ModelArts

Service Overview

Issue 01

Date 2022-09-30

HUAWEI

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


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 What Is ModelArts?

ModelArts is a one-stop AI development platform geared toward developers and data scientists of all skill levels. It enables you to rapidly build, train, and deploy models anywhere, and manage full-lifecycle AI workflows. ModelArts accelerates AI development and fosters AI innovation with key capabilities, including data preprocessing and auto labeling, distributed training, and automated model building.

ModelArts runs through all phases of AI development, including algorithm development, model training, and model deployment. The underlying technologies of ModelArts support a wide range of heterogeneous computing resources, allowing you to flexibly select and use the resources that fit your needs. ModelArts supports mainstream open-source AI development frameworks such as TensorFlow, PyTorch, and MindSpore. You can also use customized algorithm frameworks tailored to your needs.

ModelArts is designed to make AI development easier and more convenient.

Product Architecture

ModelArts supports the entire development process, including model training, management, and deployment.

ModelArts can be used in scenarios such as image classification and object detection.

Product Advantages

- **One-stop platform**

  The out-of-the-box and full-lifecycle AI development platform provides one-stop training, management, and deployment of models.

- **Easy to use**
  - Automatic optimization of hyperparameters
  - Code-free development and simplified operations

- **High performance**

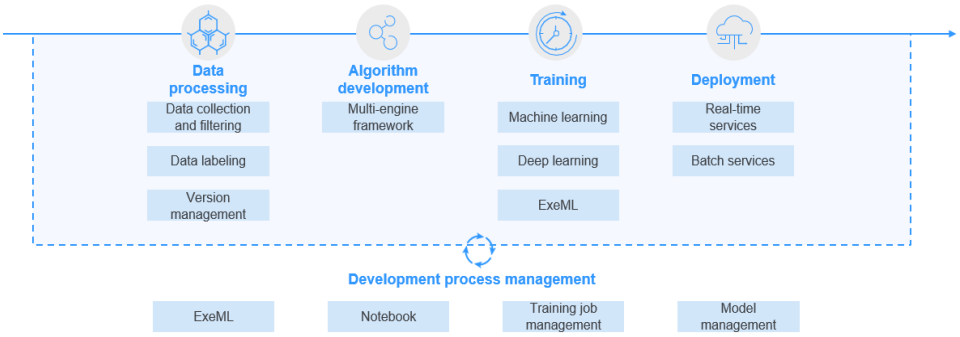
  Huawei MoXing deep learning framework for accelerated algorithm development and training

- **Flexible**
  - Mainstream open-source frameworks such as TensorFlow, PyTorch, and MindSpore
  - Mainstream GPUs
  - Exclusive use of dedicated resources
  - Custom images for custom frameworks and operators

2 Functions

AI engineers face challenges in the installation and configuration of various AI tools and model training. To address these challenges, the one-stop AI development platform ModelArts is provided. The platform integrates data preparation, algorithm development, model training, and model deployment into the production environment, allowing AI engineers to perform one-stop AI development.

Figure 2-1 Function overview



ModelArts has the following features:

Multi-scenario deployment

Models can be deployed in multiple production environments as real-time or batch inference services on the cloud.

3 Basic Knowledge

3.1 Introduction to the AI Development Lifecycle

What Is AI

Artificial intelligence (AI) is a technology capable of simulating human cognition through machines. The core capability of AI is to make a judgment or prediction based on a given input.

What Is the Purpose of AI Development

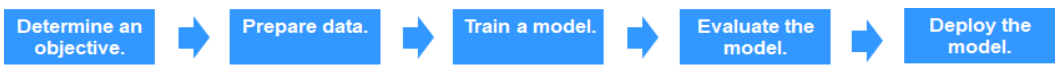
AI development aims to centrally process and extract information from volumes of data to summarize internal patterns of the study objects.

Massive volumes of collected data are computed, analyzed, summarized, and organized by using appropriate statistics, machine learning, and deep learning methods to maximize data value.

Basic Process of AI Development

The basic process of AI development includes the following steps: determining an objective, preparing data, and training, evaluating, and deploying a model.

Figure 3-1 AI development process



Step 1 Determine an objective.

Before starting AI development, determine what to analyze. What problems do you want to solve? What is the business goal? Sort out the AI development framework and ideas based on the business understanding. For example, image classification and object detection. Different projects have different requirements for data and AI development methods.

Step 2 Prepare data.

Data preparation refers to data collection and preprocessing.

Data preparation is the basis of AI development. When you collect and integrate related data based on the determined objective, the most important thing is to ensure the authenticity and reliability of the obtained data. Typically, you cannot collect all the data at the same time. In the data labeling phase, you may find that some data sources are missing and then you may need to repeatedly adjust and optimize the data.

Step 3 Train a model.

Modeling involves analyzing the prepared data to find the causality, internal relationships, and regular patterns, thereby providing references for commercial decision making. After model training, usually one or more machine learning or deep learning models are generated. These models can be applied to new data to obtain predictions and evaluation results.

Step 4 Evaluate the model.

A model generated by training needs to be evaluated. Typically, you cannot obtain a satisfactory model after the first evaluation, and may need to repeatedly adjust algorithm parameters and data to further optimize the model.

Some common metrics, such as the accuracy, recall, and area under the curve (AUC), help you effectively evaluate and obtain a satisfactory model.

Step 5 Deploy the model.

Model development and training are based on existing data (which may be test data). After a satisfactory model is obtained, the model needs to be formally applied to actual data or newly generated data for prediction, evaluation, and visualization. The findings can then be reported to decision makers in an intuitive way, helping them develop the right business strategies.

----End

3.2 Basic Concepts of AI Development

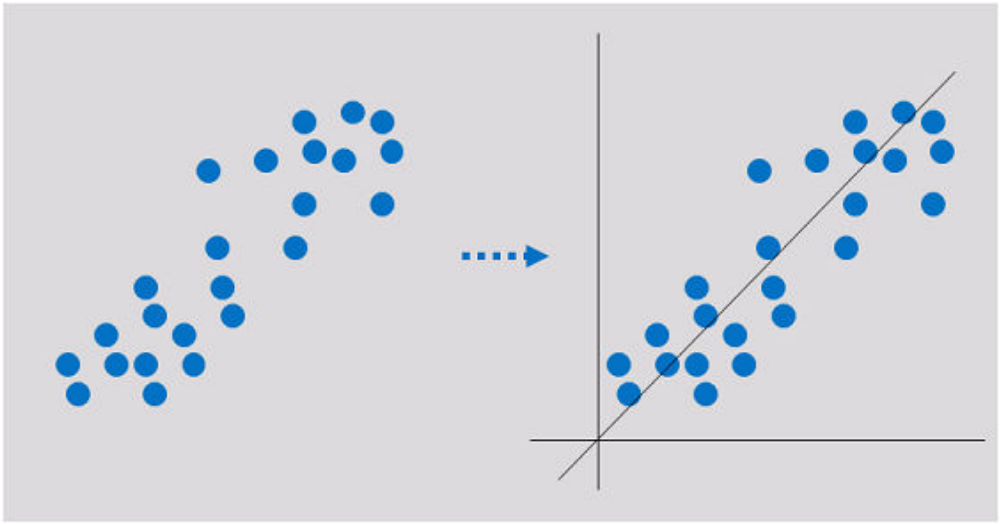
Machine learning is classified into supervised, unsupervised, and reinforcement learning.

- Supervised learning uses labeled samples to adjust the parameters of classifiers to achieve the required performance. It can be considered as learning with a teacher. Common supervised learning includes regression and classification.
- Unsupervised learning is used to find hidden structures in unlabeled data. Clustering is a form of unsupervised learning.

- Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward.

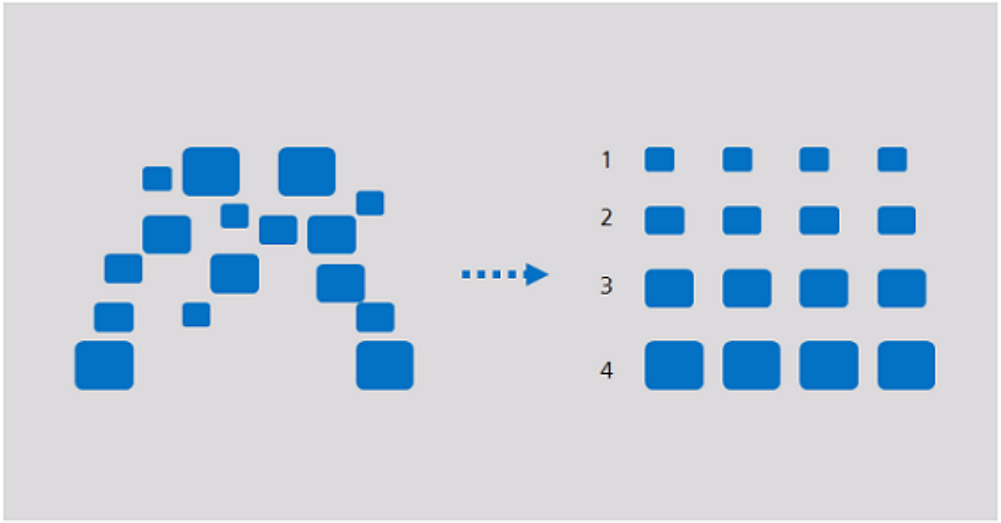
Regression

Regression reflects the time feature of data attributes and generates a function that maps one data attribute to an actual variable prediction to find the dependency between the variable and attribute. Regression mainly analyzes data and predicts data and data relationship. Regression can be used for customer development, retention, customer churn prevention, production lifecycle analysis, sales trend prediction, and targeted promotion.



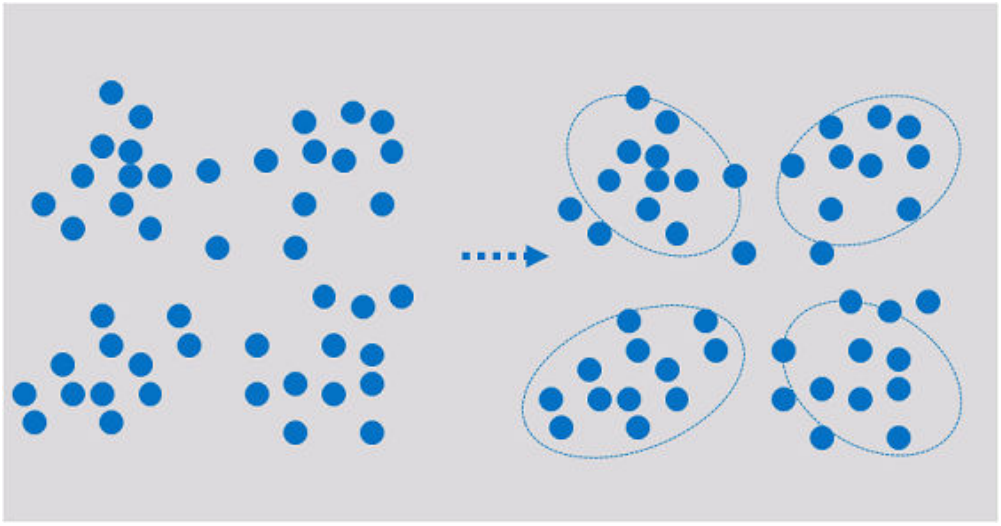
Classification

Classification involves defining a set of categories based on the common features of objects and identifying which category an object belongs to. Classification can be used for customer classification, customer properties, feature analysis, customer satisfaction analysis, and customer purchase trend prediction.



Clustering

Clustering involves grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups. Clustering can be used for customer segmentation, customer characteristic analysis, customer purchase trend prediction, and market segmentation.



Clustering analyzes data objects and produces class labels. Objects are grouped based on the maximized and minimized similarities to form clusters. In this way, objects in the same cluster are more similar to each other than to those in other clusters.

3.3 Common Concepts of ModelArts

Inference

Inference is the process of deriving a new judgment from a known judgment according to a certain strategy. In AI, machines simulate human intelligence, and complete inference based on neural networks.

Real-Time Inference

Real-time inference specifies a web service that provides an inference result for each inference request.

Batch Inference

Batch inference specifies a batch job that processes batch data for inference.

Resource Pool

ModelArts provides large-scale computing clusters for model development, training, and deployment. Both public resource pool and dedicated resource pool are available for you to select. ModelArts provides public resource pools by default. Dedicated resource pools are created separately and used exclusively.

3.4 DevEnviron

NOTE

This document describes the DevEnviron notebook functions of the new version.

Software development is a process of reducing developer costs and improving development experience. In AI development, ModelArts is dedicated to improving AI development experience and simplifying the development process. ModelArts DevEnviron uses cloud native resources and integrates the development tool chain to provide better in-cloud AI development experience for AI development, exploration, and teaching.

ModelArts notebook for seamless in-cloud and on-premises collaboration

- In-cloud JupyterLab, local IDE, and ModelArts plug-ins for remote development and debugging, tailored to your needs
- In-cloud development environment with AI compute resources, cloud storage, and built-in AI engines
- Custom runtime environment saved as an image for training and inference

Feature 1: Remote development, allowing remote access to notebook from a local IDE

The notebook of the new version provides remote development. After enabling remote SSH, you can remotely access the ModelArts notebook development environment to debug and run code from a local IDE.

Due to limited local resources, developers using a local IDE run and debug code typically on a CPU or GPU server shared between team members. Building and maintaining the CPU or GPU server are costly.

ModelArts notebook instances are out of the box with various built-in engines and flavors for you to select. You can use a dedicated container environment. Only after simple configurations, you can remotely access the environment to run and debug code from your local IDE.

ModelArts notebook can be regarded as an extension of a local development environment. The operations such as data reading, training, and file saving are the same as those performed in a local environment.

ModelArts notebook allows you to use in-cloud resources while with local coding habits unchanged.

Feature 2: Preset images that are out-of-the-box with optimized configurations and supporting mainstream AI engines

The AI engines and versions preset in each image are fixed. When creating a notebook instance, specify an AI engine and version, including the chip type.

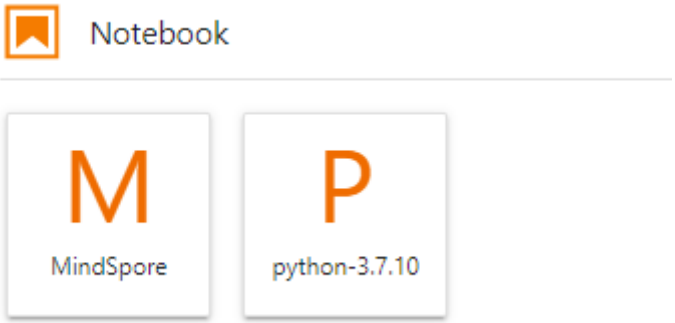
ModelArts DevEnviron provides a group of preset images, including PyTorch, TensorFlow, and MindSpore images. You can use a preset image to start your

notebook instance. After the development in the instance, submit a training job without any adaptation.

The image versions preset in ModelArts are determined based on user feedback and version stability. If your development can be carried out using the versions preset in ModelArts, for example, MindSpore 1.2, use preset images. These images have been fully verified and have many commonly-used installation packages built in. They are out-of-the-box, relieving you from configuring the environment.

The images preset in ModelArts DevEnviron include:

- Common preset packages: common AI engines such as PyTorch and MindSpore based on standard Conda, common data analysis software packages such as Pandas and Numpy, and common tool software such as CUDA and cuDNN, meeting common AI development requirements.
- Preset Conda environments: A Conda environment and basic Conda Python (excluding any AI engine) are created for each preset image. The following figure shows the Conda environment for the preset MindSpore.

  Select a Conda environment based on whether the AI engine is used for debugging.
- Notebook: a web application that enables you to code on the GUI and combine the code, mathematical equations, and visualized content into a document.
- JupyterLab plug-ins: enable flavor changing and instance stopping to improving user experience.
- Remote SSH: allows you to remotely debug a notebook instance from a local PC.

NOTE

- To simplify operations, ModelArts notebook of the new version does not support switchover between AI engines in a notebook instance.
- AI engines vary based on regions. For details about the AI engines available in a region, see the AI engines displayed on the management console.

Feature 3: JupyterLab, an online interactive development and debugging tool

ModelArts integrates open-source JupyterLab for online interactive development and debugging. You can use the notebook on the ModelArts management console to compile and debug code and train models based on the code, without concerning environment installation or configuration.

JupyterLab is an interactive development environment. It is the next-generation product of Jupyter Notebook. JupyterLab enables you to compile notebooks, operate terminals, edit Markdown text, enable interaction, and view CSV files and images.

3.5 Model Training

In addition to data and algorithms, developers spend a lot of time configuring model training parameters. Model training parameters determine the model's precision and convergence time. Parameter selection is heavily dependent on developers' experience. Improper parameter selection will affect the model's precision or significantly increase the time required for model training.

To simplify AI development and improve development efficiency and training performance, ModelArts offers visualized job management, resource management, and version management and automatically performs hyperparameter optimization based on machine learning and reinforcement learning. It provides automatic hyperparameter tuning policies such as learning rate and batch size, and integrates common models.

Currently, when most developers build models, the models usually have dozens of layers or even hundreds of layers and MB-level or GB-level parameters to meet precision requirements. As a result, the specifications of computing resources are extremely high, especially the computing power of hardware resources, memory, and ROM. The resource specifications on the device side are strictly limited. For example, the computing power on the device side is 1 TFLOPS, the memory size is about 2 GB, and the ROM space is about 2 GB, so the model size on the device side must be limited to 100 KB and the inference delay must be limited to 100 milliseconds.

Therefore, compression technologies with lossless or near-lossless model precision, such as pruning, quantization, and knowledge distillation, are used to implement automatic model compression and optimization, and automatic iteration of model compression and retraining to control the loss of model precision. The low-bit quantization technology, which eliminates the need for retraining, converts the model from a high-precision floating point to a fixed-point operation. Multiple compression and optimization technologies are used to meet the lightweight requirements of device and edge hardware resources. The model compression technology reduces the precision by less than 1% in specific scenarios.

When the training data volume is large, the training of the deep learning model is time-consuming. In computer vision technology, ImageNet-1k (a classification dataset containing 1,000 image classes, referred to as ImageNet) is a commonly used dataset. If you use a P100 GPU to train a ResNet-50 model on the dataset, it will take nearly one week. This hinders rapid development of deep learning applications. Therefore, the acceleration of deep learning training has always been an important concern to the academia and the industry.

Distributed training acceleration needs to be considered in terms of software and hardware. A single optimization method cannot meet expectations. Therefore, optimization of distributed acceleration is a system project. The distributed training architecture needs to be considered in terms of hardware and chip design. To minimize compute and communication delays, many factors need to be considered, including overall compute specifications, network bandwidth, high-

speed cache, power consumption, and heat dissipation of the system, and the relationship between compute and communication throughput.

The software design needs to combine high-performance hardware features to fully use the high-speed hardware network and implement high-bandwidth distributed communication and efficient local data caching. By using training optimization algorithms, such as hybrid parallel, gradient compression, and convolution acceleration, the software and hardware of the distributed training system can be efficiently coordinated and optimized from end to end, and training acceleration can be implemented in a distributed environment of multiple hosts and cards. ModelArts delivers an industry-leading speedup of over 0.8 for ResNet50 on the ImageNet dataset in the distributed environment with thousands of hosts and cards.

To measure the acceleration performance of distributed deep learning, the following two key indicators are used:

- Throughput, that is, the amount of data processed in a unit time
- Convergence time, that is, the time required to achieve certain precision

The throughput depends on server hardware (for example, more AI acceleration chips with higher FLOPS processing capabilities and higher communication bandwidth achieve higher throughput), data reading and caching, data preprocessing, model computing (for example, convolution algorithm selection), and communication topology optimization. Except low-bit computing and gradient (or parameter) compression, most technologies improve throughput without affecting model precision. To achieve the shortest convergence time, you need to optimize the throughput and adjust the parameters. If the parameters are not adjusted properly, the throughput cannot be optimized. If the batch size is set to a small value, the parallel performance of model training will be relatively poor. As a result, the throughput cannot be improved even if the number of compute nodes are increased.

Users are most concerned about convergence time. The MoXing framework implements full-stack optimization and significantly reduces the training convergence time. For data read and preprocessing, MoXing uses multi-level concurrent input pipelines to prevent data I/Os from becoming a bottleneck. In terms of model computing, MoXing provides hybrid precision calculation, which combines semi-precision and single-precision for the upper layer models and reduces the loss caused by precision calculation through adaptive scaling. Dynamic hyperparameter policies (such as momentum and batch size) are used to minimize the number of epochs required for model convergence.

ModelArts High-Performance Distributed Training Optimization

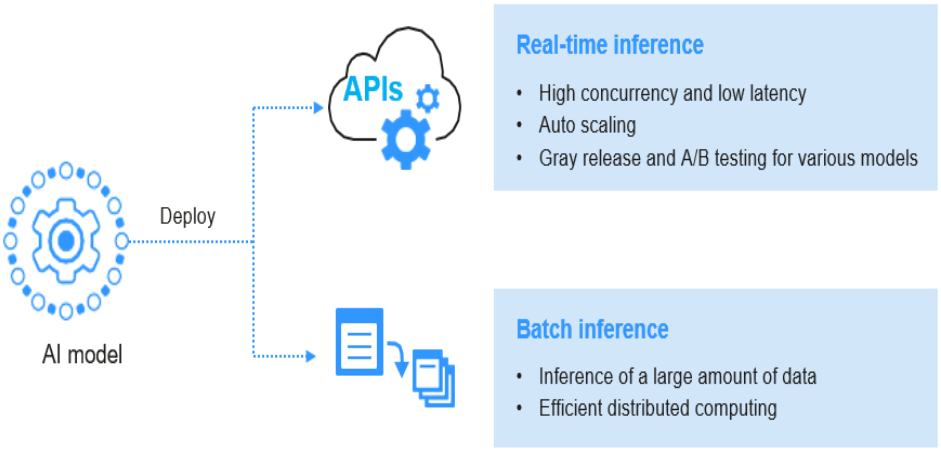
- Automatic hybrid precision to fully utilize hardware computing capabilities
- Dynamic hyperparameter adjustment technologies (dynamic batch size, image size, and momentum)
- Automatic model gradient merging and splitting
- Communication operator scheduling optimization based on BP bubble adaptive computing
- Distributed high-performance communication libraries (NStack and HCCL)
- Distributed data-model hybrid parallel

- Training data compression and multi-level caching

3.6 Model Deployment

AI model deployment and large-scale implementation are typically complex.

Figure 3-2 Process of deploying a model



Real-time inference services feature high concurrency, low latency, and elastic scaling, and support multi-model gray release and A/B testing.

3.7 Resource Pools

Overview

Both public and dedicated resource pools are available for you to select when ModelArts is used for full-process AI development.

- **Public resource pools**: provide large-scale public computing clusters, which are allocated based on job parameter settings. Resources are isolated by job.
- **Dedicated resource pools**: provide dedicated compute resources, which can be used for notebook instances, training jobs, and model deployment. The resources provided in a dedicated resource pool are exclusive, featuring higher resource efficiency than a public resource pool.

  To use a dedicated resource pool, create it and select the dedicated resource pool during AI development. For details about a dedicated resource pool, see the following:

  Dedicated Resource Pool

  Creating a Dedicated Resource Pool

  Resizing a Dedicated Resource Pool

  Deleting a Dedicated Resource Pool

Dedicated Resource Pool

- Dedicated resource pools can be used by notebook instances and training jobs, and for service deployment.

- Dedicated resource pools are classified as the pools **Dedicated for Development/Training** and the pools **Dedicated for Service Deployment**. The resource pools dedicated for development/training can only be used for notebook instances and training jobs. The resource pools dedicated for service deployment can only be used for deploying AI applications.
- Only running dedicated resource pools are available. If a dedicated resource pool is unavailable or abnormal, rectify the fault before using it.

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 a resource pool type.
3. Click **Create** in the upper left corner. The page for creating a dedicated resource pool is displayed.
4. Configure parameters by referring to **Table 3-1** or **Table 3-2**.

Table 3-1 Parameters of a resource pool dedicated for development/training

Parameter	Description
Resource Type	This parameter is not editable.
Name	Name of a dedicated resource pool Only letters, digits, hyphens (-), and underscores (_) are allowed.
Description	Brief description of a dedicated resource pool
Node Flavor	CPU or GPU
Specifications	Node specifications. GPUs offer better performance, and CPUs are more cost-effective. If a flavor is sold out, you can purchase it only after the resources are released by other users in the resource pool.
Nodes	The number of nodes in a dedicated resource pool. A more number of nodes lead to better compute performance.

Table 3-2 Parameters of a resource pool dedicated for service deployment

Parameter	Description
Resource Type	**Dedicated for Service Deployment** by default, which cannot be changed
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

Parameter	Description
Custom Network Configuration	Allows you to customize network configurations. If you enable **Custom Network Configuration**, your instance will run in the specified network and are accessible to other cloud service instances in the network. If you do not enable **Custom Network Configuration**, ModelArts will automatically allocate a dedicated network to each user and all the users are isolated from each other. If you enable **Custom Network Configuration**, configure **VPC**, **Subnet**, and **Security Group**. If no network is available, go to the VPC management console and create one.
Node Flavor	CPU or GPU
Specifications	Node specifications.
Nodes	The number of nodes in a dedicated resource pool. A more number of nodes lead to better compute performance.

5. After confirming the configurations, create the dedicated resource pool. After a dedicated resource pool is created, its status changes to **Running**.

Resizing a Dedicated Resource Pool

After a dedicated resource pool is created, you can resize it by increasing or decreasing the number of nodes to better suit your service needs.

To resize a dedicated resource pool, do as follows:

1. Switch to the dedicated resource pool list, locate the row containing the target dedicated resource pool, and click **Scale** in the **Operation** column.
2. On the **Resize Dedicated Resource Pool** page, increase or decrease the number of nodes to increase or decrease the capacity of the resource pool based on your service requirements.
   - During capacity expansion, add nodes based on service requirements.
   - During capacity reduction, delete the target nodes in the **Operation** column. To delete one node, disable it in the **Operation** column in **Node List**.

     CAUTION

     Before deleting a node from a resource pool dedicated for service deployment, ensure that there are no running instances on the node. Otherwise, the deployed services will be interrupted. If you are not sure whether there is any instance running on a node to be deleted, submit a consultation service ticket.
3. Click **Submit**. Then, you will be redirected to the dedicated resource pool management page.

NOTE

After submitting a request to decrease the capacity of a dedicated resource pool, do not repeatedly perform the operation because the capacity decreasing requires a certain period of time. Repeated deletion may lead to a resizing failure.

After submitting a request to decrease the capacity of a dedicated resource pool, view the event on the **Events** tab of the 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 on the backend.

Deleting a Dedicated Resource Pool

Delete a dedicated resource pool that is not needed to release resources.

NOTE

After a dedicated resource pool is deleted, it cannot be recovered, and the training jobs, notebook instances, real-time services, and batch services that are deployed in the resource pool will become unavailable.

1. Go to the dedicated resource pool management page and click **Delete** in the **Operation** column.
2. In the dialog box that is displayed, enter **DELETE** and click **OK**.

4 Related Services

OBS

ModelArts uses Object Storage Service (OBS) to securely and reliably store data and models at low costs. For more details, see *Object Storage Service Console Operation Guide*.

CCE

ModelArts uses Cloud Container Engine (CCE) to deploy models as real-time services. CCE enables high concurrency and provides elastic scaling. For more details, see *Cloud Container Engine User Guide*.

SWR

To use an AI framework that is not supported by ModelArts, use SoftWare Repository for Container (SWR) to customize an image and import the image to ModelArts for training or inference. For more details, see *Software Repository for Container User Guide*.

Cloud Eye

ModelArts uses Cloud Eye to monitor online services and model loads in real time and send alarms and notifications automatically. For more details, see *Cloud Eye User Guide*.

CTS

ModelArts uses Cloud Trace Service (CTS) to record operations for later query, audit, and backtrack operations. For more details, see *Cloud Trace Service User Guide*.

5 How Do I Access ModelArts?

You can access ModelArts through the web-based management console or by using HTTPS-based application programming interfaces (APIs).

- **Using the Management Console**

  ModelArts offers an easy-to-use management console with various functions such as data management, development environment, model training, AI application management, and service deployment. You can perform end-to-end AI development on the management console.

- **Using SDKs**

  If you want to integrate ModelArts into a third-party system for secondary development, call SDKs to complete the development. ModelArts SDKs encapsulate RESTful APIs provided by ModelArts to simplify secondary development. For details about the SDKs and operations, see *ModelArts SDK Reference*.

  In addition, you can directly call the ModelArts SDKs when writing code in a notebook on the management console.

- **Using APIs**

  If you want to integrate ModelArts into a third-party system for secondary development, use the APIs to access ModelArts. For details about the APIs and operations, see *ModelArts API Reference*.

