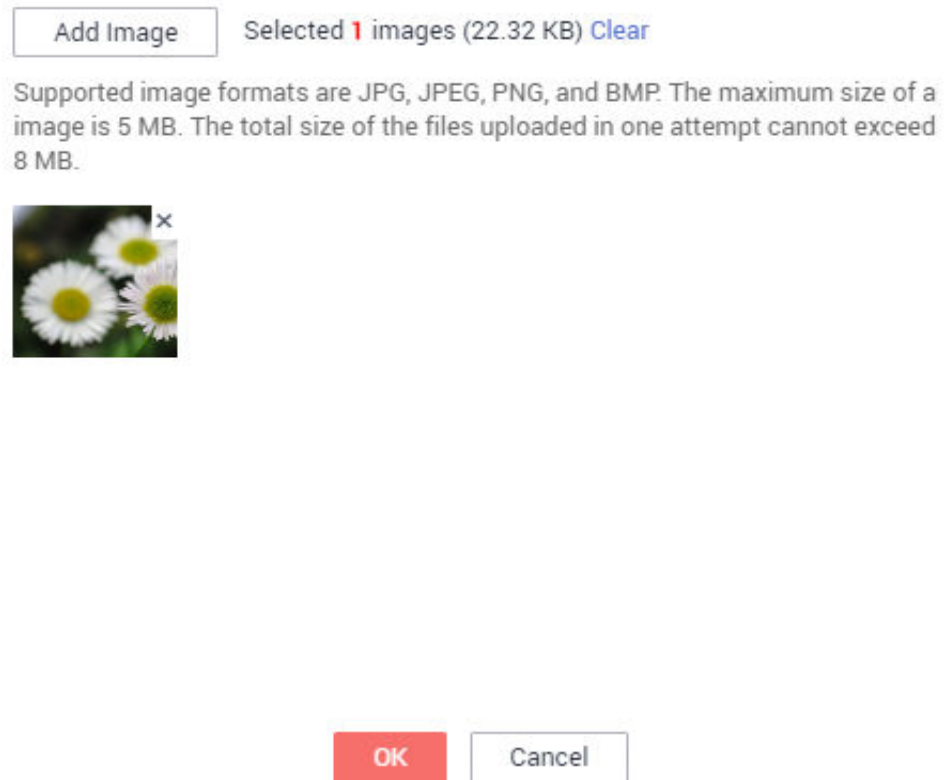
Adding Images

In addition to automatically synchronizing data from **Input Dataset Path**, you can directly add images on ModelArts for data labeling.

1. On the dataset details page, click the **All** or **Unlabeled** tab. Then click **Add**.
2. On the **Add** page that is displayed, click **Add Image**.

Select one or more images to be uploaded in the local environment. Images in JPG, JPEG, PNG, and BMP formats are supported. The size of a single image cannot exceed 5 MB, and the total size of all images uploaded at a time cannot exceed 8 MB.

After the images are selected, their thumbnails and sizes are displayed on the **Add** page.

Figure 2-25 Adding images**Add**

3. On the **Add** page, click **OK**.

The images you have added will be automatically displayed in the image list on the **Unlabeled** tab page. In addition, the images are automatically saved to the OBS directory specified by **Input Dataset Path**.

Deleting Images

You can quickly delete the images you want to discard.

On the **All**, **Unlabeled**, or **Labeled** tab page, select the images to be deleted or click **Select Images on Current Page** to select all images on the page, and click **Delete** in the upper left corner to delete the images. In the displayed dialog box, select or deselect **Delete source files** as required. After confirmation, click **OK** to delete the images.

If a tick is displayed in the upper left corner of an image, the image is selected. If no image is selected on the page, the **Delete** button is unavailable.

NOTE

If you select **Delete source files**, images stored in the OBS directory will be deleted accordingly. This operation may affect other dataset versions or datasets using those files, for example, leading to an error in page display, training, or inference. Deleted data cannot be recovered. Exercise caution when performing this operation.

2.3.4 Text Classification

Model training requires a large amount of labeled data. Therefore, before the model training, add labels to the files that are not labeled. In addition, you can modify, delete, and re-label the labeled text.

Text classification classifies text content based on labels. Before labeling text content, you need to understand the following:

- Text labeling supports multiple labels. That is, you can add multiple labels to a labeling object.
- A label name can contain a maximum of 32 characters, including letters, digits, hyphens (-), and underscores (_).

Starting Labeling

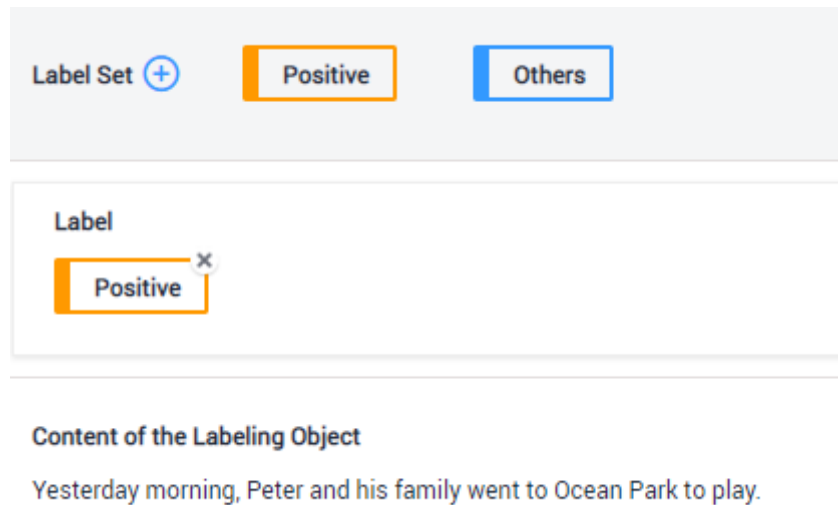
1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. In the dataset list, select the dataset to be labeled based on the labeling type, and click the dataset name to go to the **Dashboard** tab page of the dataset.
By default, the **Dashboard** tab page of the current dataset version is displayed. If you need to label the dataset of another version, click the **Versions** tab and then click **Set to Current Version** in the right pane. For details, see [Managing Dataset Versions](#).
3. On the **Dashboard** page of the dataset, click **Label** in the upper right corner. The dataset details page is displayed. By default, all data of the dataset is displayed on the dataset details page.

Labeling Content

The dataset details page displays the labeled and unlabeled text files in the dataset. The **Unlabeled** tab page is displayed by default.

1. On the **Unlabeled** tab page, the objects to be labeled are listed in the left pane. In the list, click the text object to be labeled, and select a label in the **Label Set** area in the right pane. Multiple labels can be added to a labeling object.

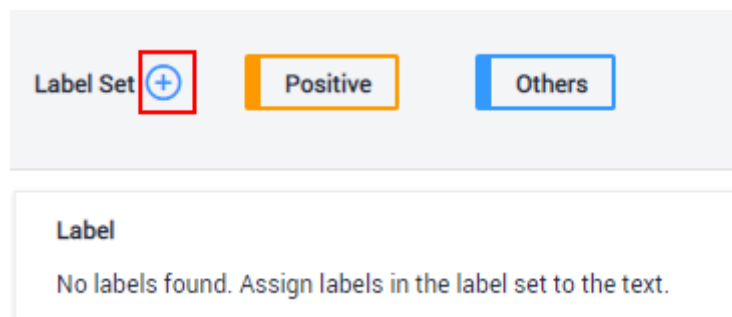
You can repeat this operation to select objects and add labels to the objects.

Figure 2-26 Labeling for text classification

2. After all objects are labeled, click **Save Current Page** at the bottom of the page to complete labeling text files on the **Unlabeled** tab page.

Adding Labels

- Adding labels on the **Unlabeled** tab page: Click the plus sign (+) next to **Label Set**. On the **Add Label** page that is displayed, add a label name, select a label color, and click **OK**.

Figure 2-27 Adding a label (1)

- Adding labels on the **Labeled** tab page: Click the plus sign (+) next to **All Labels**. On the **Add Label** page that is displayed, add a label name, select a label color, and click **OK**.

Figure 2-28 Adding a label (2)

All Labels (+)

Label	Count	Operation
Positive	2	
Others	1	

Figure 2-29 Adding a label**Add Label**

The screenshot shows a dialog box titled "Add Label". It contains a text input field with the placeholder "Enter a label name." and a color selection block with a blue square. Below the input field is a plus icon and the text "Add Label". At the bottom are "OK" and "Cancel" buttons.

Viewing the Labeled Text

On the dataset details page, click the **Labeled** tab to view the list of the labeled text. You can also view all labels supported by the dataset in the **All Labels** area on the right.

Modifying Labeled Data

After labeling data, you can modify labeled data on the **Labeled** tab page.

- **Modifying based on texts**

On the dataset details page, click the **Labeled** tab, and select the text to be modified from the text list.

In the text list, click the text. When the text background turns blue, the text is selected. If a text file has multiple labels, you can click **✕** above a label to delete the label.

- **Modifying based on labels**

On the dataset details page, click the **Labeled** tab. The information about all labels is displayed on the right.

- Batch modification: In the **All Labels** area, click the editing icon in the **Operation** column, modify the label name in the text box, select a label color, and click **OK**.
- Batch deletion: In the **All Labels** area, click the deletion icon in the **Operation** column to delete the label. In the dialog box that is displayed, select **Delete label** or **Delete label and objects with only the label**, and click **OK**.

Adding Files

In addition to automatically synchronizing data from **Input Dataset Path**, you can directly add text files on ModelArts for data labeling.

1. On the dataset details page, click the **Unlabeled** tab. Then click **Add File**.
2. In the displayed **Add File** dialog box, set the parameters as required and then select the file to be uploaded.

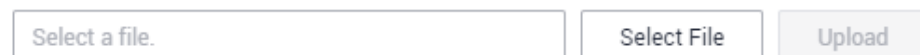
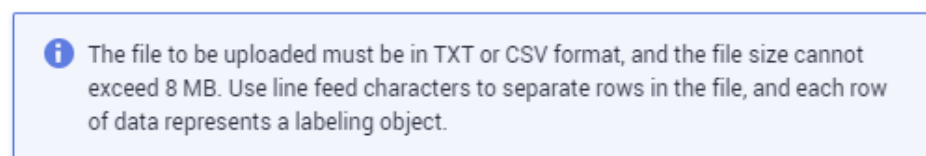
Select one or more files to be uploaded in the local environment. Only **.txt** and **.csv** files are supported. The total size of files uploaded at a time cannot

exceed 8 MB. **Text and Label Separator** and **Label Separator** must be different.

- **Pattern:** Select **Merge text objects and labels** or **Separate text objects and labels**. An example is provided. Determine the mode of the file to be added by referring to the example.
- **Text and Label Separator:** Select **Tab**, **Space**, **Semicolon**, **Comma**, or **Other**. If you select **Other**, enter a separator in the text box on the right.
- **Label Separator:** Select **Tab**, **Space**, **Semicolon**, **Comma**, or **Other**. If you select **Other**, enter a separator in the text box on the right.

Figure 2-30 Adding a file

Add File



3. In the **Add File** dialog box, click **Upload**. The files you add will be automatically displayed on the **Unlabeled** or **Labeled** tab page.

Deleting a File

You can quickly delete the files you want to discard.

- On the **Unlabeled** tab page, select the text to be deleted, and click **Delete** in the upper left corner to delete the text.
- On the **Labeled** tab page, select the text to be deleted and click **Delete**. Alternatively, you can tick **Select Images on Current Page** to select all text objects on the current page and click **Delete** in the upper left corner.

The background of the selected text is blue.

2.3.5 Named Entity Recognition

Named entity recognition assigns labels to named entities in text, such as time and locations. Before labeling, you need to understand the following:

- A label name can contain a maximum of 32 characters, including letters, digits, hyphens (-), and underscores (_).

Starting Labeling

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. In the dataset list, select the dataset to be labeled based on the labeling type, and click the dataset name to go to the **Dashboard** tab page of the dataset.

By default, the **Dashboard** tab page of the current dataset version is displayed. If you need to label the dataset of another version, click the **Versions** tab and then click **Set to Current Version** in the right pane. For details, see [Managing Dataset Versions](#).

3. On the **Dashboard** page of the dataset, click **Label** in the upper right corner. The dataset details page is displayed. By default, all data of the dataset is displayed on the dataset details page.

Labeling Content

The dataset details page displays the labeled and unlabeled text files in the dataset. The **Unlabeled** tab page is displayed by default.

1. On the **Unlabeled** tab page, the objects to be labeled are listed in the left pane. In the list, click the text object to be labeled, select a part of text displayed under **Label Set** for labeling, and select a label in the **Label Set** area in the right pane. Multiple labels can be added to a labeling object. You can repeat this operation to select objects and add labels to the objects.

Figure 2-31 Labeling for named entity recognition

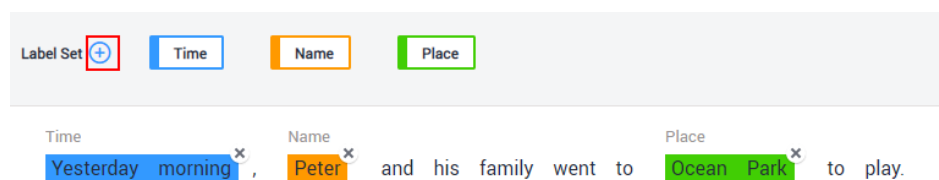


2. Click **Save Current Page** in the lower part of the page to complete the labeling.

Adding Labels

- Adding labels on the **Unlabeled** tab page: Click the plus sign (+) next to **Label Set**. On the **Add Label** page that is displayed, add a label name, select a label color, and click **OK**.

Figure 2-32 Adding a named entity label (1)



- Adding labels on the **Labeled** tab page: Click the plus sign (+) next to **All Labels**. On the **Add Label** page that is displayed, add a label name, select a label color, and click **OK**.

Figure 2-33 Adding a named entity label (2)

All Labels +


Label	Count	Operation
Time	1	
Name	1	
Place	1	

Figure 2-34 Adding a named entity label

Add Label

Label Name

Label Color (Click the color block to set the label color.)



+ Add Label

OK

Cancel

Viewing the Labeled Text

On the dataset details page, click the **Labeled** tab to view the list of the labeled text. You can also view all labels supported by the dataset in the **All Labels** area on the right.

Modifying Labeled Data

After labeling data, you can modify labeled data on the **Labeled** tab page.

On the dataset details page, click the **Labeled** tab, and modify the text information in the label information area on the right.

- **Modifying based on texts**

On the dataset details page, click the **Labeled** tab, and select the text to be modified from the text list.

Manual deletion: In the text list, click the text. When the text background turns blue, the text is selected. On the right of the page, click above a text label to delete the label.

- **Modifying based on labels**

On the dataset details page, click the **Labeled** tab. The information about all labels is displayed on the right.

- Batch modification: In the **All Labels** area, click the editing icon in the **Operation** column, add a label name in the text box, select a label color, and click **OK**.
- Batch deletion: In the **All Labels** area, click the deletion icon in the **Operation** column to delete the label. In the dialog box that is displayed, select **Delete label** or **Delete label and objects with only the label**, and click **OK**.

Adding Files

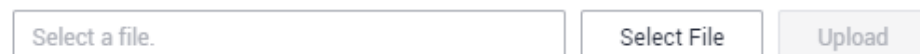
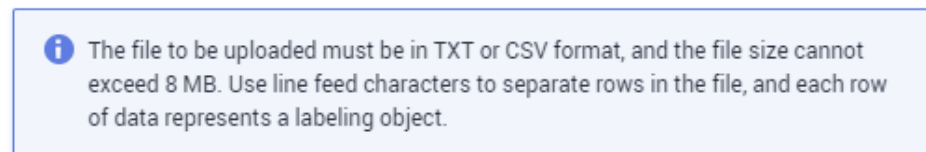
In addition to automatically synchronizing data from **Input Dataset Path**, you can directly add text files on ModelArts for data labeling.

1. On the dataset details page, click the **Unlabeled** tab. Then click **Add File**.
2. In the displayed **Add File** dialog box, set the parameters as required and then select the file to be uploaded.

Select one or more files to be uploaded in the local environment. Only **.txt** and **.csv** files are supported. The total size of files uploaded at a time cannot exceed 8 MB.

Figure 2-35 Adding a file

Add File



3. In the **Add File** dialog box, click **Upload**. The files you add will be automatically displayed on the **Unlabeled** tab page.

Deleting a File

You can quickly delete the files you want to discard.

- On the **Unlabeled** tab page, select the text to be deleted, and click **Delete** in the upper left corner to delete the text.
- On the **Labeled** tab page, select the text to be deleted and click **Delete**. Alternatively, you can tick **Select Images on Current Page** to select all text objects on the current page and click **Delete** in the upper left corner.

The background of the selected text is blue.

2.3.6 Text Triplet

Triplet labeling is suitable for scenarios where structured information, such as subjects, predicates, and objects, needs to be labeled in statements. With this

function, not only entities in statements, but also relationships between entities can be labeled. Triplet labeling is often used in natural language processing tasks such as dependency syntax analysis and information extraction.

Text triplet labeling involves two classes of important labels: **Entity Label** and **Relationship Label**. For the **Relationship Label**, you need to set its **Source entity** and **Target entity**.

- You can define multiple entity and relationship labels for a text object.
- The **Entity Label** defined during dataset creation cannot be deleted.

Precautions

Before labeling, ensure that the **Entity Label** and **Relationship Label** of a dataset have been defined. For the **Relationship Label**, you need to set its **Source entity** and **Target entity**. The **Relationship Label** must be between the defined **Source entity** and **Target entity**.

For example, if two entities are labeled as **Place**, you cannot add any relationship label between them, as shown in [Figure 2-36](#). If a relationship label cannot be added, a red cross is displayed, as shown in [Figure 2-37](#).

Figure 2-36 Example of entity and relationship labels

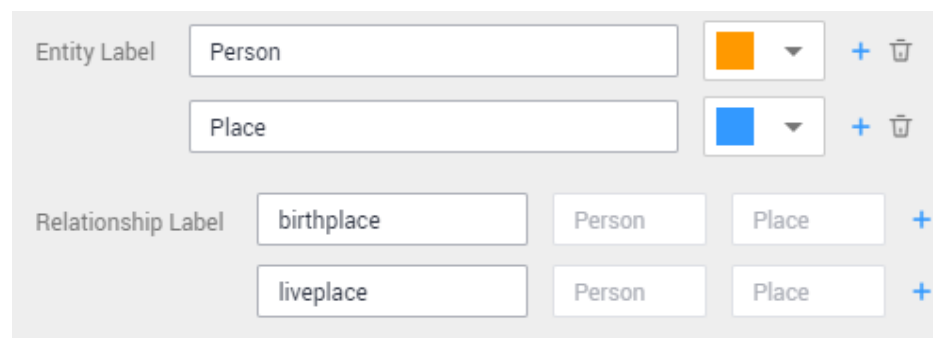
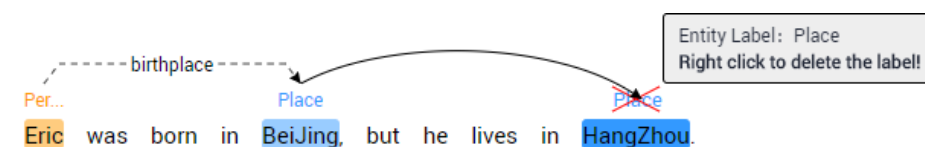


Figure 2-37 Failure of adding a relationship label



Starting Labeling

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. In the dataset list, select the dataset to be labeled based on the labeling type, and click the dataset name to go to the **Dashboard** tab page of the dataset.

By default, the **Dashboard** tab page of the current dataset version is displayed. If you need to label the dataset of another version, click the **Versions** tab and then click **Set to Current Version** in the right pane. For details, see [Managing Dataset Versions](#).

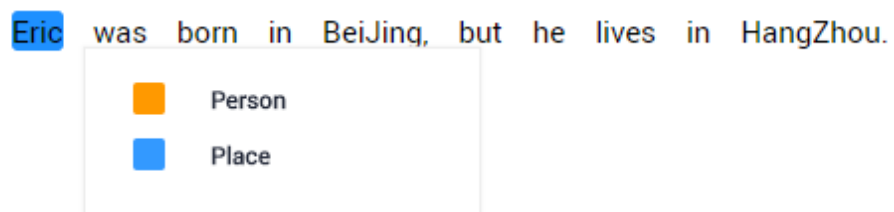
3. On the **Dashboard** page of the dataset, click **Label** in the upper right corner. The dataset details page is displayed. By default, all data of the dataset is displayed on the dataset details page.

Labeling Content

The dataset details page displays the labeled and unlabeled text objects in the dataset. The **Unlabeled** tab page is displayed by default.

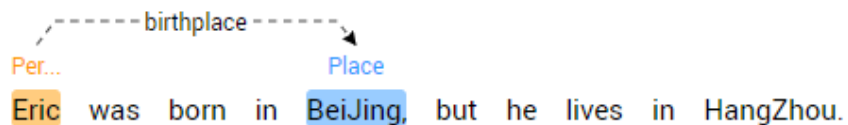
1. On the **Unlabeled** tab page, the objects to be labeled are listed in the left pane. In the list, click a text object, select the corresponding text content on the right pane, and select an entity name from the displayed entity list to label the content.

Figure 2-38 Labeling an entity



2. After labeling multiple entities, click the source entity and target entity in sequence and select a relationship type from the displayed relationship list to label the relationship.

Figure 2-39 Labeling a relationship



3. After all objects are labeled, click **Save Current Page** at the bottom of the page.

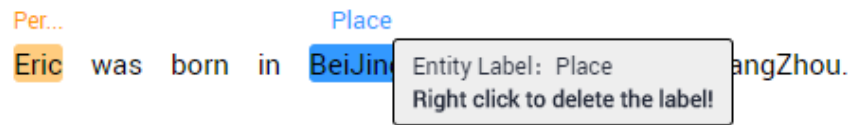
NOTE

You cannot modify the labels of a dataset in the text triplet type on the labeling page. Instead, click **Edit** to enter the **Modify Dataset** page and modify the **Entity Label** and **Relationship Label**.

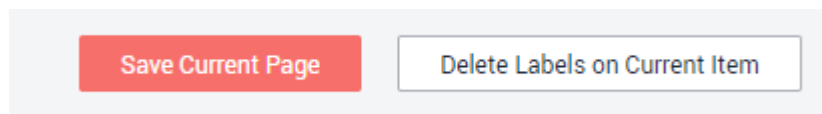
Modifying Labeled Data

After labeling data, you can modify labeled data on the **Labeled** tab page.

On the dataset details page, click the **Labeled** tab. Select a text object in the left pane and the right pane displays the detailed label information. You can move your cursor to the entity or relationship label, and right-click to delete it. You can also click the source entity and target entity in sequence to add a relationship label.

Figure 2-40 Modifying a label in the text

You can click **Delete Labels on Current Item** at the bottom of the page to delete all labels in the selected text object.

Figure 2-41 Deleting current labels

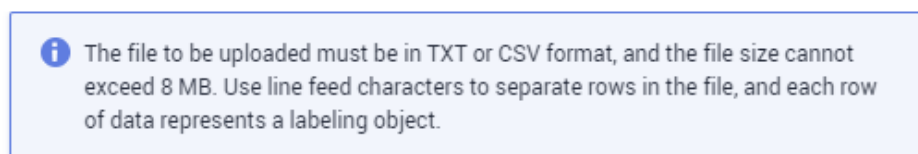
Adding a File

In addition to automatically synchronizing data from **Input Dataset Path**, you can directly add text files on ModelArts for data labeling.

1. On the dataset details page, click the **Unlabeled** tab. Then click **Add File**.
2. In the **Add File** dialog box that is displayed, select the files to be uploaded. Select one or more files to be uploaded in the local environment. Only **.txt** and **.csv** files are supported. The total size of files uploaded at a time cannot exceed 8 MB.

Figure 2-42 Add a file to be uploaded

Add File



3. In the **Add File** dialog box, click **Upload**. The files you add will be automatically displayed in the **Labeling Objects** list on the **Unlabeled** tab page.

Deleting a File

You can quickly delete the files you want to discard.

- On the **Unlabeled** tab page, select the text to be deleted, and click **Delete** in the upper left corner to delete the text.

- On the **Labeled** tab page, select the text to be deleted and click **Delete**. Alternatively, you can tick **Select Images on Current Page** to select all text objects on the current page and click **Delete** in the upper left corner.

The background of the selected text is blue. If no text is selected on the page, the **Delete** button is unavailable.

2.3.7 Sound Classification

Model training requires a large amount of labeled data. Therefore, before the model training, label the unlabeled audio files. ModelArts enables you to label audio files in batches by one click. In addition, you can modify the labels of audio files, or remove their labels and label the audio files again.

Starting Labeling

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. In the dataset list, select the dataset to be labeled based on the labeling type, and click the dataset name to go to the **Dashboard** tab page of the dataset.
By default, the **Dashboard** tab page of the current dataset version is displayed. If you need to label the dataset of another version, click the **Versions** tab and then click **Set to Current Version** in the right pane. For details, see [Managing Dataset Versions](#).
3. On the **Dashboard** page of the dataset, click **Label** in the upper right corner. The dataset details page is displayed. By default, all data of the dataset is displayed on the dataset details page.


Synchronizing the Data Source

ModelArts automatically synchronizes data and labeling information from **Input Dataset Path** to the dataset details page.

To quickly obtain the latest data in the OBS bucket, click **Synchronize Data Source** on the **Unlabeled** tab page of the dataset details page to add the data uploaded using OBS to the dataset.

Labeling Audio Files

The dataset details page displays the labeled and unlabeled audio files. The

Unlabeled tab page is displayed by default. Click  on the left of the audio to preview the audio.

1. On the **Unlabeled** tab page, select the audio files to be labeled.
 - Manual selection: In the audio list, click the target audio. If the blue check box is displayed in the upper right corner, the audio is selected. You can select multiple audio files of the same type and label them together.
 - Batch selection: If all audio files of the current page belong to one type, you can click **Select Images on Current Page** in the upper right corner of the list to select all the audio files on the page.
2. Add labels.

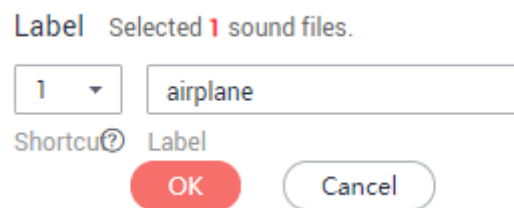
- a. In the right pane, set a label name in the **Label** text box.
Method 1 (the required label already exists): In the right pane, select a shortcut from the **Shortcut** drop-down list, select an existing label name from the **Label** text box, and click **OK**.
Method 2 (adding a label): In the right pane, select a shortcut from the **Shortcut** drop-down list, and enter a new label name in the **Label** text box.
- b. The selected audio files are automatically moved to the **Labeled** tab page. On the **Unlabeled** tab page, the labeling information is updated along with the labeling process, including the added label names and the number of audio files corresponding to each label.

NOTE

Shortcut key description: After specifying a shortcut key for a label, you can select an audio file and press the shortcut key to add a label for the audio file. Example: Specify **1** as the shortcut key for the **aa** label. Select one or more files and press **1** during data labeling. A message is displayed, asking you whether to label the files with **aa**. Click **OK**.

Each label has a shortcut key. A shortcut key cannot be specified for different labels. Shortcut keys can greatly improve the labeling efficiency.

Figure 2-43 Adding an audio label



Viewing the Labeled Audio Files

On the dataset details page, click the **Labeled** tab to view the list of the labeled audio files. Click an audio file. You can view the label information about the audio file in the **File Labels** area on the right.

Modifying Labels

After labeling data, you can modify labeled data on the **Labeled** tab page.

- **Modifying based on audio**

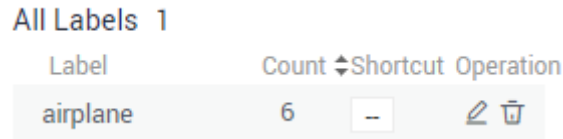
On the data labeling page, click the **Labeled** tab. Select one or more audio files to be modified from the audio list. Modify the label in the label details area on the right.



- **Modifying a label:** In the **File Labels** area, click the edit icon in the **Operation** column, enter the correct label name in the text box, and click the check mark to complete the modification.
- **Deleting a label:** In the **File Labels** area, click the delete icon in the **Operation** column to delete the label.

- **Modifying based on labels**

On the dataset details page, click the **Labeled** tab. The information about all labels is displayed on the right.

Figure 2-44 Information about all labels



Label	Count	Shortcut	Operation
airplane	6	-	 

- Modifying a label: Click the editing icon in the **Operation** column. In the dialog box that is displayed, enter the new label name and click **OK**. After the modification, the new label applies to the audio files that contain the original label.
- Deleting a label: Click the deletion icon in the **Operation** column. In the displayed dialog box, select the object to be deleted as prompted and click **OK**.

Adding Audio Files

In addition to automatically synchronizing data from **Input Dataset Path**, you can directly add audio files on ModelArts for data labeling.

1. On the dataset details page, click the **Unlabeled** tab. Then click **Add Audio** in the upper left corner.
2. In the **Add Audio** dialog box that is displayed, click **Add Audio**.
Select the audio files to be uploaded in the local environment. Only WAV audio files are supported. The size of an audio file cannot exceed 4 MB. The total size of audio files uploaded at a time cannot exceed 8 MB.
3. In the **Add Audio** dialog box, click **OK**.

The audio files you add will be automatically displayed on the **Unlabeled** tab page. In addition, the audio files are automatically saved to the OBS directory specified by **Input Dataset Path**.

Deleting Audio Files

You can quickly delete the audio files you want to discard.

On the **Unlabeled** or **Labeled** tab page, select the audio files to be deleted one by one or tick **Select Images on Current Page** to select all audio files on the page, and then click **Delete File** in the upper left corner. In the displayed dialog box, select or deselect **Delete source files** as required. After confirmation, click **OK** to delete the audio files.

If a tick is displayed in the upper right corner of an audio file, the audio file is selected. If no audio file is selected on the page, the **Delete File** button is unavailable.

 NOTE

If you select **Delete source files**, audio files stored in the corresponding OBS directory will be deleted when you delete the selected audio files. Deleting source files may affect other dataset versions or datasets using those files. As a result, the page display, training, or inference is abnormal. Deleted data cannot be recovered. Exercise caution when performing this operation.

2.3.8 Speech Labeling

Model training requires a large amount of labeled data. Therefore, before the model training, label the unlabeled audio files. ModelArts enables you to label audio files in batches by one click. In addition, you can modify the labels of audio files, or remove their labels and label the audio files again.

Starting Labeling

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. In the dataset list, select the dataset to be labeled based on the labeling type, and click the dataset name to go to the **Dashboard** tab page of the dataset.
By default, the **Dashboard** tab page of the current dataset version is displayed. If you need to label the dataset of another version, click the **Versions** tab and then click **Set to Current Version** in the right pane. For details, see [Managing Dataset Versions](#).
3. On the **Dashboard** page of the dataset, click **Label** in the upper right corner. The dataset details page is displayed. By default, all data of the dataset is displayed on the dataset details page.

Synchronizing the Data Source

ModelArts automatically synchronizes data and labeling information from **Input Dataset Path** to the dataset details page.

To quickly obtain the latest data in the OBS bucket, click **Synchronize Data Source** on the **Unlabeled** tab page of the dataset details page to add the data uploaded using OBS to the dataset.

Labeling Audio Files

The dataset details page displays the labeled and unlabeled audio files. The **Unlabeled** tab page is displayed by default.


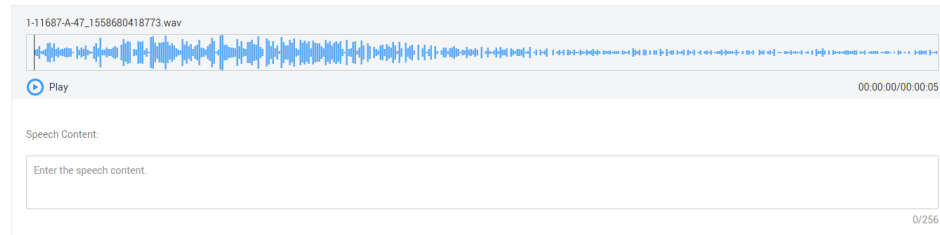
1. In the audio file list on the **Unlabeled** tab page, click the target audio file. In the area on the right, the audio file is displayed. Click  below the audio file to play the audio.
2. In **Speech Content**, enter the speech content.
3. After entering the content, click **OK** to complete the labeling. The audio file is automatically moved to the **Labeled** tab page.

Figure 2-45 Labeling an audio file

Viewing the Labeled Audio Files

On the dataset details page, click the **Labeled** tab to view the list of the labeled audio files. Click the audio file to view the audio content in the **Speech Content** text box on the right.

Modifying Labeled Data

After labeling data, you can modify labeled data on the **Labeled** tab page.

On the data labeling page, click the **Labeled** tab, and select the audio file to be modified from the audio file list. In the label information area on the right, modify the content of the **Speech Content** text box, and click **OK** to complete the modification.

Adding Audio Files

In addition to automatically synchronizing data from **Input Dataset Path**, you can directly add audio files on ModelArts for data labeling.

1. On the dataset details page, click the **Unlabeled** tab. Then click **Add Audio** in the upper left corner.
2. In the **Add Audio** dialog box that is displayed, click **Add Audio**.

Select the audio files to be uploaded in the local environment. Only WAV audio files are supported. The size of an audio file cannot exceed 4 MB. The total size of audio files uploaded at a time cannot exceed 8 MB.

3. In the **Add Audio** dialog box, click **OK**.

The audio files you add will be automatically displayed on the **Unlabeled** tab page. In addition, the audio files are automatically saved to the OBS directory specified by **Input Dataset Path**.

Deleting Audio Files

You can quickly delete the audio files you want to discard.

On the **Unlabeled** or **Labeled** tab page, select the audio files to be deleted, and then click **Delete File** in the upper left corner. In the displayed dialog box, select or deselect **Delete source files** as required. After confirmation, click **OK** to delete the audio files.

 NOTE

If you select **Delete source files**, audio files stored in the corresponding OBS directory will be deleted when you delete the selected audio files. Deleting source files may affect other dataset versions or datasets using those files. As a result, the page display, training, or inference is abnormal. Deleted data cannot be recovered. Exercise caution when performing this operation.

2.3.9 Speech Paragraph Labeling

Model training requires a large amount of labeled data. Therefore, before the model training, label the unlabeled audio files. ModelArts enables you to label audio files. In addition, you can modify the labels of audio files, or remove their labels and label the audio files again.

Starting Labeling

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. In the dataset list, select the dataset to be labeled based on the labeling type, and click the dataset name to go to the **Dashboard** tab page of the dataset.
By default, the **Dashboard** tab page of the current dataset version is displayed. If you need to label the dataset of another version, click the **Versions** tab and then click **Set to Current Version** in the right pane. For details, see [Managing Dataset Versions](#).
3. On the **Dashboard** page of the dataset, click **Label** in the upper right corner. The dataset details page is displayed. By default, all data of the dataset is displayed on the dataset details page.

Synchronizing the Data Source

ModelArts automatically synchronizes data and labeling information from **Input Dataset Path** to the dataset details page.

To quickly obtain the latest data in the OBS bucket, click **Synchronize Data Source** on the **Unlabeled** tab page of the dataset details page to add the data uploaded using OBS to the dataset.

Labeling Audio Files

The dataset details page displays the labeled and unlabeled audio files. The **Unlabeled** tab page is displayed by default.


1. In the audio file list on the **Unlabeled** tab page, click the target audio file. In the area on the right, the audio file is displayed. Click  below the audio file to play the audio.
2. Select an audio segment based on the content being played, and enter the audio file label and content in the **Speech Content** text box.

Figure 2-46 Labeling an audio file


- After entering the content, click **OK** to complete the labeling. The audio file is automatically moved to the **Labeled** tab page.

Viewing the Labeled Audio Files

On the dataset details page, click the **Labeled** tab to view the list of the labeled audio files. Click the audio file to view the audio content in the **Speech Content** text box on the right.

Modifying Labeled Data

After labeling data, you can modify labeled data on the **Labeled** tab page.

- Modifying a label: On the dataset details page, click the **Labeled** tab, and select the audio file to be modified from the audio file list. In the **Speech Content** area, modify **Label** and **Content**, and click **OK** to complete the modification.
- Deleting a label: Click  in the **Operation** column of the target number to delete the label of the audio segment. Alternatively, you can click the cross (x) icon above the labeled audio file to delete the label. Then click **OK**.

Adding Audio Files

In addition to automatically synchronizing data from **Input Dataset Path**, you can directly add audio files on ModelArts for data labeling.

- On the dataset details page, click the **Unlabeled** tab. Then click **Add Audio** in the upper left corner.
- In the **Add Audio** dialog box that is displayed, click **Add Audio**.
Select the audio files to be uploaded in the local environment. Only WAV audio files are supported. The size of an audio file cannot exceed 4 MB. The total size of audio files uploaded at a time cannot exceed 8 MB.
- In the **Add Audio** dialog box, click **OK**.
The audio files you add will be automatically displayed on the **Unlabeled** tab page. In addition, the audio files are automatically saved to the OBS directory specified by **Input Dataset Path**.

Deleting Audio Files

You can quickly delete the audio files you want to discard.

On the **Unlabeled** or **Labeled** tab page, select the audio files to be deleted, and then click **Delete File** in the upper left corner. In the displayed dialog box, select

or deselect **Delete source files** as required. After confirmation, click **OK** to delete the audio files.

NOTE

If you select **Delete source files**, audio files stored in the corresponding OBS directory will be deleted when you delete the selected audio files. Deleting source files may affect other dataset versions or datasets using those files. As a result, the page display, training, or inference is abnormal. Deleted data cannot be recovered. Exercise caution when performing this operation.

2.3.10 Video Labeling

Model training requires a large amount of labeled video data. Therefore, before the model training, label the unlabeled video files. ModelArts enables you to label video files. In addition, you can modify the labels of video files, or remove their labels and label the video files again.

Starting Labeling

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. In the dataset list, select the dataset to be labeled based on the labeling type, and click the dataset name to go to the **Dashboard** tab page of the dataset.
By default, the **Dashboard** tab page of the current dataset version is displayed. If you need to label the dataset of another version, click the **Versions** tab and then click **Set to Current Version** in the right pane. For details, see [Managing Dataset Versions](#).
3. On the **Dashboard** page of the dataset, click **Label** in the upper right corner. The dataset details page is displayed. By default, all data of the dataset is displayed on the dataset details page.

Synchronizing Data Sources

ModelArts automatically synchronizes data and labeling information from **Input Dataset Path** to the dataset details page.

To quickly obtain the latest data in the OBS bucket, click **Synchronize Data Source** on the **Unlabeled** tab page of the dataset details page to add the data uploaded using OBS to the dataset.

Labeling Video Files

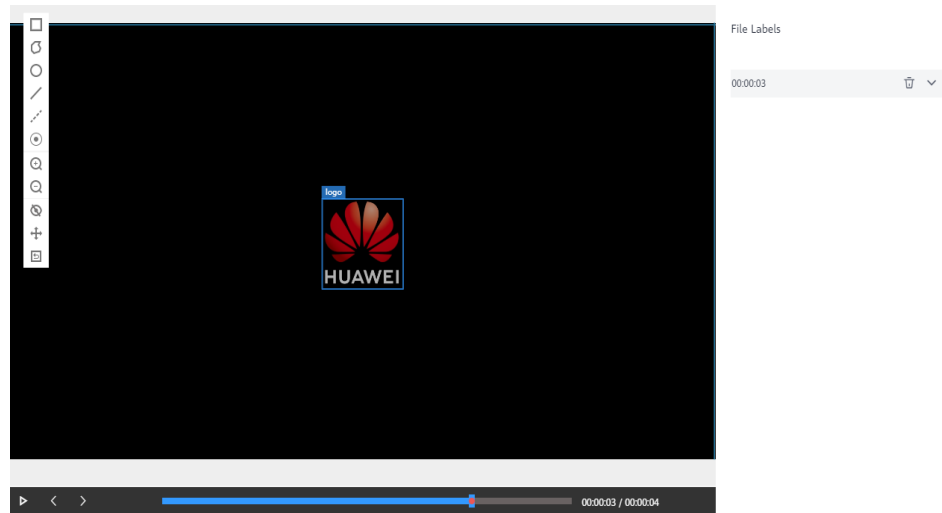
On the dataset details page, both unlabeled and labeled video files in the dataset are displayed.

1. On the **Unlabeled** tab page, click the target video file in the video list on the left. The labeling page is displayed.
2. Play the video. When the video is played to the time point to be labeled, click the pause button in the progress bar to pause the video to a specific image.
3. In the left pane, select a bounding box. By default, a rectangular box is selected. Drag the mouse to select an object in the video image, enter a new label name in the displayed **Add Label** text box, select a label color, and click

Add to label the object. Alternatively, select an existing label from the drop-down list and click **Add** to label the object. Label all objects in the image. Multiple labels can be added to an image.

The supported bounding boxes are the same as those supported by Object Detection. For details, see [Table 2-9](#) in [Object Detection](#).

Figure 2-47 Labeling video files



- After the previous image is labeled, click the play button on the progress bar to resume the playback. Then, repeat [3](#) to complete labeling on the entire video.

The labeled time points of the current video are displayed on the right of the page.


Figure 2-48 File labels



- Click **Back to Data Labeling Preview** in the upper left corner of the page. The dataset details page is displayed, and the labeled video file is displayed on the **Labeled** tab page.

Modifying Labeled Data

After labeling data, you can delete labeled data on the **Labeled** tab page.

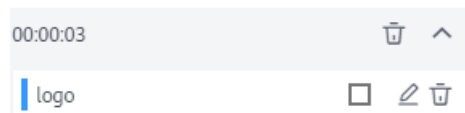
- Click  in the **Operation** column of the target number to delete the label of the video segment. Alternatively, you can click the cross (x) icon above the labeled video file to delete the label. Then click **OK**.

On the **Labeled** tab page, click the target video file. In the **All Labels** area on the right of the labeling page, click the triangle icon on the right of the time point to view details. You can modify or delete a label.

- Modifying a label: Click the edit icon on the right of a label to modify the label name.
- Deleting a label: Click the delete icon on the right of a label to delete the label. If you click the delete icon on the right of the image time, all labels on the image are deleted.

Figure 2-49 Modifying labeled data

File Labels



Deleting a Video File

You can quickly delete the video files you want to discard.

On the **All**, **Unlabeled**, or **Labeled** tab page, select the video files to be deleted or click **Select Images on Current Page** to select all video files on the page, and click **Delete** in the upper left corner to delete the video files. In the displayed dialog box, select or deselect **Delete source files** as required. After confirmation, click **OK** to delete the videos.

If a tick is displayed in the upper left corner of a video file, the video file is selected. If no video file is selected on the page, the **Delete File** button is unavailable.

NOTE

If you select **Delete source files**, video files stored in the corresponding OBS directory will be deleted when you delete the selected video files. Deleting source files may affect other dataset versions or datasets using those files. As a result, the page display, training, or inference is abnormal. Deleted data cannot be recovered. Exercise caution when performing this operation.

2.4 Importing Data

2.4.1 Import Operation

After a dataset is created, you can directly synchronize data from the dataset. Alternatively, you can import more data by importing the dataset. Data can be imported from an OBS directory or the manifest file.

Prerequisites

- You have created a dataset.
- You have stored the data to be imported in OBS. You have stored the manifest file in OBS.
- The OBS buckets and ModelArts are in the same region.

Import Modes

There are two import modes: **OBS path** and **Manifest file**.

- **OBS path**: indicates that the dataset to be imported has been stored in an OBS directory in advance. In this case, you need to select an OBS path that you can access. In addition, the directory structure in the OBS path must comply with the specifications. For details, see [Specifications for Importing Data from an OBS Directory](#). Only the following types of dataset support the **OBS path** import mode: **Image classification**, **Object detection**, **Text classification**, **Table**, and **Sound classification**.
- **Manifest file**: indicates that the dataset file is in the manifest format and data is imported from the manifest file. The manifest file defines the mapping between labeling objects and content. In addition, the manifest file has been uploaded to OBS. For details about the specifications of the manifest file, see [Specifications for Importing the Manifest File](#).

NOTE

Before importing an object detection dataset, ensure that the labeling range of the labeling file does not exceed the size of the original image. Otherwise, the import may fail.

Table 2-13 Import modes supported by datasets

Dataset Type	Importing Data from an OBS Path	Importing Data from a Manifest File
Image classification	Supported Follow the format specifications described in Image Classification .	Supported Follow the format specifications described in Image Classification .
Object detection	Supported Follow the format specifications described in Object Detection .	Supported Follow the format specifications described in Object Detection .
Image segmentation	Supported Follow the format specifications described in Image Segmentation .	Supported Follow the format specifications described in Image Segmentation .
Sound classification	Supported Follow the format specifications described in Sound Classification .	Supported Follow the format specifications described in Sound Classification .
Speech labeling	N/A	Supported Follow the format specifications described in Speech Paragraph Labeling .

Dataset Type	Importing Data from an OBS Path	Importing Data from a Manifest File
Speech paragraph labeling	N/A	Supported Follow the format specifications described in Speech Labeling .
Text classification	Supported Follow the format specifications described in Text Classification .	Supported Follow the format specifications described in Text Classification .
Named entity recognition	N/A	Supported Follow the format specifications described in Named Entity Recognition .
Text triplet	N/A	Supported Follow the format specifications described in Text Triplet .
Table	Supported You can also import data from DWS, DLI, or MRS. Follow the format specifications described in Table .	N/A
Video	N/A	Supported Follow the format specifications described in Video Labeling .
Free format	N/A	N/A

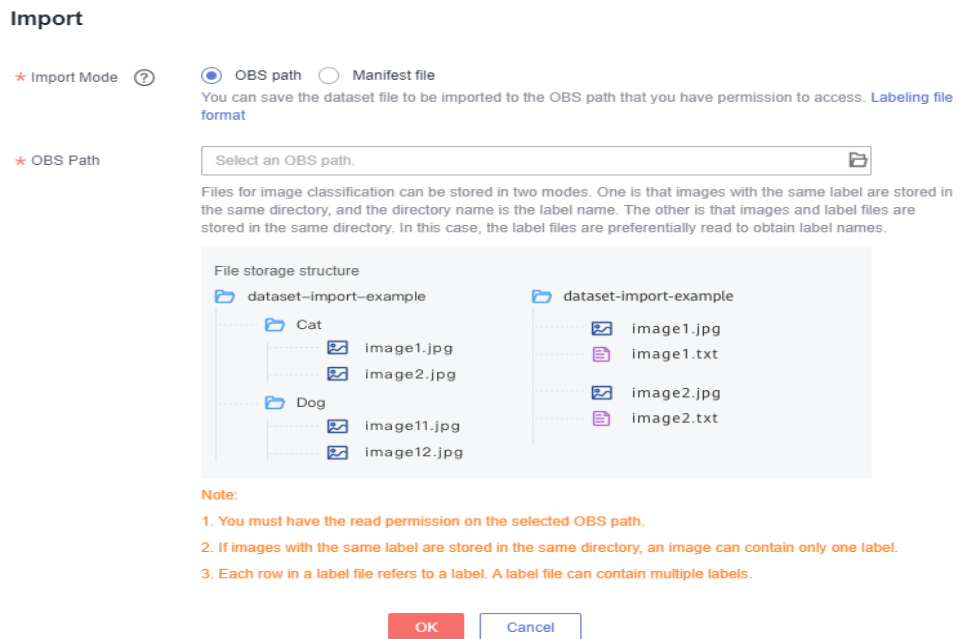
Importing Data from an OBS Path

The parameters on the GUI for data import vary according to the dataset type. The following uses a dataset of the image classification type as an example.

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. Locate the row that contains the desired dataset and choose **More > Import** in the **Operation** column.
Alternatively, you can click the dataset name to go to the **Dashboard** tab page of the dataset, and click **Import** in the upper right corner.
3. In the **Import** dialog box, set **Import Mode** to **OBS path** and set **OBS path** to the path for storing data. Then click **OK**.

NOTE

You can import a dataset of the table type from data sources such as OBS, Data Warehouse Service (DWS), Data Lake Insight (DLI), and MapReduce Service (MRS). The settings and data requirements for importing a dataset are the same as those for creating a dataset. For details about the parameters, see [Table](#).

Figure 2-50 Importing the dataset from an OBS path

After the data import is successful, the data is automatically synchronized to the dataset. On the **Datasets** page, you can click the dataset name to view its details and label the data.

Importing Data from a Manifest File

The parameters on the GUI for data import vary according to the dataset type. The following uses a dataset of the object detection type as an example. Datasets of the table type cannot be imported from the manifest file.

- Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
- Locate the row that contains the desired dataset and choose **More > Import** in the **Operation** column.

Alternatively, you can click the dataset name to go to the **Dashboard** tab page of the dataset, and click **Import** in the upper right corner.

- In the **Import** dialog box, set the parameters as follows and click **OK**.
 - Import Mode:** Select **Manifest file**.
 - Manifest file:** Select the OBS path for storing the manifest file.
 - Import by Label:** The system automatically obtains the labels of the dataset. You can click **Add Label** to add a label or click the deletion icon on the right to delete a label. This field is optional. After importing a dataset, you can add or delete labels during data labeling.

- **Import labels:** If this parameter is selected, the labels defined in the manifest file are imported to the ModelArts dataset.
- **Import only hard examples:** If this parameter is selected, only the **hard** attribute data of the manifest file is imported. Examples whose **hard** attribute is true in the manifest file are hard examples.

Figure 2-51 Importing the dataset

Import

* Import Mode ? OBS path Manifest file
The manifest file needs to define the mapping between labeling objects and content. [Manifest file specifications and examples](#)

* Manifest File 📁

Import by Label

dog	🗑️
cat	🗑️
pig	🗑️
➕ Add Label	

Import labels

Import only hard examples ?

OK Cancel

After the data import is successful, the data is automatically synchronized to the dataset. On the **Datasets** page, you can click the dataset name to go to the **Dashboard** tab page of the dataset, and click **Label** in the upper right corner. On the displayed dataset details page, view detailed data and label data.

2.4.2 Specifications for Importing Data from an OBS Directory

When a dataset is imported, the data storage directory and file name must comply with the ModelArts specifications if the data to be used is stored in OBS.

Only the following types of dataset support the **OBS path** import mode: **Image classification**, **Object detection**, **Text classification**, **Table**, and **Sound classification**. A table dataset can be imported from data sources such as OBS, DWS, DLI, and MRS.

NOTE

To import data from an OBS directory, you must have the read permission on the OBS directory.

Image Classification

- Image classification data can be in two modes. The first mode (directory mode) supports only single labels. The second mode (.txt label files) supports multiple labels.
 - Images with the same label must be stored in the same directory, and the label name is the directory name. If there are multiple levels of directories, the last level is used as the label name.

In the following example, **Cat** and **Dog** are label names.

```
dataset-import-example
├── Cat
│   ├── 10.jpg
│   ├── 11.jpg
│   └── 12.jpg
└── Dog
    ├── 1.jpg
    ├── 2.jpg
    └── 3.jpg
```

- If **.txt** files exist in the directory, the content in the **.txt** files is used as the image label. This mode is better than the previous one.

In the following example, **import-dir-1** and **import-dir-2** are the imported subdirectories:

```
dataset-import-example
├── import-dir-1
│   ├── 10.jpg
│   ├── 10.txt
│   ├── 11.jpg
│   ├── 11.txt
│   ├── 12.jpg
│   └── 12.txt
└── import-dir-2
    ├── 1.jpg
    ├── 1.txt
    ├── 2.jpg
    └── 2.txt
```

The following shows a label file for a single label, for example, the **1.txt** file:

```
Cat
```

The following shows a label file for multiple labels, for example, the **1.txt** file:

```
Cat
Dog
```

- Only images in JPG, JPEG, PNG, and BMP formats are supported. The size of a single image cannot exceed 5 MB, and the total size of all images uploaded at a time cannot exceed 8 MB.

Object Detection

- The simple mode of object detection requires users store labeled objects and their label files (in one-to-one relationship with the labeled objects) in the same directory. For example, if the name of the labeled object file is **IMG_20180919_114745.jpg**, the name of the label file must be **IMG_20180919_114745.xml**.

The label files for object detection must be in PASCAL VOC format. For details about the format, see [Table 2-21](#).

Example:

```
dataset-import-example
├── IMG_20180919_114732.jpg
├── IMG_20180919_114732.xml
├── IMG_20180919_114745.jpg
├── IMG_20180919_114745.xml
├── IMG_20180919_114945.jpg
└── IMG_20180919_114945.xml
```

A label file example is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<annotation>
```

```

<folder>NA</folder>
<filename>bike_1_1593531469339.png</filename>
<source>
  <database>Unknown</database>
</source>
<size>
  <width>554</width>
  <height>606</height>
  <depth>3</depth>
</size>
<segmented>0</segmented>
<object>
  <name>Dog</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <occluded>0</occluded>
  <bndbox>
    <xmin>279</xmin>
    <ymin>52</ymin>
    <xmax>474</xmax>
    <ymin>278</ymin>
  </bndbox>
</object>
<object>
  <name>Cat</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <occluded>0</occluded>
  <bndbox>
    <xmin>279</xmin>
    <ymin>198</ymin>
    <xmax>456</xmax>
    <ymin>421</ymin>
  </bndbox>
</object>
</annotation>

```

- Only images in JPG, JPEG, PNG, and BMP formats are supported. The size of a single image cannot exceed 5 MB, and the total size of all images uploaded at a time cannot exceed 8 MB.

Image Segmentation

- The simple mode of image segmentation requires users store labeled objects and their label files (in one-to-one relationship with the labeled objects) in the same directory. For example, if the name of the labeled object file is **IMG_20180919_114746.jpg**, the name of the label file must be **IMG_20180919_114746.xml**.

Fields **mask_source** and **mask_color** are added to the label file in PASCAL VOC format. For details about the format, see [Table 2-17](#).

Example:

```

dataset-import-example
  IMG_20180919_114732.jpg
  IMG_20180919_114732.xml
  IMG_20180919_114745.jpg
  IMG_20180919_114745.xml
  IMG_20180919_114945.jpg
  IMG_20180919_114945.xml

```

A label file example is as follows:

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<annotation>
  <folder>NA</folder>

```

```

<filename>image_0006.jpg</filename>
<source>
  <database>Unknown</database>
</source>
<size>
  <width>230</width>
  <height>300</height>
  <depth>3</depth>
</size>
<segmented>1</segmented>
<mask_source>obs://xianao/out/dataset-8153-Jmf5yLjRmSacj9KevS/annotation/V001/segmentationClassRaw/image_0006.png</mask_source>
<object>
  <name>bike</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <mask_color>193,243,53</mask_color>
  <occluded>0</occluded>
  <polygon>
    <x1>71</x1>
    <y1>48</y1>
    <x2>75</x2>
    <y2>73</y2>
    <x3>49</x3>
    <y3>69</y3>
    <x4>68</x4>
    <y4>92</y4>
    <x5>90</x5>
    <y5>101</y5>
    <x6>45</x6>
    <y6>110</y6>
    <x7>71</x7>
    <y7>48</y7>
  </polygon>
</object>
</annotation>

```

Text Classification

txt and csv files can be imported for text classification, with the text encoding format of UTF-8 or GBK.

Labeled objects and labels for text classification can be stored in two modes:

- The labeled objects and labels for text classification are in the same text file. You can specify a separator to separate the labeled objects and labels, as well as multiple labels.

For example, the following shows an example text file. The **Tab** key is used to separate the labeled object from the label.

```

It touches good and responds quickly. I don't know how it performs in the future. positive
Three months ago, I bought a very good phone and replaced my old one with it. It can operate longer
between charges. positive
Why does my phone heat up if I charge it for a while? The volume button stuck after being pressed
down. negative
It's a gift for Father's Day. The logistics is fast and I received it in 24 hours. I like the earphones
because the bass sounds feel good and they would not fall off. positive

```

- The labeled objects and labels for text classification are text files, and correspond to each other based on the rows. For example, the first row in a label file indicates the label of the first row in the file of the labeled object.

For example, the content of labeled object **COMMENTS_20180919_114745.txt** is as follows:

```

It touches good and responds quickly. I don't know how it performs in the future.
Three months ago, I bought a very good phone and replaced my old one with it. It can operate longer

```

between charges.

Why does my phone heat up if I charge it for a while? The volume button stuck after being pressed down.

It's a gift for Father's Day. The logistics is fast and I received it in 24 hours. I like the earphones because the bass sounds feel good and they would not fall off.

The content of label file **COMMENTS_20180919_114745_result.txt** is as follows:

```
positive
negative
negative
positive
```

The data format requires users to store labeled objects and their label files (in one-to-one relationship with the labeled objects) in the same directory. For example, if the name of the labeled object file is

COMMENTS_20180919_114745.txt, the name of the label file must be **COMMENTS_20180919_114745_result.txt**.

Example of data file storage:

```
dataset-import-example
├── COMMENTS_20180919_114732.txt
│   ├── COMMENTS_20180919_114732_result.txt
│   ├── COMMENTS_20180919_114745.txt
│   ├── COMMENTS_20180919_114745_result.txt
│   ├── COMMENTS_20180919_114945.txt
│   └── COMMENTS_20180919_114945_result.txt
```

Sound Classification

For sound classification, sound files with the same label must be stored in the same directory, and the label name is the directory name.

Example:

```
dataset-import-example
├── Cat
│   ├── 10.wav
│   ├── 11.wav
│   └── 12.wav
└── Dog
    ├── 1.wav
    ├── 2.wav
    └── 3.wav
```

Table

You can import data from OBS, DWS, DLI, and MRS.

Import description:

1. The prerequisite for successful import is that the schema of the data source must be the same as that specified during dataset creation. The schema indicates column names and types of a table. Once specified during dataset creation, the values cannot be changed.
2. If the data format is invalid, the data is set to null values. For details, see [Table 2-6](#).
3. When a CSV file is imported from OBS or MRS, the data type is not verified, but the number of columns must be the same as that in the schema of the dataset.

The following describes how to import data from different data sources:

- From OBS

CSV files can be imported from OBS. You need to select the directory where the files are stored. The number of columns in the CSV file must be the same as that in the dataset schema. The schema of the CSV file can be automatically obtained.

```
dataset-import-example
├── table_import_1.csv
│   ├── table_import_2.csv
│   ├── table_import_3.csv
│   └── table_import_4.csv
```

- From DWS

To import data from DWS, you need to select a DWS cluster and enter the database name, table name, username, and password. The schema (column name and type) of the imported table must be the same as that of the dataset.

- From DLI

To import data from DLI, you need to select the DLI queue, database, and table name. The schema (column name and type) of the selected table must be the same as that of the dataset. The schema of the selected table can be automatically obtained. The default queue of DLI is used only for experience. Different accounts may preempt resources. Therefore, resources need to be queued. You may not be able to obtain required resources each time to perform related operations. DLI supports schema mapping. That is, the schema field name of the imported table can be different from that of the dataset, but the type must be the same.

- From MRS

Data can be imported only from the analysis cluster. To import data in CSV format stored on HDFS from MRS, select an existing MRS cluster and select the file name or directory from the HDFS file list. The number of columns in the imported file must be the same as that of the dataset schema.

2.4.3 Specifications for Importing the Manifest File

The manifest file defines the mapping between labeling objects and content. The **Manifest file** import mode means that the manifest file is used for dataset import. The manifest file can be imported from OBS. When importing a manifest file from OBS, ensure that the current user has the permissions to access the directory housing the manifest file.

NOTE

There are many requirements on the Manifest file compilation. Import new data from OBS. Generally, Manifest file import is used for data migration of ModelArts in different regions or using different accounts. If you have labeled data in a region using ModelArts, you can obtain the manifest file of the published dataset from the output path. Then you can import the dataset using the manifest file to ModelArts of other regions or accounts. The imported data carries the labeling information and does not need to be labeled again, improving development efficiency.

The manifest file that contains information about the original file and labeling can be used in labeling, training, and inference scenarios. The manifest file that contains only information about the original file can be used in inference scenarios

or used to generate an unlabeled dataset. The manifest file must meet the following requirements:

- The manifest file uses the UTF-8 encoding format. The **source** value of text classification can contain Chinese characters. However, Chinese characters are not recommended for other parameters.

- The manifest file uses the JSON Lines format (jsonlines.org). A line contains one JSON object.

```
{ "source": "/path/to/image1.jpg", "annotation": ... }
{ "source": "/path/to/image2.jpg", "annotation": ... }
{ "source": "/path/to/image3.jpg", "annotation": ... }
```

In the preceding example, the manifest file contains multiple lines of JSON object.

- The manifest file can be generated by users, third-party tools, or ModelArts Data Labeling. The file name can be any valid file name. To facilitate the internal use of the ModelArts system, the file name generated by the ModelArts Data Labeling function consists of the following character strings: *DatasetName-VersionName.manifest*. For example, **animal-v201901231130304123.manifest**.

Image Classification

```
{
  "source": "s3://path/to/image1.jpg",
  "usage": "TRAIN",
  "hard": "true",
  "hard-coefficient": 0.8,
  "id": "0162005993f8065ef47eefb59d1e4970",
  "annotation": [
    {
      "type": "modelarts/image_classification",
      "name": "cat",
      "property": {
        "color": "white",
        "kind": "Persian cat"
      },
      "hard": "true",
      "hard-coefficient": 0.8,
      "annotated-by": "human",
      "creation-time": "2019-01-23 11:30:30"
    },
    {
      "type": "modelarts/image_classification",
      "name": "animal",
      "annotated-by": "modelarts/active-learning",
      "confidence": 0.8,
      "creation-time": "2019-01-23 11:30:30"
    }
  ],
  "inference-loc": "/path/to/inference-output"
}
```

Table 2-14 Parameters

Parameter	Mandatory	Description
source	Yes	URI of an object to be labeled. For details about data source types and examples, see Table 2-15 .

Parameter	Mandatory	Description
usage	No	By default, the parameter value is left blank. Possible values are as follows: <ul style="list-style-type: none"> ● TRAIN: The object is used for training. ● EVAL: The object is used for evaluation. ● TEST: The object is used for testing. ● INFERENCE: The object is used for inference. If the parameter value is left blank, the user decides how to use the object.
id	No	Sample ID exported from the system. You do not need to set this parameter when importing the sample.
annotation	No	If the parameter value is left blank, the object is not labeled. The value of annotation consists of an object list. For details about the parameters, see Table 2-16 .
inference-loc	No	This parameter is available when the file is generated by the inference service, indicating the location of the inference result file.

Table 2-15 Data source types

Type	Example
OBS	"source":"s3://path-to-jpg"
Content	"source":"content://I love machine learning"

Table 2-16 annotation objects

Parameter	Mandatory	Description
type	Yes	Label type. Possible values are as follows: <ul style="list-style-type: none"> ● image_classification: image classification ● text_classification: text classification ● text_entity: named entity recognition ● object_detection: object detection ● audio_classification: sound classification ● audio_content: speech labeling ● audio_segmentation: speech paragraph labeling

Parameter	Mandatory	Description
name	Yes/No	This parameter is mandatory for the classification type but optional for other types. This example uses the image classification type.
id	Yes/No	Label ID. This parameter is mandatory for triplets but optional for other types. The entity label ID of a triplet is in E+number format, for example, E1 and E2 . The relationship label ID of a triplet is in R+number format, for example, R1 and R2 .
property	No	Labeling property. In this example, the cat has two properties: color and kind.
hard	No	Indicates whether the example is a hard example. True indicates that the labeling example is a hard example, and False indicates that the labeling example is not a hard example.
annotated-by	No	The default value is human , indicating manual labeling. <ul style="list-style-type: none"> human
creation-time	No	Time when the labeling job was created. It is the time when labeling information was written, not the time when the manifest file was generated.
confidence	No	Confidence score of machine labeling. The value ranges from 0 to 1.

Image Segmentation

```
{
  "annotation": [{
    "annotation-format": "PASCAL VOC",
    "type": "modelarts/image_segmentation",
    "annotation-loc": "s3://path/to/annotation/image1.xml",
    "creation-time": "2020-12-16 21:36:27",
    "annotated-by": "human"
  }],
  "usage": "train",
  "source": "s3://path/to/image1.jpg",
  "id": "16d196c19bf61994d7deccafa435398c",
  "sample-type": 0
}
```

- The parameters such as **source**, **usage**, and **annotation** are the same as those described in [Image Classification](#). For details, see [Table 2-14](#).
- **annotation-loc** indicates the path for saving the label file. This parameter is mandatory for image segmentation and object detection but optional for other labeling types.
- **annotation-format** indicates the format of the label file. This parameter is optional. The default value is **PASCAL VOC**. Only **PASCAL VOC** is supported.

- **sample-type** indicates a sample format. Value **0** indicates image, **1** text, **2** audio, **4** table, and **6** video.

Table 2-17 PASCAL VOC format parameters

Parameter	Mandatory	Description
folder	Yes	Directory where the data source is located
filename	Yes	Name of the file to be labeled
size	Yes	Image pixel <ul style="list-style-type: none"> • width: image width. This parameter is mandatory. • height: image height. This parameter is mandatory. • depth: number of image channels. This parameter is mandatory.
segmented	Yes	Segmented or not
mask_source	No	Segmentation mask path
object	Yes	Object detection information. Multiple object{} functions are generated for multiple objects. <ul style="list-style-type: none"> • name: class of the labeled content. This parameter is mandatory. • pose: shooting angle of the labeled content. This parameter is mandatory. • truncated: whether the labeled content is truncated (0 indicates that the content is not truncated). This parameter is mandatory. • occluded: whether the labeled content is occluded (0 indicates that the content is not occluded). This parameter is mandatory. • difficult: whether the labeled object is difficult to identify (0 indicates that the object is easy to identify). This parameter is mandatory. • confidence: confidence score of the labeled object. The value ranges from 0 to 1. This parameter is optional. • bndbox: bounding box type. This parameter is mandatory. For details about the possible values, see Table 2-18. • mask_color: label color, which is represented by the RGB value. This parameter is mandatory.

Table 2-18 Bounding box types

Type	Shape	Labeling Information
polygon	Polygon	Coordinates of points <x1>100<x1> <y1>100<y1> <x2>200<x2> <y2>100<y2> <x3>250<x3> <y3>150<y3> <x4>200<x4> <y4>200<y4> <x5>100<x5> <y5>200<y5> <x6>50<x6> <y6>150<y6> <x7>100<x7> <y7>100<y7>

Example:

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<annotation>
  <folder>NA</folder>
  <filename>image_0006.jpg</filename>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>230</width>
    <height>300</height>
    <depth>3</depth>
  </size>
  <segmented>1</segmented>
  <mask_source>obs://xianao/out/dataset-8153-Jmf5ylljRmSacj9KevS/annotation/V001/segmentationClassRaw/image_0006.png</mask_source>
  <object>
    <name>bike</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <mask_color>193,243,53</mask_color>
    <occluded>0</occluded>
    <polygon>
      <x1>71</x1>
      <y1>48</y1>
      <x2>75</x2>
      <y2>73</y2>
      <x3>49</x3>
      <y3>69</y3>
      <x4>68</x4>
      <y4>92</y4>
      <x5>90</x5>
      <y5>101</y5>
      <x6>45</x6>
      <y6>110</y6>
    </polygon>
  </object>
</annotation>

```

```
<x7>71</x7>
<y7>48</y7>
</polygon>
</object>
</annotation>
```

Text Classification

```
{
  "source": "content://I like this product ",
  "id": "XGDVGS",
  "annotation": [
    {
      "type": "modelarts/text_classification",
      "name": " positive",
      "annotated-by": "human",
      "creation-time": "2019-01-23 11:30:30"
    }
  ]
}
```

The **content** parameter indicates the text to be labeled (in UTF-8 encoding format, which can be Chinese). The other parameters are the same as those described in [Image Classification](#). For details, see [Table 2-14](#).

Named Entity Recognition

```
{
  "source": "content://Michael Jordan is the most famous basketball player in the world.",
  "usage": "TRAIN",
  "annotation": [
    {
      "type": "modelarts/text_entity",
      "name": "Person",
      "property": {
        "@modelarts:start_index": 0,
        "@modelarts:end_index": 14
      },
      "annotated-by": "human",
      "creation-time": "2019-01-23 11:30:30"
    },
    {
      "type": "modelarts/text_entity",
      "name": "Category",
      "property": {
        "@modelarts:start_index": 34,
        "@modelarts:end_index": 44
      },
      "annotated-by": "human",
      "creation-time": "2019-01-23 11:30:30"
    }
  ]
}
```

The parameters such as **source**, **usage**, and **annotation** are the same as those described in [Image Classification](#). For details, see [Table 2-14](#).

[Table 2-19](#) describes the property parameters. For example, if you want to extract **Michael** from "**source**": "**content**://**Michael Jordan**", the value of **start_index** is **0** and that of **end_index** is **7**.

Table 2-19 Description of **property** parameters

Parameter	Data Type	Description
@modelarts:start_index	Integer	Start position of the text. The value starts from 0, including the characters specified by start_index .
@modelarts:end_index	Integer	End position of the text, excluding the characters specified by end_index .

Text Triplet

```
{
  "source":"content://"Three Body" is a series of long science fiction novels created by Liu Cix.",
  "usage":"TRAIN",
  "annotation":[
    {
      "type":"modelarts/text_entity",
      "name":"Person",
      "id":"E1",
      "property":{
        "@modelarts:start_index":67,
        "@modelarts:end_index":74
      },
      "annotated-by":"human",
      "creation-time":"2019-01-23 11:30:30"
    },
    {
      "type":"modelarts/text_entity",
      "name":"Book",
      "id":"E2",
      "property":{
        "@modelarts:start_index":0,
        "@modelarts:end_index":12
      },
      "annotated-by":"human",
      "creation-time":"2019-01-23 11:30:30"
    },
    {
      "type":"modelarts/text_triplet",
      "name":"Author",
      "id":"R1",
      "property":{
        "@modelarts:from":"E1",
        "@modelarts:to":"E2"
      },
      "annotated-by":"human",
      "creation-time":"2019-01-23 11:30:30"
    },
    {
      "type":"modelarts/text_triplet",
      "name":"Works",
      "id":"R2",
      "property":{
        "@modelarts:from":"E2",
        "@modelarts:to":"E1"
      },
      "annotated-by":"human",
      "creation-time":"2019-01-23 11:30:30"
    }
  ]
}
```

The parameters such as **source**, **usage**, and **annotation** are the same as those described in [Image Classification](#). For details, see [Table 2-14](#).

[Table 5 property parameters](#) describes the **property** parameters.

@modelarts:start_index and **@modelarts:end_index** are the same as those of named entity recognition. For example, when **source** is set to **content://"Three Body" is a series of long science fiction novels created by Liu Cix., Liu Cix** is an entity person, **Three Body** is an entity book, the person is the author of the book, and the book is works of the person.

Table 2-20 Description of **property** parameters

Parameter	Data Type	Description
@modelarts:start_index	Integer	Start position of the triplet entities. The value starts from 0, including the characters specified by start_index .
@modelarts:end_index	Integer	End position of the triplet entities, excluding the characters specified by end_index .
@modelarts:from	String	Start entity ID of the triplet relationship.
@modelarts:to	String	Entity ID pointed to in the triplet relationship.

Object Detection

```
{
  "source":"s3://path/to/image1.jpg",
  "usage":"TRAIN",
  "hard":"true",
  "hard-coefficient":0.8,
  "annotation": [
    {
      "type":"modelarts/object_detection",
      "annotation-loc": "s3://path/to/annotation1.xml",
      "annotation-format":"PASCAL VOC",
      "annotated-by":"human",
      "creation-time":"2019-01-23 11:30:30"
    }
  ]
}
```

- The parameters such as **source**, **usage**, and **annotation** are the same as those described in [Image Classification](#). For details, see [Table 2-14](#).
- **annotation-loc** indicates the path for saving the label file. This parameter is mandatory for object detection and image segmentation but optional for other labeling types.
- **annotation-format** indicates the format of the label file. This parameter is optional. The default value is **PASCAL VOC**. Only **PASCAL VOC** is supported.

Table 2-21 PASCAL VOC format parameters

Parameter	Mandatory	Description
folder	Yes	Directory where the data source is located

Parameter	Mandatory	Description
filename	Yes	Name of the file to be labeled
size	Yes	Image pixel <ul style="list-style-type: none"> • width: image width. This parameter is mandatory. • height: image height. This parameter is mandatory. • depth: number of image channels. This parameter is mandatory.
segmented	Yes	Segmented or not
object	Yes	Object detection information. Multiple object{} functions are generated for multiple objects. <ul style="list-style-type: none"> • name: class of the labeled content. This parameter is mandatory. • pose: shooting angle of the labeled content. This parameter is mandatory. • truncated: whether the labeled content is truncated (0 indicates that the content is not truncated). This parameter is mandatory. • occluded: whether the labeled content is occluded (0 indicates that the content is not occluded). This parameter is mandatory. • difficult: whether the labeled object is difficult to identify (0 indicates that the object is easy to identify). This parameter is mandatory. • confidence: confidence score of the labeled object. The value ranges from 0 to 1. This parameter is optional. • bndbox: bounding box type. This parameter is mandatory. For details about the possible values, see Table 2-22.

Table 2-22 Description of bounding box types

Type	Shape	Labeling Information
point	Point	Coordinates of a point <x>100<x> <y>100<y>

Type	Shape	Labeling Information
line	Line	Coordinates of points <x1>100<x1> <y1>100<y1> <x2>200<x2> <y2>200<y2>
bndbox	Rectangle	Coordinates of the upper left and lower right points <xmin>100<xmin> <ymin>100<ymin> <xmax>200<xmax> <ymin>200<ymin>
polygon	Polygon	Coordinates of points <x1>100<x1> <y1>100<y1> <x2>200<x2> <y2>100<y2> <x3>250<x3> <y3>150<y3> <x4>200<x4> <y4>200<y4> <x5>100<x5> <y5>200<y5> <x6>50<x6> <y6>150<y6>
circle	Circle	Center coordinates and radius <cx>100<cx> <cy>100<cy> <r>50<r>

Example:

```

<annotation>
  <folder>test_data</folder>
  <filename>260730932.jpg</filename>
  <size>
    <width>767</width>
    <height>959</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>point</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <occluded>0</occluded>
  </object>
</annotation>

```

```

<difficult>0</difficult>
  <point>
    <x1>456</x1>
    <y1>596</y1>
  </point>
</object>
<object>
  <name>line</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <occluded>0</occluded>
  <difficult>0</difficult>
  <line>
    <x1>133</x1>
    <y1>651</y1>
    <x2>229</x2>
    <y2>561</y2>
  </line>
</object>
<object>
  <name>bag</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <occluded>0</occluded>
  <difficult>0</difficult>
  <bndbox>
    <xmin>108</xmin>
    <ymin>101</ymin>
    <xmax>251</xmax>
    <ymin>238</ymin>
  </bndbox>
</object>
<object>
  <name>boots</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <occluded>0</occluded>
  <difficult>0</difficult>
  <hard-coefficient>0.8</hard-coefficient>
  <polygon>
    <x1>373</x1>
    <y1>264</y1>
    <x2>500</x2>
    <y2>198</y2>
    <x3>437</x3>
    <y3>76</y3>
    <x4>310</x4>
    <y4>142</y4>
  </polygon>
</object>
<object>
  <name>circle</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <occluded>0</occluded>
  <difficult>0</difficult>
  <circle>
    <cx>405</cx>
    <cy>170</cy>
    <r>100</r>
  </circle>
</object>
</annotation>

```

Sound Classification

```

{
  "source":
    "s3://path/to/pets.wav",

```

```

"annotation": [
  {
    "type": "modelarts/audio_classification",
    "name": "cat",
    "annotated-by": "human",
    "creation-time": "2019-01-23 11:30:30"
  }
]
}

```

The parameters such as **source**, **usage**, and **annotation** are the same as those described in [Image Classification](#). For details, see [Table 2-14](#).

Speech Labeling

```

{
  "source": "s3://path/to/audio1.wav",
  "annotation": [
    {
      "type": "modelarts/audio_content",
      "property": {
        "@modelarts:content": "Today is a good day."
      },
      "annotated-by": "human",
      "creation-time": "2019-01-23 11:30:30"
    }
  ]
}

```

- The parameters such as **source**, **usage**, and **annotation** are the same as those described in [Image Classification](#). For details, see [Table 2-14](#).
- The **@modelarts:content** parameter in **property** indicates speech labeling. The data type is **String**.

Speech Paragraph Labeling

```

{
  "source": "s3://path/to/audio1.wav",
  "usage": "TRAIN",
  "annotation": [
    {
      "type": "modelarts/audio_segmentation",
      "property": {
        "@modelarts:start_time": "00:01:10.123",
        "@modelarts:end_time": "00:01:15.456",
        "@modelarts:source": "Tom",
        "@modelarts:content": "How are you?"
      },
      "annotated-by": "human",
      "creation-time": "2019-01-23 11:30:30"
    },
    {
      "type": "modelarts/audio_segmentation",
      "property": {
        "@modelarts:start_time": "00:01:22.754",
        "@modelarts:end_time": "00:01:24.145",
        "@modelarts:source": "Jerry",
        "@modelarts:content": "I'm fine, thank you."
      },
      "annotated-by": "human",
      "creation-time": "2019-01-23 11:30:30"
    }
  ]
}

```

- The parameters such as **source**, **usage**, and **annotation** are the same as those described in [Image Classification](#). For details, see [Table 2-14](#).
- [Table 2-23](#) describes the **property** parameters.

Table 2-23 Description of **property** parameters

Parameter	Data Type	Description
@modelarts:start_time	String	Start time of the sound. The format is hh:mm:ss.SSS . hh indicates the hour, mm indicates the minute, ss indicates the second, and SSS indicates the millisecond.
@modelarts:end_time	String	End time of the sound. The format is hh:mm:ss.SSS . hh indicates the hour, mm indicates the minute, ss indicates the second, and SSS indicates the millisecond.
@modelarts:source	String	Sound source
@modelarts:content	String	Sound content

Video Labeling

```
{
  "annotation": [{
    "annotation-format": "PASCAL VOC",
    "type": "modelarts/object_detection",
    "annotation-loc": "s3://path/to/annotation1_t1.473722.xml",
    "creation-time": "2020-10-09 14:08:24",
    "annotated-by": "human"
  }],
  "usage": "train",
  "property": {
    "@modelarts:parent_duration": 8,
    "@modelarts:parent_source": "s3://path/to/annotation1.mp4",
    "@modelarts:time_in_video": 1.473722
  },
  "source": "s3://input/path/to/annotation1_t1.473722.jpg",
  "id": "43d88677c1e9a971eeb692a80534b5d5",
  "sample-type": 0
}
```

- The parameters such as **source**, **usage**, and **annotation** are the same as those described in [Image Classification](#). For details, see [Table 2-14](#).
- **annotation-loc** indicates the path for saving the label file. This parameter is mandatory for object detection but optional for other labeling types.
- **annotation-format** indicates the format of the label file. This parameter is optional. The default value is **PASCAL VOC**. Only **PASCAL VOC** is supported.
- **sample-type** indicates a sample format. Value **0** indicates image, **1** text, **2** audio, **4** table, and **6** video.

Table 2-24 property parameters

Parameter	Data Type	Description
@modelarts:parent_duration	Double	Duration of the labeled video, in seconds
@modelarts:time_in_video	Double	Timestamp of the labeled video frame, in seconds
@modelarts:parent_source	String	OBS path of the labeled video

Table 2-25 PASCAL VOC format parameters

Parameter	Mandatory	Description
folder	Yes	Directory where the data source is located
filename	Yes	Name of the file to be labeled
size	Yes	Image pixel <ul style="list-style-type: none">• width: image width. This parameter is mandatory.• height: image height. This parameter is mandatory.• depth: number of image channels. This parameter is mandatory.
segmented	Yes	Segmented or not

Parameter	Mandatory	Description
object	Yes	<p>Object detection information. Multiple object{} functions are generated for multiple objects.</p> <ul style="list-style-type: none"> • name: class of the labeled content. This parameter is mandatory. • pose: shooting angle of the labeled content. This parameter is mandatory. • truncated: whether the labeled content is truncated (0 indicates that the content is not truncated). This parameter is mandatory. • occluded: whether the labeled content is occluded (0 indicates that the content is not occluded). This parameter is mandatory. • difficult: whether the labeled object is difficult to identify (0 indicates that the object is easy to identify). This parameter is mandatory. • confidence: confidence score of the labeled object. The value ranges from 0 to 1. This parameter is optional. • bndbox: bounding box type. This parameter is mandatory. For details about the possible values, see Table 2-26.

Table 2-26 Bounding box types

Type	Shape	Labeling Information
point	Point	Coordinates of a point <x>100<x> <y>100<y>
line	Line	Coordinates of points <x1>100<x1> <y1>100<y1> <x2>200<x2> <y2>200<y2>
bndbox	Rectangle	Coordinates of the upper left and lower right points <xmin>100<xmin> <ymin>100<ymin> <xmax>200<xmax> <ymin>200<ymin>

Type	Shape	Labeling Information
polygon	Polygon	Coordinates of points <x1>100<x1> <y1>100<y1> <x2>200<x2> <y2>100<y2> <x3>250<x3> <y3>150<y3> <x4>200<x4> <y4>200<y4> <x5>100<x5> <y5>200<y5> <x6>50<x6> <y6>150<y6>
circle	Circle	Center coordinates and radius <cx>100<cx> <cy>100<cy> <r>50<r>

Example:

```

<annotation>
  <folder>test_data</folder>
  <filename>260730932_t1.473722.jpg.jpg</filename>
  <size>
    <width>767</width>
    <height>959</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>point</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <occluded>0</occluded>
    <difficult>0</difficult>
    <point>
      <x1>456</x1>
      <y1>596</y1>
    </point>
  </object>
  <object>
    <name>line</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <occluded>0</occluded>
    <difficult>0</difficult>
    <line>
      <x1>133</x1>
      <y1>651</y1>
      <x2>229</x2>
      <y2>561</y2>
    </line>
  </object>

```

```
<object>
  <name>bag</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <occluded>0</occluded>
  <difficult>0</difficult>
  <bndbox>
    <xmin>108</xmin>
    <ymin>101</ymin>
    <xmax>251</xmax>
    <ymax>238</ymax>
  </bndbox>
</object>
<object>
  <name>boots</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <occluded>0</occluded>
  <difficult>0</difficult>
  <hard-coefficient>0.8</hard-coefficient>
  <polygon>
    <x1>373</x1>
    <y1>264</y1>
    <x2>500</x2>
    <y2>198</y2>
    <x3>437</x3>
    <y3>76</y3>
    <x4>310</x4>
    <y4>142</y4>
  </polygon>
</object>
<object>
  <name>circle</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <occluded>0</occluded>
  <difficult>0</difficult>
  <circle>
    <cx>405</cx>
    <cy>170</cy>
    <r>100</r>
  </circle>
</object>
</annotation>
```

2.5 Exporting Data

A dataset includes labeled and unlabeled data. You can select images or filter data based on the filter criteria and export to a new dataset or the specified OBS directory. In addition, you can view the task history to learn about the export records.

NOTE

Only datasets of image classification, object detection, image segmentation, and free format types can be exported.

- For image classification datasets, only the label files in TXT format can be exported.
- For object detection datasets, only XML label files in Pascal VOC format can be exported.
- For image segmentation datasets, only XML label files in Pascal VOC format and mask images can be exported.
- For free format datasets, all files of the datasets can be exported.

Exporting Data to a New Dataset

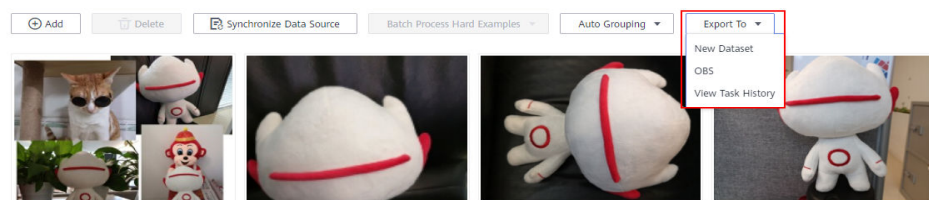
1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. In the dataset list, select the dataset of the object detection or image classification type and click the dataset name to go to the **Dashboard** tab page of the dataset.

NOTE

For a dataset of the free format type, you can click the dataset name to directly access the dataset details page and go to 4.

3. On the **Dashboard** page of the dataset, click **Label** in the upper right corner. The dataset details page is displayed.
4. On the dataset details page, select or filter data to be exported. Click **Export To** and choose **New Dataset** from the drop-down list.

Figure 2-52 Selecting or filtering images to be exported



5. In the displayed **Export to New Dataset** dialog box, enter the related information and click **OK**.

Name: name of the new dataset

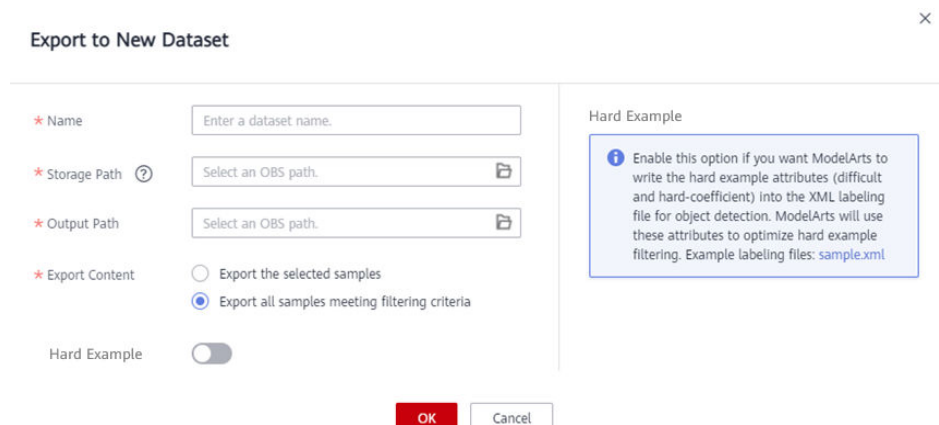
Storage Path: input path of the new dataset, that is, the OBS path where the data to be exported is stored

Output Path: output path of the new dataset, that is, the output path after labeling is complete. The output path cannot be the same as the storage path, and the output path cannot be a subdirectory of the storage path.

Export Content: The options are **Export the selected samples** and **Export all samples meeting filtering criteria**.

Hard Example Filtering: Select whether to enable hard example filtering.

Figure 2-53 Exporting to a new dataset



- After the data is exported, you can view the new dataset in the dataset list.

Exporting Data to OBS

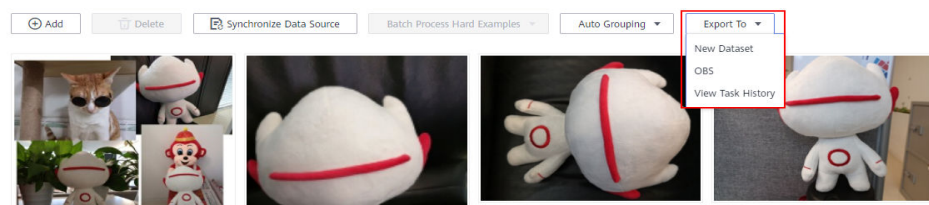
- Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
- In the dataset list, select the dataset of the object detection or image classification type and click the dataset name to go to the **Dashboard** tab page of the dataset.

NOTE

For a dataset of the free format type, you can click the dataset name to directly access the dataset details page and go to 4.

- On the **Dashboard** page of the dataset, click **Label** in the upper right corner. The dataset details page is displayed.
- On the dataset details page, select or filter data to be exported. Click **Export To** and choose **OBS** from the drop-down list.

Figure 2-54 Selecting or filtering images to be exported



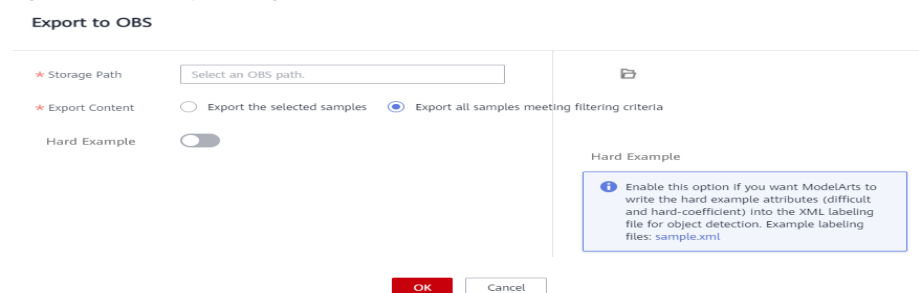
- In the displayed **Export to OBS** dialog box, enter the related information and click **OK**.

Storage Path: path where the data to be exported is stored. You are advised not to save data to the input or output path of the current dataset.

Export Content: The options are **Export the selected samples** and **Export all samples meeting filtering criteria**.

Hard Example: Select whether to enable hard examples.

Figure 2-55 Exporting to OBS



- After the data is exported, you can view it in the specified path.

Viewing the Task History

When you export data to a new dataset or OBS, you can view the export task details in the **View Task History** dialog box.

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. In the dataset list, select the dataset of the object detection or image classification type and click the dataset name to go to the **Dashboard** tab page of the dataset.


NOTE

For a dataset of the free format type, you can click the dataset name to directly access the dataset details page and go to [4](#).

3. On the **Dashboard** page of the dataset, click **Label** in the upper right corner. The dataset details page is displayed.
4. On the dataset details page, select or filter data to be exported. Click **Export To** and choose **View Task History** from the drop-down list.
5. In the **View Task History** dialog box, view the export task history of the current dataset. Information about **Task ID**, **Created**, **Type**, **Path**, **Total**, and **Status** is included.

Figure 2-56 Viewing the task history

View Task History

Task ID	Created	Type	Path	Total	Status
WDRhZmJZnPpSS...	Mar 21, 2020 16:5...	OBS	/test-modelarts2/d...	--	 Running [...

2.6 Modifying a Dataset

For a created dataset, you can modify its basic information to match service changes.

Prerequisites

You have created a dataset.

Modifying the Basic Information About a Dataset

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. In the dataset list, choose **More > Modify** in the **Operation** column. Alternatively, you can click the dataset name to go to the **Dashboard** tab page of the dataset, and click **Modify** in the upper right corner.
3. Modify basic information about the dataset and then click **OK**. Refer to [Table 2-27](#) for details.

Figure 2-57 Modifying a dataset

Modify Dataset

Name

Description

0/256

Label Set

▼ + 🗑️

+ Add Label

OK Cancel

Table 2-27 Parameters

Parameter	Description
Name	Enter the name of the dataset. A dataset name can contain only letters, digits, underscores (_), and hyphens (-).
Description	Enter a brief description for the dataset.
Label Set	The label set varies depending on the dataset type. For details about how to modify the label set, see the parameters of different dataset types in Creating a Dataset (Old Version) . There is no limit on the number of label sets.

2.7 Publishing a Dataset

ModelArts distinguishes data of the same source according to versions labeled at different time, which facilitates the selection of dataset versions during subsequent model building and development. After labeling the data, you can publish the dataset to generate a new dataset version.

About Dataset Versions

- For a newly created dataset (before publishing), there is no dataset version information. The dataset must be published before being used for model development or training.
- The default naming rules of dataset versions are V001 and V002 in ascending order. You can customize the version number during publishing.
- You can set any version to the current directory. Then the details of the version are displayed on the dataset details page.

- You can obtain the dataset in the manifest file format corresponding to each dataset version based on the value of **Storage Path**. The dataset can be used when you import data or filter hard examples.
- The version of a table dataset cannot be changed.

Publishing a Dataset

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. In the dataset list, click **Publish** in the **Operation** column.
Alternatively, you can click the dataset name to go to the **Dashboard** tab page of the dataset, and click **Publish** in the upper right corner.
3. In the displayed dialog box, set the parameters and click **OK**.

Table 2-28 Parameters for publishing a dataset

Parameter	Description
Version Name	The naming rules of V001 and V002 in ascending order are used by default. A version name can be customized. Only letters, digits, hyphens (-), and underscores (_) are allowed.
Format	Only table datasets support version format setting. Available values are CSV and CarbonData . NOTE If the exported CSV file contains any command starting with =, +, -, or @, ModelArts automatically adds the Tab setting and escapes the double quotation marks (") for security purposes.
Splitting	Only image classification, object detection, text classification, and sound classification datasets support data splitting. By default, this function is disabled. After this function is enabled, you need to set the training and validation ratios. Enter a value ranging from 0 to 1 for Training Set Ratio . After the training set ratio is set, the validation set ratio is determined. The sum of the training set ratio and the validation set ratio is 1. The training set ratio is the ratio of sample data used for model training. The validation set ratio is the ratio of the sample data used for model validation. The training and validation ratios affect the performance of training templates.
Description	Description of the current dataset version.
Hard Example	Only image classification and object detection datasets support hard example attributes. By default, this function is disabled. After this function is enabled, information such as the hard example attributes of the dataset are written to the corresponding manifest file.

Figure 2-58 Publishing a dataset

Publish New Version

* Version

Splitting

Description

Hard Example

Hard Example

Enable this option if you want ModelArts to write the hard example attributes (difficult, hard-coefficient, and hard-reasons) into the XML and manifest labeling files. ModelArts will use these attributes to optimize hard example filtering. Example labeling files: [sample.manifest](#)

After the version is published, you can go to the **Version Manager** tab page to view the detailed information. By default, the system sets the latest version to the current directory.

Directory Structure of Related Files After the Dataset Is Published

Datasets are managed based on OBS directories. After a new version is published, the directory is generated based on the new version in the output dataset path.

Take an image classification dataset as an example. After the dataset is published, the directory structure of related files generated in OBS is as follows:

```
|-- user-specified-output-path
|  |-- DatasetName-datasetId
|     |-- annotation
|        |-- VersionMame1
|           |-- VersionMame1.manifest
|        |-- VersionMame2
|           ...
|        |-- ...
```

The following uses object detection as an example. If a manifest file is imported to the dataset, the following provides the directory structure of related files after the dataset is published:

```
|-- user-specified-output-path
|  |-- DatasetName-datasetId
|     |-- annotation
|        |-- VersionMame1
|           |-- VersionMame1.manifest
|           |-- annotation
|              |-- file1.xml
|        |-- VersionMame2
|           ...
|        |-- ...
```

Take video labeling as an example. After the dataset is published, the labeling result file (XML) is stored in the dataset output directory.

```
|-- user-specified-output-path
|  |-- DatasetName-datasetId
```

```
|-- annotation
  |-- VersionMame1
    |-- VersionMame1.manifest
    |-- annotations
    |-- images
      |-- videoName1
        |-- videoName1.timestamp.xml
      |-- videoName2
        |-- videoName2.timestamp.xml
    |-- VersionMame2
  ...
|-- ...
```

The key frames for video labeling are stored in the dataset input directory.

```
|-- user-specified-input-path
  |-- images
    |-- videoName1
      |-- videoName1.timestamp.jpg
    |-- videoName2
      |-- videoName2.timestamp.jpg
```

2.8 Deleting a Dataset

If a dataset is no longer in use, you can delete it to release resources.

NOTE

After a dataset is deleted, if you need to delete the data in the dataset input and output paths in OBS to release resources, delete the data and the OBS folders on the OBS Console.

Procedure

1. In the left navigation pane, choose **Data Management > Datasets**. On the **Datasets** page, choose **More > Delete** in the **Operation** column of the dataset.
2. In the displayed dialog box, click **OK**.

NOTE

After a dataset is deleted, some functions such as dataset version management become unavailable. Exercise caution when performing this operation. However, the original data and labeling data of the dataset are still stored in OBS.

2.9 Managing Dataset Versions

After labeling data, you can publish the dataset to multiple versions for management. For the published versions, you can view the dataset version updates, set the current version, and delete versions. For details about dataset versions, see [About Dataset Versions](#).

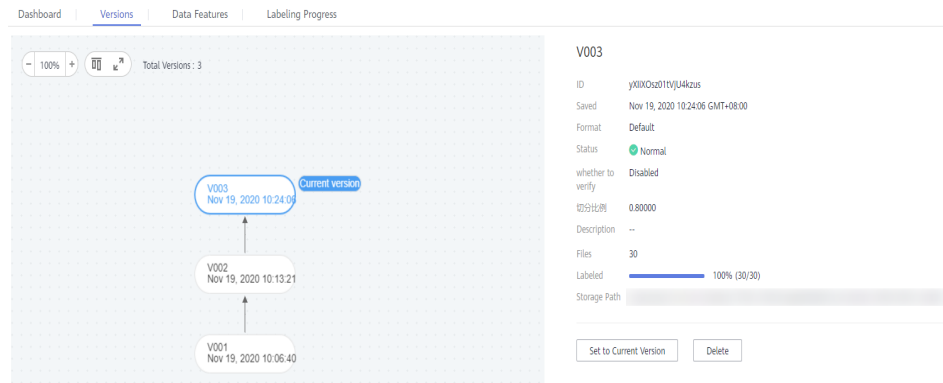
For details about how to publish a new version, see [Publishing a Dataset](#).

Viewing Dataset Version Updates

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. In the dataset list, choose **More > Manage Version** in the **Operation** column. The **Manage Version** tab page is displayed.

You can view basic information about the dataset, and view the versions and publish time on the left.

Figure 2-59 Viewing dataset versions



Setting to Current Version

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. In the dataset list, choose **More > Manage Version** in the **Operation** column. The **Manage Version** tab page is displayed.
3. On the **Manage Version** tab page, select the desired dataset version, and click **Set to Current Version** in the basic information area on the right. After the setting is complete, **Current version** is displayed to the right of the version name.

NOTE

Only the version in **Normal** status can be set to the current version.

Figure 2-60 Setting to current version



Deleting a Dataset Version

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. In the dataset list, choose **More > Manage Version** in the **Operation** column. The **Manage Version** tab page is displayed.

3. Locate the row that contains the target version, and click **Delete** in the **Operation** column. In the dialog box that is displayed, click **OK**.

NOTE

Deleting a dataset version does not remove the original data. Data and its labeling information are still stored in the OBS directory. However, if it is deleted, you cannot manage the dataset versions on the ModelArts management console. Exercise caution when performing this operation.

2.10 Auto Labeling

In addition to manual labeling, ModelArts also provides the auto labeling function to quickly label data, reducing the labeling time by more than 70%. Auto labeling means learning and training are performed based on the selected labels and images and an existing model is selected to quickly label the remaining images.

Background

- Only datasets of image classification and object detection types support the auto labeling function.
- To enable **Auto Labeling**, add at least two types of labels to the dataset and add each type of the label to at least 5 objects.
- At least one unlabeled image must exist when you enable **Auto Labeling**.
- Before enabling **Auto Labeling**, ensure that no auto labeling task is in progress in the system.
- Check the image data used for labeling and ensure that no RGBA four-channel image exists in the image data. If four-channel images exist, the auto labeling task will fail. Therefore, delete the four-channel images from the dataset and then start the auto labeling task.

Auto Labeling

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. In the dataset list, select a dataset of the object detection or image classification type and click **Auto Labeling** in the **Operation** column to start an intelligent labeling job.
3. On the **Enable Auto Labeling** page, select **Active learning** or **Pre-labeling**. For details, see [Table 2-29](#) and [Table 2-30](#).

Table 2-29 Active learning

Parameter	Description
Auto Labeling Type	Active learning: The system uses semi-supervised learning and hard example filtering to perform auto labeling, reducing manual labeling workload and helping you find hard examples.

Parameter	Description
Algorithm Type	For a dataset of the image classification type, you need to set the following parameters: Fast: Use the labeled samples for training. Precise: Use labeled and unlabeled samples for semi-supervised training, which improves the model precision.

Table 2-30 Pre-labeling

Parameter	Description
Auto Labeling Type	Pre-labeling: Select an AI application created on the AI Applications page. Ensure that the model type matches the dataset labeling type. After the pre-labeling is complete, if the labeling result complies with the standard labeling format defined by the platform, the system filters hard examples. This step does not affect the pre-labeling result.
Model and Version	<ul style="list-style-type: none"> My AI Applications: You can select a model based on site requirements. Click the drop-down arrow on the left of the target AI application and select a proper version. For details about how to create an AI application, see Creating an AI Application.
Specifications	In the drop-down list, you can select the node specifications supported by ModelArts.
Compute Nodes	The default value is 1 . You can select a value based on site requirements. The maximum value is 5 .

 **NOTE**

- For datasets of the object detection type, only rectangular boxes can be recognized and labeled when **Active Learning** is selected.
- If there are too many auto labeling jobs in the system, the jobs may be queued. As a result, the jobs are always in the labeling state. The system will complete labeling jobs in sequence.


Figure 2-61 Enabling auto labeling (image classification)

Enable Auto Labeling

Auto Labeling Type

Active learning Pre-labeling

The system uses semi-supervised learning and hard example filtering to perform auto labeling, reducing manual labeling workload and helping you find hard examples.

 Auto Labeling can identify and add only rectangle labeling boxes.

Limited-time free

Auto Labeling is billed based on the training duration. [Pricing details](#)

Submit


Figure 2-62 Enabling auto labeling (object detection)

Enable Auto Labeling

Auto Labeling Type

Active learning Pre-labeling

The system uses semi-supervised learning and hard example filtering to perform auto labeling, reducing manual labeling workload and helping you find hard examples.

 Auto Labeling can identify and add only rectangle labeling boxes.

Limited-time free

Auto Labeling is billed based on the training duration. [Pricing details](#)

Submit

Figure 2-63 Enabling auto labeling (pre-labeling)

Enable Auto Labeling ×

Auto Labeling Type

Active learning Pre-labeling

Select a model created on the Model Management page. Ensure that the model type matches the dataset labeling type. After the pre-labeling is complete, if the labeling result complies with the standard labeling format defined by the platform, the system filters hard examples. This step does not affect the pre-labeling result.

* Model and Version

Select

* Specifications

* Compute Nodes

Limited-time free

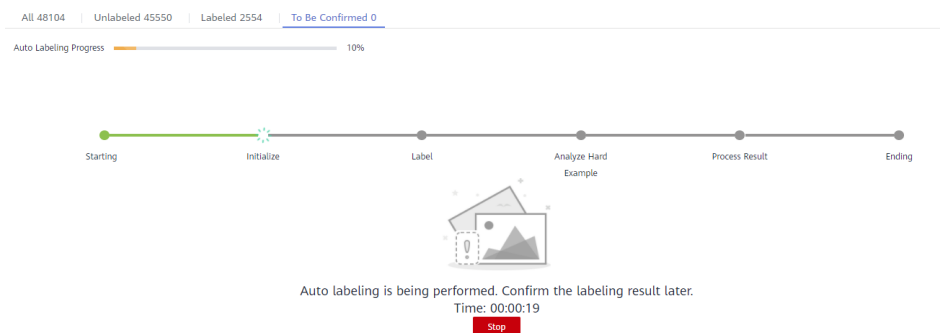
Auto Labeling is billed based on the training duration. [Pricing details](#)

Submit

4. After setting the parameters, click **Submit** to enable auto labeling.
5. In the dataset list, click a dataset name to go to the **Dashboard** page.
6. On the **Dashboard** page of the dataset, click **Label** in the upper right corner. The dataset details page is displayed.
7. On the dataset details page, click the **To be Confirmed** tab to view the auto labeling progress.

You can also enable auto labeling or view the auto labeling history on this tab page.

Figure 2-64 Labeling progress



8. After auto labeling is complete, all the labeled images are displayed on the **To Be Confirmed** page.
 - Datasets of the image classification type

On the **To Be Confirmed** page, check whether labels are correct, select the correctly labeled images, and click **Labeled** to confirm the auto labeling results. The confirmed image will be categorized to the **Labeled** page.

You can manually modify the labels of the images marked as hard examples based on site requirements. For details, see [For datasets of the image classification type](#).
 - Datasets of the object detection type

On the **To Be Confirmed** page, click images to view their labeling details and check whether labels and target bounding boxes are correct. For the correctly labeled images, click **Labeled** to confirm the auto labeling results. The confirmed image will be categorized to the **Labeled** page.

You can manually modify the labels or target bounding boxes of the images marked as hard examples based on site requirements. For details, see [For datasets of the object detection type](#).

2.11 Confirming Hard Examples

In a labeling task that processes a large amount of data, auto labeling results cannot be directly used for training because the labeled images are insufficient at the initial stage of labeling. It takes a lot of time and manpower to adjust and confirm all unlabeled data one by one. To accelerate labeling progress, ModelArts embeds an auto hard example detection function for labeling unlabeled data in an

auto labeling task. This function provides suggestions on labeling priorities for remaining unlabeled images. The auto labeling result of an image with high labeling priority is not as expected. Therefore, this case is called a hard example.

The auto hard example detection function is used to automatically label hard examples during auto labeling and data collection and filtering. Further confirm and label hard example data, and add labeling results to the training dataset to obtain a trained model with higher precision. No manual intervention is required for hard example detection, and you only need to confirm and modify the labeled data, improving data management and labeling efficiency. In addition, you can supplement data similar to hard examples to improve the variety of the dataset and further improve the model training precision. Hard example management involves the following scenarios.

- [Confirming Hard Examples After Auto Labeling](#)
- [Labeling Data in a Dataset as Hard Examples](#)

NOTE

Only datasets of image classification and object detection types support the auto hard example detection function.

Confirming Hard Examples After Auto Labeling

During the execution of an auto labeling task, ModelArts automatically detects and labels hard examples. After the auto labeling task is complete, the labeling results of hard examples are displayed on the **To Be Confirmed** tab page. Modify hard example data and confirm the labeling result.

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
2. In the dataset list, select the dataset of the object detection or image classification type and click the dataset name to go to the **Dashboard** tab page of the dataset.
3. On the **Dashboard** page of the dataset, click **Label** in the upper right corner. The dataset details page is displayed.
4. On the dataset details page, click the **To Be Confirmed** tab to view and confirm hard examples.

NOTE

Labeling data is displayed on the **To Be Confirmed** tab page only after the auto labeling task is complete. Otherwise, no data is available on the tab page. For details about auto labeling, see [Auto Labeling](#).

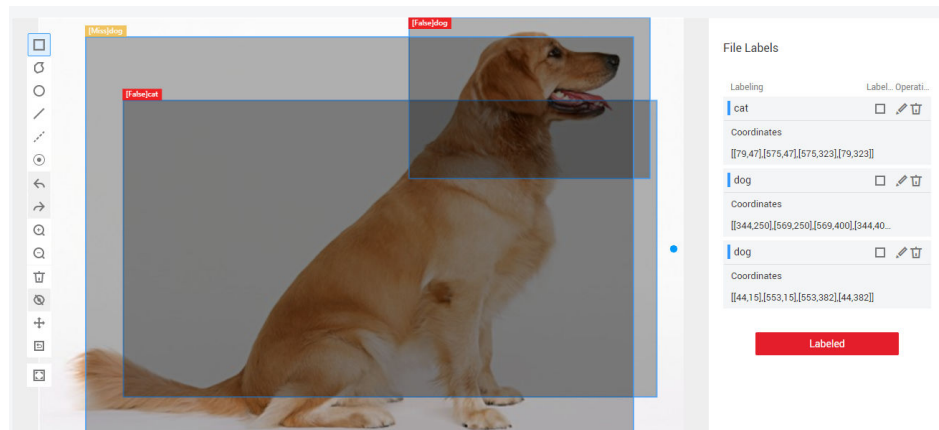
- For datasets of the object detection type

On the **To Be Confirmed** tab page, click an image to expand its labeling details. Check whether labeling information is correct, for example, whether the label is correct and whether the target bounding box is correctly added to the right position. If the auto labeling result is inaccurate, manually adjust the label or target bounding box and click **Labeled**. Then, the re-labeled data is displayed on the **Labeled** tab page.

In the hard example shown in [Figure 2-65](#), the position of the target bounding box for the **dog** label is incorrect. Use a bounding box to re-label the image again, for example, the **miss** bounding box shown in the

following figure. Then, delete the incorrect labeling bounding box, that is, the **false** bounding box. After manual adjustment, click **Labeled** to confirm the hard example.

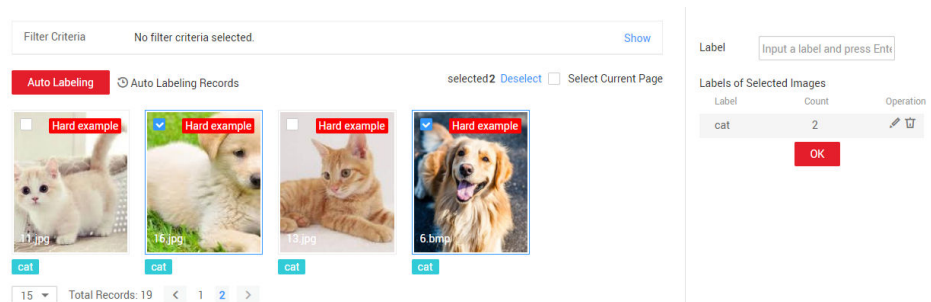
Figure 2-65 Confirming a hard example for object detection



- For datasets of the image classification type
On the **To Be Confirmed** tab page, check whether labels added to images with the **Hard example** mark are correct. Select the images that are incorrectly labeled, delete the incorrect labels, and add correct labels in **Label** on the right. Click **OK**. The selected images and its labeling details are displayed on the **Labeled** tab page.

As shown in **Figure 2-66**, the selected images are incorrectly labeled. Delete the incorrect labels on the right, add the **dog** label in **Label**, and click **OK** to confirm the hard examples.

Figure 2-66 Confirming hard examples for image classification



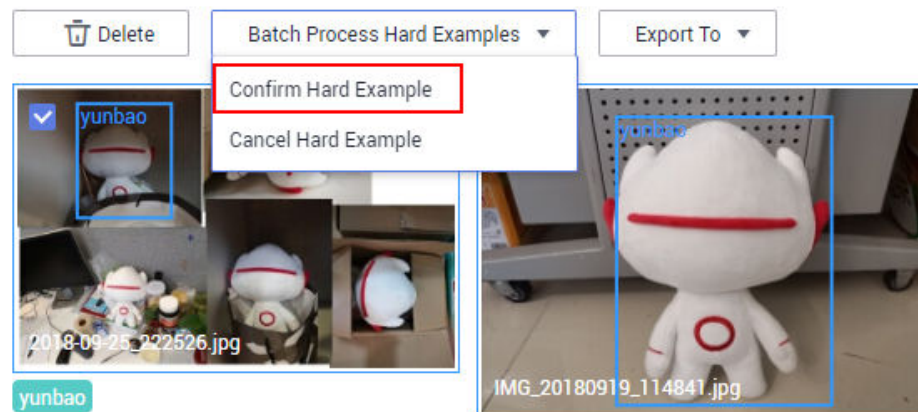
Labeling Data in a Dataset as Hard Examples

In a dataset, labeled or unlabeled image data can be labeled as hard examples. Data labeled as hard examples can be used to improve model precision through built-in rules during subsequent model training.

- Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Datasets**. The **Datasets** page is displayed.
- In the dataset list, select the dataset of the object detection or image classification type and click the dataset name to go to the **Dashboard** tab page of the dataset.

3. On the **Dashboard** page of the dataset, click **Label** in the upper right corner. The dataset details page is displayed.
4. On the dataset details page, click the **Labeled**, **Unlabeled**, or **All** tab, select the images to be labeled as hard examples, and choose **Batch process Hard Examples** > **Confirm Hard Example**. After the labeling is complete, a **Hard example** mark will be displayed in the upper right corner of a preview image.

Figure 2-67 Confirming hard examples



2.12 Auto Grouping

To improve the precision of auto labeling algorithms, you can evenly label multiple classes. ModelArts provides built-in grouping algorithms. You can enable auto grouping to improve data labeling efficiency.

Auto grouping can be understood as data labeling preprocessing. Clustering algorithms are used to cluster unlabeled images, and images are labeled or cleaned by group based on the clustering result.

For example, a user searches for *XX* through a search engine, downloads and uploads related images to the dataset, and then uses the auto grouping function to classify *XX* images, such as papers, posters, images confirmed as *XX*, and others. The user can quickly remove unwanted images from a group or select all images of a type and add labels to the images.

NOTE

Only datasets of image classification, object detection, and image segmentation types support the auto grouping function.

Starting Auto Grouping Tasks

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management** > **Datasets**. The **Datasets** page is displayed.
2. In the dataset list, select the dataset of the object detection or image classification type and click the dataset name to go to the **Dashboard** tab page of the dataset.
3. On the **Dashboard** page of the dataset, click **Label** in the upper right corner. The dataset details page is displayed.

4. On the **All** tab page of the dataset details page, choose **Auto Grouping > Start Task**.

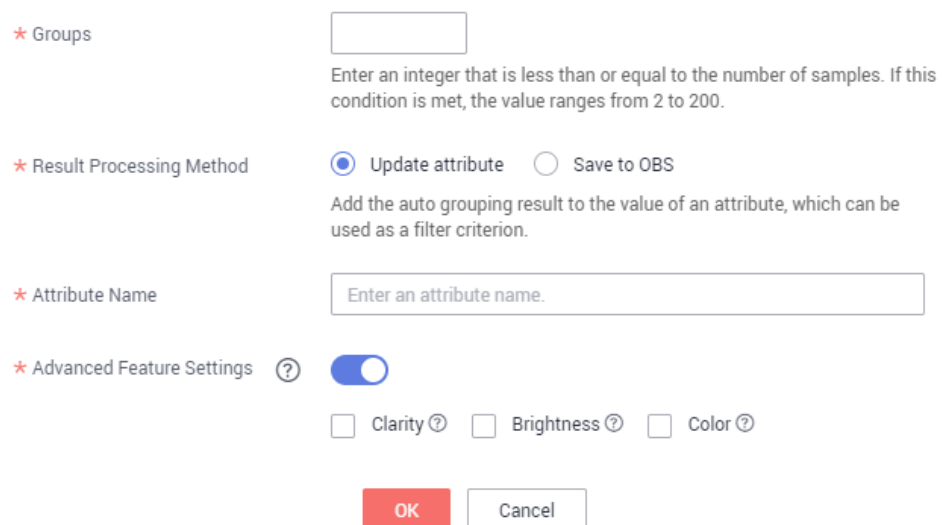
 **NOTE**

You can start auto group tasks or view task history only on the **All** tab page.

5. In the displayed **Auto Grouping** dialog box, set parameters and click **OK**.
 - **Groups**: Enter an integer from 2 to 200. The parameter value indicates the number of groups into which images are divided.
 - **Result Processing Method**: Select **Update attribute** or **Save to OBS**.
 - **Attribute Name**: If you select **Update attribute**, you need to enter an attribute name.
 - **Result Storage Path**: If you select **Save to OBS**, specify an OBS path.
 - **Advanced Feature Settings**: After this function is enabled, you can select **Clarity**, **Brightness**, and **Color** dimensions for the auto grouping function so that the grouping is based on the image brightness, color, and clarity. You can select multiple options.

Figure 2-68 Auto grouping

Auto Grouping



* Groups

Enter an integer that is less than or equal to the number of samples. If this condition is met, the value ranges from 2 to 200.

* Result Processing Method Update attribute Save to OBS

Add the auto grouping result to the value of an attribute, which can be used as a filter criterion.

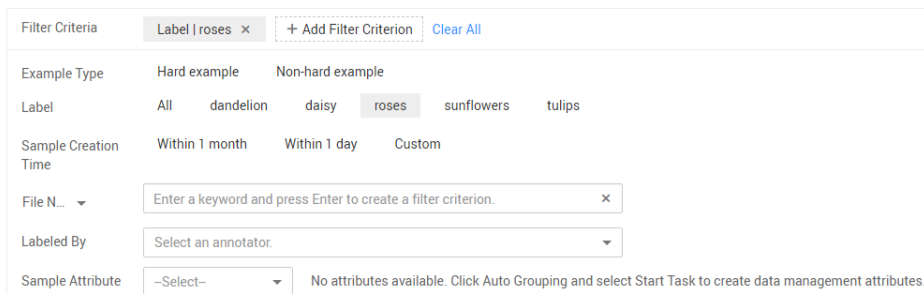
* Attribute Name

* Advanced Feature Settings Clarity Brightness Color

6. After the task is submitted, the task progress is displayed in the upper right corner of the page. After the task is complete, you can view the history of the auto grouping tasks to learn task status.

Viewing the Auto Grouping Result

On the **All** tab page of the dataset details page, expand **Filter Criteria**, set **Sample Attribute** to the attribute name of the auto grouping task, and set the sample attribute value to filter the grouping result.

Figure 2-69 Viewing the auto grouping result


Viewing Auto Grouping Task History

On the **All** tab page of the dataset details page, choose **Auto Grouping > View Task History**. In the **View Task History** dialog box, basic information about the auto grouping tasks of the current dataset is displayed.

Figure 2-70 Auto grouping task history

View Task History

If the Result Processing Method is Update attribute, you can select attribute values based on the sample attribute as a filter criterion to obtain the result. If the Result Processing Method is Save to OBS, you can view or download the grouping result in the storage path.

Created	Groups	Result Processin...	Storage Path/Att...	Status	Opera...
2020-03-13 09:20...	2	Update attribute	sunflowers	Running[The j...]	Stop

2.13 Data Features

Images or target bounding boxes are analyzed based on image features, such as blurs and brightness to draw visualized curves to help process datasets.

You can also select multiple versions of a dataset to view their curves for comparison and analysis.

Background

- Data feature analysis is only available for datasets whose labeling type is **Object detection** or **Image classification**.
- Data feature analysis is only available for the published datasets. The published dataset versions in **Default** format support data feature analysis.
- A data scope for feature analysis varies depending on the dataset type.
 - In a dataset of the object detection type, if the number of labeled samples is 0, the **Data Features** tab page is unavailable and data features are not displayed after a version is published. After the images are labeled and the version is published, and the data features of the labeled images are displayed.
 - In a dataset of the image classification type, if the number of labeled samples is 0, the **Data Features** tab page is unavailable and data features are not displayed after a version is published. After the images

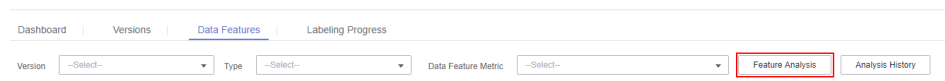
are labeled and the version is published, and the data features of all images are displayed.

- The analysis result is valid only when the number of images in a dataset reaches a certain level. Generally, more than 1,000 images are required.
- Image classification supports the following data feature metrics: **Resolution**, **Aspect Ratio**, **Brightness**, **Saturation**, **Blur Score**, and **Colorfulness**. Object detection supports all data feature metrics. [Supported Data Feature Metrics](#) provides all data feature metrics supported by ModelArts.

Data Feature Analysis

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management** > **Datasets**. The **Datasets** page is displayed.
2. Select a dataset and click **Data Features** in the **Operation** column. The **Data Features** tab page of the dataset page is displayed.
You can also click a dataset name to go to the dataset page and click the **Data Features** tab.
3. By default, feature analysis is not started for published datasets. You need to manually start feature analysis tasks for each dataset version. On the **Data Features** tab page, click **Feature Analysis**.

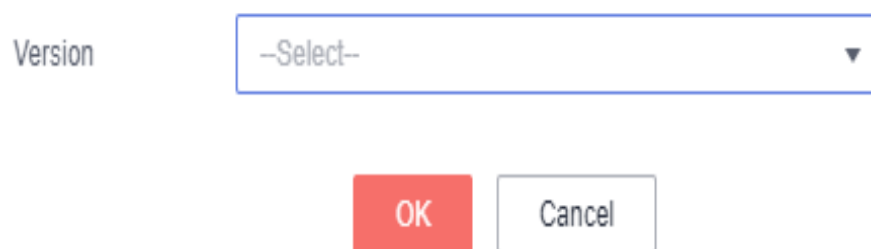
Figure 2-71 Feature Analysis



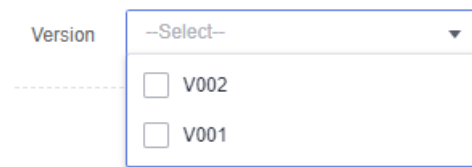
4. In the dialog box that is displayed, configure the dataset version for feature analysis and click **OK** to start analysis.

Version: Select a published version of the dataset.

Figure 2-72 Starting a data feature analysis task



5. After a data feature analysis task is started, it takes a certain period of time to complete, depending on the data volume. If the selected version is displayed in the **Version** drop-down list and can be selected, the analysis is complete.

Figure 2-73 Selecting a version for which feature analysis has been performed

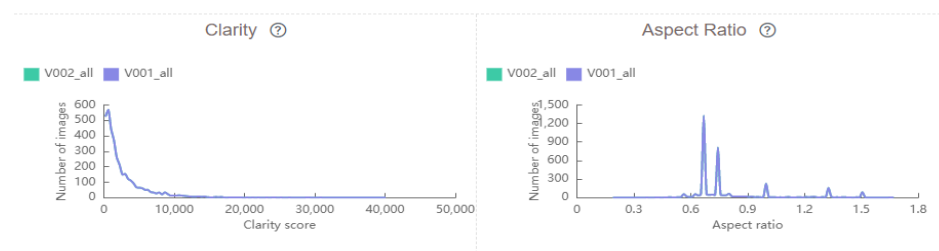
6. View the data feature analysis result.

Version: Select the version to be compared from the drop-down list. You can also select only one version.

Type: Select the type to be analyzed. The value can be **all**, **train**, **eval**, or **inference**.

Data Feature Metric: Select metrics to be displayed from the drop-down list. For details, see [Supported Data Feature Metrics](#).

Then, the selected version and metrics are displayed on the page, as shown in [Figure 2-74](#). The displayed chart helps you understand data distribution for better data processing.

Figure 2-74 Data feature analysis

7. View historical records of the analysis task.

After data feature analysis is complete, you can click **Task History** on the right of the **Data Features** tab page to view historical analysis tasks and their statuses in the dialog box that is displayed.

Figure 2-75 Viewing the task history

View Task History

Dataset Vers...	Task ID	Created	Duration(hh:...	Status
V002	Rdnjwum33T...	Apr 03, 2020 ...	00:01:50	Successful
V001	gpw2hG6D6...	Apr 02, 2020 ...	00:02:23	Successful

Supported Data Feature Metrics

Table 2-31 Data feature metrics

Metric	Description	Explanation
Resolution	Image resolution. An area value is used as a statistical value.	Metric analysis results are used to check whether there is an offset point. If an offset point exists, you can resize or delete the offset point.
Aspect Ratio	An aspect ratio is a proportional relationship between an image's width and height.	The chart of the metric is in normal distribution, which is generally used to compare the difference between the training set and the dataset used in the real scenario.

Metric	Description	Explanation
Brightness	Brightness is the perception elicited by the luminance of a visual target. A larger value indicates better image brightness.	The chart of the metric is in normal distribution. You can determine whether the brightness of the entire dataset is high or low based on the distribution center. You can adjust the brightness based on your application scenario. For example, if the application scenario is night, the brightness should be lower.
Saturation	Color saturation of an image. A larger value indicates that the entire image color is easier to distinguish.	The chart of the metric is in normal distribution, which is generally used to compare the difference between the training set and the dataset used in the real scenario.
Blur Score Clarity	Image clarity, which is calculated using the Laplace operator. A larger value indicates clearer edges and higher clarity.	You can determine whether the clarity meets the requirements based on the application scenario. For example, if data is collected from HD cameras, the clarity must be higher. You can sharpen or blur the dataset and add noises to adjust the clarity.
Colorfulness	Horizontal coordinate: Colorfulness of an image. A larger value indicates richer colors. Vertical coordinate: Number of images	Colorfulness on the visual sense, which is generally used to compare the difference between the training set and the dataset used in the real scenario.
Bounding Box Number	Horizontal coordinate: Number of bounding boxes in an image Vertical coordinate: Number of images	It is difficult for a model to detect a large number of bounding boxes in an image. Therefore, more images containing many bounding boxes are required for training.

Metric	Description	Explanation
<p>Std of Bounding Boxes Area Per Image</p> <p>Standard Deviation of Bounding Boxes Per Image</p>	<p>Horizontal coordinate: Standard deviation of bounding boxes in an image. If an image has only one bounding box, the standard deviation is 0. A larger standard deviation indicates higher bounding box size variation in an image.</p> <p>Vertical coordinate: Number of images</p>	<p>It is difficult for a model to detect a large number of bounding boxes with different sizes in an image. You can add data for training based on scenarios or delete data if such scenarios do not exist.</p>
<p>Aspect Ratio of Bounding Boxes</p>	<p>Horizontal coordinate: Aspect ratio of the target bounding boxes</p> <p>Vertical coordinate: Number of bounding boxes in all images</p>	<p>The chart of the metric is generally in Poisson distribution, which is closely related to application scenarios. This metric is mainly used to compare the differences between the training set and the validation set. For example, if the training set is a rectangle, the result will be significantly affected if the validation set is close to a square.</p>
<p>Area Ratio of Bounding Boxes</p>	<p>Horizontal coordinate: Area ratio of the target bounding boxes, that is, the ratio of the bounding box area to the entire image area. A larger value indicates a higher ratio of the object in the image.</p> <p>Vertical coordinate: Number of bounding boxes in all images</p>	<p>The metric is used to determine the distribution of anchors used in the model. If the target bounding box is large, set the anchor to a large value.</p>

Metric	Description	Explanation
Marginalization Value of Bounding Boxes	<p>Horizontal coordinate: Marginalization degree, that is, the ratio of the distance between the center point of the target bounding box and the center point of the image to the total distance of the image. A larger value indicates that the object is closer to the edge.</p> <p>Vertical coordinate: Number of bounding boxes in all images</p>	<p>Generally, the chart of the metric is in normal distribution. The metric is used to determine whether an object is at the edge of an image. If a part of an object is at the edge of an image, you can add a dataset or do not label the object.</p>
Overlap Score of Bounding Boxes Overlap Score of Bounding Boxes	<p>Horizontal coordinate: Overlap degree, that is, the part of a single bounding box overlapped by other bounding boxes. The value ranges from 0 to 1. A larger value indicates that more parts are overlapped by other bounding boxes.</p> <p>Vertical coordinate: Number of bounding boxes in all images</p>	<p>The metric is used to determine the overlapping degree of objects to be detected. Overlapped objects are difficult to detect. You can add a dataset or do not label some objects based on your needs.</p>
Brightness of Bounding Boxes Brightness of Bounding Boxes	<p>Horizontal coordinate: Brightness of the image in the target bounding box. A larger value indicates brighter image.</p> <p>Vertical coordinate: Number of bounding boxes in all images</p>	<p>Generally, the chart of the metric is in normal distribution. The metric is used to determine the brightness of an object to be detected. In some special scenarios, the brightness of an object is low and may not meet the requirements.</p>
Blur Score of Bounding Boxes Clarity of Bounding Boxes	<p>Horizontal coordinate: Clarity of the image in the target bounding box. A larger value indicates higher image clarity.</p> <p>Vertical coordinate: Number of bounding boxes in all images</p>	<p>The metric is used to determine whether the object to be detected is blurred. For example, a moving object may become blurred during collection and its data needs to be collected again.</p>

2.14 Team Labeling

2.14.1 Introduction to Team Labeling

Generally, a small data labeling task can be completed by an individual. However, team work is required to label a large dataset. ModelArts provides the team labeling function. A labeling team can be formed to manage labeling for the same dataset.


NOTE

The team labeling function supports only datasets for image classification, object detection, text classification, named entity recognition, text triplet, and speech paragraph labeling.

How to Enable Team Labeling


- When creating a dataset, enable **Team Labeling** and select a team or task manager.

Figure 2-76 Enabling during dataset creation

Team Labeling 

* Name

Type Team Task Manager

Select Team  Select at least one labeler

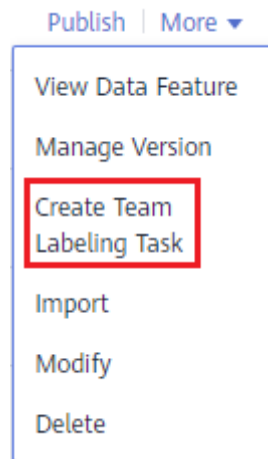
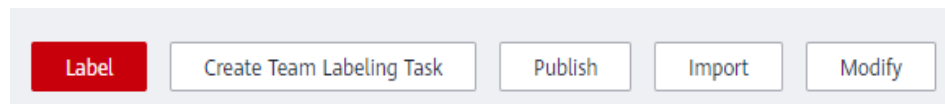
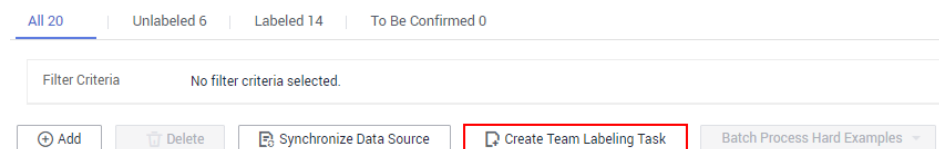
Immediately assign the unlabeled files in the dataset to a specified team member for labeling and reviewing. After receiving an email from the system, the team member labels and reviews the files as instructed.

<input type="checkbox"/>	Member Name	Role	Created
<input type="checkbox"/>	member02@huawei.com	Labeler	--
<input type="checkbox"/>	member01@huawei.com	Labeler	--

Automatically synchronize new files to the team labeling task.
New files in the dataset will be automatically synchronized to the labeling task that has been started.

Automatically load the intelligent labeling results to files that need to be labeled.
Files are automatically labeled. Labelers can then accept or modify the labels.

- If team labeling is not enabled for a dataset that has been created, create a team labeling task to enable team labeling. For details about how to create a team labeling task, see [Creating Team Labeling Tasks](#).

Figure 2-77 Creating a team labeling task in a dataset list**Figure 2-78** Creating a team labeling task**Figure 2-79** Creating a team labeling task on the dataset details page

NOTE

You can receive the email only when you create a team labeling task. No email will be sent when you create a labeling team or add members to a labeling team. Additionally, after all samples are labeled, no email will be sent when you create a team labeling task.

Operations Related to Team Labeling

- [Team Management](#)
- [Member Management](#)
- [Managing Team Labeling Tasks](#)

2.14.2 Team Management

Team labeling is managed in a unit of teams. To enable team labeling for a dataset, a team must be specified. Multiple members can be added to a team.

Background

- An account can have a maximum of 10 teams.
- An account must have at least one team to enable team labeling for datasets. If the account has no team, add a team by referring to [Adding a Team](#).

Adding a Team

1. In the left navigation pane of the ModelArts management console, choose **Data Management > Labeling Teams**. The **Labeling Teams** page is displayed.
2. On the **Labeling Teams** page, click **Add Team**.
3. In the displayed **Add Team** dialog box, enter a team name and description and click **OK**. The labeling team is added.

Figure 2-80 Adding a team

Add Team

* Name

Description

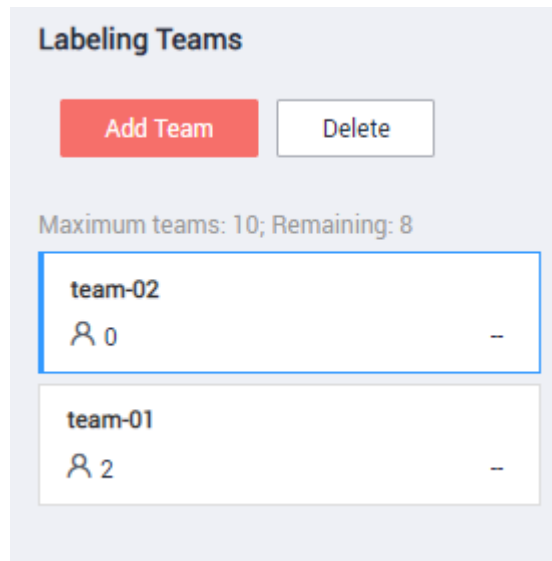
0/256

The new team is displayed on the **Labeling Teams** page. You can view team details in the right pane. There is no member in the new team. Add members to the new team by referring to [Adding a Member](#).

Deleting a Team

You can delete a team that is no longer used.

On the **Labeling Teams** page, select the target team and click **Delete**. In the dialog box that is displayed, click **OK**.

Figure 2-81 Deleting a team

2.14.3 Member Management

There is no member in a new team. You need to add members who will participate in a team labeling task.

A maximum of 100 members can be added to a team. If there are more than 100 members, add them to different teams for better management.

Adding a Member

1. In the left navigation pane of the ModelArts management console, choose **Data Management > Labeling Teams**. The **Labeling Teams** page is displayed.
2. On the **Labeling Teams** page, select a team from the team list on the left and click a team name. The team details are displayed in the right pane.
3. In the **Team Details** area, click **Add Member**.
4. In the displayed **Add Member** dialog box, enter an email address, description, and a role for a member and click **OK**.

An email address uniquely identifies a team member. Different members cannot use the same email address. The email address you enter will be recorded and saved in ModelArts. It is used only for ModelArts team labeling. After a member is deleted, the email address will also be deleted.

Possible values of **Role** are **Labeler**, **Reviewer**, and **Team Manager**. Only one **Team Manager** can be set.

Information about the added member is displayed in the **Team Details** area.

Modifying Member Information

You can modify member information if it is changed.

1. In the **Team Details** area, select the desired member.
2. In the row containing the desired member, click **Modify** in the **Operation** column. In the displayed dialog box, modify the description or role.

The email address of a member cannot be changed. To change the email address of a member, delete the member, and set a new email address when adding a member.

Possible values of **Role** are **Labeler**, **Reviewer**, and **Team Manager**. Only one **Team Manager** can be set.

Deleting Members

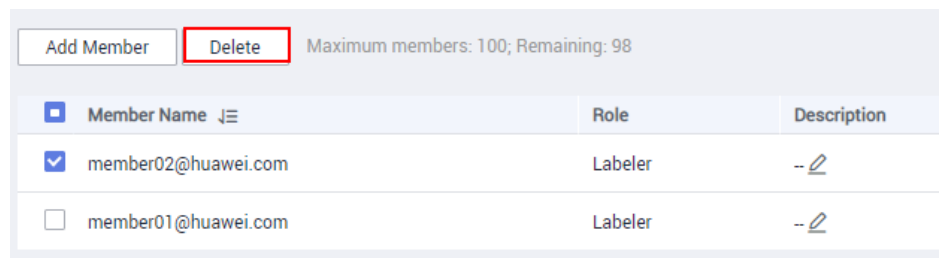
- **Deleting a single member**

In the **Team Details** area, select the desired member, and click **Delete** in the **Operation** column. In the dialog box that is displayed, click **OK**.

- **Batch Deletion**

In the **Team Details** area, select members to be deleted and click **Delete**. In the dialog box that is displayed, click **OK**.

Figure 2-82 Batch deletion



2.14.4 Managing Team Labeling Tasks

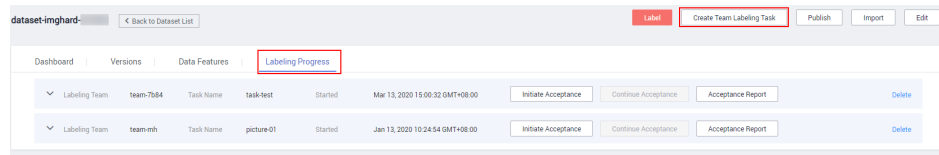
For datasets with team labeling enabled, you can create team labeling tasks and assign the labeling tasks to different teams so that team members can complete the labeling tasks together. During data labeling, members can initiate acceptance, continue acceptance, and view acceptance reports.

Creating Team Labeling Tasks

If you enable team labeling when creating a dataset and assign a team to label the dataset, the system creates a labeling task based on the team by default. After the dataset is created, you can view the labeling task on the **Labeling Progress** tab page of the dataset.

You can also create a team marking task and assign it to different members in the same team or to other labeling teams.

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management** > **Datasets**. A dataset list is displayed.
2. In the dataset list, select a dataset that supports team labeling, and click the dataset name to go to the **Dashboard** tab page of the dataset.
3. Click the **Labeling Progress** tab to view existing labeling tasks of the dataset. Click **Create Team Labeling Task** in the upper right corner to create a task.

Figure 2-83 Labeling tasks

4. In the displayed **Create Team Labeling Task** dialog box, set related parameters and click **OK**.
 - **Name:** Enter a task name.
 - **Type:** Select a task type, **Team** or **Task Manager**.
 - **Select Team:** If **Type** is set to **Team**, you need to select a team and members for labeling. The **Select Team** drop-down list lists the labeling teams and members created by the current account. For details about team management, see [Introduction to Team Labeling](#).
 - **Select Task Manager:** If **Type** is set to **Task Manager**, you need to select one **Team Manager** member from all teams as the task manager.
 - **Label Set:** All existing labels and label attributes of the dataset are displayed. You can also select **Automatically synchronize new images to the team labeling task** or **Automatically load the intelligent labeling results to images that need to be labeled** under **Label Set**.

NOTE

The process of loading auto labeling results to a team labeling task is as follows:

- If you set **Type** to **Team**, you are required to create a team labeling task before executing the task.
- If you set **Type** to **Task Manager**, select a team labeling job on the **My Participations** tab page and click **Assign Task**.

Figure 2-84 Creating a team labeling task**Create Team Labeling Task**

* Name

Type Team Task Manager

Select Team ! Select at least one labeler

Immediately assign the unlabeled files in the dataset to a specified team member for labeling and reviewing. After receiving an email from the system, the team member labels and reviews the files as instructed.

<input checked="" type="checkbox"/>	Member Name	Role	Created
<input checked="" type="checkbox"/>	member02@huawei.com	Labeler	--
<input checked="" type="checkbox"/>	member01@huawei.com	Labeler	--

Label Set

Label Name

Automatically synchronize new files to the team labeling task.
New files in the dataset will be automatically synchronized to the labeling task that has been started.

Automatically load the intelligent labeling results to files that need to be labeled

After the task is created, you can view the new task on the **Labeling Progress** tab page.

Labeling (Team Member)

After a labeling task is created, the team member to which the task is assigned receives a labeling notification email.

In the email, click the labeling job link to go to the data labeling job details page on the management console. Then, label the unlabeled data.

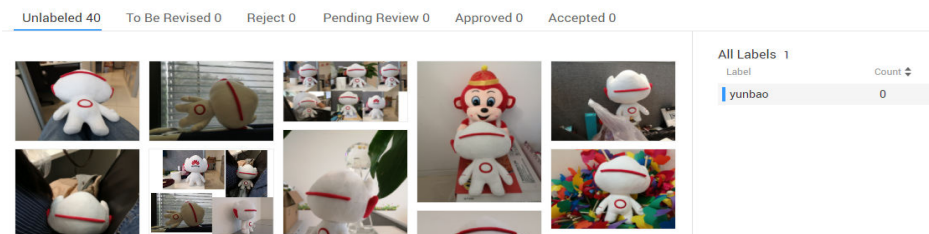
The labeling method varies depending on the dataset type. For details, see the following:

- [Image Classification](#)
- [Object Detection](#)
- [Text Classification](#)
- [Named Entity Recognition](#)
- [Text Triplet](#)

On the labeling page, each member can view the images that are not labeled, to be corrected, rejected, to be reviewed, approved, and accepted. Pay attention to the images rejected by the administrator and the images to be corrected.

If the Reviewer role is assigned for a team labeling task, the labeling result needs to be reviewed. After the labeling result is reviewed, it is submitted to the administrator for acceptance.

Figure 2-85 Labeling platform



Task Acceptance (Administrator)

- **Initiating acceptance**

After team members complete data labeling, the dataset creator can initiate acceptance to check labeling results. The acceptance can be initiated only when a labeling member has labeled data. Otherwise, the acceptance initiation button is unavailable.

- On the **Labeling Progress** tab page, click **Initiate Acceptance** to accept tasks.
- In the displayed dialog box, set **Sample Policy** to **By percentage** or **By quantity**. Click **OK** to start the acceptance.

By percentage: Sampling is performed based on a percentage for acceptance.

By quantity: Sampling is performed based on quantity for acceptance.

Figure 2-86 Initiating acceptance

Initiate Acceptance

Samples for Acceptance 0

Sampling Policy

By percentage

%

By quantity

OK

Cancel

- After the acceptance is initiated, an acceptance report is displayed on the console in real time. In the **Acceptance Result** area on the right, select **Pass** or **Reject**.

If you select **Pass**, set **Rating** to **A**, **B**, **C**, or **D**. Option **A** indicates the highest score. See [Figure 2-88](#). If you select **Reject**, enter your rejection reasons in the text box. See [Figure 2-89](#).

Figure 2-87 Viewing a real-time acceptance report

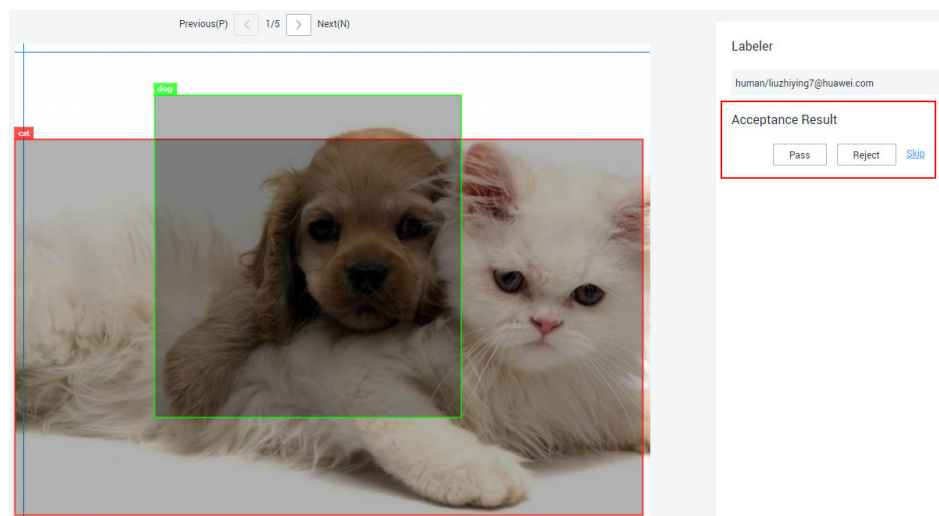


Figure 2-88 Pass

Acceptance Result

Rating: A B C D

[Skip](#)

Figure 2-89 Reject

Acceptance Result

8/256

[Skip](#)

- **Continuing acceptance**

You can continue accepting tasks whose acceptance is not completed. For tasks for which an acceptance process is not initiated, the **Continue Acceptance** button is unavailable.

On the **Labeling Progress** tab page, click **Continue Acceptance** to continue accepting tasks. The **Real-Time Acceptance Report** page is displayed. You can continue to accept the images that are not accepted.

- **Finishing acceptance**

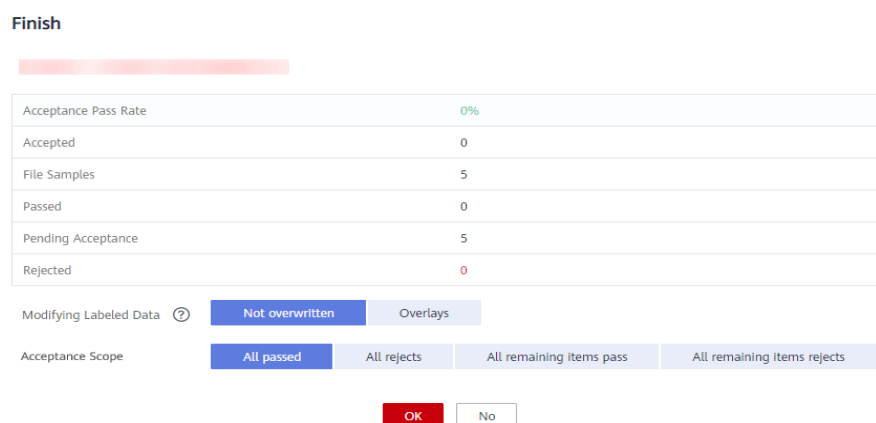
In the acceptance completion window, you can view dataset acceptance details, such as the number of sample files, set the following parameters, and perform acceptance. The labeling information is synchronized to the **Labeled** tab page of the dataset only after the acceptance is complete.

Once the labeled data is accepted, team members cannot modify the labeling information. Only the dataset creator can modify the labeling information.

Table 2-32 Parameters for finishing acceptance

Parameter	Description
Modifying Labeled Data	<ul style="list-style-type: none"> ● Not overwrite: For the same data, do not overwrite the existing data with the labeling result of the current team. ● Overlays: For the same data, overwrite the existing data with the labeling result of the current team. Overwritten data cannot be recovered. Exercise caution when performing this operation.
Acceptance Scope	<ul style="list-style-type: none"> ● All: all data that has been labeled by the current team, including Accepted, Pending Acceptance, and Rejected data. It refers to all sample files in the dataset. ● All rejects: rejects all data that has been labeled by the current team. That is, all labeled data is rejected to the labeling personnel. ● Accepted and pending acceptance: accepts the data that passes the acceptance or is in the Pending Acceptance state in the sample files and rejects the data that fails the acceptance to the labeling personnel. ● Accepted: accepts the data that has passed the acceptance in the sample files and rejects the data that is in the Pending Acceptance state or fails the acceptance to the labeling personnel.

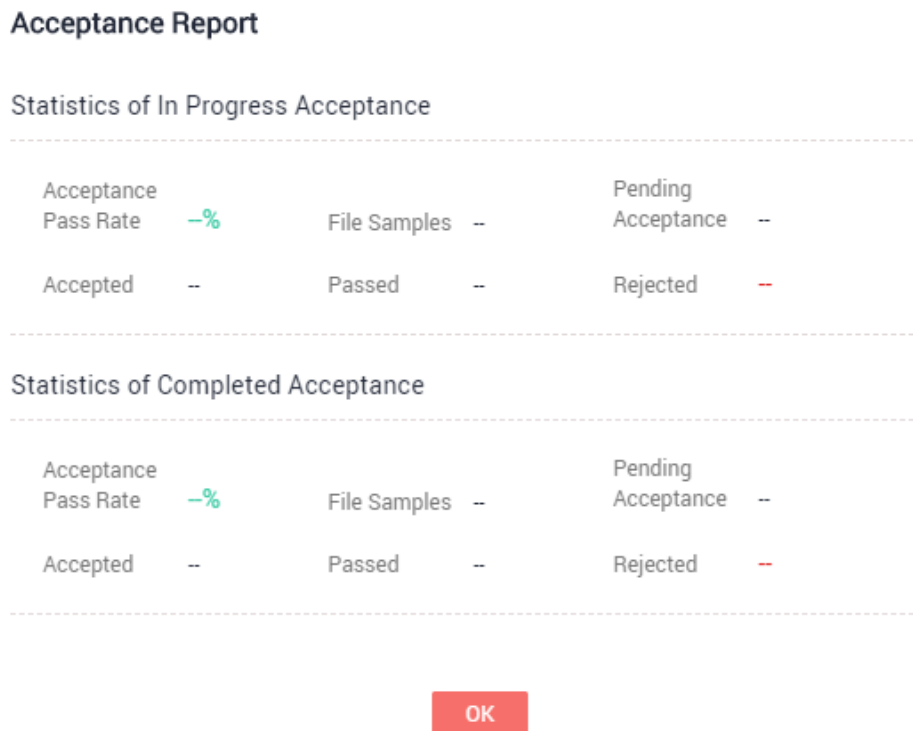
Figure 2-90 Finishing acceptance



Viewing an Acceptance Report

You can view the acceptance report of an ongoing or finished labeling task. On the **Labeling Progress** tab page, click **Acceptance Report**. In the displayed **Acceptance Report** dialog box, view report details.

Figure 2-91 Viewing an acceptance report



Deleting a Labeling Task

On the **Labeling Progress** tab page, click **Delete** in the row where a labeling task to be deleted. After a task is deleted, the labeling details that are not accepted will be lost. Exercise caution when performing this operation. However, the original data in the dataset and the labeled data that has been accepted are still stored in the corresponding OBS bucket.

2.15 Data Processing

2.15.1 Introduction to Data Processing

ModelArts provides the data processing function to extract valuable and meaningful data from a large amount of disordered and difficult-to-understand data. After data is collected and accessed, the data cannot directly meet the training requirements. To ensure data quality and avoid negative impact on subsequent operations (such as data labeling and model training), the data needs to be processed. Common data processing types are as follows:

- **Data validation:** Generally, data needs to be verified after being collected to ensure data validity.

Data validation is a process of determining and verifying data availability. Generally, the collected data cannot be further processed due to some format problems. Take image recognition as an example. Users often find some images from the Internet for training, and the image quality cannot be ensured. The name, path, and extension of the images may not meet the requirements of the training algorithm. Images may also be partially damaged. As a result, the images cannot be decoded or processed by the algorithm. Therefore, data validation is very important. It can help AI developers detect data problems in advance and effectively prevent algorithm precision deterioration or training failures caused by noisy data.

- **Data cleansing:** Data cleansing refers to the process of removing, correcting, or supplementing data.

Data cleansing is to check data consistency based on data validation and correct some invalid values. For example, in the deep learning field, data may be cleansed based on a positive sample and a negative sample that are input by a user, to retain a category that the user wants and remove a category that the user does not want.

- **Data selection:** Data selection refers to the process of selecting data subsets from full data.

Data can be selected based on the similarity or deep learning algorithm. Data selection can avoid problems such as repeated images and similar images introduced during manual image collection. Among a batch of inference data input to an old model, data selection using built-in rules can further improve the precision of the old model.

- Data augmentation:

Data amplification: Data amplification increases data volumes directly or indirectly through simple data amplification operations such as scaling, cropping, transformation, and composition.

Image generation: Image generation increases data volumes by applying deep learning models, learning the original dataset, and generating a new dataset.

2.15.2 Creating a Data Processing Task

You can create a data processing task to verify, cleanse, select, or augment existing data.

Prerequisites

- Data has been prepared. Specifically, a dataset has been created or data has been uploaded to OBS.
- The OBS directory you use and ModelArts are in the same region.

Procedure

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management > Process Data**. The **Process Data** page is displayed.
2. On the **Process Data** page, click **Create**.
3. On the page for creating a data processing task, set required algorithm parameters.

- a. Set the basic information, including **Name**, **Version**, and **Description**. For **Version**, the system automatically creates a version number, which is named according to a certain rule, for example, **V0001** and **V0002**. The version number cannot be changed.

Specify **Name** and **Description** according to actual requirements.

Figure 2-92 Basic information for creating a data processing task

★ Name

Version V0001 (System-defined version number)

Description 0/256

- b. Set the scenario type. You can set the scenario type to image classification or object detection.
- c. Set the data processing type. Data processing types include data cleaning, data validation, data selection, and data augmentation.

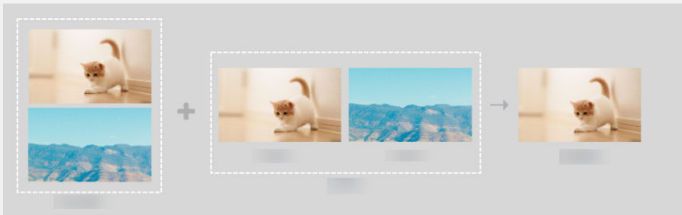
Set the operator parameters based on the data processing type. For details about the operator parameters, see [Built-in Operators](#).

Figure 2-93 Setting the scenario type and data processing type

★ Scenario Image Classification Object detection

★ Data Processing Type

Algorithms



Parameter =

The positive sample directory for data cleaning. The directory should store picture files. The algorithm will filter the data in the algorithm input according to these pictures as positive examples, that is, keep the data with high similarity to the prototype_sample_path directory

Advanced

- d. Set the input and output. Select **Datasets** or **OBSCatalog** based on the site requirements. If you select **Datasets**, you need to enter the dataset name and dataset version. If you select **OBSCatalog**, you need to enter the correct OBS path.

Figure 2-94 Input and output settings - Datasets

* Input

Datasets OBSCatalog

Select a dataset. Select a dataset version.

* Output ?

Datasets OBSCatalog

Select a dataset. Enter a dataset version.

Figure 2-95 Input and output settings - OBSCatalog

* Input

Datasets **OBSCatalog**

Select an OBS path.

Storage Structure Images and labels ? Only images ?

input-example

- Cat
 - IMG_0821.jpg
 - IMG_0826.jpg
- Dog
 - IMG_0837.jpg
 - IMG_0855.jpg

input-example

- IMG_0821.jpg
- IMG_0823.jpg
- IMG_0838.jpg
- IMG_0845.jpg

* Output ?

Datasets **OBSCatalog**

Select an OBS path.

- e. After confirming that the parameter settings are correct, click **Next**.

2.15.3 Managing and Viewing Data Processing Tasks

Deleting a Data Processing Task

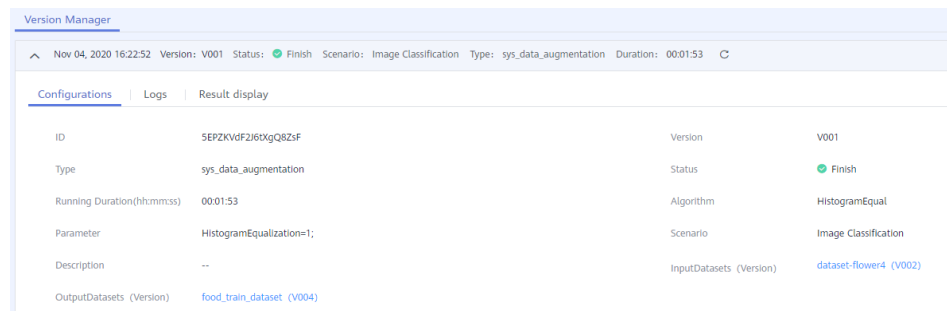
If an existing data processing task is no longer used, you can delete it.

For a training job in the **Completed**, **Failed**, **Stopped**, **Running Failed**, or **Deploying** state, you can click **Delete** in the **Operation** column to delete the corresponding data processing task.

Viewing Data Processing Task Details

1. Log in to the ModelArts management console. In the left navigation pane, choose **Data Management** > **Process Data**. The **Process Data** page is displayed.
2. In the data processing task list, click the data processing task name. The version management page of the data processing task is displayed. You can modify and delete the data processing task on this page.

Figure 2-96 Data processing version management



3. Switch between tab pages on the version management page to view the configuration information, logs, and results.

Figure 2-97 Logs page

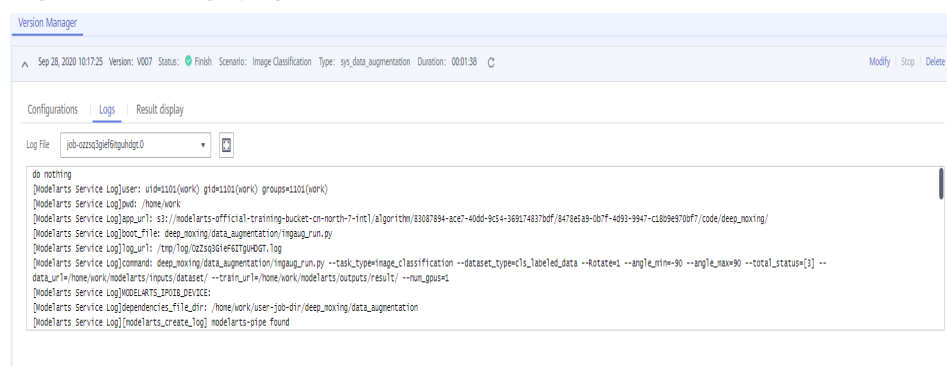
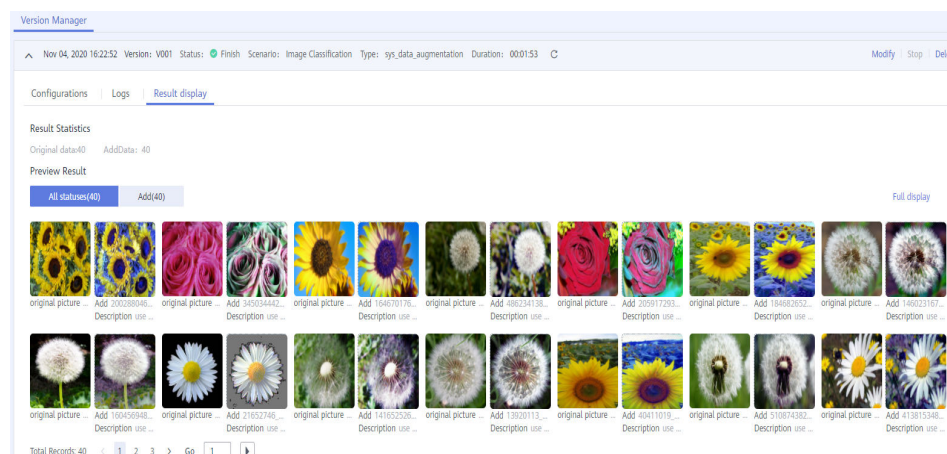


Figure 2-98 Result display page



2.15.4 Built-in Operators

2.15.4.1 Data Validation

MetaValidation Operator Overview

ModelArts data validation is implemented by the MetaValidation operator. ModelArts supports the following image formats: JPG, JPEG, BMP, and PNG. The

object detection scenario supports the XML labeling format but does not support the non-rectangular box labeling format. The MetaValidation operator supports data validation for images and XML files based on the dataset you provide.

Table 2-33 Image data validation

Exception	Solution
The images are damaged and cannot be decoded.	Filters out images that cannot be decoded.
The image channel can be channel 1 or channel 2. Channel 3 is not commonly used.	Converts images into RGB three-channel images.
The image format is not supported by ModelArts.	Converts the image format to JPG.
The image suffix is inconsistent with the actual format, but the format is supported by ModelArts.	Converts the suffix to the actual format.
The image suffix is inconsistent with the actual format and the format is not supported by ModelArts.	Converts the image format to JPG.
The image resolution is too high.	The image width and height are cropped based on the specified size and ratio.

Table 2-34 Labeling file data validation

Exception	Solution
The XML structure is incomplete and cannot be parsed.	Filters XML files.
No labeled object is in the XML file.	Filters XML files.
The XML file does not contain rectangle bndbox .	Filters XML files.
Some labeled objects do not have rectangle bndbox .	Filters labeled objects.
After an image is cropped, the width and height of the image are inconsistent with those in the XML file.	Changes the values of the width and height parameters to the actual width and height of the image.
No width and height fields exist in XML files.	Supplements the width and height fields and values in the XML file based on the actual width and height of the image.

Exception	Solution
After an image is cropped, its size is inconsistent with the size of rectangle bndbox in the XML file.	Change the value of bndbox in the XML file based on the image cropping ratio.
The width or height of rectangle bndbox in the XML file is too small and is displayed as a line.	If the difference between the width and height of the rectangle is less than 2, remove the current object.
In the XML file, the minimum value of rectangle bndbox is greater than the maximum value.	Removes the current object.
Rectangle bndbox exceeds the image boundaries, and the excess part occupies more than 50% of the frame area.	Removes the current object.
Rectangle bndbox exceeds the image boundaries, and the excess part occupies less than 50% of the frame area.	Rectangle bndbox is pulled back to the image boundaries.

NOTE

Original data is not changed during data validation. The newly validated image or XML file is saved in the specified output path.

Parameters

Table 2-35 Parameters of the data validation-MetaValidation operator

Parameter	Mandatory	Default Value	Description
image_max_width	No	-1	Maximum width of an input image. If the width of an input image exceeds the configured value, the image is cropped based on the ratio. The unit is pixel. The default value -1 indicates that the image is not cropped.
image_max_height	No	-1	Maximum length of an input image. If the length of an input image exceeds the configured value, the image is cropped based on the ratio. The unit is pixel. The default value -1 indicates that the image is not cropped.

Operator Input Requirements

The following two types of operator input are available:

- **Datasets:** Select a dataset and its version created on the ModelArts console from the drop-down list. The dataset type must be the same as the scenario type selected in this task.
- **OBSCatalog:** Select either of the following storage structures:
 - **Only images:** If the directory contains only images, the JPG, JPEG, PNG, and BMP formats are supported, and all images in the nested subdirectories are read.
 - **Images and labels:** The structure varies depending on the scenario type.

The following shows the directory structure in the image classification scenario. The following directory structure supports only single-label scenarios.

```
input_path/
--label1/
----1.jpg
--label2/
----2.jpg
--./
```

The following shows the directory structure in the object detection scenario. Images in JPG, JPEG, PNG, and BMP formats are supported. XML files are standard PACAL VOC files.

```
input_path/
--1.jpg
--1.xml
--2.jpg
--2.xml
...
```

Output Description

- **Image classification**

The output directory structure is as follows:

```
output_path/
--Data/
----class1/ # If the input data has labeling information, the information is also output. class1
indicates the labeling class.
-----1.jpg
-----2_checked.jpg
----class2/
-----3.jpg
-----4_checked.jpg
----5_checked.jpg
--output.manifest
```

The following shows a manifest file example. The verification attribute "**property":{"@modelarts:data_checked":true}**" is added for each data record.

```
{
  "id": "xss",
  "source": "obs://hard_example_path/Data/fc8e2688015d4a1784dcbda44d840307_14_checked.jpg",
  "property": {
    "@modelarts:data_checked": true
  },
  "usage": "train",
  "annotation": [
    {
```

```

    "name": "Cat",
    "type": "modelarts/image_classification"
  }
]
}

```

- **Object detection**

The output directory structure is as follows:

```

output_path/
--Data/
  ----1_checked.jpg
  ----1_checked.xml # If the input data is converted during verification, '_checked' is added to the
file name.
  ----2.jpg      # If the input data is not converted, the file is saved with the original name.
  ----2.xml
--output.manifest

```

The following shows a manifest file example. The verification attribute **"property":{"@modelarts:data_checked":true}** is added for each data record.

```

{
"source": "obs://hard_example_path/Data/be462ea9c5abc09f_checked.jpg",
"property": {
  "@modelarts:data_checked": true
},
"annotation": [
  {
    "annotation-loc": "obs://hard_example_path/Data/be462ea9c5abc09f_checked.xml",
    "type": "modelarts/object_detection",
    "annotation-format": "PASCAL VOC",
    "annotated-by": "modelarts/hard_example_algo"
  }
]
}
}

```

2.15.4.2 Data Cleansing

PCC Operator Overview

ModelArts data cleansing is implemented by the PCC operator. The dataset used for image classification or object detection may contain images that do not belong to the required categories. These images need to be removed to avoid interference to labeling and model training.

Figure 2-99 PCC operator effect



Parameters

Table 2-36 Parameters of the data cleansing-PCC operator

Parameter	Mandatory	Default Value	Description
prototype_sample_path	Yes	None	<p>Directory for storing positive data cleansing samples. The directory stores positive sample image files. The algorithm filters input data based on the positive sample images. That is, the data that is highly similar to the images in the prototype_sample_path directory is retained.</p> <p>Enter an existing OBS directory. The directory contains the provided positive sample images and starts with obs://, for example, <i>obs://obs_bucket_name/folder_name</i>.</p>
criticism_sample_path	No	None	<p>Directory for storing negative data cleansing samples. The directory stores negative sample image files. The algorithm filters input data based on the negative sample images. That is, the data that is less similar to the images in the criticism_sample_path directory is retained.</p> <p>It is recommended that this parameter be used together with prototype_sample_path to improve the accuracy of data cleansing.</p> <p>Enter an existing OBS directory that starts with obs://, for example, <i>obs://obs_bucket_name/folder_name</i>.</p>
n_clusters	No	auto	<p>Number of data sample types. The default value is auto. You can enter an integer less than the total number of samples or auto. auto indicates that the number of images in the positive sample directory is used as the number of data sample types.</p>
similarity_threshold	No	0.9	<p>Similarity threshold. If the similarity between two images exceeds the threshold, the two images are regarded as similar. Otherwise, they are regarded as dissimilar. The value ranges from 0 to 1.</p>
embedding_distance	No	0.2	<p>Distance between sample features. If the feature distance between two images is smaller than the specified value, the two images are regarded as similar. Otherwise, the two images are regarded as dissimilar. The value ranges from 0 to 1.</p>

Parameter	Mandatory	Default Value	Description
do_validation	No	True	Indicates whether to validate data. The value can be True or False . True indicates that data is validated before cleansing. False indicates that data is cleansed only.

Operator Input Requirements

The following two types of operator input are available:

- **Datasets:** Select a dataset and its version created on the ModelArts console from the drop-down list. Ensure that the dataset type be the same as the scenario type selected in this task.
- **OBSCatalog:** Select either of the following storage structures:
 - **Only images:** If the directory contains only images, the JPG, JPEG, PNG, and BMP formats are supported, and all images in the nested subdirectories are read.
 - **Images and labels:** The structure varies depending on the scenario type.

The following shows the directory structure in the image classification scenario. The following directory structure supports only single-label scenarios.

```
input_path/
--label1/
----1.jpg
--label2/
----2.jpg
--./
```

The following shows the directory structure in the object detection scenario. Images in JPG, JPEG, PNG, and BMP formats are supported. XML files are standard PACAL VOC files.

```
input_path/
--1.jpg
--1.xml
--2.jpg
--2.xml
...
```

Output Description

- **Image classification**

The output directory structure is as follows:

```
output_path/
--Data/
----class1/ # If the input data has labeling information, the information is also output. class1
indicates the labeling class.
-----1.jpg
----class2/
-----2.jpg
----3.jpg
--output.manifest
```

The following shows a manifest file example.

```
{
  "id": "xss",
  "source": "obs://home/fc8e2688015d4a1784dcba44d840307_14.jpg",
  "usage": "train",
  "annotation": [
    {
      "name": "Cat",
      "type": "modelarts/image_classification"
    }
  ]
}
```

- **Object detection**

The output directory structure is as follows:

```
output_path/
--Data/
  ----1.jpg
  ----1.xml # If the input data has labeling information, the information is also output. xml
            indicates the label file.
  ----2.jpg
  ----3.jpg
--output.manifest
```

The following shows a manifest file example.

```
{
  "source":"obs://fake/be462ea9c5abc09f.jpg",
  "annotation":[
    {
      "annotation-loc":"obs://fake/be462ea9c5abc09f.xml",
      "type":"modelarts/object_detection",
      "annotation-format":"PASCAL VOC",
      "annotated-by":"modelarts/hard_example_algo"
    }
  ]
}
```

2.15.4.3 Data Selection

Overview of Data Selection Operators

ModelArts provides the following data selection operators:

- The SimDeduplication operator can implement image deduplication based on the similarity threshold you set. Image deduplication is a common method for image data processing. Image duplication means that the image content is completely the same, or the scale, displacement, color, or brightness changes slightly, or a small amount of other content is added.

Table 2-37 Advanced parameters

Parameter	Mandatory	Default Value	Description
similarity_threshold	No	0.9	Similarity threshold. When the similarity between two images is greater than the threshold, one of the images is filtered out as a duplicate image. The value ranges from 0 to 1.

Parameter	Mandatory	Default Value	Description
do_validation	No	True	Indicates whether to validate data. The value can be True or False . True indicates that data is validated before deduplication. False indicates that data is deduplicated only.

- **RRD**: The data with the largest difference can be removed based on the preset proportion.

Table 2-38 Advanced parameters

Parameter	Mandatory	Default Value	Description
sample_ratio	No	0.9	Percentage of reserved data. The value ranges from 0 to 1. For example, 0.9 indicates that 90% of the original data is reserved.
n_clusters	auto	auto	Number of data sample types. The default value is auto , indicating that the total number of types is obtained based on the number of images in the directory. For example, you can specify the number of types to 4 .
do_validation	No	True	Indicates whether to validate data. The value can be True or False . True indicates that data is validated before deredundancy. False indicates that data is deduplicated only.

Operator Input Requirements

The following two types of operator input are available:

- **Datasets**: Select a dataset and its version created on the ModelArts console from the drop-down list. Ensure that the dataset type be the same as the scenario type selected in this task.
- **OBSCatalog**: Select either of the following storage structures:
 - **Only images**: If the directory contains only images, the JPG, JPEG, PNG, and BMP formats are supported, and all images in the nested subdirectories are read.
 - **Images and labels**: The structure varies depending on the scenario type. The following shows the directory structure in the image classification scenario. The following directory structure supports only single-label scenarios.

```
input_path/
--label1/
----1.jpg
--label2/
----2.jpg
--./
```

The following shows the directory structure in the object detection scenario. Images in JPG, JPEG, PNG, and BMP formats are supported. XML files are standard PACAL VOC files.

```
input_path/
--1.jpg
--1.xml
--2.jpg
--2.xml
...
```

Output Description

- **Image classification**

The output directory structure is as follows:

```
output_path/
--Data/
----class1/ # If the input data has labeling information, the information is also output. class1
indicates the labeling class.
-----1.jpg
----class2/
-----2.jpg
-----3.jpg
--output.manifest
```

The following shows a manifest file example.

```
{
  "id": "xss",
  "source": "obs://home/fc8e2688015d4a1784dcba44d840307_14.jpg",
  "usage": "train",
  "annotation": [
    {
      "name": "Cat",
      "type": "modelarts/image_classification"
    }
  ]
}
```

- **Object detection**

The output directory structure is as follows:

```
output_path/
--Data/
----1.jpg
----1.xml # If the input data has labeling information, the information is also output. xml
indicates the label file.
----2.jpg
----3.jpg
--output.manifest
```

The following shows a manifest file example.

```
{
  "source":"obs://fake/be462ea9c5abc09f.jpg",
  "annotation":[
    {
      "annotation-loc":"obs://fake/be462ea9c5abc09f.xml",
      "type":"modelarts/object_detection",
      "annotation-format":"PASCAL VOC",
      "annotated-by":"modelarts/hard_example_algo"
    }
  ]
}
```

2.15.4.4 Data Selection (Hard Examples)

Algorithm Overview

In actual service scenarios, model maintenance is a long-term process. For example, data retraining is performed weekly or monthly, or periodic retraining is required when data accumulates to a certain volume. Full-data retraining often consumes a large amount of labeling manpower and training time. To improve the model maintenance efficiency, hard examples-based retraining can be adopted.

A hard example filtering algorithm is used to analyze and filter full data and output only a small amount of valuable data for model maintenance. In this case, retraining with the filtered data effectively reduces the labeling manpower and training time.

Multiple methods are integrated in the hard example filtering algorithm. To achieve the optimal effect, you need to select some or all methods and adjust their weights based on your needs.

Parameters

Parameter	Man dato ry	Default Value	Description
source_service	Y	inference	Preset data source of a hard example filtering task. Only inference is supported. The parameter value cannot be changed.
filter_func	Y	comprehensive_mining	Set the hard example filtering algorithm to comprehensive_mining . The parameter value cannot be changed.
checkpoint_path	Y	/home/work/ user-job-dir/ data_filter/ resnet_v1_50	Model directory used for feature extraction. Only pre-trained resnet_v1_50 model based on ImageNet is supported. The parameter value cannot be changed.
model_serving_url	N	None	Inference model path, that is, the output path of a training job. This model is used for inference after data augmentation in the aug_consistent_mining algorithm. Enter an existing OBS directory, for example, obs://obs_bucket_name/folder_name/ .

Parameter	Mandatory	Default Value	Description
train_data_path	N	None	<p>Training dataset, which is the training data used by the model_serving_url model. The manifest file generated by the dataset version needs to be entered.</p> <p>Enter an existing OBS directory, for example, obs://obs_bucket_name/folder_name/v001.manifest.</p>
comprehensive_algo_config	N	clustering_mining:0.2020+aug_consistent_mining:0.4265+feature_distribution_mining:0.0451+sequential_mining:0.425+image_similarity_mining:0.0949+predict_score_mining:0.3900+anomaly_detection_mining:0.2020	<p>Algorithm and its weight. By default, the optimal parameter after the system experiment is used. You can also configure this parameter with different data.</p> <p>Example: predict_score_mining:0.3900+anomaly_detection_mining:0.2020</p>
algo_hard_threshold	N	0.1	<p>Threshold of the filtering coefficient. The value ranges from 0 to 1.</p> <p>If the threshold is set too high, the output result may be 0. Set this parameter to a proper value.</p>
aug_op_config	N	crop:0.1+fliplr:0.1+gaussianblur:0.1	<p>Data augmentation method used in the aug_consistent_mining algorithm. The value can be crop, fliplr, gaussianblur, flipud, scale, translate, shear, superpixels, sharpen, add, or invert.</p>

Parameter	Mandatory	Default Value	Description
feature_op_config	N	image_aspect_ratio:0.5+image_brightness:1.0+image_saturation:0.5+image_resolution:0.5+image_colorfulness:0.5+ambiguity:1.0+bbbox_num:1.0+bbbox_iou:1.0+bbbox_std:0.5+bbbox_bright:0.5+bbbox_ambiguity:0.5+bbbox_aspect_ratio:1.0+bbbox_area_ratio:0.5+bbbox_edge_value:0.5	Feature defined in the feature_distribution_mining algorithm. The weight can be modified.
score_threshold_up	N	0.6	Maximum confidence value defined in the predict_score_mining algorithm. The value ranges from 0 to 1.
score_threshold_low	N	0.3	Minimum confidence value defined in the predict_score_mining algorithm. The value ranges from 0 to 1.
margin	N	0.8	Top 2 confidence difference. If the difference is greater than this parameter value, this sample is a hard example. The value ranges from 0 to 1. The default value is 0.8 .
similarity_sample_ratio	N	1.0	Similarity ratio in the image_similarity_mining algorithm. The value ranges from 0 to 1. The default value is 1.0 .
task_summary_file	N	None	Log output path and log file of the algorithm simplification. Enter an existing OBS directory that starts with obs:// . The file name can be custom. Example: obs://obs_bucket_name/folder_name/xxx.log

Parameter	Mandatory	Default Value	Description
output_data_set_type	N	manifest	<p>The options are as follows:</p> <ul style="list-style-type: none"> • directory: outputs the raw image and label to the Data folder in the result directory. • manifest: only outputs the manifest file. <p>This parameter is automatically filled on the data processing page based on your configurations.</p>

Operator Input Requirements

The following two types of operator input are available:

- **Datasets**: Select a dataset and its version created on the ModelArts console from the drop-down list. Ensure that the dataset type be the same as the scenario type selected in this task.
- **OBSCatalog**: The directory must contain the raw image for inference and the inference result file.

The directory structure is as follows:

```
input_path/
--images/ #The folder name must be images.
  ----1.jpg
  ----2.jpg
--inference_results/ # The folder name must be inference_results.
  ----1.jpg_result.txt
  ----2.jpg_result.txt
```

The .txt inference result file must meet the following requirements: If you use the model trained by the ModelArts built-in algorithm for inference, the default inference result meets the requirements.

– Image classification

```
{
  "predicted_label": "dog",
  "scores": [
    [
      "dog",
      "0.589"
    ],
    [
      "cat",
      "0.411"
    ]
  ]
}
```

– Object detection

```
{
  "detection_classes": [
    "cat",
    "cat"
  ],
  "detection_boxes": [
```

```

    [
      117.56356048583984,
      335.9902648925781,
      270.50848388671875,
      469.0136413574219
    ],
    [
      18.747316360473633,
      13.10757064819336,
      217.25146484375,
      108.3551025390625
    ]
  ],
  "detection_scores": [
    0.5179755091667175,
    0.46941104531288147
  ]
}

```

Output Description

- Object detection

The output directory structure is as follows:

```

output_path :
  --Data
    ----1.jpg
    ----1.xml # Export the filtering result to this directory.
  --output.manifest

```

A manifest file example is as follows:

```

{"source":"/tmp/test_out/object_detection/images/be462ea9c5abc09f.jpg",
"hard":"True",
"hard-reasons":"0", # Reason why the sample is determined as a hard example. The specific reason is
displayed only in the auto labeling module.
"hard-coefficient":"1.0", # Hard example coefficient obtained using the hard example algorithm. A
larger value indicates a higher probability that the sample is a hard example.
"annotation":[
{"annotation-loc":"/tmp/test_out/object_detection/annotations/be462ea9c5abc09f.xml",
"type":"modelarts/object_detection",
"annotation-format":"PASCAL VOC",
"annotated-by":"modelarts/hard_example_algo"}}]

```

- Image classification

The output directory structure is as follows:

```

output_path :
  --Data
    ----class1
      -----1.jpg
    ----class2
      -----2.jpg
  --output.manifest

```

A manifest file example is as follows:

```

{"source":"/obs://obs_bucket_name/folder_name/catDog/5.jpg",
"hard":true,
"hard-reasons":"1-20-2-19-21-3",
"hard-coefficient":1.0,
"annotation":[
{"name":"cat",
"type":"modelarts/image_classification",
"confidence":0.599,
"annotated-by":"modelarts/hard_example_algo"}}]

```

Log File Description

task_summary_file is the output file path of the simplified log. The content is as follows:

```
{  
  "task_status": 'SUCCEED', # Algorithm execution status  
  "total_sample": integer, # Total input samples  
  "hard_sample": integer # Total output samples  
}
```

or

```
{  
  "task_status": 'FAILED',  
  "error_message": 'xxxxxx' # Error information that causes the algorithm execution failure  
}
```

2.15.4.5 Data Augmentation (Data Amplification)

Overview of Data Amplification Operators

Data amplification is mainly used in scenarios where the training dataset is insufficient or simulation is required. You can transform the labeled dataset to increase the number of images trained and generate corresponding labels. In the deep learning field, augmentation is of great significance. It can improve model generalization and enhance anti-disturbance. Original data is not changed during data amplification. The newly amplified image or XML file is saved in the specified output path.

ModelArts provides the following data amplification operators:

Table 2-39 Description of data amplification operators

Operator	Description	Advanced
AddNoise	Adds noises to simulate the noises that may be generated when common capture devices capture images.	<ul style="list-style-type: none"> • noise_type: noise distribution type. Gauss indicates Gaussian noise. Laplace indicates Laplace noise. Poisson indicates Poisson noise. Impulse indicates impulse noise. SaltAndPepper indicates salt and pepper noise. The default value is Gauss. • loc: average noise distribution. This parameter is valid only in Gauss and Laplace. The default value is 0. • scale: standard deviation of noise distribution. This parameter is valid only in Gauss and Laplace. The default value is 1. • lam: lambda coefficient of Poisson distribution. This parameter is valid only for Poisson. The default value is 2. • p: probability of pulse noise or salt-and-pepper noise for each pixel. This parameter is valid only for Impulse and SaltAndPepper. The default value is 0.01. • do_validation: indicates whether to validate data before data amplification. The default value is True.
Blur	Uses filters to filter images and sometimes to simulate imaging of imaging devices.	<ul style="list-style-type: none"> • blur_type: The value can be Gauss or Average, which indicates Gaussian filtering and average filtering, respectively. The default value is Gauss. • do_validation: indicates whether to validate data before data amplification. The default value is True.

Operator	Description	Advanced
Crop	Randomly crops a part of an image as a new image.	<ul style="list-style-type: none"> • crop_percent_min: minimum value in the value range of the cropping ratio of each edge. The default value is 0.0. • crop_percent_max: maximum value in the value range of the cropping ratio of each edge. The default value is 0.2. • do_validation: indicates whether to validate data before data amplification. The default value is True.
CutOut	Random erase, which is a common method used in deep learning to simulate an object that is blocked by an obstacle.	<ul style="list-style-type: none"> • do_validation: indicates whether to validate data before data amplification. The default value is True.
Flip	Flips along the horizontal or vertical axis of an image, which is a very common enhancement method.	<ul style="list-style-type: none"> • lr_ud: flipping direction. lr indicates horizontal flipping, and ud indicates vertical flipping. The default value is lr. • flip_p: flipping probability. The default value is 1. • do_validation: indicates whether to validate data before data amplification. The default value is True.
Grayscale	Changes a three-channel color image to a three-channel grayscale image.	<ul style="list-style-type: none"> • do_validation: indicates whether to validate data before data amplification. The default value is True.
HistogramEqual	Indicates the histogram equalization, which is mainly used to improve the visual effect of images. It is used in some scenarios.	<ul style="list-style-type: none"> • do_validation: indicates whether to validate data before data amplification. The default value is True.
LightArithmetic	Implements linear enhancement on luminance space.	<ul style="list-style-type: none"> • do_validation: indicates whether to validate data before data amplification. The default value is True.

Operator	Description	Advanced
LightContrast	Enhances luminance contrast. A certain non-linear function is used to change the luminance value of the luminance space.	<p>func: The default value is gamma.</p> <ul style="list-style-type: none"> • gamma: Gamma correction. Its formula is $255*((v/255)**gamma)'$. • sigmoid: S-shaped curve function. Its formula is $255*1/(1+exp(gain*(cutoff-l_{ij}/255)))'$. • log: logarithmic function. Its formula is $255*gain*log_2(1+v/255)$. • linear: linear function. Its formula is $127 + alpha*(v-127)'$. <p>do_validation: indicates whether to validate data before data amplification. The default value is True.</p>
MotionBlur	Indicates the motion blur generated when an object moves.	<p>do_validation: indicates whether to validate data before data amplification. The default value is True.</p>
Padding	Pads an image with black edges.	<ul style="list-style-type: none"> • px_top: number of pixel lines added at the top of the image. The default value is 1. • px_right: number of pixel lines added at the right of the image. The default value is 1. • px_left: number of pixel lines added at the left of the image. The default value is 1. • px_bottom: number of pixel lines added at the bottom of the image. The default value is 1. • do_validation: indicates whether to validate data before data amplification. The default value is True.
Resize	Resizes an image.	<ul style="list-style-type: none"> • height: height of the image after conversion. The default value is 224. • width: width of the image after conversion. The default value is 224. • do_validation: indicates whether to validate data before data amplification. The default value is True.

Operator	Description	Advanced
Rotate	Rotates an image around the center point. After the operation is complete, the original shape of the image remains unchanged, and the blank part is filled with black.	<ul style="list-style-type: none"> ● angle_min: minimum value in the range of rotation angles. Each image randomly obtains a value from the range. The default value is 90°. ● angle_max: maximum value in the range of rotation angles. Each image randomly obtains a value from the range. The default value is -90°. ● do_validation: indicates whether to validate data before data amplification. The default value is True.
Saturation	Enhances chrominance and saturation. The H and S spaces in the HSV of an image are changed linearly to change the chrominance and saturation of the image.	<ul style="list-style-type: none"> ● do_validation: indicates whether to validate data before data amplification. The default value is True.
Scale	Zooms in or out an image. The length or width of an image is randomly zoomed in or out.	<ul style="list-style-type: none"> ● scaleXY: scaling direction. X indicates horizontal, and Y indicates vertical. The default value is X. ● scale_min: lower limit of the random scaling ratio range. The default value is 0.5. ● scale_max: upper limit of the random scaling ratio range. The default value is 1.5. ● do_validation: indicates whether to validate data before data amplification. The default value is True.
Sharpen	Indicates image sharpening, which is used to sharpen the edges of objects.	<ul style="list-style-type: none"> ● do_validation: indicates whether to validate data before data amplification. The default value is True.

Operator	Description	Advanced
Shear	Indicates image shearing, which is used for geometric transformation of images. Pixels are mapped using linear functions.	<ul style="list-style-type: none"> ● shearXY: shearing direction. X indicates horizontal, and Y indicates vertical. The default value is X. ● shear_min: lower limit of the random shearing angle range. The default value is -30. ● shear_max: upper limit of the random shearing angle range. The default value is 30. ● do_validation: indicates whether to validate data before data amplification. The default value is True.
Translate	Moves an image along the x-axis or y-axis, discards the part that exceeds the original image, and fills the blank part with black.	<ul style="list-style-type: none"> ● translateXY: translation direction. X indicates horizontal, and Y indicates vertical. The default value is X. ● do_validation: indicates whether to validate data before data amplification. The default value is True.
Weather	Adds weather information to simulate the weather effect.	<p>weather_mode: weather mode. The default value is Rain.</p> <ul style="list-style-type: none"> ● Rain: rain ● Fog: fog ● Snow: snow ● Clouds: cloud <p>do_validation: indicates whether to validate data before data amplification. The default value is True.</p>

Operator Input Requirements

The following two types of operator input are available:

- **Datasets**: Select a dataset and its version created on the ModelArts console from the drop-down list. Ensure that the dataset type be the same as the scenario type selected in this task.
- **OBS Catalog**: The storage structure supports **Images and labels**.

Images and labels: The structure varies depending on the scenario type.

The following shows the directory structure in the image classification scenario. The following directory structure supports only single-label scenarios.

```
input_path/
--label1/
```

```
----1.jpg
--label2/
----2.jpg
--./
```

The following shows the directory structure in the object detection scenario. Images in JPG, JPEG, PNG, and BMP formats are supported. XML files are standard PACAL VOC files.

```
input_path/
--1.jpg
--1.xml
--2.jpg
--2.xml
...
```

Output Description

Some data will be discarded due to some operations of the operators. Therefore, the output folder may not contain the full dataset. For example, **Rotate** will discard the images whose bounding boxes exceed the image boundaries.

The following shows the output directory structure. In this structure, the **Data** folder stores newly generated images and labeling information. The **manifest** file stores the structure of images in the folder and can be directly imported to the dataset in Data Management.

```
|----data_url
|----Data
|----xxx.jpg
|----xxx.xml(xxx.txt)
|----output.manifest
```

A manifest file example is as follows:

```
{
  "id": "xss",
  "source": "obs://home/fc8e2688015d4a1784dcba44d840307_14.jpg",
  "usage": "train",
  "annotation": [
    {
      "name": "Cat",
      "type": "modelarts/image_classification"
    }
  ]
}
```

2.15.4.6 Data Augmentation (Image Generation)

Image Generation Operator Overview

The image generation operator uses a generative adversarial network (GAN) to generate a new dataset with the existing dataset. A GAN is a network that contains the generator and discriminator. The generator randomly selects samples from a latent space as the input and outputs results similar to the real samples in the training set. The discriminator distinguishes the outputs of the generative network from the real samples by inputting real samples or outputs of the generative network. The generative network's training objective is to increase the error rate of the discriminative network (for example, "fool" the discriminative network). The two networks compete with each other and continuously adjust parameters to achieve the final purpose, that is, make the discriminative network

unable to distinguish whether the output of the generative network is true. The generative network obtained during training can be used to generate images similar to the input images, which can be used as new datasets for training. New datasets generated based on GANs have no labels. Original data is not changed during image generation. The newly generated image or XML file is saved in the specified output path.

ModelArts provides two types of image generation operators:

- CycleGan operator: generates images for domain transfer based on CycleGAN, that is, converts one type of images into another, or converts samples in the X space into samples in the Y space. CycleGAN can use non-paired data for training. During model training, two inputs are supported, indicating the source domain and target domain of data, respectively. After the training is complete, all images transferred from the source domain to the target domain are generated.

Table 2-40 Advanced parameters of the CycleGan operator

Parameter	Default Value	Description
do_validation	True	Whether to verify data. The default value is True , which indicates that data is validated before being generated. False indicates data is generated directly.
image_channel	3	Number of channels of the image
image_height	256	Image height. The value must be 2 to the power of n .
image_width	256	Image width. The value must be 2 to the power of n .
batch_size	1	Training parameter: number of samples for batch training.
max_epoch	100	Training parameter: number of dataset epochs during training.
g_learning_rate	0.0001	Training parameter: learning rate for the generator training.
d_learning_rate	0.0001	Training parameter: learning rate for the discriminator training.
log_frequency	5	Training parameter: logging frequency (counted by step).
save_frequency	5	Training parameter: model save frequency (counted by epoch)
predict	False	Whether to perform inference and prediction. The default value is False . If this parameter is set to True , you need to set resume to the OBS path where the trained model is stored.

Parameter	Default Value	Description
resume	empty	If predict is set to True , enter the OBS path where the TensorFlow model file is stored for inference and prediction. Currently, only models in .pb format are supported. Example: obs://xxx/xxxx.pb . The default value is empty .

- StyleGAN operator: randomly generates similar images based on StyleGAN2 when a dataset is small. StyleGAN has a new generator architecture that can control the high-level attributes such as hairstyle and freckles of the generated images. These generated images perform even better in certain aspects. In addition, StyleGAN is equipped with the data augmentation algorithm, which can generate new satisfactory samples even with a small number of samples. However, there must be at least 70 samples to have rich image styles.

Table 2-41 Advanced parameters of the StyleGan operator

Parameter	Default Value	Description
resolution	256	Height and width of the generated square image. The value must be 2 to the power of n .
batch-size	8	Number of samples for batch training
total-kimg	300	The total number of trained images is obtained by multiplying this parameter value by 1,000.
generate_number	300	Number of generated images. If the generated images have multiple classes, the value is the number of generated images of each class.
predict	False	Whether to perform inference and prediction. The default value is False . If this parameter is set to True , you need to set resume to the OBS path where the trained model is stored.
resume	empty	If predict is set to True , enter the OBS path where the TensorFlow model file is stored for inference and prediction. Currently, only models in .pb format are supported. Example: obs://xxx/xxxx.pb . The default value is empty .
do_validation	True	Whether to verify data. The default value is True , which indicates that data is validated before being generated. False indicates data is generated directly.

Data Input

The following two types of operator input are available:

- **Datasets:** Select a dataset and its version created on the ModelArts console from the drop-down list. Ensure that the dataset type be the same as the scenario type selected in this task.
- **OBSCatalog:** Operator image_generation does not require labeling information and its input supports single-level or dual-level directories, and the storage structure supports single-level or dual-level mode.

The single-level directory structure is as follows:

```
image_folder---0001.jpg
  ---0002.jpg
  ---0003.jpg
  ...
  ---1000.jpg
```

The dual-level directory structure is as follows:

```
image_folder---sub_folder_1---0001.jpg
  ---0002.jpg
  ---0003.jpg
  ...
  ---0500.jpg
  ---sub_folder_2---0001.jpg
    ---0002.jpg
    ---0003.jpg
    ...
    ---0500.jpg
  ...
  ---sub_folder_100---0001.jpg
    ---0002.jpg
    ---0003.jpg
    ...
    ---0500.jpg
```

Output Description

The output directory structure is as follows: Folder **model** stores model **frozen PB** for inference, folder **samples** stores the output images during training, and folder **Data** stores the images generated by the training model.

```
train_url---model---CYcleGan_epoch_10.pb
  ---CYcleGan_epoch_20.pb
  ...
  ---CYcleGan_epoch_1000.pb
  ---samples---0000_0.jpg
  ---0000_1.jpg
  ...
  ---0100_15.jpg
  ---Data---CYcleGan_0_0.jpg
  ---CYcleGan_0_1.jpg
  ...
  ---CYcleGan_16_8.jpg
  ---output_0.manifest
```

A manifest file example is as follows:

```
{
  "id": "xss",
  "source": "obs://home/fc8e2688015d4a1784dcbda44d840307_14.jpg",
  "usage": "train",
  "annotation": [
```

```
{  
  "name": "Cat",  
  "type": "modelarts/image_classification"  
}
```

3 Training Management (Old Version)

3.1 Introduction to Model Training

 NOTE

ModelArts provides model training of both the new and old versions. Training management of the old version is only available for its existing users.

This tutorial applies only to training management of the old version, which will be discontinued soon. Use the tutorial for the new version. For details, see, [Introduction to Model Development](#).

For details about the differences between the old and new versions, see [FAQs](#).

ModelArts provides model training for you to view the training effect, based on which you can adjust your model parameters. You can select resource pools with different instance flavors for model training.

Model Training Features

Table 3-1 Features

Feature	Description	Reference
Training job management	You can create training jobs, manage training job versions, and view details of training jobs, and evaluation details.	Introduction to Training Jobs Managing Training Job Versions Viewing Job Details

Feature	Description	Reference
Job parameter management	You can save the parameter settings of a training job (including the data source, algorithm source, running parameters, resource pool parameters, and more) as a job parameter, which can be directly used when you create a training job, eliminating the need to set parameters one by one. As such, the configuration efficiency can be greatly improved.	Managing Job Parameters
Model training visualization	TensorBoard and MindInsight effectively display the computational graph of a model in the running process, the trend of all metrics in time, and the data used in the training.	Managing Visualization Jobs

3.2 Frequently-used Frameworks

This section describes frequently-used AI frameworks supported by ModelArts and how to use these frameworks to compile training code for creating training jobs.

Frequently-used AI Frameworks for Training Management

ModelArts supports the following AI engines and versions.

NOTE

- MoXing is a distributed training acceleration framework developed by the ModelArts team. It is built on the open-source deep learning engines TensorFlow, MXNet, PyTorch, and Keras. If you use MoXing to compile a training script, select the corresponding AI engine and version based on your selected API when you create a training job.

Developing Training Code Using Frequently-used Frameworks

When creating a training job using a common framework, set the code directory, boot file, input path, and output path. These settings enable the interaction between you and ModelArts.

- Code directory
Specify the code directory in the OBS bucket and upload training data such as training code, dependency installation packages, or pre-generated model to the directory. After you create the training job, ModelArts downloads the code directory and its subdirectories to the container.
- Boot file
The boot file in the code directory is used to start the training. Only Python boot files are supported.
- OBS path for storing training data

Do not store training data in the code directory. When the training job starts, the data stored in the code directory will be downloaded to the backend. A large amount of training data may lead to a download failure.

After the training job is started, ModelArts mounts a disk to the **/cache** directory. You can use this directory to store temporary files. For details about the size of the **/cache** directory, see [What Are Sizes of the /cache Directories for Different Resource Specifications in the Training Environment?](#)

You must upload the training data to another OBS path rather than the code directory and parse **data_url** to download the training data to the **/cache** directory. Ensure that you have the read permission to the OBS bucket.

- Output path

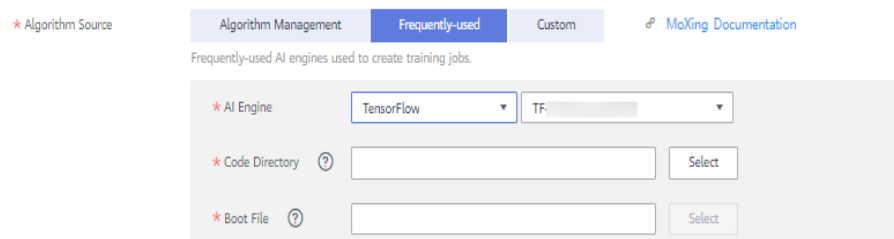
You are advised to set an empty directory as the training output path. In the training code, you must parse **train_url** to upload the training output to the output path. Ensure that you have the write and read permissions to the OBS bucket.

When you use a frequently-used framework to create a training job, develop training code. The following figure shows the process of developing training code.

1. (Optional) Introduce dependencies.

If your model references other dependencies, place the corresponding file or installation package in **Code Directory** you set during training job creation.

Figure 3-1 Selecting a frequently-used framework and specifying the model boot file



2. Parse mandatory parameters **data_url** and **train_url**.

When using a frequently-used framework to create a training job, set job parameters on the page for creating a training job.

data_url: Training data is mandatory for developing training code. When creating a training job, set **Data Source**. In the training code, **data_url** indicates the path of the data source.

train_url: After model training is complete, the trained model and the output information must be stored in an OBS path. When creating a training job, set **Training Output Path**. In the training code, **train_url** indicates the OBS path specified for **Training Output Path**.

Figure 3-2 Job parameters

The screenshot shows a configuration interface for job parameters. It includes the following elements:

- Data Source:** A dropdown menu with 'Dataset' and 'Data path' options. Below it is a 'Data Path' input field with a 'Select' button and a trash icon.
- Training Output Path:** An input field with a 'Select' button. A note below it says: "We recommend you select an empty directory as the output path."
- Running Parameter:** An 'Add Running Parameter' button.
- Job Log Path:** An input field with 'Select' and 'Clear' buttons. A note below it says: "By default, logs are stored in the service and will be deleted irregularly. Select a path for storing logs."

In the training code, **data_url** and **train_url** must be parsed. Use the following parameter parsing methods for ModelArts:

```
import argparse
# Create a parsing task.
parser = argparse.ArgumentParser(description="train mnist",
                                formatter_class=argparse.ArgumentDefaultsHelpFormatter)
# Add parameters.
parser.add_argument('--train_url', type=str, default='obs://obs-test/ckpt/mnist',
                    help='the path model saved')
parser.add_argument('--data_url', type=str, default='obs://obs-test/data/', help='the training data')
# Parse the parameters.
args, unknown = parser.parse_known_args()
```

3. Import training data from **data_url**.

Training data is stored in **data_url**. Use the MoXing API to download the data to the **cache** directory.

```
import moxing as mox
mox.file.copy_parallel(args.data_url, "/cache")
```

4. Compile training code and save the model.

Training code and the code for saving the model are closely related to the AI engine you use. The following uses the TensorFlow framework as an example. In the training code, the TensorFlow API **tf.flags.FLAGS** is used to receive CLI parameters.

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import os

import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

import moxing as mox

tf.flags.DEFINE_integer('max_steps', 1000, 'number of training iterations.')
tf.flags.DEFINE_string('data_url', '/home/jnn/nfs/mnist', 'dataset directory.')
tf.flags.DEFINE_string('train_url', '/home/jnn/temp/delete', 'saved model directory.')

FLAGS = tf.flags.FLAGS

def main(*args):
    mox.file.copy_parallel(FLAGS.data_url, '/cache/data_url')

    # Train model
    print('Training model...')
    mnist = input_data.read_data_sets('/cache/data_url', one_hot=True)
```

```

sess = tf.InteractiveSession()
serialized_tf_example = tf.placeholder(tf.string, name='tf_example')
feature_configs = {'x': tf.FixedLenFeature(shape=[784], dtype=tf.float32),}
tf_example = tf.parse_example(serialized_tf_example, feature_configs)
x = tf.identity(tf_example['x'], name='x')
y_ = tf.placeholder('float', shape=[None, 10])
w = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
sess.run(tf.global_variables_initializer())
y = tf.nn.softmax(tf.matmul(x, w) + b, name='y')
cross_entropy = -tf.reduce_sum(y_ * tf.log(y))

tf.summary.scalar('cross_entropy', cross_entropy)

train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entropy)

correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, 'float'))
tf.summary.scalar('accuracy', accuracy)
merged = tf.summary.merge_all()
test_writer = tf.summary.FileWriter('/cache/train_url', flush_secs=1)

for step in range(FLAGS.max_steps):
    batch = mnist.train.next_batch(50)
    train_step.run(feed_dict={x: batch[0], y_: batch[1]})
    if step % 10 == 0:
        summary, acc = sess.run([merged, accuracy], feed_dict={x: mnist.test.images, y_:
mnist.test.labels})
        test_writer.add_summary(summary, step)
        print('training accuracy is:', acc)
    print('Done training!')

builder = tf.saved_model.builder.SavedModelBuilder(os.path.join('/cache/train_url', 'model'))

tensor_info_x = tf.saved_model.utils.build_tensor_info(x)
tensor_info_y = tf.saved_model.utils.build_tensor_info(y)

prediction_signature = (
    tf.saved_model.signature_def_utils.build_signature_def(
        inputs={'images': tensor_info_x},
        outputs={'scores': tensor_info_y},
        method_name=tf.saved_model.signature_constants.PREDICT_METHOD_NAME))

builder.add_meta_graph_and_variables(
    sess, [tf.saved_model.tag_constants.SERVING],
    signature_def_map={
        'predict_images':
            prediction_signature,
    },
    main_op=tf.tables_initializer(),
    strip_default_attrs=True)

builder.save()

print('Done exporting!')

mox.file.copy_parallel('/cache/train_url', FLAGS.train_url)

if __name__ == '__main__':
    tf.app.run(main=main)

```

5. Export the trained model to **train_url**.

The training output path is **train_url**. You are advised to use the MoXing API to export the output from **/cache/train_url** to **train_url**.

```
mox.file.copy_parallel("/cache/train_url", args.train_url)
```

3.3 Creating a Training Job

3.3.1 Introduction to Training Jobs

ModelArts supports multiple types of training jobs during the entire AI development process. Select a creation mode based on the algorithm source.

Algorithm Sources of Training Jobs

- **Frequently-used**
If you have used some frequently-used frameworks to develop algorithms locally, you can select a frequently-used framework and create a training job to build a model. For details, see [Using Frequently-used Frameworks to Train Models](#).
- **Custom**
If the framework used for algorithm development is not a frequently-used framework, you can build an algorithm into a custom image and use the custom image to create a training job. For details about the operation guide to create a training job, see [Using Custom Images to Train Models](#). For details about the specifications and description of custom images, see [Specifications for Custom Images Used for Training Jobs](#).

3.3.2 Using Existing Algorithms to Train Models

You can quickly use a created algorithm to create a training job and build a model.

Prerequisites

- Data has been prepared. Specifically, you have created an available dataset in ModelArts, or you have uploaded the dataset used for training to the OBS directory.
- The algorithm used for training is available on the **Algorithm Management** page. The algorithm can be obtained by creating a custom script. The newly created algorithms are supported only in the new version of training management. For details, see [Creating an Algorithm](#).
- At least one empty folder has been created in OBS for storing the training output.
- The account is not in arrears because resources are consumed when training jobs are running.
- The OBS directory you use and ModelArts are in the same region.

Precautions

In the dataset directory specified for a training job, the names of the files (such as the image file, audio file, and label file) containing data used for training contain 0 to 255 characters. If the names of certain files in the dataset directory contain over 255 characters, the training job will ignore these files and use data in the

valid files for training. If the names of all files in the dataset directory contain over 255 characters, no data is available for the training job and the training job fails.

Creating a Training Job

1. Log in to the ModelArts management console. In the left navigation pane, choose **Training Management > Training Jobs**. By default, the system switches to the **Training Jobs** page.
2. In the upper left corner of the training job list, click **Create** to switch to the **Create Training Job** page.
3. Set related parameters and click **Next**.
 - a. Enter the basic information, including **Name** and **Description**. For **Version**, the system automatically creates a version number, which is named according to a certain rule, for example, **V0001** and **V0002**. The version number cannot be changed.
 - b. Set job parameters, including the data source, algorithm source, and more. For details, see [Table 3-2](#). The value range of the data source is consistent with the constraints of existing algorithms.

Table 3-2 Job parameters

Parameter	Sub-Parameter	Description
Algorithm Source	Algorithm Management	Select Algorithm Management and click Select on the right of the algorithm name. The Algorithm Management page is displayed. <ul style="list-style-type: none"> • On the My Algorithms tab page, you can select a created algorithm. The newly created algorithms are supported only in the new version of training management. For details, see Creating an Algorithm.
Training Input	Data Source > Dataset	Select an available dataset and its version from the ModelArts Data Management module. <ul style="list-style-type: none"> • Dataset: Select a published dataset from the drop-down list. If no dataset is available in ModelArts, no result will be displayed in the drop-down list. • Version: Select a version based on the selected dataset.
	Data Source > Data path	Select the training data from your OBS bucket. On the right of the Data path text box, click Select . In the dialog box that is displayed, select an OBS folder for storing data.

Parameter	Sub-Parameter	Description
Training Output	Model Output	Select the storage path of the training result (OBS path). To minimize errors, select an empty directory for Model Output . Do not use the dataset storage directory as the training output location.
Hyperparameters	-	The value of this parameter varies according to the selected algorithm. If tuning parameters are defined for the created or subscribed algorithm, you need to set the parameters when you create a training job. You can click Add Hyperparameter to add multiple hyperparameters.
Job Log Path	-	Select a path for storing log files generated during job running.

- c. Select resources for the training job. The value range of the training parameters is consistent with the constraints of existing algorithms.

Table 3-3 Resource parameters

Parameter	Description
Resource Pool	Select resource pools for the job. For training jobs, Public resource pools and Dedicated resource pools are available.
Instance Flavor	Select a resource flavor based on the resource type. The GPU resource delivers better performance, and the CPU resource is more cost effective. If your algorithm has been defined to use CPUs or GPUs, you can select a proper resource flavor based on the constraints of existing algorithms. Invalid options are grayed out. The data disk capacity varies depending on the resource type. For details, see What Are Sizes of the /cache Directories for GPU and CPU Resources in the Training Environment?
Compute Nodes	Set the number of compute nodes. The default value is 1 .

- d. Configure the subscription function and set whether to save the parameter settings of the training job.

Figure 3-3 Configuring notifications for the training job

Table 3-4 Parameters related to the subscription function and parameter configuration saving

Parameter	Description
Notification	<p>Select the resource pool status to be monitored from the event list, and SMN sends a notification message when the event occurs.</p> <p>This parameter is optional. You can choose whether to enable subscription based on actual requirements. If you enable subscription, set the following parameters as required:</p> <ul style="list-style-type: none"> • Topic: indicates the topic name. You can create a topic on the SMN console. • Event: indicates the event to be subscribed to. The options are OnJobRunning, OnJobSucceeded, and OnJobFailed, indicating that the job is in progress, successful, and failed, respectively.
Saving Training Parameters	<p>If you select this option, the parameter settings of the current training job will be saved to facilitate subsequent job creation.</p> <p>Select Save Training Parameters and specify Configuration Name and Description. After a training job is created, you can switch to the Job Parameters tab page to view your saved job parameter settings. For details, see Managing Job Parameters.</p>

- e. After setting the parameters, click **Next**.
4. On the **Confirm** page that is displayed, confirm that the information is correct and click **Submit**. Generally, training jobs run for a period of time, which may be several minutes or tens of minutes depending on the amount of your selected data and resources.

NOTE

After a training job is created, it is started immediately. During the running, you will be charged based on your selected resources.

You can switch to the training job list to view the basic information about training jobs. In the training job list, **Status** of the newly created training job is **Initializing**. If the status changes to **Successful**, the training job ends and the model generated is stored in the location specified by **Training Output Path**. If the status of a training job changes to **Running failed**. Click the name of the training job and view the job logs. Troubleshoot the fault based on the logs.

3.3.3 Using Frequently-used Frameworks to Train Models

If you use frequently-used frameworks, such as TensorFlow and MindSpore, to develop algorithms locally, you can select **Frequently-used** to create training jobs and build models.

Prerequisites

- Data has been prepared. Specifically, you have created an available dataset in ModelArts, or you have uploaded the dataset used for training to the OBS directory.
- If you select **Frequently-used** for **Algorithm Source**, prepare the training script and upload it to the OBS directory.
- At least one empty folder has been created in OBS for storing the training output.
- The account is not in arrears because resources are consumed when training jobs are running.
- The OBS directory you use and ModelArts are in the same region.

Precautions

- In the dataset directory specified for a training job, the names of the files (such as the image file, audio file, and label file) containing data used for training contain 0 to 255 characters. If the names of certain files in the dataset directory contain over 255 characters, the training job will ignore these files and use data in the valid files for training. If the names of all files in the dataset directory contain over 255 characters, no data is available for the training job and the training job fails.
- In the training script, the **Data Source** and **Training Output Path** parameters must be set to the OBS path. To perform read and write operations in the path, use MoXing APIs.

Creating a Training Job

1. Log in to the ModelArts management console. In the left navigation pane, choose **Training Management > Training Jobs**. By default, the system switches to the **Training Jobs** page.
2. In the upper left corner of the training job list, click **Create** to switch to the **Create Training Job** page.
3. Set related parameters and click **Next**.
 - a. Set the basic information, Specify **Name** and **Description** according to actual requirements. For **Version**, the system automatically creates a version number, which is named according to a certain rule, for example, **V0001** and **V0002**. The version number cannot be changed.

- b. Set job parameters, including the data source, algorithm source, and more. For details, see [Table 3-5](#).

Figure 3-4 Frequently-used as the algorithm source

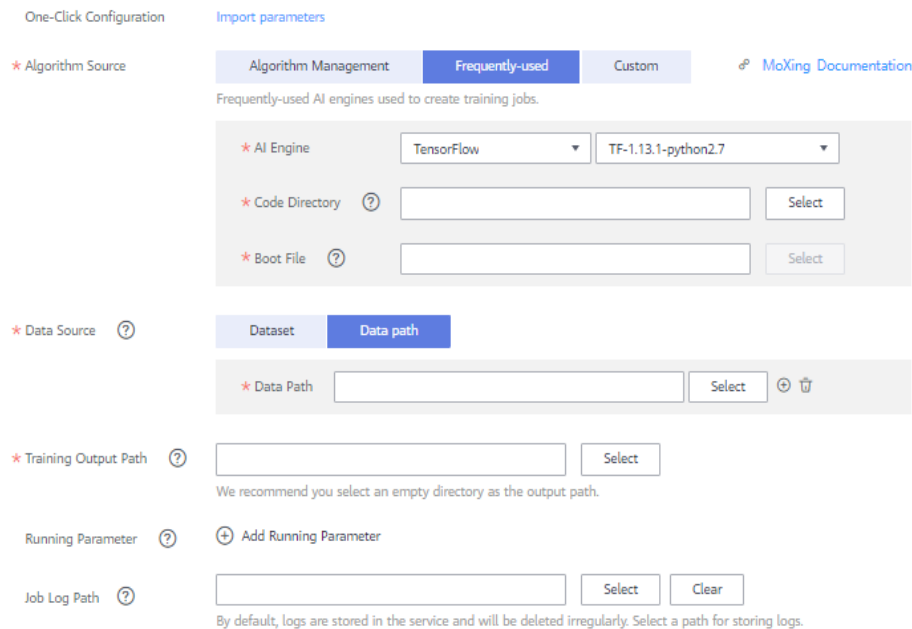






Table 3-5 Job parameters

Parameter	Sub-Parameter	Description
One-Click Configuration	N/A	If you have saved job parameter configurations in ModelArts, click One-Click Configuration and select an existing job parameter configuration as prompted to quickly complete parameter setting for the job.
Algorithm Source	Frequently-used	Select an AI engine and its version and specify Code Directory and Boot File . The framework selected for the AI engine must be the same as the one you select when compiling training code. For example, if TensorFlow is used in your training code, select TensorFlow when you create a training job. If your model requires Python dependencies, place the dependency package and its configuration file in the code directory based on the requirements defined in ModelArts. For details, see How Do I Create a Training Job When a Dependency Package is Referenced in a Model .

Parameter	Sub-Parameter	Description
Data Source	Dataset	<p>Select an available dataset and its version from the ModelArts Data Management module.</p> <ul style="list-style-type: none"> • Dataset: Select an existing dataset from the drop-down list. If no dataset is available in ModelArts, no result will be displayed in the drop-down list. • Version: Select a version according to the Dataset setting. <p>For a training job, you can select multiple datasets by clicking . To delete a dataset, click  in the row for it.</p>
	Data path	<p>Select the training data from your OBS bucket. On the right of the Data path text box, click Select. In the dialog box that is displayed, select an OBS folder for storing data.</p> <p>If you set Algorithm Source to Frequently-used, you can select multiple data storage paths for a training job by clicking . To delete a data storage path, click  in the row for it.</p>
Training Output Path	N/A	<p>Select a path for storing the training result.</p> <p>NOTE To minimize errors, select an empty directory for Training Output Path. Do not select the directory used for storing the dataset for Training Output Path.</p>
Running Parameter	N/A	<p>Set the command line parameters in the code based on the algorithm code logic. Make sure that the parameter names are the same as those in the code.</p> <p>For example, train_steps = 10000, where train_steps is a passing parameter in code.</p>
Job Log Path	N/A	<p>Select a path for storing log files generated during job running.</p>

- c. Select resources for the training job.

Table 3-6 Resource parameters

Parameter	Description
Resource Pool	Select resource pools for the job. Instances in the public resource pool can be of the CPU or GPU type. Pricing standards for resource pools with different instance types are different. For details, see Product Pricing Details .
Type	If Resource Pool is set to Public resource pools , select a resource type. Available resource types are CPU and GPU . The GPU resource delivers better performance, and the CPU resource is more cost effective. If the selected algorithm has been defined to use the CPU or GPU, the resource type is automatically displayed on the page. Select the resource type as required. The data disk capacity varies depending on the resource type. For details, see What Are Sizes of the / cache Directories for GPU and CPU Resources in the Training Environment? NOTE If GPU resources are used in training code, you must select a GPU cluster when selecting a resource pool. Otherwise, the training job may fail.
Specifications	Select a resource flavor based on the resource type.
Compute Nodes	Set the number of compute nodes. If you set Compute Nodes to 1 , the standalone computing mode is used. If you set Compute Nodes to a value greater than 1, the distributed computing mode is used. Select a computing mode based on the actual requirements. When Frequently-used of Algorithm Source is set to Caffe , only standalone training is supported, that is, Compute Nodes must be set to 1 . For other options of Frequently-used , you can select the standalone or distributed mode based on service requirements.

- d. Configure **Notification** and select whether to save the training job parameters.

Figure 3-5 Configuring notifications for the training job

Table 3-7 Parameters related to notification and parameter configuration saving

Parameter	Description
Notification	<p>Select the resource pool status to be monitored from the event list, and SMN sends a notification message when the event occurs.</p> <p>This parameter is optional. You can choose whether to enable subscription based on actual requirements. If you enable subscription, set the following parameters as required:</p> <ul style="list-style-type: none"> • Topic: indicates the topic name. You can create a topic on the SMN console. • Event: indicates the event to be subscribed to. The options are OnJobRunning, OnJobSucceeded, and OnJobFailed, indicating that training is in progress, successful, and failed, respectively.
Saving Training Parameters	<p>If you select this option, the parameter settings of the current training job will be saved to facilitate subsequent job creation.</p> <p>Select Save Training Parameters and specify Configuration Name and Description. After a training job is created, you can switch to the Job Parameters tab page to view your saved job parameter settings. For details, see Managing Job Parameters.</p>

- e. After setting the parameters, click **Next**.
4. Confirm that the information is correct on the **Confirm** page that is displayed and click **Submit**. Generally, training jobs run for a period of time, which may be several minutes or tens of minutes depending on the amount of your selected data and resources.

NOTE

After a training job is created, it is started immediately. During the running, you will be charged based on your selected resources.

You can switch to the training job list to view the basic information about training jobs. In the training job list, **Status** of the newly created training job is **Initializing**. If the status changes to **Successful**, the training job ends and the model generated is stored in the location specified by **Training Output Path**. If the status of a training job changes to **Running failed**, click the name of the training job and view the job logs. Troubleshoot the fault based on the logs.

3.3.4 Using Custom Images to Train Models

If the framework used for algorithm development is not a frequently-used framework, you can build an algorithm into a custom image and use the custom image to create a training job.

Prerequisites

- Data has been prepared. Specifically, you have created an available dataset in ModelArts, or you have uploaded the dataset used for training to the OBS directory.
- If the algorithm source is **Custom**, create an image and upload the image to SWR. For details, see [Specifications for Custom Images Used for Training Jobs](#).
- The training script has been uploaded to the OBS directory.
- At least one empty folder has been created in OBS for storing the training output.
- The account is not in arrears because resources are consumed when training jobs are running.
- The OBS directory you use and ModelArts are in the same region.

Precautions

- In the dataset directory specified for a training job, the names of the files (such as the image file, audio file, and label file) containing data used for training contain 0 to 255 characters. If the names of certain files in the dataset directory contain over 255 characters, the training job will ignore these files and use data in the valid files for training. If the names of all files in the dataset directory contain over 255 characters, no data is available for the training job and the training job fails.
- In the training script, the **Data Source** and **Training Output Path** parameters must be set to the OBS path. To perform read and write operations in the path, use MoXing APIs.

Creating a Training Job

1. Log in to the ModelArts management console. In the left navigation pane, choose **Training Management > Training Jobs**. By default, the system switches to the **Training Jobs** page.
2. In the upper left corner of the training job list, click **Create** to switch to the **Create Training Job** page.
3. Set related parameters and click **Next**.

- a. Set the basic information,
Specify **Name** and **Description** according to actual requirements.
- b. Set job parameters, including the data source, algorithm source, and more. For details, see [Table 3-8](#).

Table 3-8 Job parameters

Parameter	Sub-Parameter	Description
One-Click Configuration	-	If you have saved job parameter configurations in ModelArts, click One-Click Configuration and select an existing job parameter configuration as prompted to quickly complete parameter setting for the job.
Algorithm Source	Custom	For details about custom image specifications, see Specifications for Custom Images Used for Training Jobs . <ul style="list-style-type: none"> ● Image Path: SWR URL after the image is uploaded to SWR. ● Code Directory: OBS path for storing the training code file. ● Boot Command: Command to boot the training job after the image is started. Set this parameter based on site requirements.
Data Source	Dataset	Select an available dataset and its version from the ModelArts Data Management module. <ul style="list-style-type: none"> ● Dataset: Select an existing dataset from the drop-down list. If no dataset is available in ModelArts, no result will be displayed in the drop-down list. ● Version: Select a version according to the Dataset setting.
	Data path	Select the training data from your OBS bucket. On the right of the Data path text box, click Select . In the dialog box that is displayed, select an OBS folder for storing data.
Training Output Path	-	Storage path of the training result NOTE To minimize errors, select an empty directory for Training Output Path . Do not select the directory used for storing the dataset for Training Output Path .
Environment Variable	-	Add environment variables based on your image file. This parameter is optional. You can click Add Environment Variable to add multiple variable parameters.

Parameter	Sub-Parameter	Description
Job Log Path	-	Select a path for storing log files generated during job running.

- c. Select resources for the training job.

Table 3-9 Resource parameters

Parameter	Description
Resource Pool	Select resource pools for the job. Instances in the public resource pool can be of the CPU or GPU type. Pricing standards for resource pools with different instance types are different. For details, see Product Pricing Details .
Type	If Resource Pool is set to Public resource pools , select a resource type. Available resource types are CPU and GPU . The GPU resource delivers better performance, and the CPU resource is more cost effective. If the selected algorithm has been defined to use the CPU or GPU, the resource type is automatically displayed on the page. Select the resource type as required. The data disk capacity varies depending on the resource type. For details, see What Are Sizes of the / cache Directories for GPU and CPU Resources in the Training Environment?
Specifications	Select a resource flavor based on the resource type.
Compute Nodes	Set the number of compute nodes. If you set Compute Nodes to 1 , the standalone computing mode is used. If you set Compute Nodes to a value greater than 1 , the distributed computing mode is used. Select a computing mode based on the actual requirements.

- d. Configure **Notification** and select whether to save the training job parameters.

Figure 3-6 Configuring notifications for the training job

Table 3-10 Parameters related to notification and parameter configuration saving

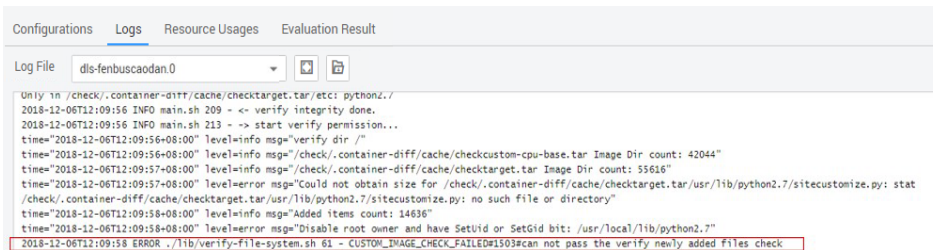
Parameter	Description
Notification	<p>Select the resource pool status to be monitored from the event list, and SMN sends a notification message when the event occurs.</p> <p>This parameter is optional. You can choose whether to enable subscription based on actual requirements. If you enable subscription, set the following parameters as required:</p> <ul style="list-style-type: none"> • Topic: indicates the topic name. You can create a topic on the SMN console. • Event: indicates the event to be subscribed to. The options are OnJobRunning, OnJobSucceeded, and OnJobFailed, indicating that training is in progress, successful, and failed, respectively.
Saving Training Parameters	<p>If you select this option, the parameter settings of the current training job will be saved to facilitate subsequent job creation.</p> <p>Select Save Training Parameters and specify Configuration Name and Description. After a training job is created, you can switch to the Job Parameters tab page to view your saved job parameter settings. For details, see Managing Job Parameters.</p>

- e. After setting the parameters, click **Next**.
4. Confirm that the information is correct on the **Confirm** page that is displayed and click **Submit**. Generally, training jobs run for a period of time, which may be several minutes or tens of minutes depending on the amount of your selected data and resources.

After a custom image job is created, the system authorizes ModelArts to obtain and run the image by default. When you run a custom image job for the first time, ModelArts checks the custom image. For details about the

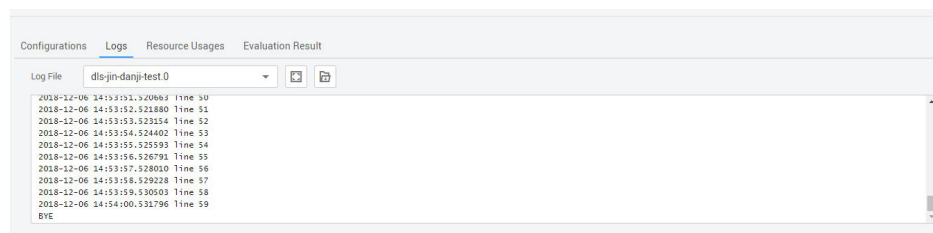
check, see [Specifications for Custom Images Used for Training Jobs](#). You can view the cause of the check failure in the log and modify the custom image based on the log.

Figure 3-7 Failed to check the image



After the image is checked, the background starts the custom image container to run the custom image training job. You can switch to the training job list to view the basic information about training jobs. In the training job list, **Status** of the newly created training job is **Initializing**. If the status changes to **Successful**, the training job ends and the model generated is stored in the location specified by **Training Output Path**. If the status of a training job changes to **Running failed**. Click the name of the training job and view the job logs. Troubleshoot the fault based on the logs.

Figure 3-8 Run log



NOTE

- After a training job is created, it is started immediately. During the running, you will be charged based on your selected resources.
- After an image is reviewed, the image does not need to be reviewed again when being used to create training jobs again.
- The default user of a custom image must be the user whose UID is **1101**.

3.4 Stopping or Deleting a Job

Stopping a Training Job

In the training job list, click **Stop** in the **Operation** column for a training job in the **Running** state to stop a running training job.

After the training job is stopped, its billing stops on ModelArts. If you have selected **Save Training Parameters** for a stopped training job, the job's parameter settings will be saved to the **Job Parameter Mgmt** page.

You cannot stop a training job that has stopped running, for example the job in the **Successful** or **Running failed** state. Only training jobs in the **Running** state can be stopped.

Deleting a Training Job

If an existing training job is no longer used, you can delete it.

For a training job in the **Running**, **Successful**, **Running failed**, **Canceled**, or **Deploying** state, click **Delete** in the **Operation** column to delete it.

If you have selected **Save Training Parameters** for a deleted training job, the job's parameter settings will be saved to the **Job Parameter Mgmt** page.

3.5 Managing Training Job Versions

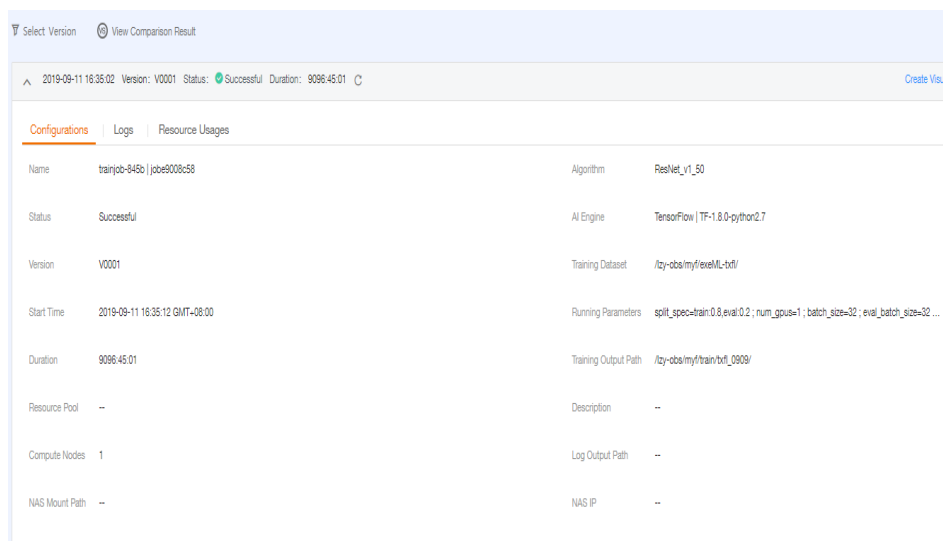
During model building, you may need to frequently tune the data, training parameters, or the model based on the training results to obtain a satisfactory model. ModelArts allows you to manage training job versions to effectively train your model after the tuning. Specifically, ModelArts generates a version each time when a training is performed. You can quickly get the difference between different versions.

Viewing Training Job Versions

1. Log in to the ModelArts management console. In the left navigation pane, choose **Training Management > Training Jobs**. By default, the system switches to the **Training Jobs** page.
2. In the training job list, click the name of a training job.

By default, the basic information about the latest version is displayed. If there are multiple available versions, click **Select Version** in the upper left corner to view a certain version. Click the downward arrow to the left of the version to display job details. For details about the training job, see [Training Job Details](#).

Figure 3-9 Viewing training job versions



Comparing Versions of a Training Job

On the **Version Manager** page, click **View Comparison Result** to view the comparison of all or selected versions of the current training job. The comparison result involves the following information: **Running Parameter**, **F1 Score**, **Recall**, **Precision**, and **Accuracy**.

 **NOTE**

The **F1 Score**, **Recall**, **Precision**, and **Accuracy** parameters of a training job are displayed only when the job is created using a built-in algorithm. For training jobs created using frequently-used frameworks or custom images, define the output of these parameters in your training script code. These parameters cannot be viewed on the GUI.

Figure 3-10 Comparing versions of a training job

View Comparison Result

Vers...	Running Parameter	F1 Score	Recall	Precision	Accuracy
V0005	data_path_suffix=codes				
V0004	data_path_suffix=codes				
V0003	data_path_suffix=codes				
V0002	data_path_suffix=codes				
V0001	data_path_suffix=codes				

Shortcut Operations Based on Training Job Versions

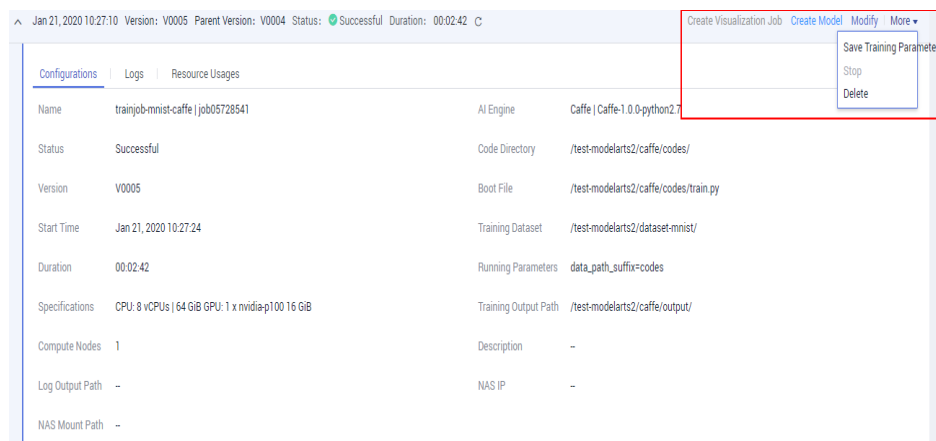
On the **Version Manager** page, ModelArts provides certain shortcut operation buttons for you to quickly enter the subsequent steps after model training is complete.

Table 3-11 Shortcut operation button description

Shortcut Operation Button	Description
Create Visualization Job	Creates a visualization job for the current training version. For details, see Managing Visualization Jobs .

Shortcut Operation Button	Description
Create Model	Creates a model for the current training version. For details, see Creating an AI Application . You can only create models for training jobs in the Running status.
Modify	If the training result of the current version does not meet service requirements or the training job fails, click Modify to switch to the page where you can modify the job parameter settings. For details about the parameters of the training job, see Creating a Training Job . After modifying the job parameter settings as required, click OK to start the training job of a new version.
Save Training Parameters	You can save the job parameter settings of this version as a job parameter configuration, which will be displayed on the Job Parameter Mgmt page. Click More > Save Training Parameters to switch to the Training Parameter page. After confirming that the settings are correct, click OK . For details about training parameter management, see Managing Job Parameters .
Stop	Click More > Stop to stop the training job of the current version. Only training jobs in the Running state can be stopped.
Delete	Click More > Delete to delete the training job of the current version.

Figure 3-11 Shortcut operation buttons



3.6 Viewing Job Details

After a training job finishes, you can manage the training job versions and check whether the training result of the job is satisfactory by viewing the [job details](#) and [Viewing the Evaluation Result](#).

Training Job Details

In the left navigation pane of the ModelArts management console, choose **Training Management > Training Jobs** to switch to the **Training Jobs** page. In the training job list, click a job name to view the job details.

Table 3-12 lists parameters of the training job of each version.

Figure 3-12 Training job details

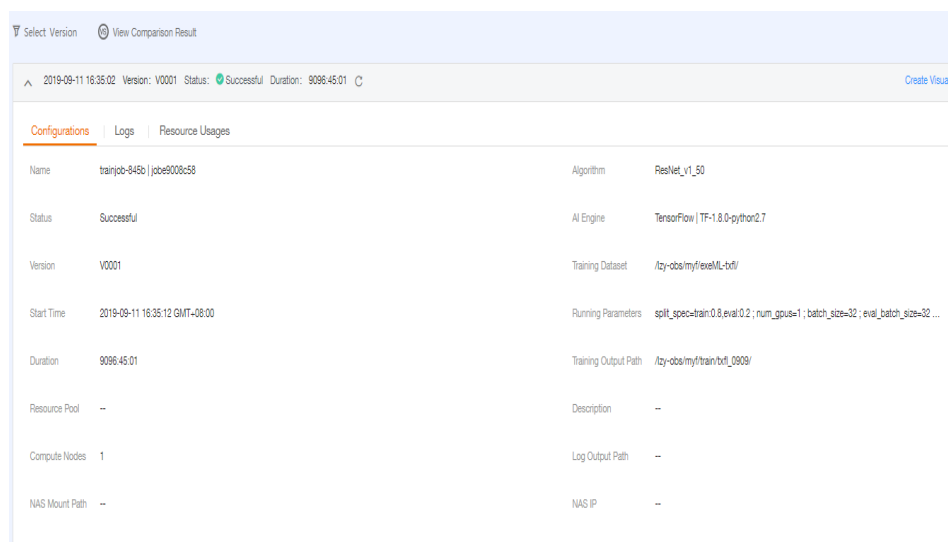


Table 3-12 Training job details

Parameter	Description
Version	Version of a training job, which is automatically defined by the system, for example, V0001 and V0002 .
Status	Status of a training job,
Duration	Running duration of a training job
Configurations	Details about the parameters of the current training job version
Logs	Logs of the current training job version. If you set Log Output Path when creating a training job, you can click the download button on the Logs tab page to download the logs stored in the OBS bucket to the local host.
Resource Usages	Usage of resources of the current training version, including CPUs, GPUs, NPUs, and memory

Viewing the Evaluation Result

When you use a built-in algorithm published by ModelArts to create a training job, you can view the evaluation result of the training job. If the evaluation code is added to the training script based on the ModelArts specifications, you can view

the evaluation result on the job details page after the training job is complete. For details about how to add the evaluation code, see [Adding the Evaluation Code](#).

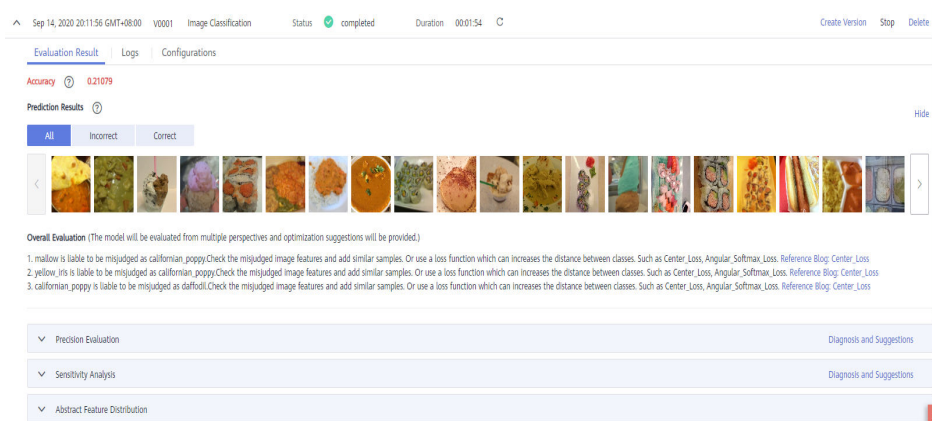
NOTE

You can view evaluation results of models using algorithms such as **Image Classification-ResNet_v1_50**, **Object Detection-FasterRCNN_ResNet50**, and **Object Detection-EfficientDet** built-in algorithms.

After a training job is executed and its status is **Successful**, you can view the evaluation details on the **Evaluation Result** tab page of the job details page. See [Figure 3-13](#).

The evaluation results include the evaluation overview, precision evaluation, sensitivity analysis, and common model indicators. The system automatically provides optimization suggestions based on your model metrics. Read the suggestions and guidance on the page carefully to further optimize your model.

Figure 3-13 Viewing the evaluation result



3.7 Managing Job Parameters

You can store the parameter settings in ModelArts during job creation so that you can use the stored settings to create follow-up training jobs, which makes job creation more efficient.

During the operations of creating, editing, and viewing training jobs, the saved job parameter settings are displayed on the **Job Parameter Mgmt** page.

Using a Job Parameter Configuration

- Method 1: Using a job parameter configuration on the **Job Parameter Mgmt** page

Log in to the ModelArts management console. In the left navigation pane, choose **Training Management > Training Jobs**. On the displayed page, click the **Job Parameter Mgmt** tab. In the job parameter list, click **Creating Training Job** for a job parameter configuration to create a training job based on the job parameter configuration.

- Method 2: Using a job parameter configuration on the **Creating Training Job** page

On the **Creating Training Job** page, click **One-Click Configuration**. In the displayed dialog box, select the required job parameter configuration to quickly create an available training job.

Editing a Job Parameter Configuration

1. Log in to the ModelArts management console. In the left navigation pane, choose **Training Management > Training Jobs**. On the displayed page, click the **Job Parameter Mgmt** tab.
2. In the job parameter configuration list, click **Edit** in the **Operation** column in a row.
3. On the displayed page, modify related parameters by referring to [Creating a Training Job](#) and click **OK** to save the job parameter settings.

In the existing job parameter settings, the job name cannot be changed.

Deleting a Training Job Parameter Configuration

1. Log in to the ModelArts management console. In the left navigation pane, choose **Training Management > Training Jobs**. On the displayed page, click the **Job Parameter Mgmt** tab.
2. In the job parameter list, click **Delete** in the **Operation** column in a row.
3. In the displayed dialog box, click **OK**.

NOTE

Deleted job parameter configurations cannot be recovered. Therefore, exercise caution when performing this operation.

3.8 Adding the Evaluation Code

After a training job is executed, ModelArts automatically evaluates your model and provides optimization diagnosis and suggestions. For details, see [Viewing the Evaluation Result](#).

- When you use a built-in algorithm to create a training job, you can view the evaluation result without any configurations.
- For a training job created by compiling a training script or using a custom image, you need to add the evaluation code to the training code so that you can view the evaluation result and diagnosis suggestions after the training job is complete.

NOTE

1. Only validation sets of the image type are supported.
2. You can add the evaluation code only when the training scripts of the following frequently-used frameworks are used:
 - TF-1.13.1-python3.6
 - TF-2.1.0-python3.6
 - PyTorch-1.4.0-python3.6

This section describes how to use the evaluation code in a training job. To adapt and modify the training code, three steps are involved, [adding the output path](#), [copying the dataset to the local host](#), and [mapping the dataset path to OBS](#).

Adding the Output Path

The code for adding the output path is simple. That is, add a path for storing the evaluation result file to the code, which is called **train_url**, that is, the training output path on the console. Add **train_url** to the analysis function and use **save_path** to obtain **train_url**. The sample code is as follows:

```
FLAGS = tf.app.flags.FLAGS
tf.app.flags.DEFINE_string('model_url', '', 'path to saved model')
tf.app.flags.DEFINE_string('data_url', '', 'path to output files')
tf.app.flags.DEFINE_string('train_url', '', 'path to output files')
tf.app.flags.DEFINE_string('adv_param_json',
                           '{"attack_method":"FGSM","eps":40}',
                           'params for adversarial attacks')
FLAGS(sys.argv, known_only=True)

...

# analyse
res = analyse(
    task_type=task_type,
    pred_list=pred_list,
    label_list=label_list,
    name_list=file_name_list,
    label_map_dict=label_dict,
    save_path=FLAGS.train_url)
```

Copying the Dataset to the Local Host

Copying a dataset to the local host is to prevent the OBS connection from being interrupted due to long-time access. Therefore, copy the dataset to the local host before performing operations.

Datasets can be copied in two modes. Use the OBS path to copy datasets.

- OBS path (recommended)
Call the `copy_parallel` API of MoXing to copy the corresponding OBS path.
- Dataset in ModelArts data management (manifest file format)
Call the `copy_manifest` API of MoXing to copy the file to the local host and obtain the path of the new manifest file. Then, use SDK to parse the new manifest file.

```
if data_path.startswith('obs://'):
    if '.manifest' in data_path:
        new_manifest_path, _ = mox.file.copy_manifest(data_path, '/cache/data/')
        data_path = new_manifest_path
    else:
        mox.file.copy_parallel(data_path, '/cache/data/')
        data_path = '/cache/data/'
    print('----- download dataset success -----')
```

Mapping the Dataset Path to OBS

The actual path of the image file, that is, the OBS path, needs to be entered in the JSON body. Therefore, after analysis and evaluation are performed on the local host, the original local dataset path needs to be mapped to the OBS path, and the new list needs to be sent to the analysis interface.

If the OBS path is used as the input of **data_url**, you only need to replace the character string of the local path.

```
if FLAGS.data_url.startswith('obs://'):
    for idx, item in enumerate(file_name_list):
        file_name_list[idx] = item.replace(data_path, FLAGS.data_url)
```

If the manifest file is used, the original manifest file needs to be parsed again to obtain the list and then the list is sent to the analysis interface.

```
if or FLAGS.data_url.startswith('obs://'):
    if 'manifest' in FLAGS.data_url:
        file_name_list = []
        manifest, _ = get_sample_list(
            manifest_path=FLAGS.data_url, task_type='image_classification')
        for item in manifest:
            if len(item[1]) != 0:
                file_name_list.append(item[0])
```

The sample code for image classification that adapts to a training job is as follows:

```
import json
import logging
import os
import sys
import tempfile

import h5py
import numpy as np
from PIL import Image

import mxing as mx
import tensorflow as tf
from deep_mxing.framework.manifest_api.manifest_api import get_sample_list
from deep_mxing.model_analysis.api import analyse, tmp_save
from deep_mxing.model_analysis.common.constant import TMP_FILE_NAME

logging.basicConfig(level=logging.DEBUG)

FLAGS = tf.app.flags.FLAGS
tf.app.flags.DEFINE_string('model_url', '', 'path to saved model')
tf.app.flags.DEFINE_string('data_url', '', 'path to output files')
tf.app.flags.DEFINE_string('train_url', '', 'path to output files')
tf.app.flags.DEFINE_string('adv_param_json',
    '{"attack_method": "FGSM", "eps": 40}',
    'params for adversarial attacks')
FLAGS(sys.argv, known_only=True)

def _preprocess(data_path):
    img = Image.open(data_path)
    img = img.convert('RGB')
    img = np.asarray(img, dtype=np.float32)
    img = img[np.newaxis, :, :, :]
    return img

def softmax(x):
    x = np.array(x)
    orig_shape = x.shape
    if len(x.shape) > 1:
        # Matrix
        x = np.apply_along_axis(lambda x: np.exp(x - np.max(x)), 1, x)
        denominator = np.apply_along_axis(lambda x: 1.0 / np.sum(x), 1, x)
        if len(denominator.shape) == 1:
            denominator = denominator.reshape((denominator.shape[0], 1))
        x = x * denominator
    else:
        # Vector
        x_max = np.max(x)
        x = x - x_max
        numerator = np.exp(x)
        denominator = 1.0 / np.sum(numerator)
```

```

    x = numerator.dot(denominator)
    assert x.shape == orig_shape
    return x

def get_dataset(data_path, label_map_dict):
    label_list = []
    img_name_list = []
    if 'manifest' in data_path:
        manifest, _ = get_sample_list(
            manifest_path=data_path, task_type='image_classification')
        for item in manifest:
            if len(item[1]) != 0:
                label_list.append(label_map_dict.get(item[1][0]))
                img_name_list.append(item[0])
            else:
                continue
    else:
        label_name_list = os.listdir(data_path)
        label_dict = {}
        for idx, item in enumerate(label_name_list):
            label_dict[str(idx)] = item
            sub_img_list = os.listdir(os.path.join(data_path, item))
            img_name_list += [
                os.path.join(data_path, item, img_name) for img_name in sub_img_list
            ]
            label_list += [label_map_dict.get(item)] * len(sub_img_list)
    return img_name_list, label_list

def deal_ckpt_and_data_with_obs():
    pb_dir = FLAGS.model_url
    data_path = FLAGS.data_url

    if pb_dir.startswith('obs://'):
        mox.file.copy_parallel(pb_dir, '/cache/ckpt/')
        pb_dir = '/cache/ckpt'
        print('----- download success -----')
    if data_path.startswith('obs://'):
        if 'manifest' in data_path:
            new_manifest_path, _ = mox.file.copy_manifest(data_path, '/cache/data/')
            data_path = new_manifest_path
        else:
            mox.file.copy_parallel(data_path, '/cache/data/')
            data_path = '/cache/data/'
        print('----- download dataset success -----')
    assert os.path.isdir(pb_dir), 'Error, pb_dir must be a directory'
    return pb_dir, data_path

def evaluation():
    pb_dir, data_path = deal_ckpt_and_data_with_obs()
    index_file = os.path.join(pb_dir, 'index')
    try:
        label_file = h5py.File(index_file, 'r')
        label_array = label_file['labels_list'][:].tolist()
        label_array = [item.decode('utf-8') for item in label_array]
    except Exception as e:
        logging.warning(e)
        logging.warning('index file is not a h5 file, try json.')
        with open(index_file, 'r') as load_f:
            label_file = json.load(load_f)
            label_array = label_file['labels_list'][:].tolist()
    label_map_dict = {}
    label_dict = {}
    for idx, item in enumerate(label_array):
        label_map_dict[item] = idx
        label_dict[idx] = item
    print(label_map_dict)

```

```

print(label_dict)

data_file_list, label_list = get_dataset(data_path, label_map_dict)

assert len(label_list) > 0, 'missing valid data'
assert None not in label_list, 'dataset and model not match'

pred_list = []
file_name_list = []
img_list = []

for img_path in data_file_list:
    img = _preprocess(img_path)
    img_list.append(img)
    file_name_list.append(img_path)

config = tf.ConfigProto()
config.gpu_options.allow_growth = True
config.gpu_options.visible_device_list = '0'
with tf.Session(graph=tf.Graph(), config=config) as sess:
    meta_graph_def = tf.saved_model.loader.load(
        sess, [tf.saved_model.tag_constants.SERVING], pb_dir)
    signature = meta_graph_def.signature_def
    signature_key = 'predict_object'
    input_key = 'images'
    output_key = 'logits'
    x_tensor_name = signature[signature_key].inputs[input_key].name
    y_tensor_name = signature[signature_key].outputs[output_key].name
    x = sess.graph.get_tensor_by_name(x_tensor_name)
    y = sess.graph.get_tensor_by_name(y_tensor_name)
    for img in img_list:
        pred_output = sess.run([y], {x: img})
        pred_output = softmax(pred_output[0])
        pred_list.append(pred_output[0].tolist())

label_dict = json.dumps(label_dict)
task_type = 'image_classification'

if FLAGS.data_url.startswith('obs://'):
    if 'manifest' in FLAGS.data_url:
        file_name_list = []
        manifest, _ = get_sample_list(
            manifest_path=FLAGS.data_url, task_type='image_classification')
        for item in manifest:
            if len(item[1]) != 0:
                file_name_list.append(item[0])
        for idx, item in enumerate(file_name_list):
            file_name_list[idx] = item.replace(data_path, FLAGS.data_url)
    # analyse
    res = analyse(
        task_type=task_type,
        pred_list=pred_list,
        label_list=label_list,
        name_list=file_name_list,
        label_map_dict=label_dict,
        save_path=FLAGS.train_url)

if __name__ == "__main__":
    evaluation()

```

3.9 Managing Visualization Jobs

You can create visualization jobs of TensorBoard and MindInsight types on ModelArts.

TensorBoard and MindInsight can effectively display the change trend of a training job and the data used in the training.

- TensorBoard
TensorBoard effectively displays the computational graph of TensorFlow in the running process, the trend of all metrics in time, and the data used in the training. TensorBoard supports only the training jobs based on the TensorFlow or MXNet engine. For more information about TensorBoard, see [TensorBoard official website](#).
- MindInsight
MindInsight visualizes information such as scalars, images, computational graphs, and model hyperparameters during training. It also provides functions such as training dashboard, model lineage, data lineage, and performance debugging, helping you train and debug models efficiently. MindInsight supports only the training jobs based on the MindSpore engine. For more information about MindInsight, see [MindSpore official website](#).

You can use the **summary** file generated during model training to create a visualization job.

Prerequisites

To ensure that the **summary** file is generated in the training result, add the related code to the training script.

- Using the TensorFlow engine:
When using the TensorFlow-based MoXing, in **mox.run**, set **save_summary_steps>0** and **summary_verbosity≥1**.
If you want to display other metrics, add tensors to **log_info** in the return value **mox.ModelSpec** of **model_fn**. Only the rank-0 tensors (scalars) are supported. The added tensors are written into the **summary** file. If you want to write tensors of higher ranks in the **summary** file, use the native **tf.summary** of TensorFlow in **model_fn**.
- Using the MindSpore engine:
MindSpore allows you to save data to the **summary** log file and display the data on the GUI. For details, see [Collecting Summary Record](#).
- Using the MXNet engine:
Add the following code to the script:

```
batch_end_callbacks.append(mx.contrib.tensorboard.LogMetricsCallback('OBS path'))
```

Precautions

- You will be charged as long as your visualization jobs are in the **Running** status. We recommend you to stop the visualization jobs when you no longer need them to prevent unnecessary fees. Visualization jobs can be automatically stopped at the specified time. To prevent unnecessary fees, enable this function.
- By default, CPU resources are used to run visualization jobs and cannot be changed to other resources.
- The OBS directory you use and ModelArts are in the same region.

Creating a Visualization Job

1. Log in to the ModelArts management console. In the left navigation pane, choose **Training Jobs**. On the displayed page, click the **Visualization Jobs** tab.
2. In the upper left corner of the visualization job list, click **Create** to switch to the **Create Visualization Job** page.
3. Set **Billing Mode** to **Pay-per-use** and **Job Type** to **TensorBoard** and **MindInsight**. Enter the visualization job name and description as required, set the **Training Output Path** and **Auto Stop** parameters.
 - **Training Output Path**: Select the training output path specified when the training job is created.
 - **Auto Stop**: Enable or disable the auto stop function. A running visualization job will be billed. To prevent unnecessary fees, enable the auto stop function to automatically stop the visualization job at the specified time. The options are **1 hour later**, **2 hours later**, **4 hours later**, **6 hours later**, and **Custom**. If you select **Custom**, you can enter any integer within 1 to 24 hours in the text box on the right.

Figure 3-14 Creating a visualization job

4. Click **Next**.
5. After confirming the specifications, click **Next**.
In the visualization job list, when the status changes to **Running**, the virtualization job has been created. You can click the name of the visualization job to view its details.

Opening a Visualization Job

In the visualization job list, click the name of the target visualization job. The page is displayed. Only the visualization job in the **Running** status can be opened.

Figure 3-15 TensorBoard page

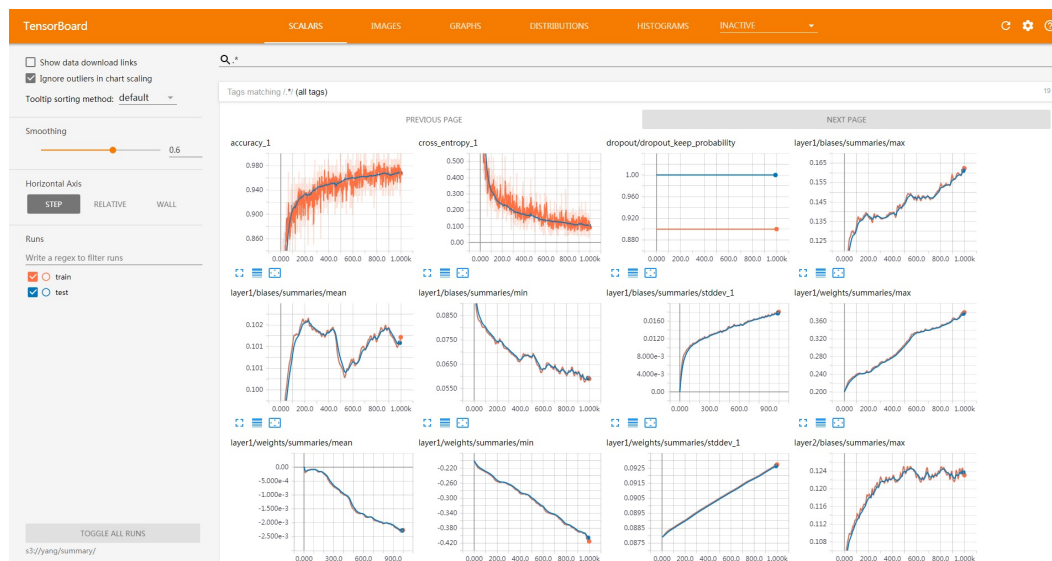
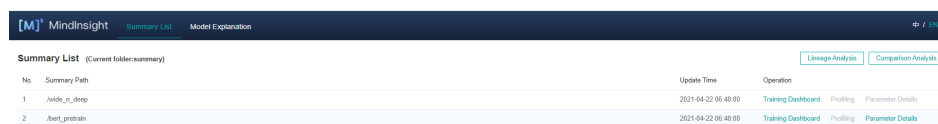


Figure 3-16 MindInsight GUI



Running or Stopping a Visualization Job

- **Stopping a visualization job:** Stop a running visualization job when it is no longer needed to stop billing. In the visualization job list, click **Stop** in the **Operation** column to stop the visualization job.
- **Running a visualization job:** You can run and use a visualization job in the **Canceled** status again. In the visualization job list, click **Run** in the **Operation** column to run the visualization job.

Deleting a Visualization Job

If your visualization job is no longer used, you can delete it to release resources. In the visualization job list, click **Delete** in the **Operation** column to delete the visualization job.

NOTE

A deleted visualized job cannot be recovered. Create a new visualization job if you want to use it. Exercise caution when performing this operation.

4 Resource Pools (Old Version to Be Terminated)

ModelArts Resource Pools

NOTE

New-version dedicated resource pools are now available in ModelArts. Experience the new pools. For more details, see [Resource Pool Management](#).

This section describes only old-version dedicated resource pools, which will be discontinued soon. For details about the differences between the old and new dedicated resource pools, see [Resource Pool](#).

When using ModelArts for full-process AI development, you can use two different resource pools.

- **Public Resource Pool:** provides public large-scale computing clusters, which are allocated based on job parameter settings. Resources are isolated by job. You are charged based on resource specifications, service duration, and the number of instances, regardless of the job type (training job, deployment job, or development job). Public resource pools are provided by ModelArts by default. You do not need to create or configure them separately. During AI development, select a public resource pool.
- **Dedicated Resource Pool:** provides exclusive compute resources, which can be used for notebook instances, training jobs, and model deployment. It delivers higher efficiency and cannot be shared with other users.

Buy a dedicated resource pool and select the dedicated resource pool during AI development. For details about the dedicated resource pool, see the following:

[Dedicated Resource Pool](#)

[Creating a Dedicated Resource Pool](#)

[Scaling a Dedicated Resource Pool](#)

[Deleting a Dedicated Resource Pool](#)

Dedicated Resource Pool

- Dedicated resource pools can be used by notebook instances, training jobs, TensorBoard, or for model deployment.

- Dedicated resource pools are classified into two types: **Dedicated for Development/Training** and **Dedicated for Service Deployment**. The **Dedicated for Development/Training** type can be used only for notebook instances, TensorBoard, and training. The **Dedicated for Service Deployment** type can be used only for AI application deployment.
- Dedicated resource pools are available only when they are in the **Running** state. If a dedicated resource pool is unavailable or abnormal, rectify the fault before using it.
- After a dedicated resource pool is created, the billing starts based on the selected specifications.
- Dedicated resource pools can be billed in pay-per-use or yearly/monthly mode.

Creating a Dedicated Resource Pool

1. Log in to the ModelArts management console and choose **Dedicated Resource Pools** on the left.
2. On the **Dedicated Resource Pools** page, select **Dedicated for Development/Training** or **Dedicated for Service Deployment**.
3. Click **Create** in the upper left corner. The page for creating a dedicated resource pool is displayed.
4. Set the parameters on the page. For details about how to set parameters, see [Table 4-1](#) and [Table 4-2](#).

Table 4-1 Parameters of the **Dedicated for Development/Training** type

Parameter	Description
Resource Type	The default value is Dedicated for Development/Training and cannot be changed.
Billing Mode	Select a billing mode, Yearly/Monthly or Pay-per-use .
Name	Name of a dedicated resource pool. The value can contain letters, digits, hyphens (-), and underscores (_).
Description	Brief description of a dedicated resource pool.
Nodes	Select the number of nodes in a dedicated resource pool. More nodes mean higher computing performance and a higher cost.
Specifications	Required specifications. The GPU delivers better performance, and the CPU is more cost-effective. If a flavor is sold out, you can purchase it again only after other users delete the resource pool.

Parameter	Description
Required Duration	<ul style="list-style-type: none"> Select the time length when you want to use the resource pool. This parameter is mandatory only when the Yearly/Monthly billing mode is selected. The duration ranges from one month to one year. ModelArts provides a 1-year preference package, which allows you to enjoy the product for 1 year by paying for only 10 months. Auto renewal. With auto-renewal, the system automatically renews your product before the product expires.

Table 4-2 Parameters of the **Dedicated for Service Deployment** type

Parameter	Description
Resource Type	The default value is Dedicated for Service Deployment and cannot be changed.
Billing Mode	Select a billing mode, Yearly/Monthly or Pay-per-use . The Yearly/Monthly billing mode is supported only in CN North-Beijing4.
Name	Name of a dedicated resource pool. Enter 4 to 24 characters starting with a lowercase letter and not ending with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed.
Description	Brief description of a dedicated resource pool.
Custom Network Configuration	If you enable Custom Network Configuration , the service instance runs on the specified network and can communicate with other cloud service resource instances on the network. If you do not enable Custom Network Configuration , ModelArts allocates a dedicated network to each user and isolates users from each other. If you enable Custom Network Configuration , set VPC , Subnet , and Security Group . If no network is available, go to the VPC service and create a network.
AZ	You can select Random , AZ 1 , AZ 2 , or AZ 3 based on site requirements. An AZ is a physical region where resources use independent power supplies and networks. AZs are physically isolated but interconnected through an internal network. To enhance workload availability, create nodes in different AZs.
Nodes	Select the number of nodes in a dedicated resource pool. More nodes mean higher computing performance and a higher cost.

Parameter	Description
Specifications	Required specifications. The GPU delivers better performance, and the CPU is more cost-effective.
Required Duration	<ul style="list-style-type: none"> Select the time length when you want to use the resource pool. This parameter is mandatory only when the Yearly/Monthly billing mode is selected. The duration ranges from one month to 11 months. Auto renewal. With auto-renewal, the system automatically renews your product before the product expires.

- After confirming that the specifications are correct, create a dedicated resource pool as prompted. After a dedicated resource pool is created, its status changes to **Running**.

NOTICE

A newly purchased dedicated resource pool can be used only after its development environment is initialized.

(Optional) Interconnecting a VPC

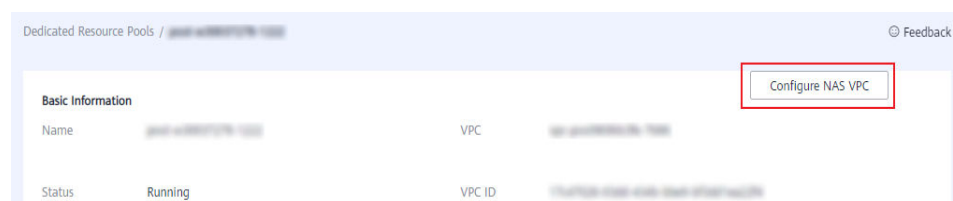
After a resource pool is created, you can interconnect a VPC on the resource pool details page. The procedure is as follows:

NOTE

Only resource pools dedicated for development/training can be interconnected with a VPC.

- On the **Dedicated Resource Pools** page, click the target resource pool.
- On the dedicated resource pool details page, click **Configure NAS VPC**.

Figure 4-1 Configure NAS VPC





- On the **Configure NAS VPC** page, enable **NAS VPC Connection** and set **NAS VPC** and **NAS Subnet**.



Figure 4-2 Configuring NAS VPC

Configure NAS VPC

NAS VPC Connection ON

* NAS VPC   [Create VPC](#)

The VPC must be in the 192.168.XXX.XXX network segment and its subnet cannot overlap 192.168.20.0/24.

* NAS Subnet   [Create Subnet](#)

- If no VPC is available, click **Create VPC** on the right to create a VPC.
- If no subnet is available, click **Create Subnet** on the right to create a subnet.

4. Click **OK**.

Scaling a Dedicated Resource Pool

After a dedicated resource pool is used for a period of time, you can scale out or in the capacity of the resource pool by increasing or decreasing the number of nodes.

The procedure for scaling is as follows:

1. Go to the dedicated resource pool management page, locate the row that contains the desired dedicated resource pool, and click **Scale** in the **Operation** column.
2. On the scaling page, increase or decrease the number of nodes. Increasing the node quantity scales out the resource pool whereas decreasing the node quantity scales in the resource pool. Scale the capacity based on service requirements.
 - During capacity expansion, increase the number of nodes according to the quota of your account. If the number exceeds the quota, the capacity expansion will fail.
 - During capacity reduction, delete the target nodes in the **Operation** column. To reduce one node, switch off the node in **Node List** to delete the node. See [Figure 4-3](#).

CAUTION

Before reducing the capacity of a resource pool for real-time inference, ensure that there are no running instances on the nodes to be released. Otherwise, the real-time services will be interrupted. If you are not sure whether any instance is running on the node to be released, submit a consultation service ticket.

Figure 4-3 Switching off a node to delete the node during scale-in

Name	Status	Published	CPUs	Memory	GPUs	Operation
modelarts-1586432843818	Creating		7.1099997	27940.629 MB	0	<input type="checkbox"/>

3. Click **Submit**. After the request is submitted, the dedicated resource pool management page is displayed.

NOTE

The node is not deleted immediately after the request is submitted. In this case, do not delete nodes from the dedicated resource pool list. Otherwise, the deletion may fail.

You can view the event list on the dedicated resource pool details page. "Begin to delete resource node %s" indicates that the node deletion starts. "Resource node %s deleted" indicates that the node has been deleted in the background.

Switching the Dedicated Resource Pool Billing Mode

- Converting the billing mode of a dedicated resource pool from pay-per-use to yearly/monthly

ModelArts allows you to change the billing mode of a dedicated resource pool to be trained from pay-per-use to yearly/monthly. The prerequisite is that use a pay-per-use dedicated resource pool for more than one consumption. If you do not have any consumption record of using the purchased pay-per-use dedicated resource pool, an error will be reported when you change the billing mode.

To change the billing mode to yearly/monthly in the next subscription period, perform the following steps:

- a. Go to the dedicated resource pool management page, locate the row that contains the desired dedicated resource pool, and click **Change Billing Mode** in the **Operation** column. The page for changing the billing mode from pay-per-use to yearly/monthly is displayed.
- b. Select the yearly/monthly subscription duration as prompted and select whether to automatically renew the subscription. After the payment is complete, the conversion is successful.

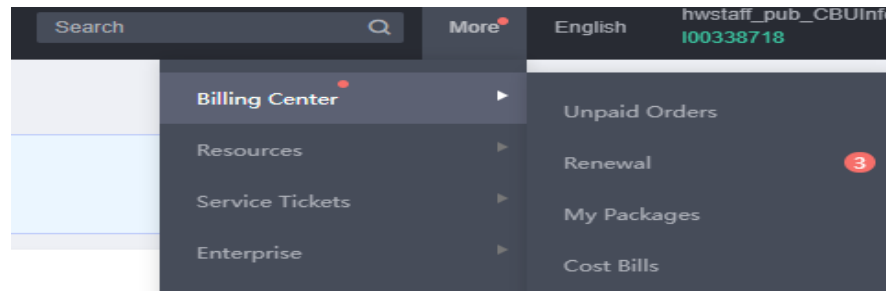
- Converting the billing mode of a dedicated resource pool from yearly/monthly to pay-per-use

For a dedicated resource pool used for training with the billing mode of yearly/monthly, the billing mode can be set to pay-per-use after the yearly/monthly billing period expires.

To change the billing mode to pay-per-use, perform the following steps:

- a. Log in to the ModelArts console and choose **More > Billing Center > Renewal**.

Figure 4-4 Renewal



- b. On the **Renewals** page, locate the row that contains the dedicated resource pool, and click **More** and choose **Change to Pay-per-Use After Expiration** in the **Operation** column.

Deleting a Dedicated Resource Pool

If a dedicated resource pool is no longer needed for AI service development, you can delete the resource pool to release resources.

NOTE

- After a dedicated resource pool is deleted, the training jobs, notebook instances, real-time services, and batch services that depend on the resource pool will become unavailable. A dedicated resource pool cannot be restored after being deleted. Exercise caution when performing this operation.
1. Go to the dedicated resource pool management page and release resources.
 - For a yearly/monthly dedicated resource pool, click **Unsubscribe** in the **Operation** column. After the unsubscription is complete, the resource is automatically deleted.
 - For a pay-per-use dedicated resource pool, click **Delete** in the **Operation** column.
 - For a dedicated resource pool failing to be created, click **Delete** in the **Operation** column to delete it.
 2. In the dialog box that is displayed, click **OK**.

5 Custom Images

5.1 Introduction to Custom Images

ModelArts provides multiple frequently-used built-in engines. However, when users have special requirements for the deep learning engine and development library, the built-in AI engines cannot meet user requirements. ModelArts provides the custom image function to allow users to customize engines.

The bottom layer of ModelArts uses the container technology. Custom images refer to that users create container images and run them on ModelArts. The custom image function supports command line parameters and environment variables in free-text format. The custom images are highly flexible and support the job boot requirements of any computing engine.

Associated Services

Using a custom image may require the following cloud services: Software Repository for Container (SWR), Object Storage Service (OBS), and Elastic Cloud Server (ECS).

- SWR

SWR provides easy, secure, and reliable management over Docker container images throughout their lifecycle, facilitating the deployment of containerized applications. You can upload, download, and manage container images through SWR console, SWR APIs, or community CLI.

Obtain the custom images used by ModelArts for training or creating AI applications from the SWR service management list. Upload the custom images you create to SWR.

Figure 5-1 Obtaining the image list



- **OBS**
OBS is a cloud storage service optimized for storing massive amounts of data. It provides unlimited, secure, and highly reliable storage capabilities at a relatively low cost.
Creating an AI application or creating a training job requires large volumes of data. You can store the data in OBS.
- **ECS**
An ECS is a basic computing component that consists of CPUs, memory, OS, and elastic volume service (EVS). After creating an ECS, you can use it like your local computer or physical server.
You can create a custom image using a local environment or an ECS.

 **NOTE**

When you use a custom image, ModelArts may need to access dependent services, such as SWR and OBS. The custom image can be used only after the access to these dependent services is authorized. It is a good practice to use an agency for authorization. After the agency is configured, the permissions to access dependent services are delegated to ModelArts so that ModelArts can use the dependent services and perform operations on resources on your behalf. For details, see [Configuring Agency Authorization](#).

Application Scenarios of Custom Images

- **For Training Models**
If you have developed a model or training script locally and the AI engine you use is not supported by ModelArts, you can create a custom image based on the basic image packages provided by ModelArts and upload the custom image to SWR. Then, you can use the custom image to create a training job on ModelArts and use the resources provided by ModelArts to train models.
- **For Creating AI Applications**
If you use an AI engine that is not supported by ModelArts to develop a model, you can create a custom image, import the image to ModelArts and use it to create an AI application for unified management, and deploy the AI application as a service.

Process of Creating a Custom Image

1. Use a local host or purchase an ECS to set up the Docker environment.
2. Obtain the basic image from the local environment.
3. Write a Dockerfile based on your requirements to build a custom image. For details about how to efficiently write a Dockerfile, see [Writing a Quality Dockerfile](#).
 - For details about how to use a custom image for a training job, see [Specifications for Custom Images Used for Training Jobs](#).
 - For details about how to use a custom image for creating an AI application, see [Custom Image Specifications for Creating AI Applications](#)
4. After a custom image is created, upload the image to your SWR by referring to [Uploading an Image Through a Container Engine Client](#).

5.2 Creating and Uploading a Custom Image

ModelArts allows you to use custom images to create training jobs and AI applications. Before creating and uploading a custom image, understand the following information:

- Software Repository for Container (SWR)
SWR provides easy, secure, and reliable management over Docker container images throughout their lifecycle, facilitating the deployment of containerized applications. You can push, pull, and manage container images through SWR console, SWR APIs, or community Command Line Interface (CLI).
Obtain the custom images used by ModelArts for training or creating AI applications from the SWR service management list. Upload the custom images you create to SWR.
- Specifications for custom images. For details about how to use a custom image for a training job, see [Specifications for Custom Images Used for Training Jobs](#). For details about how to use a custom image for creating an AI application, see [Custom Image Specifications for Creating AI Applications](#).

Creating and Uploading a Custom Image

1. Use a local host or purchase a HUAWEI CLOUD cloud server to set up the Docker environment.
2. Obtain the basic image from the local environment.
3. Compile a Dockerfile based on your requirements to build a custom image. For details about how to efficiently compile Docker files, see [Software Repository for Container Best Practices](#).
 - For details about how to create a custom image for a training job, see [Example: Creating a Training Job Using a Custom Image](#).
4. After a custom image is created, upload the image to your SWR by referring to [Uploading an Image Through a Container Engine Client](#).

5.3 Using Custom Images to Train Models (Old Version to Be Terminated)

5.3.1 Specifications for Custom Images Used for Training Jobs

NOTE

This section describes how to use a custom image to train a model based on the training module of the old version. The training module of the old version is only available for its existing users. You are advised to use the new version for model training. For details, see [New-Version Training](#).

When creating an image using locally developed models and training scripts, ensure that they meet the specifications defined by ModelArts.

Specifications

- Custom images cannot contain malicious code.
- Part of content in the basic images cannot be changed, including all the files in **/bin**, **/sbin**, **/usr**, and **/lib(64)**, some important configuration files in **/etc**, and the ModelArts tools in **\$HOME**.
- A file cannot be added whose owner is **root** and has permission **setuid** or **setgid**.
- The size of a custom image cannot exceed 9.5 GB.
- To ensure that the log content can be displayed normally, the logs must be standard output.
- The default user of a custom image must be the user whose UID is **1101**.
- Custom images can be developed based on basic ModelArts images. For details about the supported basic images, see [Overview of a Basic Image Package](#).

Overview of a Basic Image Package

To facilitate code download, training log output, and log file upload to OBS, ModelArts provides basic image packages for creating custom images. The basic images provided by ModelArts have the following features:

- Some necessary tools are available in the basic image. You need to create a custom image based on the basic images provided by ModelArts.
- ModelArts continuously updates the basic image versions. For compatible updates, after the basic images are updated, you can still use the old images. For incompatible updates, the custom images created based on the old version cannot run on ModelArts, but the approved custom images can still be used.
- If a custom image fails to be approved and the audit log contains an error message indicating that the basic image does not match, you need to use a new basic image to create an image.

Run the following command to obtain a ModelArts image:

```
docker pull <Address for obtaining a basic image>
```

After customizing an image, upload it to SWR. Make sure that you have created an organization and obtained the password for logging in to SWR. For details, see "Image Management" > "Uploading an Image Through SWR Console" in *Software Repository for Container User Guide*.

```
docker push swr.<region>.myhuaweicloud.com/<Organization to which the target image belongs>/<Image name>
```

Obtain basic images based on chip requirements:

- [CPU-based Basic Images](#)
- [GPU-based Basic Images](#)

CPU-based Basic Images

Address for obtaining a basic image

```
swr.<region>.myhuaweicloud.com/modelarts-job-dev-image/custom-cpu-base:1.3
```

Table 5-1 Optional parameters

Parameter	Optional Value	Description
<region>	<ul style="list-style-type: none"> ap-southeast-1 	Region where the image resides. The possible values are described as follows: <ul style="list-style-type: none"> CN-Hong Kong

Table 5-2 and **Table 5-3** list the components and tools used by basic images.

Table 5-2 Components

Component	Description
run_train.sh	Training boot script. You can download the code directory, run training commands, redirect training log output, and upload log files to OBS after training commands are executed.

Table 5-3 Tool list

Tool	Description
utils.sh	Tool script. The run_train.sh script depends on this script. It provides methods such as SK decryption, code directory download, and log file upload.
ip_mapper.py	Script for obtaining NIC addresses. By default, the IP address of the ib0 NIC is obtained. Training code can use the IP address of the ib0 NIC to accelerate network communications.
dls-downloader.py	OBS download script. The utils.sh script depends on this script.

GPU-based Basic Images

- Image of the CUDA 10.0, 10.1, or 10.2 version, using Ubuntu 18.04 as the basic image and with MoXing pre-installed by default
`swr.<region>.myhuaweicloud.com/modelarts-job-dev-image/custom-base-<cuda version>-<python version>-<os>-<arch>:<image tag>`
- Image of the CUDA 8, 9, or 92 version, with MoXing pre-installed by default
`swr.<region>.myhuaweicloud.com/modelarts-job-dev-image/custom-gpu-<cuda version>-inner-moxing-<python version>:<image tag>`
- Image of the CUDA 8, 9, or 92 version
`swr.<region>.myhuaweicloud.com/modelarts-job-dev-image/custom-gpu-<cuda version>-base:<image tag>`

Table 5-4 Optional parameters

Parameter	Possible Value	Description
<region>	<ul style="list-style-type: none"> ap-southeast-1 	Region where the image resides. The possible values are described as follows: <ul style="list-style-type: none"> CN-Hong Kong
<cuda version>	<ul style="list-style-type: none"> cuda92 cuda9 cuda8 cuda10.0 cuda10.1 cuda10.2 	CUDA version installed in the image NOTE Check the CUDA version. After the version is specified, it cannot be changed. Otherwise, the training will fail.
<image tag>	<ul style="list-style-type: none"> 1.1 1.3 	Image version <ul style="list-style-type: none"> Version 1.3 available for CUDA 8, 9, or 92 version Version 1.1 available for CUDA 10.0, 10.1, or 10.2 version
python version	<ul style="list-style-type: none"> cp27 cp36 	Python environment
os	ubuntu18.04	Operating system
arch	x86	Architecture

[Table 5-2](#) and [Table 5-3](#) list the components and tools used by basic images.

Table 5-5 Components

Component	Description
run_train.sh	Training boot script. You can download the code directory, run training commands, redirect training log output, and upload log files to OBS after training commands are executed.

Table 5-6 Tool list

Tool	Description
utils.sh	Tool script. The run_train.sh script depends on this script. It provides methods such as SK decryption, code directory download, and log file upload.

Tool	Description
ip_mapper.py	Script for obtaining NIC addresses. By default, the IP address of the ib0 NIC is obtained. Training code can use the IP address of the ib0 NIC to accelerate network communications.
dls-downloader.py	OBS download script. The utils.sh script depends on this script.

5.3.2 Creating a Training Job Using a Custom Image (GPU)

NOTE

Training a model based on a custom image applies only to the training module of the old version, which is only available for its existing users. For details about the new-version training, see [Using a Custom Image to Train Models \(New-Version Training\)](#).

After creating and uploading a custom image to SWR, you can use the image to create a training job on the ModelArts management console to complete model training.

Prerequisites

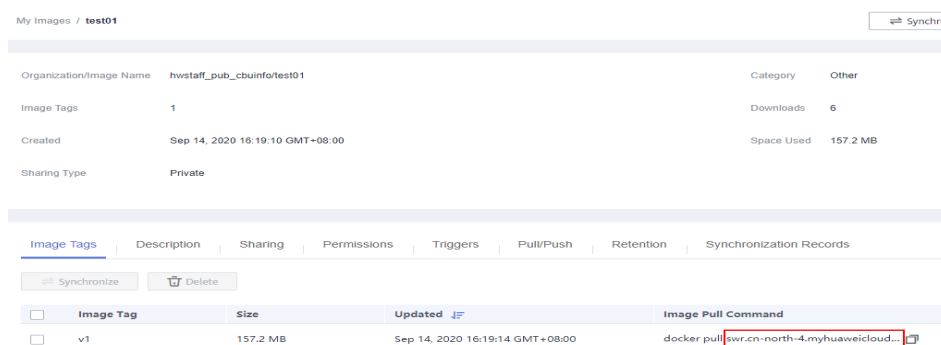
- You have created a custom image package based on ModelArts specifications. For details about the specifications you need to comply with when using a custom image to create training jobs, see [Specifications for Custom Images Used for Training Jobs](#).
- You have uploaded the custom image to SWR. For details, see [Creating and Uploading a Custom Image](#).

Creating a Training Job

Log in to the ModelArts management console and create a training job. When using a custom image to create a job, pay attention to the settings of **Algorithm Source**, **Environment Variable**, and **Resource Pool**.

- Algorithm Source**
Select **Custom**.
 - Image Path:** SWR URL after the image is uploaded to SWR

Figure 5-2 SWR image address



- **Code Directory:** OBS path for storing the training code file.
- **Boot Command:** boot command after the image is started. The basic format is as follows:

```
bash /home/work/run_train.sh {UserCommand}
```

```
bash /home/work/run_train.sh [python/bash/..] {file_location}  
{file_parameter}
```

run_train.sh is the training boot script. After this script is executed, ModelArts recursively downloads all content in the code directory to the local path of the container. The local path is in the format of **/home/work/user-job-dir/\${Name of the last level in the code directory}**.

For example, if the OBS path of the training code file is **obs://obs-bucket/new/train.py** and the code directory is **obs://obs-bucket/new/**, the local path of the container is **/home/work/user-job-dir/new/**. The local training code path of the container is **/home/work/user-job-dir/new/train.py**. Then, you can set the boot command to the following:
bash /home/work/run_train.sh python /home/work/user-job-dir/new/train.py {python_file_parameter}

 **NOTE**

If you create a training job using a custom image, ModelArts allows you to customize the boot command. The following are two basic formats for the boot command:

```
bash /home/work/run_train.sh {UserCommand}
```

```
bash /home/work/run_train.sh [python/bash/..] {file_location}  
{file_parameter}
```

run_train.sh is the training boot script. When creating a custom image, you can implement the training boot script or place the training code in the custom image environment in advance to customize the boot command (in the basic formats or any other formats).

- **Environment Variable**

After the container is started, besides the environment variables added by configuring **Environment Variable** during training job creation, [Table 5-7](#) lists other environment variables to be loaded. You can determine whether to use these environment variables in your own Python training script, or run the **{python_file_parameter}** command to pass the required parameters.

Table 5-7 Optional environment variables

Environment Variable	Description
DLS_TASK_INDEX	Container index, starting from 0.
DLS_TASK_NUMBER	Number of containers, corresponding to Compute Nodes
DLS_APP_URL	Code directory, corresponding to Code Dir with the protocol name added. For example, you can use \$DLS_APP_URL/*.py to read files in OBS.

Environment Variable	Description
DLS_DATA_URL	Dataset path, corresponding to Data Source with the protocol name added
DLS_TRAIN_URL	Training output path, corresponding to Training Output Path with the protocol name added
BATCH_{jobName}.0_HOSTS (standalone)	<p>For standalone training, that is, when the number of compute nodes is 1, the environment variable is BATCH_{jobName}.0_HOSTS.</p> <p>The format of the HOSTS environment variable is hostname:port. A container can view the HOSTS of all containers in the same job, such as BATCH_CUSTOM0_HOSTS and BATCH_CUSTOM1_HOSTS, varying according to the indexes. If the resource pool is a dedicated resource pool with the 8GPU specifications, the network type of the container is a host network, and the host IB network can be used to accelerate communications. If other resource pools are used, the network is a container network.</p> <p>NOTE When the host IB network is used for communication acceleration, the ip_mapper.py tool is required to obtain the IP address of the ib0 NIC for using the IPoIB feature.</p>

- **Resource Pool**

If you select a resource pool of the GPU type, ModelArts mounts NVME SSDs to the **/cache** directory. You can use this directory to store temporary files.

Figure 5-3 Creating a training job

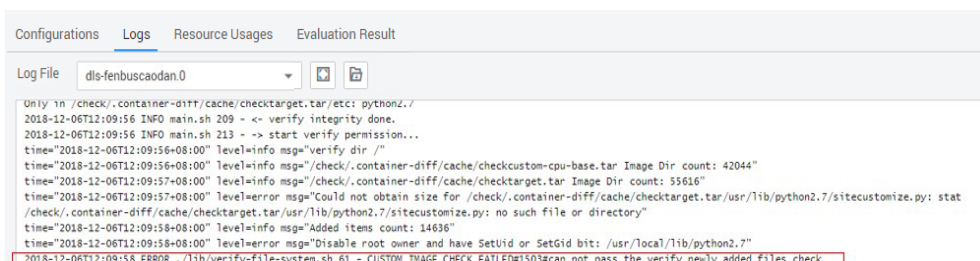
The screenshot displays the 'Import parameters' section of the ModelArts configuration interface. It includes several configuration fields:

- Algorithm Source:** Set to 'Custom'. Below it, a note states: 'A custom image can be used to create the training job. By default, SWR images can be accessed through ModelArts. Create a custom image'. There is a link for 'MoXing Documentation'.
- Image Path:** Text input field containing '/Namespace/Repository:Tag'.
- Code Directory:** Text input field with a 'Select' button.
- Boot Command:** Text input field.
- Data Source:** Two tabs: 'Dataset' (selected) and 'Data path'. Under 'Dataset', there are dropdown menus for 'Dataset' (selected 'dataset-5766 (Image class...') and 'Version' (selected 'V003').
- Training Output Path:** Text input field with a 'Select' button. A note below says: 'We recommend you select an empty directory as the output path.'
- Environment Variable:** A section with a '+' icon and the text 'Add Environment Variable'.
- Job Log Path:** Text input field with 'Select' and 'Clear' buttons. A note below says: 'By default, logs are stored in the service and will be deleted irregularly. Select a path for storing logs.'

Running a Training Job Created Using a Custom Image

After a custom image is uploaded to SWR, ModelArts is authorized to obtain and run the image by default when you create a training job using the custom image. When a custom image is run for the first time, the image is checked first. For details about the check, see [Specifications for Custom Images Used for Training Jobs](#). The check failure cause is outputted in the log, and you can modify the image based on the log.

Figure 5-4 Failed to check the image



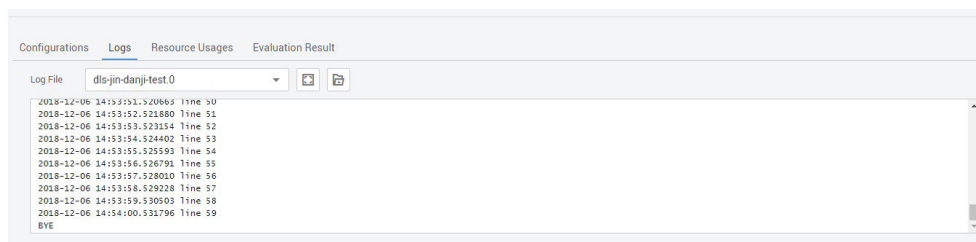
```
Configurations Logs Resource Usages Evaluation Result
Log File dls-fenbuscaodan.0
Unly in /check/.container-diff/cache/checktarget.tar/etc: python2./
2018-12-06T12:09:56 INFO main.sh 209 - <- verify integrity done.
2018-12-06T12:09:56 INFO main.sh 213 - -> start verify permission...
time="2018-12-06T12:09:56+08:00" level=info msg="verify dir ./"
time="2018-12-06T12:09:56+08:00" level=info msg="/check/.container-diff/cache/checkcustom-cpu-base.tar Image Dir count: 42044"
time="2018-12-06T12:09:57+08:00" level=info msg="/check/.container-diff/cache/checktarget.tar Image Dir count: 55616"
time="2018-12-06T12:09:57+08:00" level=error msg="Could not obtain size for /check/.container-diff/cache/checktarget.tar/usr/lib/python2.7/sitecustomize.py: stat
/check/.container-diff/cache/checktarget.tar/usr/lib/python2.7/sitecustomize.py: no such file or directory"
time="2018-12-06T12:09:58+08:00" level=info msg="Added items count: 14636"
time="2018-12-06T12:09:58+08:00" level=error msg="Disable root owner and have Setuid or SetGid bit: /usr/local/lib/python2.7"
2018-12-06T12:09:58 ERROR ./lib/verify-file-system.sh 61 - CUSTOM_IMAGE_CHECK_FAILED#1503#can not pass the verify newly added files check
```

After the image is checked, the backend starts the custom image container to run the training job. You can view the training status based on the log.

NOTE

After an image is reviewed, the image does not need to be reviewed again when being used to create training jobs again.

Figure 5-5 Run log



```
Configurations Logs Resource Usages Evaluation Result
Log File dls-jin-danji-test.0
2018-12-06 14:53:53.520663 line 50
2018-12-06 14:53:52.521880 line 51
2018-12-06 14:53:53.523154 line 52
2018-12-06 14:53:54.524402 line 53
2018-12-06 14:53:55.525599 line 54
2018-12-06 14:53:56.526791 line 55
2018-12-06 14:53:57.528010 line 56
2018-12-06 14:53:58.529228 line 57
2018-12-06 14:53:59.530503 line 58
2018-12-06 14:54:00.531796 line 59
BYE
```

5.3.3 Example: Creating a Training Job Using a Custom Image

NOTE

This section describes how to use a custom image to train a model based on the training module of the old version. The training module of the old version is only available for its existing users. For details about how to use custom images in training of the new version, see [Using a Custom Image to Train Models \(New-Version Training\)](#).

The files required in this example are stored in [GitHub](#). This example uses the MNIST dataset downloaded from the [MNIST official website](#).

- `mnist_softmax.py`: standalone training script

Creating and Uploading a Custom Image

In this example, the Dockerfile file is used to customize an image.

A Linux x86_x64 host is used here. You can purchase an ECS of the same specifications or use an existing local host to create a custom image.

1. Install Docker. For details, see <https://docs.docker.com/engine/install/binaries/#install-static-binaries>.

The following uses the Linux x86_64 OS as an example to describe how to obtain the Docker installation package. Run the following command to install the Docker software:

```
curl -fsSL get.docker.com -o get-docker.sh  
sh get-docker.sh
```

If the **docker images** command is successfully executed, Docker has been installed. In this case, skip this step.

2. Obtain a basic image.

A custom image used for a training job must be compiled based on a basic image. For details about the formats of basic image names, see [Overview of a Basic Image Package](#). Run the following command to obtain a basic image for custom images:

```
docker pull swr.<region>.myhuaweicloud.com/<image org>/<image name>
```

In addition, you can run the **docker images** command to view the local image list.

3. Compile a Dockerfile for building a custom image.

This example uses a TensorFlow 1.13.2 image. The file name is **tf-1.13.2.dockerfile**. Run the **vi tf-1.13.2.dockerfile** command to switch to the Dockerfile.

For details about how to compile Dockerfile, see [Dockerfile Reference](#).

```
FROM swr.cn-north-4.myhuaweicloud.com/modelarts-job-dev-image/custom-base-cuda10.0-cp36-ubuntu18.04-x86:1.1  
# Configure the HUAWEI CLOUD source and install TensorFlow.  
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \  
sed -i "s@http://.*archive.ubuntu.com@http://repo.myhuaweicloud.com@g" /etc/apt/sources.list && \  
sed -i "s@http://.*security.ubuntu.com@http://repo.myhuaweicloud.com@g" /etc/apt/sources.list && \  
pip install --trusted-host https://repo.huaweicloud.com -i https://repo.huaweicloud.com/repository/  
pypi/simple tensorflow==1.13.2  
# Configure environment variables.  
ENV PATH=/root/miniconda3/bin:$PATH
```

4. Create a custom image.

In the following example, the image is in the **cn-north-4** region and belongs to the **deep-learning-diy** organization. Run the following command in the directory where the **tf-1.13.2.dockerfile** file resides:

```
docker build -f tf-1.13.2.dockerfile . -t swr.cn-north-4.myhuaweicloud.com/deep-learning-diy/  
tf-1.13.2:latest
```

5. Push the image to SWR. For details about how to upload an image, see [Software Repository for Container User Guide](#).

The prerequisite is that you have [created an organization](#) and [obtained the SWR login command](#). In the following example, the image is in the **cn-north-4** region and belongs to the **deep-learning-diy** organization. Run the following command to push the image to SWR:

```
docker push swr.cn-north-4.myhuaweicloud.com/deep-learning-diy/tf-1.13.2:latest
```

swr.cn-north-4.myhuaweicloud.com/deep-learning-diy/tf-1.13.2:latest is the SWR URL of the custom image.

Standalone Training

1. Upload training code **mnist_softmax.py** and training data to OBS. Store the code and data in the code root directory so that they can be directly downloaded to the container.

The root directory **obs://deep-learning/new/mnist/** is used as an example.

The training code file is **obs://deep-learning/new/mnist/**.

The data is stored in **obs://deep-learning/new/mnist/mnist_data**.

2. Create a training job using a custom image. Set **Data Storage Location** and **Training Output Path** based on site requirements. Set **Image Path**, **Code Directory**, and **Boot Command** as follows:

- **Image Path**: Enter the SWR URL of the uploaded image.
- **Code Directory**: Enter the OBS path for storing the training code, that is, the code root directory in **1**.

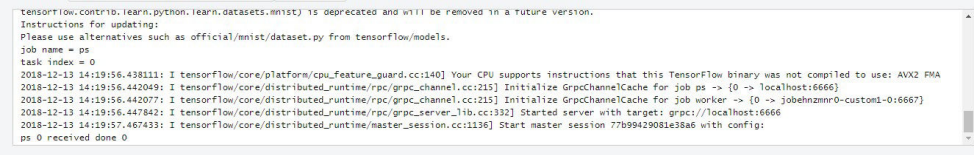
Before a training job is started, ModelArts automatically recursively downloads all content in the code directory to the local path of the container. The local path of the container is **/home/work/user-job-dir/\$ {Last level of the code root directory}**. For example, if **Code Directory** is set to **obs://deep-learning/new/mnist**, the local path is **/home/work/user-job-dir/mnist/**, and the code boot file is **/home/work/user-job-dir/mnist/mnist_softmax.py**.

- **Boot Command**: **bash /home/work/run_train.sh python /home/work/user-job-dir/mnist/mnist_softmax.py --data_url /home/work/user-job-dir/mnist/mnist_data**

/home/work/user-job-dir/mnist/mnist_softmax.py is the code boot file, and **--data_url /home/work/user-job-dir/mnist/mnist_data** is the data storage path.

3. After the training job is created, the code directory is downloaded, the custom image is reviewed, and the training job is completed in the background. Generally, training jobs run for a period of time, which may be several minutes or tens of minutes depending on the amount of data and resources you select. After the program is executed successfully, the log similar to the following is outputted:

Figure 5-6 Run log information



```
tensorflow.contrib.learn.python.learn.datasets.mnist is deprecated and will be removed in a future version.
Instructions for updating:
Please use alternatives such as official/mnist/dataset.py from tensorflow/models.
job name = ps
task index = 0
2018-12-13 14:19:56.438111: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
2018-12-13 14:19:56.442049: I tensorflow/core/distributed_runtime/rpc/grpc_channel.cc:215] Initialize GrpcChannelCache for job ps -> [0 -> localhost:6666]
2018-12-13 14:19:56.442077: I tensorflow/core/distributed_runtime/rpc/grpc_channel.cc:215] Initialize GrpcChannelCache for job worker -> [0 -> jobehzmn0-custom1-0:6667]
2018-12-13 14:19:56.447842: I tensorflow/core/distributed_runtime/rpc/grpc_server_lib.cc:332] Started server with target: grpc://localhost:6666
2018-12-13 14:19:57.467433: I tensorflow/core/distributed_runtime/master_session.cc:1136] Start master session 77b99429081e38a6 with config:
ps 0 received done 0
```

6 Permissions Management

6.1 Creating a User and Granting Permissions

This section describes how to use [IAM](#) to implement fine-grained permissions control for your ModelArts resources. With IAM, you can:

- Create IAM users for employees based on the organizational structure of your enterprise. Each IAM user has their own security credentials, providing access to ModelArts resources.
- Grant only the permissions required for users to perform a task.
- Entrust a HUAWEI CLOUD account or cloud service to perform professional and efficient O&M on your ModelArts resources.

If your HUAWEI CLOUD account does not require individual IAM users, you can skip this section.

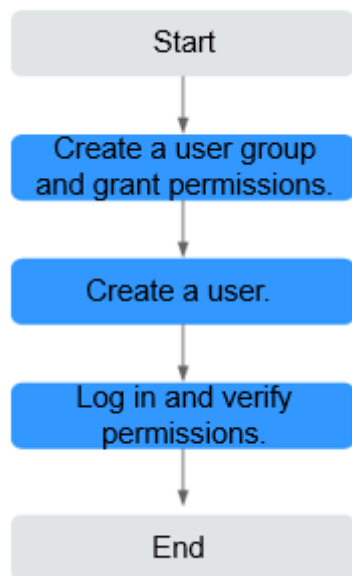
This section describes the procedure for granting permissions (see [Figure 6-1](#)).

Prerequisites

- You have learnt about the permissions supported by ModelArts and understood how to choose policies or roles according to your requirements. For details, see [ModelArts Permissions](#).
- The permissions to use ModelArts depend on OBS authorization. Therefore, you need to grant OBS system permissions to users. For details, see [OBS Permissions](#).
- For the system policies of other services, see [System Permissions](#).

Process Flow

Figure 6-1 Process for granting ModelArts permissions



1. **Create a user group and assign permissions to it.**

Create a user group on the IAM console, and assign the **ModelArts CommonOperations** policy to the group.

The use of ModelArts depends on OBS permissions. Therefore, assign the **Tenant Administrator** policy that takes effect for global services to the user group.

2. **Create a user and add it to a user group.**

Create a user on the IAM console and add the user to the group created in 1.

3. **Log in** and verify permissions.

Log in to the ModelArts console by using the newly created user, and verify that the user only has read permissions for ModelArts.

- Choose **Service List > ModelArts**. On the ModelArts management console, choose **Dedicated Resource Pools > Create**. If the creation fails (assume that the current permission contains only **ModelArts CommonOperations**), the **ModelArts CommonOperations** policy has already taken effect.
- Choose any other service in **Service List**. If a message appears indicating that you have insufficient permissions to access the service (assume that the current permission contains only **ModelArts CommonOperations**), the **ModelArts CommonOperations** policy has already taken effect.
- Choose **Service List > ModelArts**. On the ModelArts management console, choose **Data Management > Datasets > Create Dataset**. If the corresponding OBS path can be accessed, the **Tenant Administrator** policy for global services has already taken effect.

6.2 Creating a Custom Policy

Custom policies can be created as a supplement to the system policies of ModelArts. For the actions that can be added for custom policies, see [ModelArts API Reference > Permissions Policies and Supported Actions](#).

You can create custom policies in either of the following ways:

- Visual editor: Select cloud services, actions, resources, and request conditions. This does not require knowledge of policy syntax.
- JSON: Edit JSON policies from scratch or based on an existing policy.

For details, see [Creating a Custom Policy](#). This section describes [example custom policies of OBS \(a dependent service of ModelArts\)](#) and [ModelArts](#).

Precautions

- The permissions to use ModelArts depend on OBS authorization. Therefore, you need to grant OBS system permissions to users.
- A custom policy can contain actions of multiple services that are globally accessible or accessible through region-specific projects.
- To define permissions required to access both global and project-level services, create two custom policies and specify the scope as **Global services** and **Project-level services**. Then grant the two policies to the users.
- By default, an IAM user created under the current account has all the operation permissions for ModelArts. To control the permissions of an IAM user, for example, to deny a user the permission to use a function, create a custom policy of ModelArts.

Example Custom Policies of OBS

ModelArts is a project-level service, and OBS is a global service. Therefore, you need to create custom policies for the two services respectively and grant them to users. The permissions to use ModelArts depend on OBS authorization. The following example shows the minimum permissions for OBS, including the permissions for OBS buckets and objects. After being granted the minimum permissions for OBS, users can access OBS from ModelArts without restrictions.

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "obs:object:PutObjectAcl",
        "obs:bucket:PutBucketAcl",
        "obs:bucket:PutBucketPolicy",
        "obs:bucket:HeadBucket",
        "obs:bucket:ListAllMyBuckets",
        "obs:bucket:ListBucket",
        "obs:object:DeleteObjectVersion",
        "obs:object:AbortMultipartUpload",
        "obs:object:DeleteObject",
        "obs:object:PutObject",
        "obs:bucket:CreateBucket",
        "obs:object:GetObject",

```

```

        "obs:bucket:GetBucketLocation",
        "obs:object:GetObjectVersionAcl",
        "obs:bucket:GetBucketAcl",
        "obs:object:ListMultipartUploadParts",
        "obs:bucket:ListBucketVersions",
        "obs:object:GetObjectVersion",
        "obs:object:GetObjectAcl",
        "obs:bucket:GetBucketPolicy"
    ]
}
]
}

```

Example Custom Policies of ModelArts

By default, an IAM user created under the current account has all the operation permissions for ModelArts. To control the permissions of an IAM user, for example, to deny a user the permission to use a function, set the parameters as follows:

- Example: Denying ExeML project deletion

A deny policy must be used in conjunction with other policies to take effect. If the permissions assigned to a user contain both Allow and Deny actions, the Deny actions take precedence over the Allow actions.

The following method can be used if you need to assign permissions of the **ModelArts FullAccess** policy to a user but also forbid the user from deleting ExeML projects. Create a custom policy for denying ExeML project deletion, and assign both policies to the group the user belongs to. Then the user can perform all operations on ModelArts except deleting ExeML projects. The following is an example deny policy:

```

{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "modelarts:exeMLProject:delete"
      ]
    }
  ]
}

```

- Example: Denying use of development environment functions

The following is a policy configuration example for this user:

```

{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "modelarts:notebook:list",
        "modelarts:notebook:create",
        "modelarts:notebook:get",
        "modelarts:notebook:update",
        "modelarts:notebook:delete",
        "modelarts:notebook:action",
        "modelarts:notebook:access"
      ]
    }
  ]
}

```

7 Audit Logs

7.1 Key Operations Recorded by CTS

With CTS, you can record operations associated with ModelArts for later query, audit, and backtrack operations.

Prerequisites

CTS has been enabled.

Key Data Management Operations Recorded by CTS

Table 7-1 Key data management operations recorded by CTS

Operation	Resource Type	Trace Name
Creating a dataset	dataset	createDataset
Deleting a dataset	dataset	deleteDataset
Updating a dataset	dataset	updateDataset
Publishing a version of a dataset	dataset	publishDatasetVersion
Deleting a dataset version	dataset	deleteDatasetVersion
Synchronizing the data source	dataset	syncDataSource
Exporting a dataset	dataset	exportDataFromDataset
Creating an auto labeling task	dataset	createAutoLabelingTask
Creating an auto grouping task	dataset	createAutoGroupingTask

Operation	Resource Type	Trace Name
Creating an automatic deployment task	dataset	createAutoDeployTask
Importing samples to a dataset	dataset	importSamplesToDataset
Creating a dataset label	dataset	createLabel
Updating a dataset label	dataset	updateLabel
Deleting a dataset label	dataset	deleteLabel
Deleting a dataset label and its samples	dataset	deleteLabelWithSamples
Adding samples	dataset	uploadSamples
Deleting samples	dataset	deleteSamples
Stopping an auto labeling task	dataset	stopTask
Creating a team labeling task	dataset	createWorkforceTask
Deleting a team labeling task	dataset	deleteWorkforceTask
Starting the acceptance of team labeling	dataset	startWorkforceSampling-Task
Approving/rejecting/canceling acceptance	dataset	updateWorkforceSamplingTask
Submitting sample review comments for acceptance	dataset	acceptSamples
Adding a label to a sample	dataset	updateSamples
Sending an email to labeling team members	dataset	sendEmails
Starting the team labeling task as the contact person	dataset	startWorkforceTask
Updating a team labeling task	dataset	updateWorkforceTask
Adding a label to a team-labeled sample	dataset	updateWorkforceTask-Samples
Reviewing team labeling results	dataset	reviewSamples

Operation	Resource Type	Trace Name
Creating a labeling team member	workforce	createWorker
Updating a labeling team member	workforce	updateWorker
Deleting a labeling team member	workforce	deleteWorker
Batch deleting labeling team members	workforce	batchDeleteWorker
Creating a labeling team	workforce	createWorkforce
Updating a labeling team	workforce	updateWorkforce
Deleting a labeling team	workforce	deleteWorkforce
Automatically creating an IAM agency	IAM	createAgency
Logging in to the labeling console as a labeling team member	labelConsoleWorker	workerLoginLabelConsole
Logging out of the labeling console as a labeling team member	labelConsoleWorker	workerLogoutLabelConsole
Changing the password of the labeling console as a labeling team member	labelConsoleWorker	workerChangePassword
Forgetting the password of the labeling console as a labeling team member	labelConsoleWorker	workerForgetPassword
Resetting the password of the labeling console through the URL as a labeling team member	labelConsoleWorker	workerResetPassword

Key DevEnviron Operations Recorded by CTS

Table 7-2 Key DevEnviron operations recorded by CTS

Operation	Resource Type	Trace Name
Creating a notebook instance	Notebook	createNotebook
Deleting a notebook instance	Notebook	deleteNotebook
Opening a notebook instance	Notebook	openNotebook
Starting a notebook instance	Notebook	startNotebook
Stopping a notebook instance	Notebook	stopNotebook
Updating a notebook instance	Notebook	updateNotebook
Deleting a NotebookApp	NotebookApp	deleteNotebookApp
Switching CodeLab specifications	NotebookApp	updateNotebookApp

Key Training Job Operations Recorded by CTS

Table 7-3 Key training job operations recorded by CTS

Operation	Resource Type	Trace Name
Creating a training job	ModelArtsTrainJob	createModelArtsTrainJob
Creating a version of a training job	ModelArtsTrainJob	createModelArtsTrainVersion
Stopping a training job	ModelArtsTrainJob	stopModelArtsTrainVersion
Modifying the description of a training job	ModelArtsTrainJob	updateModelArtsTrainDesc
Deleting a training job version	ModelArtsTrainJob	deleteModelArtsTrainVersion
Deleting a training job	ModelArtsTrainJob	deleteModelArtsTrainJob
Creating a training job configuration	ModelArtsTrainConfig	createModelArtsTrainConfig

Operation	Resource Type	Trace Name
Modifying a training job configuration	ModelArtsTrainConfig	updateModelArtsTrain-Config
Deleting a training job configuration	ModelArtsTrainConfig	deleteModelArtsTrain-Config
Creating a visualization job	ModelArtsTensorboard-Job	createModelArtsTensorboardJob
Deleting a visualization job	ModelArtsTensorboard-Job	deleteModelArtsTensorboardJob
Modifying the description of a visualization job	ModelArtsTensorboard-Job	updateModelArtsTensorboardDesc
Stopping a visualization job	ModelArtsTensorboard-Job	stopModelArtsTensorboardJob
Restarting a visualization job	ModelArtsTensorboard-Job	restartModelArtsTensorboardJob

Key AI Application Management Operations Recorded by CTS

Table 7-4 Key AI application management operations recorded by CTS

Operation	Resource Type	Trace Name
Creating an AI application	model	addModel
Updating an AI application	model	updateModel
Deleting an AI application	model	deleteModel
Adding a model conversion task	convert	addConvert
Updating a model conversion task	convert	updateConvert
Deleting a model conversion task	convert	deleteConvert

Key Service Management Operations Recorded by CTS

Table 7-5 Key service management operations recorded by CTS

Operation	Resource Type	Trace Name
Deploying a model as a service	service	addService
Deleting a service	service	deleteService
Updating a service	service	updateService
Starting/stopping a service	service	startOrStopService
Adding an access key	service	addAkSk
Deleting an access key	service	deleteAkSk
Creating a dedicated resource pool	cluster	createCluster
Deleting a dedicated resource pool	cluster	deleteCluster
Adding a node to a dedicated resource pool	cluster	addClusterNode
Deleting a node from a dedicated resource pool	cluster	deleteClusterNode
Getting a result from the dedicated resource pool creation	cluster	createClusterResult

Key AI Gallery Operations Recorded by CTS

Table 7-6 Key AI Gallery operations recorded by CTS



Operation	Resource Type	Trace
Publishing an asset	ModelArts_Market	create_content
Modifying asset information	ModelArts_Market	modify_content
Publishing an asset version	ModelArts_Market	add_version
Subscribing to an asset	ModelArts_Market	subscription_content
Removing an asset from favorites	ModelArts_Market	cancel_star_content

Operation	Resource Type	Trace
Liking an asset	ModelArts_Market	like_content
Unliking an asset	ModelArts_Market	cancel_like_content
Publishing an activity	ModelArts_Market	publish_activity
Signing up an activity	ModelArts_Market	regist_activity
Modifying user information	ModelArts_Market	update_user

7.2 Viewing Audit Logs

After CTS is enabled, CTS starts recording operations related to ModelArts. The CTS management console stores the last seven days of operation records. This section describes how to query operation records of the last seven days on the CTS management console.

Procedure

1. Log in to the CTS management console.
2. Click  in the upper left corner of the page and select a region.
3. In the left navigation pane, click **Trace List**.
4. Specify the filter criteria used for querying traces. The following four filter criteria are available:
 - **Trace Source, Resource Type, and Search By**
Select a filter criterion from the drop-down list.
If you select **Trace name** for **Search By**, you need to select a specific trace name.
If you select **Resource ID** for **Search By**, you need to enter a specific resource ID.
If you select **Resource name** for **Search By**, you need to select or enter a specific resource name.
 - **Operator**: Select a specific operator (a user rather than an account).
 - **Trace Status**: Available options include **All trace statuses**, **normal**, **warning**, and **incident**. You can only select one of them.
 - **Time Range**: You can view traces generated during any time range of the last seven days.
5. Click  on the left of a trace to expand its details.
6. Click **View Trace** in the **Operation** column. In the displayed **View Trace** dialog box, the trace structure details are displayed.
For details about the key fields in the CTS trace structure, see [Cloud Trace Service User Guide](#).

A Change History

Released On	Description
2023-03-30	Deleted "DevEnviron (Old Version)". Deleted section "Creating AI Applications" in "Custom Image".
2023-03-01	Moved the content in "Audit Logs" to Audit Logs in "Resource Management". Moved the content in "Permissions Management" to Permissions Management in "Best Practices". Removed "Operation Guide".
2022-12-30	Moved the content in "AI Application Management", "Deploying a Service", "Model Package Specifications", "Model Version", "Examples of Custom Scripts", and "Monitoring" to Inference Deployment .
2022-05-19	Added hitless rolling upgrade for inference services.
2022-01-26	Removed "Hard Example Filtering" in "Deploying a Service".
2022-01-04	Discontinued one-click model deployment in "Data Management".
2021-12-20	Discontinued built-in algorithms in the training management of the old version. Canceled "Evaluating Models" because model evaluation has been changed to a Closed Beta Test (CBT) function.
2021-12-15	Changed "Model Management" to "AI Application Management".
2021-10-21	Added "Differences Between the New and Old Versions of Training".
2021-06-26	Added "Training Management (New Version)".

Released On	Description
2020-01-14	<ul style="list-style-type: none"> Optimized the description for the following section: Creating and Uploading a Custom Image Added the machine learning inference code example and configuration file description in model package specifications. Optimized the PyTorch code example in the customized script.
2020-11-10	<ul style="list-style-type: none"> Optimized the display of the development environment when a notebook is created.
2020-09-21	<ul style="list-style-type: none"> Added the application authentication function for real-time services.
2020-08-06	<ul style="list-style-type: none"> Updated the description of the notebook function, optimized the document structure, and added common JupyterLab operations. Added the dataset of the table type, which supports the DWS data source input. Added guidance on how to integrate real-time service APIs.
2020-07-06	<p>Optimized auto search jobs and added one sample.</p> <p>Added the support for the GitHub code library in the development environment.</p> <p>Added the support for the PyTorch 1.4.0 engine in development environments and model import.</p> <p>Added the sample code of the TensorFlow 2.X custom script.</p>
2020-06-08	<p>Added table dataset and CSV file import and release.</p> <p>Enhanced DevEnviron by introducing JupyterLab and TensorFlow 2.1, integrated multiple basic capabilities of ModelArts, and improved user experience.</p> <p>Optimized auto search job: Added hyperparameter search with fix_norm, data augmentation with adv_aug, and multiple NAS methods such as BETANAS to achieve the optimal precision in mobile setting on ImageNet. Multisearch is available using at least five lines of code.</p> <p>Added TensorFlow 2.1. When creating a notebook instance or using a frequently-used framework to create a training job, TensorFlow 2.1 is ready to use.</p> <ul style="list-style-type: none"> Using Frequently-used Frameworks to Train Models <p>Added the Python 3.7 runtime environment to Model Management.</p>

Released On	Description
2020-03-24	<ul style="list-style-type: none"> ● Optimized the description of training jobs based on different algorithm sources. <ul style="list-style-type: none"> – Using Frequently-used Frameworks to Train Models – Using Custom Images to Train Models ● Added support for team labeling for datasets of the text triplet, text classification, and named entity types, and updated the description of the data labeling function. ● Updated all screenshots in this document because ModelArts UI is upgraded.
2020-01-07	<ul style="list-style-type: none"> ● Hid the old data management module.
2019-12-03	<ul style="list-style-type: none"> ● Added four sections for the model import function based on scenarios. ● Supported team labeling task management for datasets that support team labeling. ● Added the text triplet dataset type. ● Optimized the sections about data management. ● Added the function of managing auto search jobs.
2019-10-17	<ul style="list-style-type: none"> ● Optimized and added functions for data management based on software changes. Updated descriptions in all related sections and added the following content: ● Added sample code of custom scripts (including frequently-used engines). ● Added monitoring description.
2019-09-30	<ul style="list-style-type: none"> ● Enriched and optimized the description of model templates. ● Optimized the description of model package specifications and provided more model package examples.
2019-08-20	Added data management and model templates.
2019-06-13	This is the first official release.