LeNet5手写数字识别实验



华为技术有限公司

|  |
| --- |
| 版权所有 © 华为技术有限公司 2022。 保留一切权利。  非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。  商标声明  C:\Users\jwx341670\Desktop\华为标志 Huawei Logo 2018\竖版标志Vertical Version\PNG\HW_POS_RBG_Vertical-150ppi.png 和其他华为商标均为华为技术有限公司的商标。  本文档提及的其他所有商标或注册商标，由各自的所有人拥有。  注意  您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。  由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。 |

|  |  |
| --- | --- |
| 华为技术有限公司 | |
| 地址： | 深圳市龙岗区坂田华为总部办公楼 邮编：518129 |
| 网址： | http://[e](http://e.huawei.com/).huawei.com |

目录

[1 LeNet5手写数字识别实验 2](#_Toc132203510)

[1.1 实验介绍 2](#_Toc132203511)

[1.1.1 实验背景 2](#_Toc132203512)

[1.1.2 实验目的 2](#_Toc132203513)

[1.2 实验环境 2](#_Toc132203514)

[1.3 ECS环境准备 3](#_Toc132203515)

[1.4 项目工程准备 4](#_Toc132203516)

[1.5 实验代码流程 4](#_Toc132203517)

[1.6 开发步骤以及关键代码解析 5](#_Toc132203518)

[1.6.1 运行管理资源申请 6](#_Toc132203519)

[1.6.2 加载模型文件，构建模型输出内存 6](#_Toc132203520)

[1.6.3 读取本地图像数据并进行预处理 7](#_Toc132203521)

[1.6.4 构建模型输入数据，进行模型推理 8](#_Toc132203522)

[1.6.5 模型推理 8](#_Toc132203523)

[1.6.6 解析模型推理结果 9](#_Toc132203524)

[1.6.7 资源释放 9](#_Toc132203525)

[1.7 项目推理 10](#_Toc132203526)

[1.7.1 模型转换 10](#_Toc132203527)

[1.7.2 模型推理 14](#_Toc132203528)

[1.8 实验总结 15](#_Toc132203529)

[1.9 拓展实验 15](#_Toc132203530)

[2 附录环境准备 16](#_Toc132203531)

[2.1 购买ECS服务-Ai1资源——推理用 16](#_Toc132203532)

[2.1.1 获取镜像 16](#_Toc132203533)

[2.1.2 创建ECS弹性云服务器 17](#_Toc132203534)

[2.1.3 MobaXterm连接ECS 21](#_Toc132203535)

[2.1.4 关闭/删除ECS弹性云服务器 23](#_Toc132203536)

# LeNet5手写数字识别实验

## 实验介绍

### 实验背景

LeNet是1998年提出的一种典型的卷积神经网络。它被用于数字识别并取得了巨大的成功。本实验使用昇思MindSpore深度学习框架搭建LeNet网络，使用MNIST数据集，利用华为云上昇腾910资源进行训练，得到LeNet模型；之后将该模型转换成离线模型，部署在华为云ECS弹性云服务器上，使用昇腾310资源进行推理，通过本实验可以体验LeNet网络在手写数字识别的应用。

### 实验目的

本实验使用MindSpore深度学习框架构建LeNet网络，进行模型训练，训练完成后，部署在ECS上进行物体识别实验。通过本实验学员能够：

掌握LeNet网络模型结构。

熟悉Linux 操作命令。

掌握ECS弹性云服务器模型推理的流程。

## 实验环境

ECS模型推理：在华为云ECS的弹性云服务器中执行。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验环境 | 实验平台 | AI 计算框架 | AI处理器/算力 | 软件 |
| ECS推理 | 华为云 ECS | MindSpore1.7 | Ascend 310 | Python3.7.5，  npu-drive 21.0， CANN 5.0.4，  MobaXterm 21.2 |

环境要求

实验平台介绍：

华为云 ECS平台：<https://www.huaweicloud.com/product/ecs.html>

弹性云服务器（Elastic Cloud Server, ECS）是一种云上可随时自助获取、可弹性伸缩的计算服务，可帮助您打造安全、可靠、灵活、高效的应用环境。

AI计算框架介绍：

MindSpore深度学习框架：<https://www.mindspore.cn/mindspore>

昇思MindSpore是一个全场景深度学习框架，旨在实现易开发、高效执行、全场景覆盖三大目标，提供支持异构加速的张量可微编程能力，支持云、服务器、边和端多种硬件平台。

NPU芯片介绍：

Ascend 910：<https://e.huawei.com/cn/products/cloud-computing-dc/atlas/ascend-910>

昇腾910是一款具有超高算力的AI处理器，其最大功耗为310W，华为自研的达芬奇架构大大提升了其能效比。八位整数精度（INT8）下的性能达到640TOPS，16位浮点数（FP16）下的性能达到320 TFLOPS。

Ascend 310：<https://e.huawei.com/cn/products/cloud-computing-dc/atlas/ascend-310>

昇腾310是一款高效、灵活、可编程的AI处理器。基于典型配置，八位整数精度（INT8）下的性能达到22TOPS，16位浮点数（FP16）下的性能达到11 TFLOPS，而其功耗仅为8W。昇腾310芯片采用华为自研的达芬奇架构，集成了丰富的计算单元，在各个领域得到广泛应用。随着全AI业务流程的加速，昇腾310芯片能够使智能系统的性能大幅提升，部署成本大幅降低。

其他资源介绍：

CANN（异构计算架构）：<https://www.hiascend.com/zh/software/cann>

AI场景的异构计算架构，通过提供多层次的编程接口，支持用户快速构建基于昇腾平台的AI应用和业务。

npu-driver（固件与驱动）：<https://www.hiascend.com/hardware/firmware-drivers?tag=community>

使用华为提供的固件包、驱动包完成昇腾设备的基础环境部署。

MobaXterm：<https://mobaxterm.mobatek.net/>

MobaXterm是一款功能强大的远程连接工具。

## ECS环境准备

本实验基于ECS 弹性服务器中完成。请参考本实验手册“2 附录环境准备” 章节，完成ECS 弹性服务器-Ascend310环境搭建。

## 项目工程准备

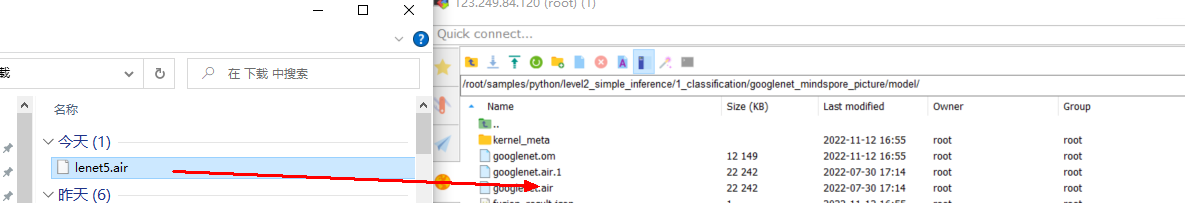
下载项目代码

git clone https://gitee.com/ascend/samples.git

本实验项目代码位于root/samples/python/level2\_simple\_inference/1\_classification/lenet\_mindspore\_picture。

上传模型文件

将[下载模型](#shuq)文件（air）上传到model下。



或者直接下载已有模型

cd samples/python/level2\_simple\_inference/1\_classification/lenet\_mindspore\_picture/model #切换目录

wget <https://obs-9be7.obs.cn-east-2.myhuaweicloud.com/003_Atc_Models/AE/ATC%20Model/lenet/mnist.air> #下载模型

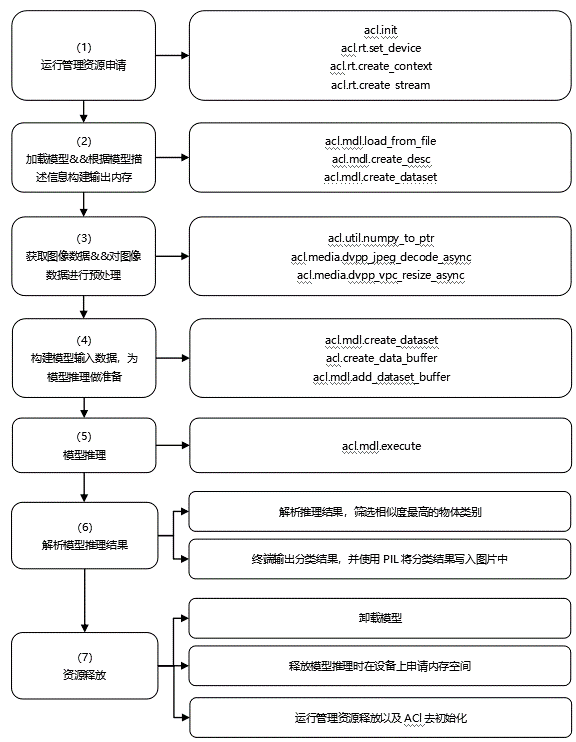
上传测试测试图片

cd ../data

wget https://obs-9be7.obs.cn-east-2.myhuaweicloud.com/models/lenet\_mindspore/test\_image/test1.png

## 实验代码流程

实验流程图如下所示：



代码流程图

## 开发步骤以及关键代码解析

将模型部署的步骤一般是这样的：

首先，模型转换，得到在昇腾AI处理器上能够跑起来的离线模型，经过上面的实验准备环节，已经得到了om模型。

然后，基于ACL接口进行应用开发。

### 运行管理资源申请

函数源码如下所示，ACL相关功能接口的调用已在函数中进行了说明：

def \_init\_resource(self):

print("[Sample] init resource stage:")

ret = acl.init()

check\_ret("acl.rt.set\_device", ret)

ret = acl.rt.set\_device(self.device\_id)

check\_ret("acl.rt.set\_device", ret)

self.context, ret = acl.rt.create\_context(self.device\_id)

check\_ret("acl.rt.create\_context", ret)

self.stream, ret = acl.rt.create\_stream()

check\_ret("acl.rt.create\_stream", ret)

self.run\_mode, ret = acl.rt.get\_run\_mode()

check\_ret("acl.rt.get\_run\_mode", ret)

print("Init resource stage success")

### 加载模型文件，构建模型输出内存

加载本地om模型文件到内存中，并创建并获取模型的描述信息。

函数定义及相关源码注释如下所示：

def init\_resource(self):

print("[Model] class Model init resource stage:")

self.model\_id, ret = acl.mdl.load\_from\_file(self.model\_path)

check\_ret("acl.mdl.load\_from\_file", ret)

self.model\_desc = acl.mdl.create\_desc()

ret = acl.mdl.get\_desc(self.model\_desc, self.model\_id)

check\_ret("acl.mdl.get\_desc", ret)

output\_size = acl.mdl.get\_num\_outputs(self.model\_desc)

self.\_gen\_output\_dataset(output\_size)

print("[Model] class Model init resource stage success")

self.\_get\_output\_info(output\_size)

return SUCCESS

根据模型的描述信息，获取模型的每路输出数据在设备上所需的空间大小。

ACL库内置数据类型说明：aclmdlDataset主要用于描述模型推理时的输入数据或输出数据，模型可能存在多个输入、多个输出，每个输入或输出的内存地址、内存大小用aclDataBuffer类型的数据来描述。

函数定义及相关源码注释如下所示：

def \_gen\_output\_dataset(self, size):

print("[Model] create model output dataset:")

dataset = acl.mdl.create\_dataset()

for i in range(size):

temp\_buffer\_size = acl.mdl.\

get\_output\_size\_by\_index(self.model\_desc, i)

temp\_buffer, ret = acl.rt.malloc(temp\_buffer\_size,

ACL\_MEM\_MALLOC\_NORMAL\_ONLY)

check\_ret("acl.rt.malloc", ret)

dataset\_buffer = acl.create\_data\_buffer(temp\_buffer,

temp\_buffer\_size)

\_, ret = acl.mdl.add\_dataset\_buffer(dataset, dataset\_buffer)

if ret:

acl.destroy\_data\_buffer(dataset)

check\_ret("acl.destroy\_data\_buffer", ret)

self.output\_dataset = dataset

print("[Model] create model output dataset success")

### 读取本地图像数据并进行预处理

读取本地图像数据后，使用DVPP的JPEGD功能将图片数据解码为YUV420SP并使用DVPP的resize功能对图像数据进行缩放至模型要求的输入图像分辨率；将读取到的图像数据拷贝至设备侧申请的内存空间中，为接下来构建模型输入数据做好准备。

函数定义及相关源码注释如下所示：

def jpegd(self, image):

print('[Dvpp] vpc decode stage:')

device\_image = image.copy\_to\_device(self.\_run\_mode)

output\_desc, out\_buffer = self.\_gen\_jpegd\_out\_pic\_desc(image)

ret = acl.media.dvpp\_jpeg\_decode\_async(self.\_dvpp\_channel\_desc,

device\_image.data(),

image.size,

output\_desc,

self.\_stream)

if ret != ACL\_ERROR\_NONE:

print("dvpp\_jpeg\_decode\_async failed ret={}".format(ret))

return None

ret = acl.rt.synchronize\_stream(self.\_stream)

if ret != ACL\_ERROR\_NONE:

print("dvpp\_jpeg\_decode\_async failed ret={}".format(ret))

return None

width = align\_up16(image.width)

height = align\_up2(image.height)

return AclImage(out\_buffer, width, height, yuv420sp\_size(width, height))

函数定义及相关源码注释如下所示：

def resize(self, image, resize\_width, resize\_height):

print('[Dvpp] vpc resize stage:')

input\_desc = self.\_gen\_input\_pic\_desc(image)

output\_desc, out\_buffer, out\_buffer\_size = \

self.\_gen\_resize\_out\_pic\_desc(resize\_width, resize\_height)

ret = acl.media.dvpp\_vpc\_resize\_async(self.\_dvpp\_channel\_desc,

input\_desc,

output\_desc,

self.\_resize\_config,

self.\_stream)

check\_ret("acl.media.dvpp\_vpc\_resize\_async", ret)

ret = acl.rt.synchronize\_stream(self.\_stream)

check\_ret("acl.rt.synchronize\_stream", ret)

print('[Dvpp] vpc resize stage success')

stride\_width = align\_up16(image.width)

stride\_height = align\_up2(image.height)

return AclImage(out\_buffer, stride\_width,

stride\_height, out\_buffer\_size)

### 构建模型输入数据，进行模型推理

函数定义及相关源码注释如下所示：

def \_gen\_input\_dataset(self, data, data\_size):

self.input\_dataset = acl.mdl.create\_dataset()

input\_dataset\_buffer = acl.create\_data\_buffer(data, data\_size)

\_, ret = acl.mdl.add\_dataset\_buffer(

self.input\_dataset,

input\_dataset\_buffer)

if ret:

ret = acl.destroy\_data\_buffer(self.input\_dataset)

check\_ret("acl.destroy\_data\_buffer", ret)

### 模型推理

根据已经加载到内存中，要进行推理的模型ID、构建好的模型推理输入数据，调用ACL库中模型推理接口进行模型推理。

函数定义及相关源码注释如下所示：

def execute(self, data, data\_size):

self.\_gen\_input\_dataset(data, data\_size)

ret = acl.mdl.execute(self.model\_id,

self.input\_dataset,

self.output\_dataset)

check\_ret("acl.mdl.execute", ret)

return self.\_output\_dataset\_to\_numpy()

### 解析模型推理结果

模型的输出为10个float型数据，10个float型数据代表10个类别，每个float型数据为归一化数据范围0 ~ 1。代表该类识别的置信度，选取其中置信度最高的类别，作为分类结果。

函数定义及相关源码注释如下所示：

def postprocess(infer\_output, image\_file):

"""

post\_process

"""

print("post process")

data = infer\_output[0]

vals = data.flatten()

max\_val=np.max(vals)

vals = np.exp(vals - max\_val)

sum\_val = np.sum(vals)

vals /= sum\_val

top\_k = vals.argsort()[-1:-6:-1]

print("images:{}".format(image\_file))

print("======== top5 inference results: =============")

for n in top\_k:

object\_class = image\_net\_classes.get\_image\_net\_class(n)

print("label:%d confidence: %f, class: %s" % (n, vals[n], object\_class))

(filepath, tempfilename) = os.path.split(image\_file)

(filename, extension) = os.path.splitext(tempfilename)

output\_path = os.path.join(os.path.join(SRC\_PATH, "../outputs"), filename + ".txt")

with open(output\_path, "w", encoding="utf-8") as fp:

fp.write(image\_net\_classes.get\_image\_net\_class(top\_k[0]))

### 资源释放

推理接收后要卸载模型，并释放与模型推理相关的资源。

函数定义及相关源码注释如下所示：

def \_\_del\_\_(self):

self.\_release\_dataset()

if self.model\_id:

ret = acl.mdl.unload(self.model\_id)

check\_ret("acl.mdl.unload", ret)

if self.model\_desc:

ret = acl.mdl.destroy\_desc(self.model\_desc)

check\_ret("acl.mdl.destroy\_desc", ret)

print("Model release source success")

函数定义及相关源码注释如下所示：

# 释放存放图像数据的内存空间

def destroy(self):

if self.\_memory\_type == MEMORY\_DEVICE:

acl.rt.free(self.\_data)

elif self.\_memory\_type == MEMORY\_HOST:

acl.rt.free\_host(self.\_data)

elif self.\_memory\_type == MEMORY\_DVPP:

acl.rt.free\_dvpp(self.\_data)

函数定义及相关源码注释如下所示：

def \_\_del\_\_(self):

if self.\_model:

del self.\_model

if self.\_dvpp:

del self.\_dvpp

if self.stream:

acl.rt.destroy\_stream(self.stream)

if self.context:

acl.rt.destroy\_context(self.context)

acl.rt.reset\_device(self.device\_id)

acl.finalize()

print("[Sample] class Samle release source success")

## 项目推理

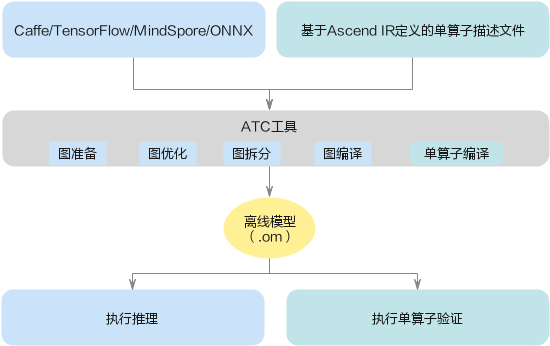
### 模型转换

**转换工具--ATC工具简介**

ATC即模型和算子编译器。用于离线模型转换、自定义算子开发、IR构图开发等场景。

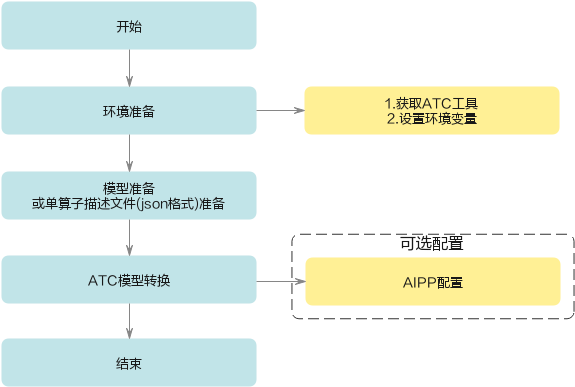
用户可以将开源框架网络模型或Ascend IR定义的单算子描述文件（json格式）通过ATC工具转换成适配昇腾AI处理器的离线模型。

ATC工具功能架构：



ATC工具功能架构

ATC工具运行流程：



ATC工具运行流程

ATC工具总体约束：

在进行模型转换前，请务必查看如下约束要求：

如果要将FasterRCNN等网络模型转成适配昇腾AI处理器的离线模型，则务必参见[定制网络修改（Caffe）](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0100.html)先修改prototxt模型文件。

支持原始框架类型为Caffe、TensorFlow、MindSpore、ONNX的模型转换，当原始框架类型为Caffe、MindSpore、ONNX时，输入数据类型为FP32、FP16（通过设置入参--input\_fp16\_nodes实现，MindSpore框架不支持该参数）、UINT8（通过配置数据预处理实现）；当原始框架类型为TensorFlow时，输入数据类型为FP16、FP32、UINT8、INT32、INT64、BOOL（原始框架类型为TensorFlow时，不支持输入输出数据类型为INT64，需要用户自行将INT64的数据类型修改为INT32类型）。

当原始框架类型为Caffe时，模型文件（.prototxt）和权重文件（.caffemodel）的op name、op type必须保持名称一致（包括大小写）。

当原始框架类型为TensorFlow时，只支持FrozenGraphDef格式。

不支持动态shape的输入，例如：NHWC输入为[?,?,?,3]多个维度可任意指定数值。模型转换时需指定固定数值。

对于Caffe框架网络模型：输入数据最大支持四维，转维算子（reshape、expanddim等）不能输出五维。

模型中的所有层算子除const算子外，输入和输出需要满足dim!=0。

只支持算子规格参考中的算子，并需满足算子限制条件。

ATC参数概览：

须知：

如果通过atc --help命令查询出的参数未解释在表1，则说明该参数预留或适用于其他芯片版本，用户无需关注。

使用atc命令进行模型转换转换时，命令有两种方式，用户根据实际情况进行选择：

* + atc param1=value1 param2=value2 ...（value值前面不能有空格，否则会导致截断，param取的value值为空）
  + atc param1 value1 param2 value2 ...

参数是否必选以--mode为0和3为准。

ATC参数

|  |  |  |  |
| --- | --- | --- | --- |
| ATC参数名称 | 参数简述（具体说明见参数描述章节） | 是否必选 | 默认值 |
| [--help或--h](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0040.html) | 显示帮助信息。 | 否 | 不涉及 |
| [--mode](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0041.html) | 运行模式。 | 否 | 0 |
| [--model](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0043.html) | 原始模型文件路径与文件名。 | 是 | 不涉及 |
| [--weight](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0044.html) | 权重文件路径与文件名。 | 否 | 不涉及 |
| [--om](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0045.html) | 需要转换为json格式的离线模型或原始模型文件的路径和文件名。 | 否 | 不涉及 |
| [--framework](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0046.html) | 原始框架类型。 | 是 | 不涉及 |
| [--input\_format](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0047.html) | 输入数据格式。 | 否 | Caffe默认为NCHW； |
| TensorFlow默认为NHWC |
| [--input\_shape](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0048.html) | 模型输入数据的shape。 | 否 | 不涉及 |
| [--input\_shape\_range](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0049.html) | 指定模型输入数据的shape范围。 | 否 | 不涉及 |
| [--dynamic\_batch\_size](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0050.html) | 设置动态batch档位参数，适用于执行推理时，每次处理图片数量不固定的场景。 | 否 | 不涉及 |
| [--dynamic\_image\_size](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0051.html) | 设置输入图片的动态分辨率参数。适用于执行推理时，每次处理图片宽和高不固定的场景。 | 否 | 不涉及 |
| [--dynamic\_dims](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0052.html) | 设置ND格式下动态维度的档位。适用于执行推理时，每次处理任意维度的场景。 | 否 | 不涉及 |
| [--singleop](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0053.html) | 单算子定义文件，将单个算子Json文件转换成适配昇腾AI处理器的离线模型。 | 否 | 不涉及 |
| [--output](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0055.html) | 如果是开源框架的网络模型，存放转换后的离线模型的路径以及文件名。 | 是 | 不涉及 |
| 如果是单算子描述文件，存放转换后的单算子模型的路径。 |
| [--output\_type](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0056.html) | 指定网络输出数据类型或指定某个输出节点的输出类型。 | 否 | 不涉及 |
| [--check\_report](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0057.html) | 预检结果保存文件路径和文件名。 | 否 | check\_result.json |
| [--json](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0058.html) | 离线模型或原始模型文件转换为json格式文件的路径和文件名。 | 否 | 不涉及 |
| [--soc\_version](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0060.html) | 模型转换时指定芯片版本。 | 是 | 不涉及 |
| [--aicore\_num](https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0062.html) | 设置模型编译时使用的aicore数目。 | 否 | 默认值为最大值 |

有关更多ATC工具的初级功能和高级功能请参考：

初级功能：<https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0007.html>

高级功能：<https://support.huaweicloud.com/atctool-cann502alpha3infer/atlasatc_16_0013.html>

python进入项目文件lenet\_mindspore\_picture/model内

model文件夹内有网络模型lenet.air，我们可以调用ATC模型转换工具将其转换成om模型。

cd /root/samples/python/level2\_simple\_inference/1\_classification/lenet\_mindspore\_picture/model/

执行模型转换命令

atc --model=./mnist.air --framework=1 --output=./mnist --soc\_version=Ascend310

输出“ATC run success”即代表ATC模型转换成功。

### 模型推理

切换到src文件夹内

cd ../src

执行模型推理代码

classify.py为模型推理代码，对data文件夹内的图片进行推理。

python3 classify.py ../data/

输出

init resource stage:

Init resource success

Init model resource start...

[Model] create model output dataset:

malloc output 0, size 40

Create model output dataset success

Init model resource success

it is not a picture, .keep, ignore this file and continue,

(32, 32)

post process

images:test1.png

======== top5 inference results: =============

label:1 confidence: 0.999887, class: 1

label:4 confidence: 0.000043, class: 4

label:8 confidence: 0.000022, class: 8

label:9 confidence: 0.000017, class: 9

label:7 confidence: 0.000013, class: 7

acl resource release all resource

Model release source success

acl resource release stream

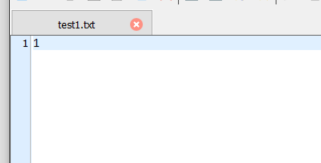
acl resource release context

Reset acl device 0

Release acl resource success

查看推理结果

推理完成后在/root/samples/python/level2\_simple\_inference/1\_classification/enet\_mindspore\_picture/outputs文件夹内得到推理结果。



关闭/删除ECS弹性云服务器

实验完成之后，请及时关闭/删除华为云ECS弹性云服务器，避免产生不必要的资源浪费。

## 实验总结

本实验将训练环节获得的模型转换成离线模型，部署在华为云ECS弹性云服务器上，使用昇腾310资源进行推理。

## 拓展实验

【在线实验体验】

[基于昇腾CANN的推理应用开发快速体验（Python)](https://www.hiascend.com/zh/college/onlineExperiment/codeLabAcl/acl)

[基于昇腾CANN的推理应用开发快速体验（C++）](https://www.hiascend.com/zh/college/onlineExperiment/detail/664597)

【进阶实验】

请参考[Ascend/samples](https://gitee.com/ascend/samples/tree/master/python/level2_simple_inference/1_classification/googlenet_imagenet_picture)代码仓库 在ECS上完成googlenet\_imagenet\_picture 推理实验

# 附录环境准备

## 购买ECS服务-Ai1资源——推理用

### 获取镜像

本节指导如何获取华为云账号下项目ID，以便方便导入实验所需镜像。

获取项目ID

进入华为云网址：<https://www.huaweicloud.com/>

点击 “账号中心”



账号中心

点击“我的凭证”

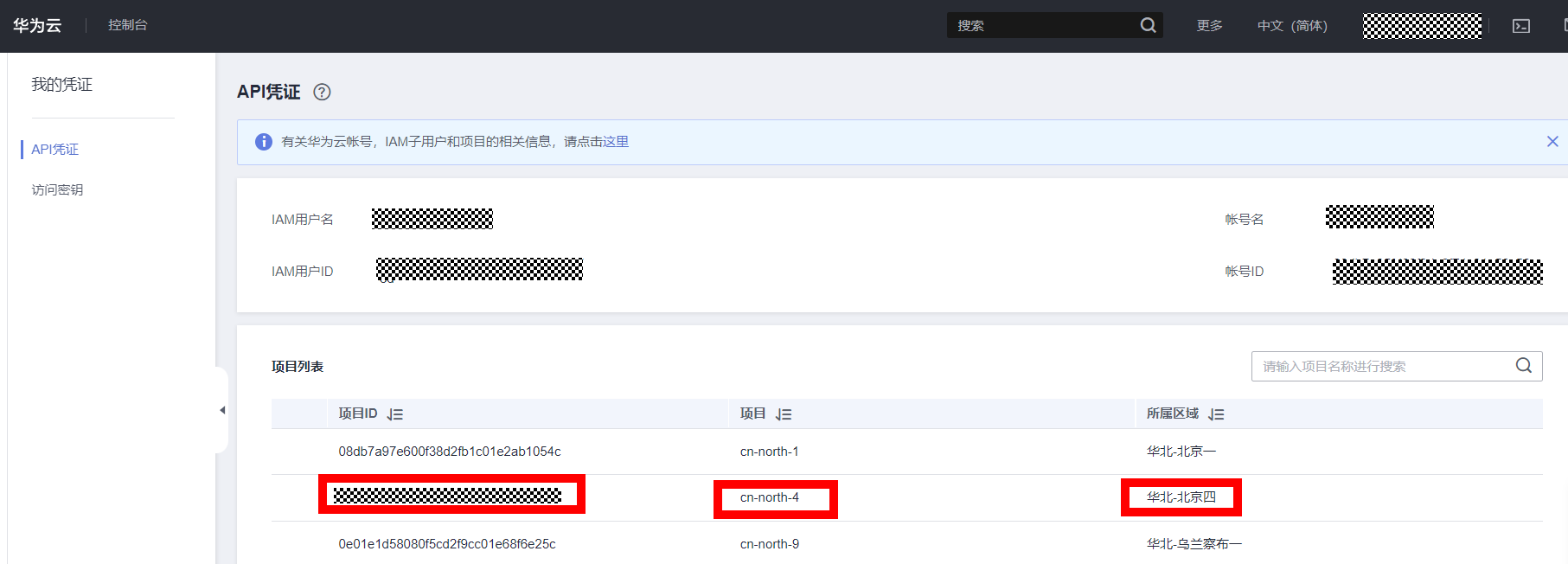


我的凭证

获取项目ID

选择项目为“cn-north-4”的项目ID，华为技术支持给高校老师分发镜像 “CANN5.1”。

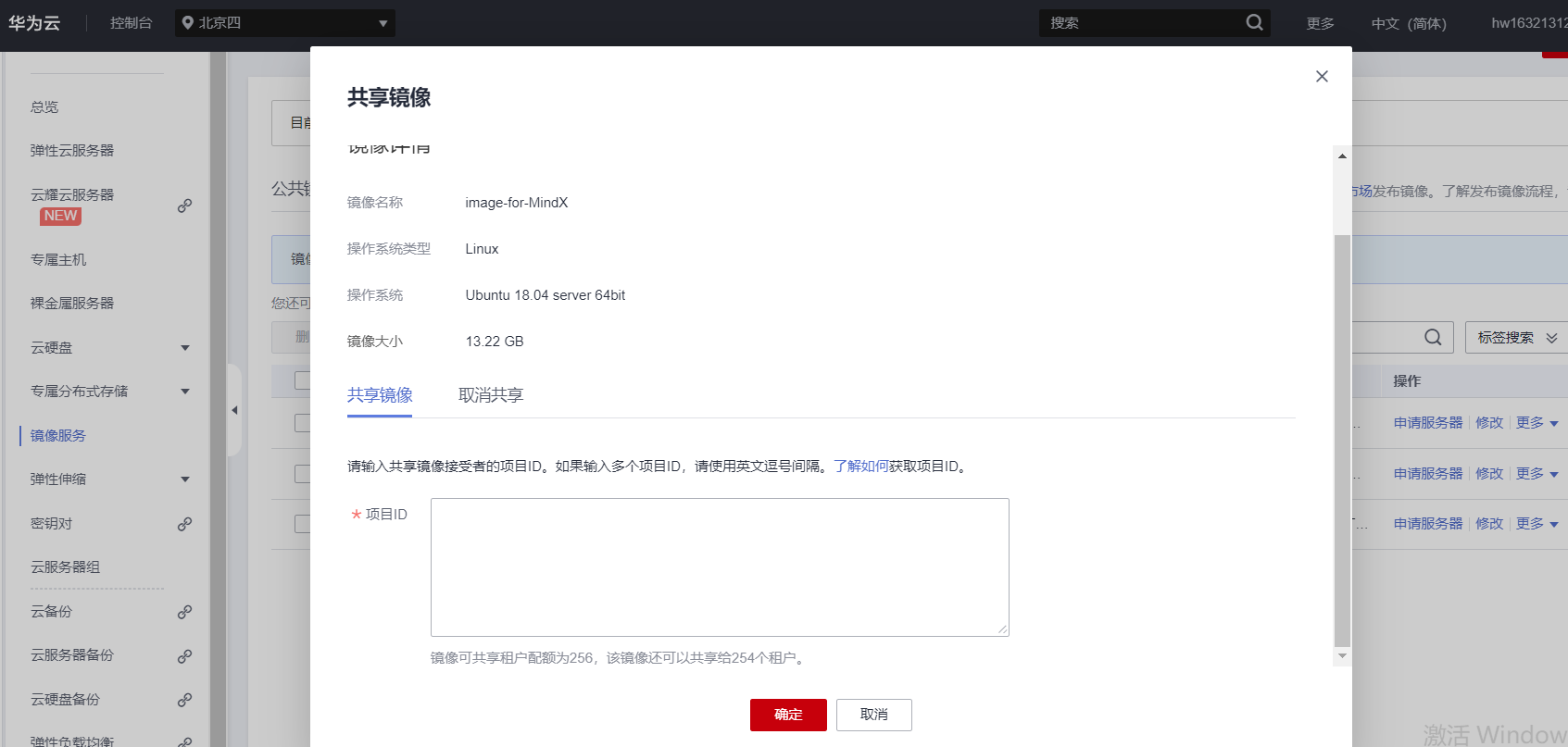
备注：如果有很多学员需要分享，可在班级群发起腾讯文档或接龙，由学员填写完毕后，统一分享.



项目ID

共享镜像

（镜像分享者需要操作）把项目ID复制到项目ID 里,点击确定。



### 创建ECS弹性云服务器

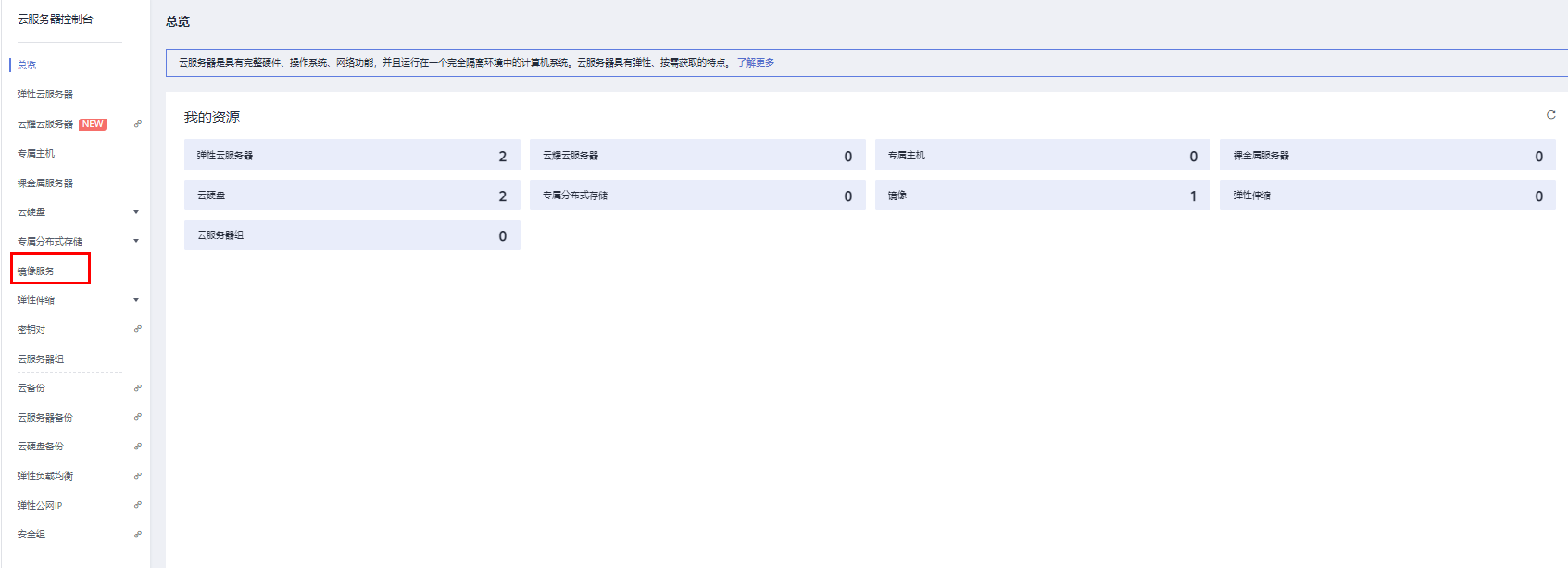
进入华为云ECS控制台

在[华为云ECS主页](https://www.huaweicloud.com/product/ecs.html)，点击“管理控制台”进入ECS的管理页面。



华为云ECS主页

接受共享镜像

在左侧菜单栏中选择“镜像服务”

镜像服务

选择“共享镜像”，“全部接受”。

共享镜像

创建弹性云服务器

控制台区域选择“华北－北京四”，在左侧菜单栏中选择“弹性云服务器”，在右上角“购买弹性云服务器”。



ECS 控制台

在“基础配置”里，选择如下配置：

计费模式：按需计费。

区域：华北-北京四。

可用区：随机分配。

CPU架构：x86计算。

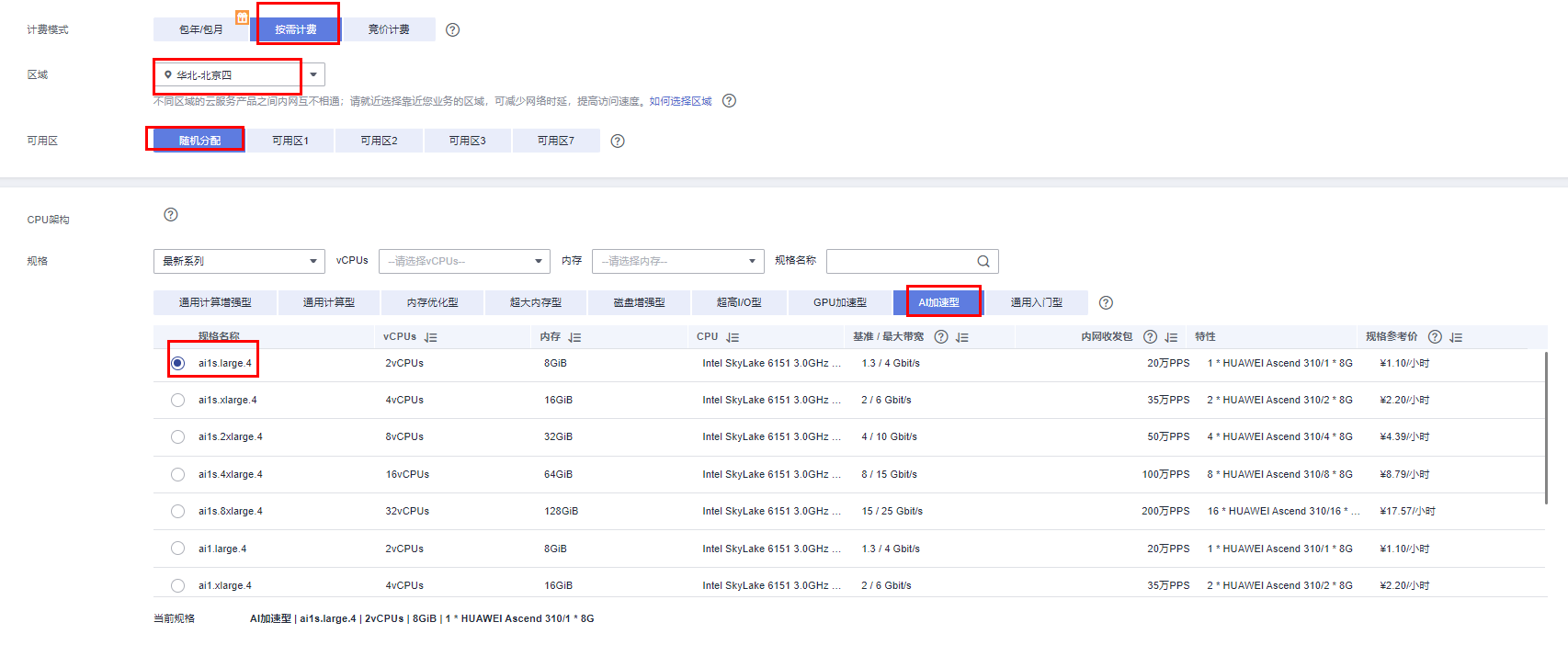
规格： AI加速型 | ai1s.large.4 | 2vCPUs | 8GiB | 1 \* HUAWEI Ascend 310/1 \* 8G。

镜像：公享镜像，CANN5.1。

规格：Ascend: 1\*Ascend910|CPU: 24核 96GB。

系统盘：通用型SSD，40GB。

如图所示：





ECS基础配置

基础配置完成之后，选择“下一步:网络配置”。

在“网络配置”里，选择如下配置：

网络：可以前往控制台创建新的虚拟私有云。

拓展网卡：无。

安全组：可以新建安全组。

弹性公网IP：现在购买。

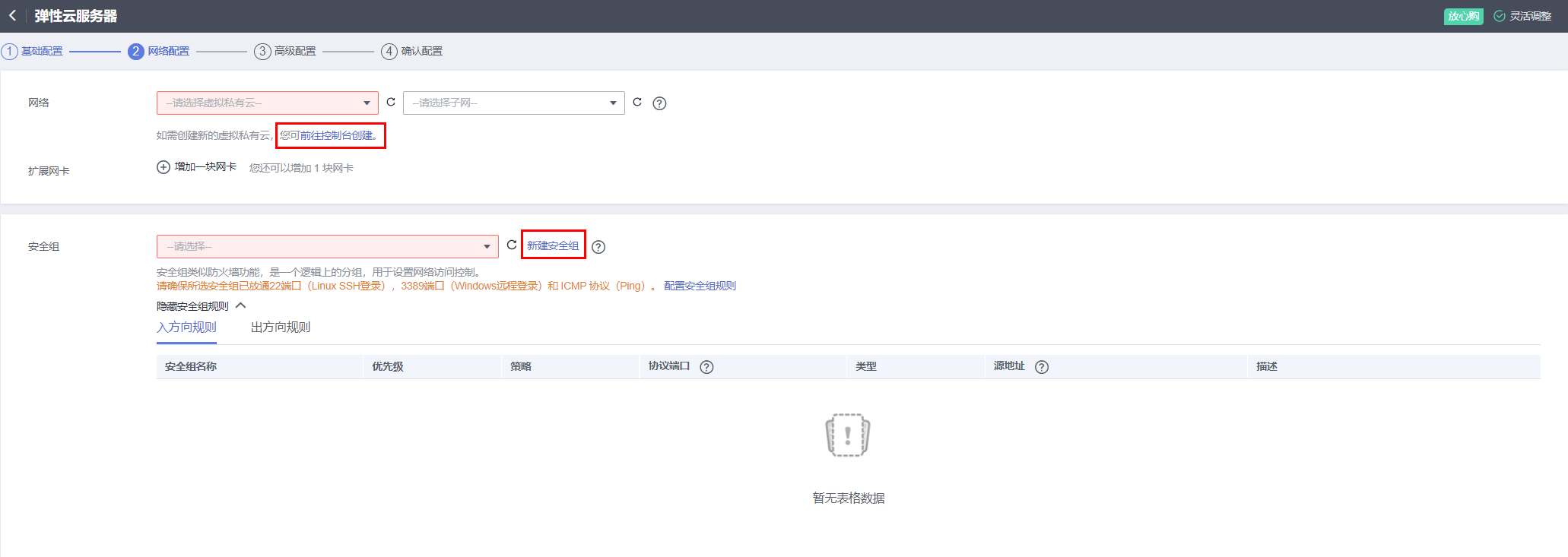
线路：全动态BGP。

公网带宽：按流量计费。

宽带大小：自定义，200Mbit/s。

释放行为：勾选随实例释放。

如图所示：





ECS网络配置

网络配置完成之后，选择“下一步”，进行高级配置。

在“高级配置”里，选择如下配置：

云服务器名称：可以自定义。

登录凭证：密码。

用户名：root。

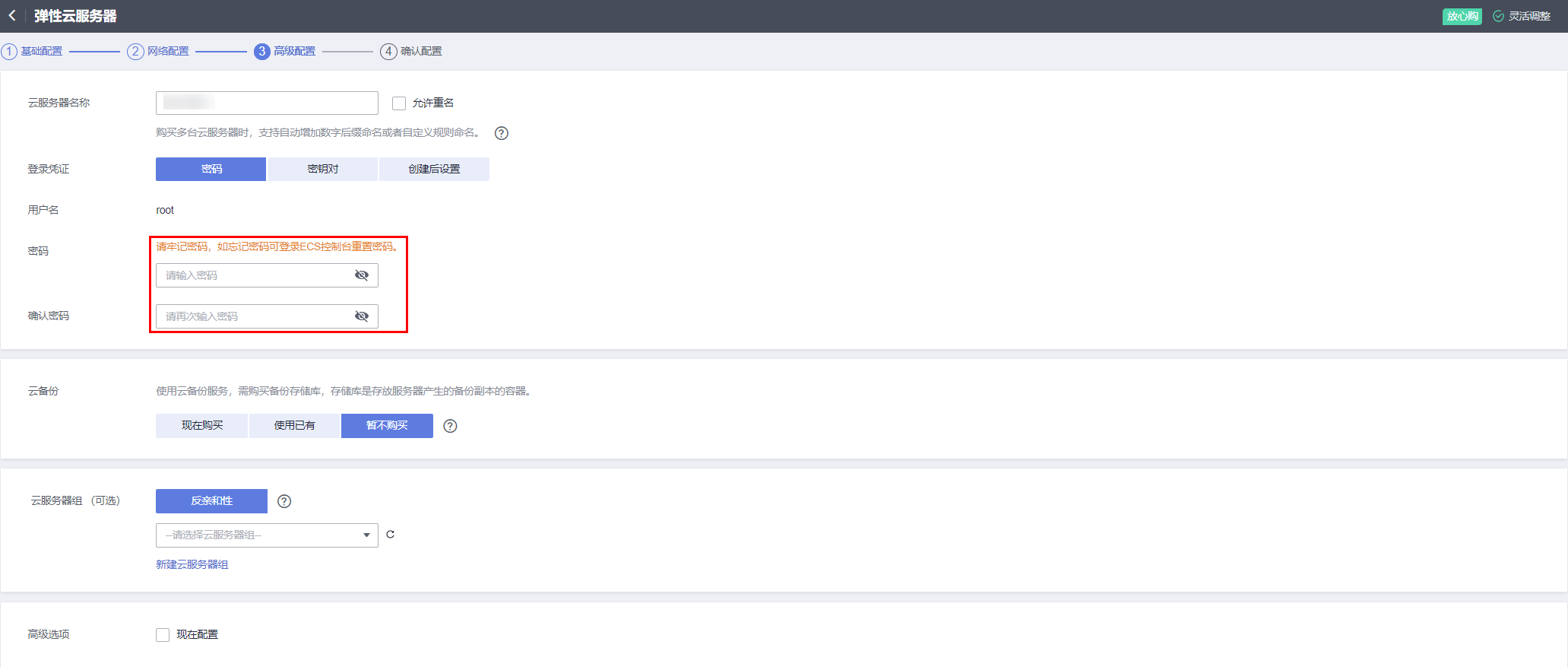
密码：自定义（后续登录使用，需谨记）。

云备份：暂不购买。

云服务器组：无。

高级选项：无。

如图所示：

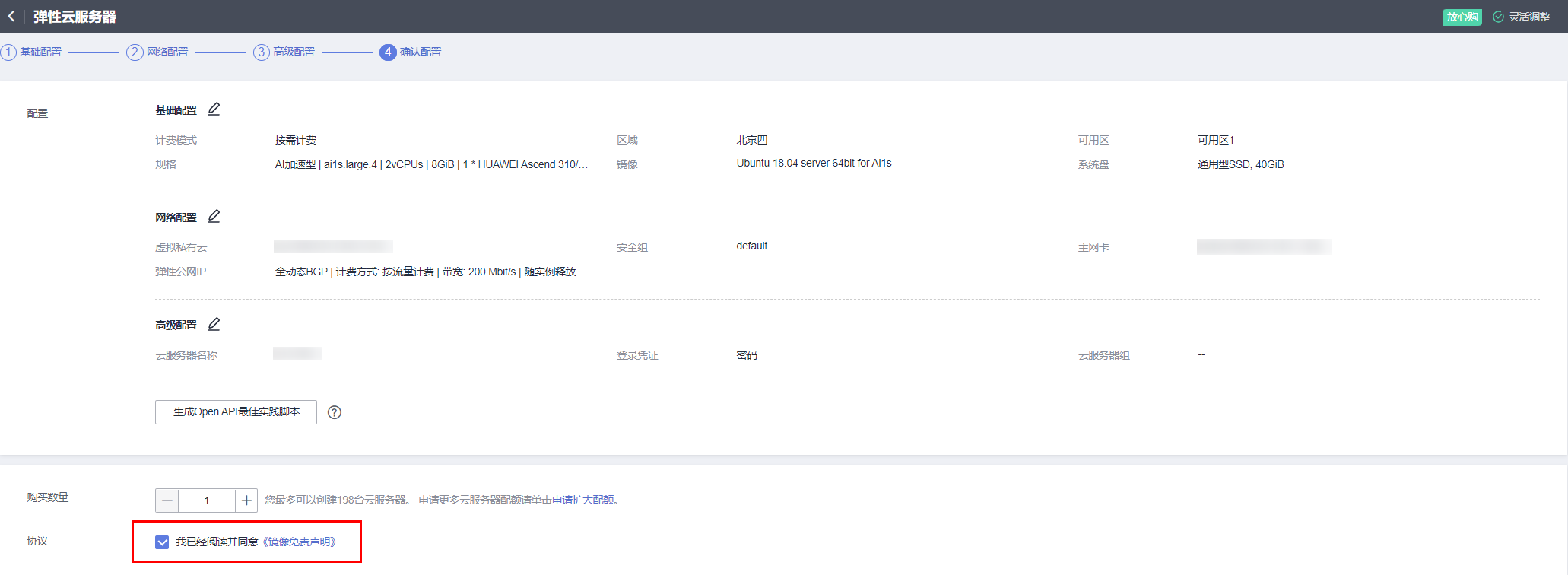


ECS高级配置

高级配置完成之后，选择“下一步”，确认配置。

在“确认配置”里，选择如下配置：

协议：勾选我已阅读并同意《镜像免责声明》。

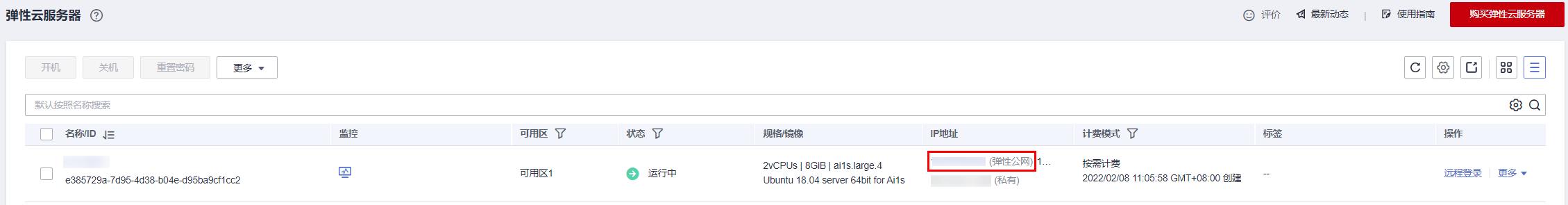


ECS 确认配置

确认配置完成之后，选择“立即购买”。

“任务提交成功”之后，选择“返回服务器列表”即可回到弹性云服务器的管理控制台，看到已创建的ECS弹性云服务器正在运行中。

\* 注意在“IP地址”显示的弹性公网IP地址，后续会用到。



ECS弹性云服务器创建成功

### MobaXterm连接ECS

下载MobaXterm （21.2及以上版本皆可）

进入MobaXterm 的官网主页：[**https://mobaxterm.mobatek.net/**](https://mobaxterm.mobatek.net/)

选择“Home Edition”，下载“MobaXterm Home Edition v21.x （Portable edition）”。

下载完成之后解压MobaXterm\_Portable\_v21.x.zip文件。

图形用户界面, 应用程序

描述已自动生成

下载界面

使用MobaXterm远程连接弹性云服务器

进入解压后的MobaXterm\_Portable\_v21.x文件夹，打开MobaXterm\_Personal\_21.x.exe文件，选择菜单栏的“Session”；

之后进入“Session settings”页面，远程链接选择“SSH”协议，输入创建ECS弹性云服务器后显示的弹性公网IP地址，选择指定用户名“Specify username”，用户名为“root”，配置完成之后选择“OK”提交。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

MobaXterm远程连接

MobaXterm 登录ECS需输入密码，在创建ECS弹性云服务器的步骤2中，高级配置里已自定义了弹性云服务器root用户密码，在此输入即可。

电脑萤幕的截图

描述已自动生成

MobaXterm 登录ECS需输入密码

MobaXterm远程链接弹性云服务器成功，后续还需进一步配置弹性云服务器的云上环境。

文本

描述已自动生成

MobaXterm 登录成功界面

### 关闭/删除ECS弹性云服务器

实验完成之后，请及时关闭/删除华为云ECS弹性云服务器，避免产生不必要的资源浪费。

关闭ECS弹性云服务器：

登录[ECS弹性云服务器控制台](https://console.huaweicloud.com/ecm/?region=cn-north-4#/ecs/manager/vmList)，在“操作”栏的“更多”，选择“关机”操作。

如下图所示：



关闭ECS弹性云服务器

删除ECS弹性云服务器：

登录[ECS弹性云服务器控制台](https://console.huaweicloud.com/ecm/?region=cn-north-4#/ecs/manager/vmList)，在“操作”栏的“更多”，选择“删除”操作，由于“未释放的弹性公网IP、磁盘会继续收费”，因此需要勾选“释放云服务器绑定的弹性公网IP地址”和“删除云服务器挂载的数据盘”。

如下图所示：



删除ECS弹性云服务器