基于MindSpore的图像识别全流程代码实战

版本：2.0



华为技术有限公司

目录

[1 实验介绍 2](#_Toc66871014)

[1.1 实验清单 2](#_Toc66871015)

[1.2 开发平台介绍 2](#_Toc66871016)

[1.3 背景知识 2](#_Toc66871017)

[2 基于MindSpore的图像识别全流程代码实战 6](#_Toc66871018)

[2.1 实验介绍 6](#_Toc66871019)

[2.2 实验环境要求 6](#_Toc66871020)

[2.3 实验总体设计 11](#_Toc66871021)

[2.4 实验过程 11](#_Toc66871022)

[2.4.1 数据集获取 12](#_Toc66871023)

[2.4.2 导入实验环境 12](#_Toc66871024)

[2.4.3 读取数据集 13](#_Toc66871025)

[2.4.4 模型构建训练 15](#_Toc66871026)

[2.4.5 模型预测 24](#_Toc66871027)

[2.4.6 模型保存和转换 25](#_Toc66871028)

[2.4.7 模型部署上线 29](#_Toc66871029)

[2.5 实验总结 35](#_Toc66871030)

# 实验介绍

卷积网络，也叫做卷积神经网络，是专门用来处理具有类似网格结构的数据的神经网络。例如时间序列数据（可以认为是在时间轴上有规律的采样形成的一维网格）和图像数据（可以看作二维的像素网格）。卷积在诸多应用领域都表现优异。本章主要围绕深度学习的卷积网络而开设的实验。

本章实验的主要目的是掌握卷及网络相关基础知识点。掌握不同神经网络架构的设计原理，熟悉使用MindSpore框架实验的一般流程，以及最后将模型部署上线。

## 实验清单

表格：实验、简述、难度、软件环境、硬件环境。

|  |  |  |  |
| --- | --- | --- | --- |
| 实验 | 简述 | 软件环境 | 开发环境 |
| 图像识别全流程代码实战 | 训练并部署上线花卉图像分类卷积网络分类识别实验。 | MindSpore-1.1.1-python3.7 | ModelArts |

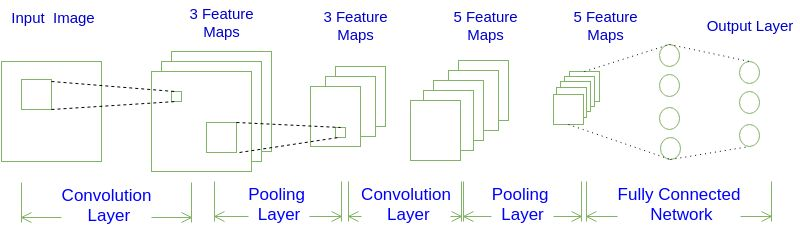
## 开发平台介绍

* MindSpore 最佳匹配昇腾芯片的开源AI计算框架，支持Asend、GPU、CPU平台。MindSpore官网：https://www.mindspore.cn
* ModelArts是面向开发者的一站式AI开发平台，为机器学习与深度学习提供海量数据预处理及半自动化标注、大规模分布式Training、自动化模型生成，及端-边-云模型按需部署能力，帮助用户快速创建和部署模型，管理全周期AI工作流。

## 背景知识

卷积神经网络结构

卷积神经网络是深度学习与神经网络算法中主流算法之一，主要用于图像识别。其结构图如下：

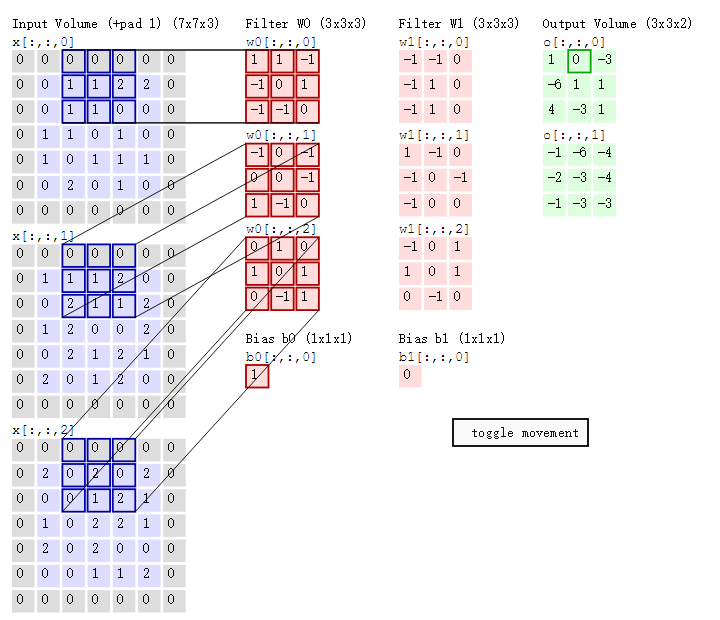


卷积神经网络结构

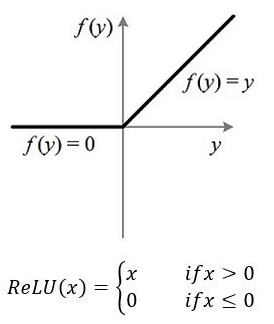
卷积层

在卷积层计算过程中，输入是一定区域大小(width\*height)的数据，和滤波器filter（带着一组固定权重的神经元）做内积后等到新的二维数据。

具体来说，滤波器filter（带着一组固定权重的神经元）通过滑动窗口的方式，对输入图像进行扫描，扫描过程中，对输入图像中的像素值进行点乘，不同的滤波器filter会得到不同的输出数据，比如颜色深浅、轮廓。相当于如果想提取图像的不同特征，则用不同的滤波器filter，提取想要的关于图像的特定信息：颜色深浅或轮廓。



激活函数：



ReLU函数其实是分段线性函数，把所有的负值都变为0，而正值不变，这种操作被成为单侧抑制。可别小看这个简单的操作，正因为有了这单侧抑制，才使得神经网络中的神经元也具有了稀疏激活性。

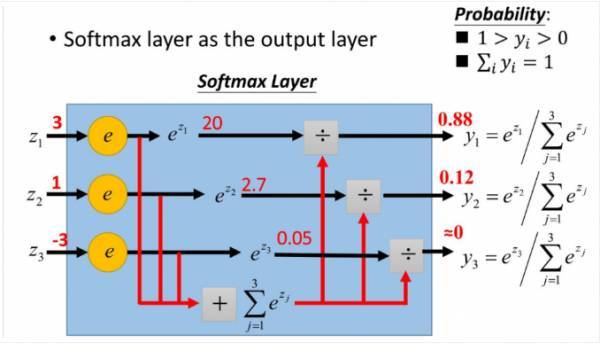
此外，相比于其它激活函数来说，ReLU有以下优势：对于线性函数而言，ReLU的表达能力更强，尤其体现在深度网络中；而对于sigmoid等激活函数而言，ReLU由于非负区间的导数为1，因此可以缓解梯度消失问题(Vanishing Gradient Problem)，使得模型的收敛速度维持在一个稳定状态。这里稍微描述一下什么是梯度消失问题：当梯度小于1时，预测值与真实值之间的误差每传播一层会衰减一次，如果在深层模型中使用sigmoid作为激活函数，这种现象尤为明显，将导致模型收敛停滞不前。

池化层：



上图所展示的是区域最大，即上图左边部分中 左上角2x2的矩阵中6最大，右上角2x2的矩阵中8最大，左下角2x2的矩阵中3最大，右下角2x2的矩阵中4最大，所以得到上图右边部分的结果：6 8 3 4。

全连接层：



softmax直白来说就是将原来输出是3,1,-3通过softmax函数一作用，就映射成为(0,1)的值， 而这些值的累和为1（满足概率的性质），那么我们就可以将它理解成概率。在训练网络时可以通过对预测值和标签值计算交叉熵损失。

# 基于MindSpore的图像识别全流程代码实战

## 实验介绍

图像分类在我们的日常生活中广泛使用，比如拍照识物，还有手机的AI拍照，在学术界，每年也有很多图像分类的比赛，本实验将会利用一个开源数据集来帮助大家学习如何构建自己的图像识别模型。本实验会使用MindSpore来构建图像识别模型，然后将模型部署ModelArts上提供在线预测服务。

## 实验环境要求

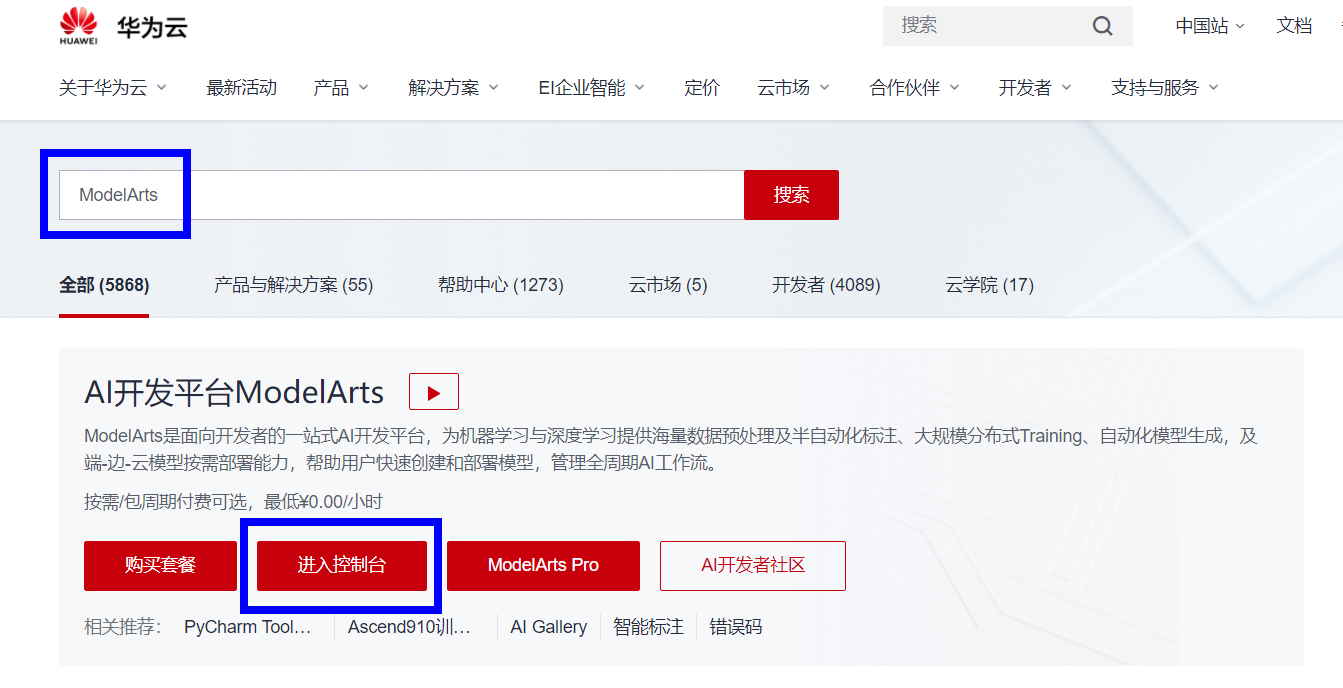
* ModelArts平台：MindSpore-1.1.1

### 实验环境创建

登录华为云网站

<https://www.huaweicloud.com/?ticket=ST-4217587-ZMyac1H4moJ0pjSOHq4ubtbh-sso&locale=zh-cn>

在搜索框输入ModelArts，进入ModelArts的控制台：



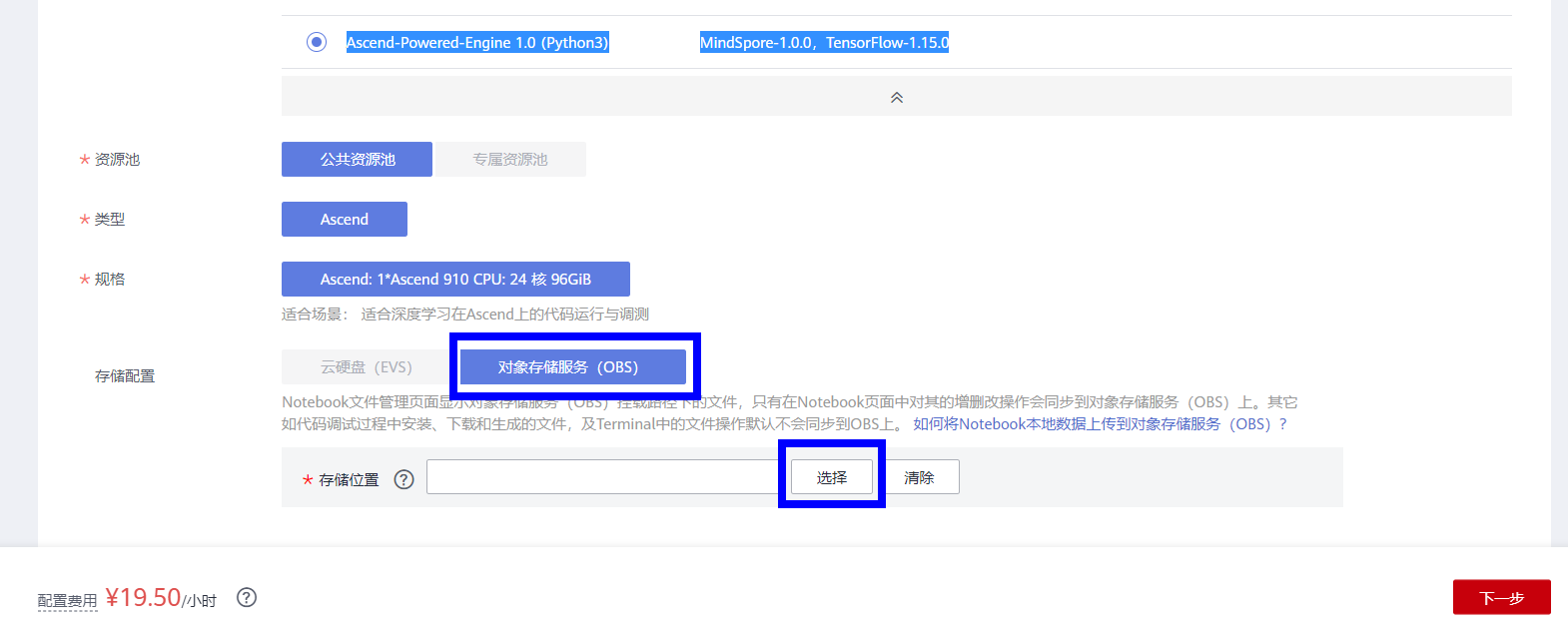
点击开发环境，进入到Notebook页面，点击创建，创建实验所需的环境：



项目自动停止时间设置为2小时，实验环境选择第4个，Ascend-Powered-Engine 1.0 (Python3) MindSpore-1.1.1



选择OBS对象存储服务，在存储位置处点击选择：



如果没有创建桶，点击新建对象存储服务（OBS）桶，新建一个OBS桶；如果已有桶，可以选择已创建的桶：



创建一个新桶：区域选择华北-北京四，桶名称需设置为全局唯一，桶默认私有，点击立即创建



桶创建成功后，在创建项目的页面，点击选择，就可以看到新建的桶，选择该桶：



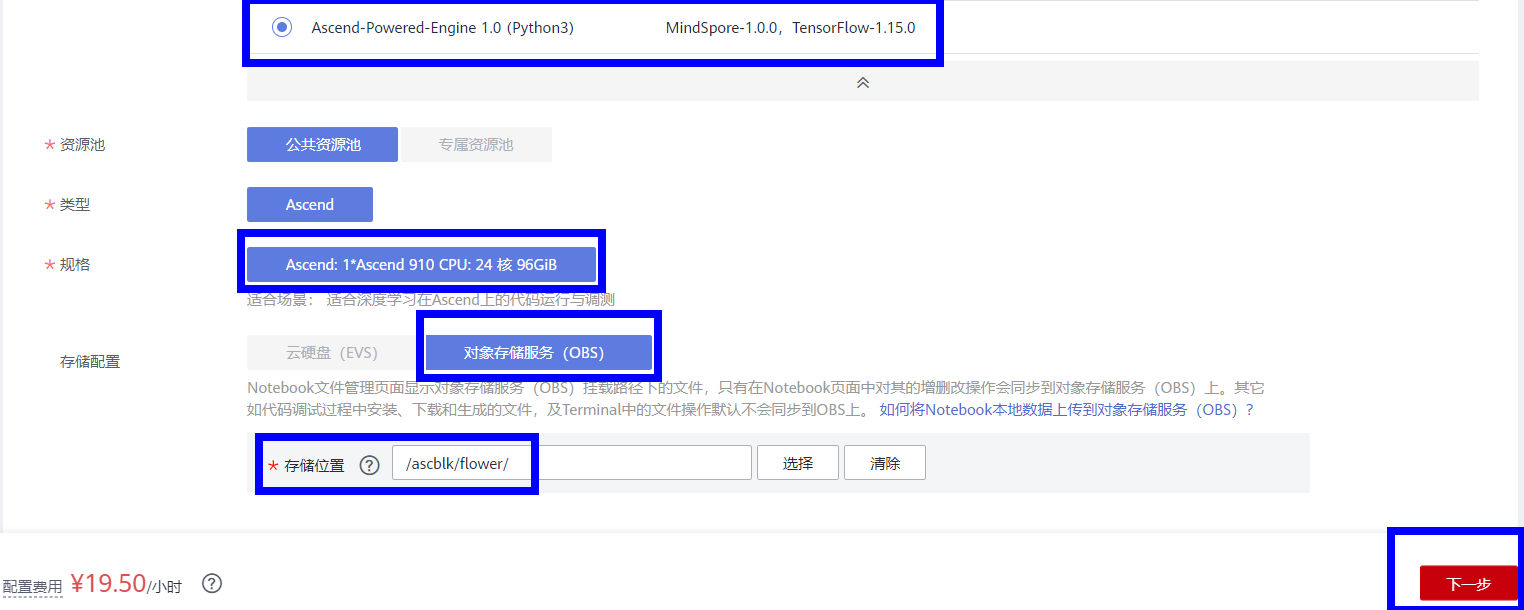
点击新建文件夹，将我们的项目存放在文件夹里面：



创建新文件夹，点击确定：



确认好各项环境无误之后，创建新环境，点击下一步：



点击提交后，打开创建的notebook工作环境：

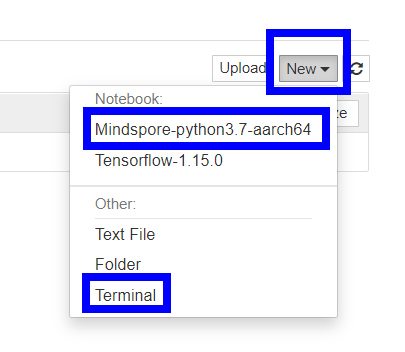




打开notebook工作环境之后，左上角显示项目名称，右上角显示自动停止时间，可以进行修改；我们选择OBS桶作为存储对象，所以数据需要同步Sync OBS；点击upload可以上传数据和代码；

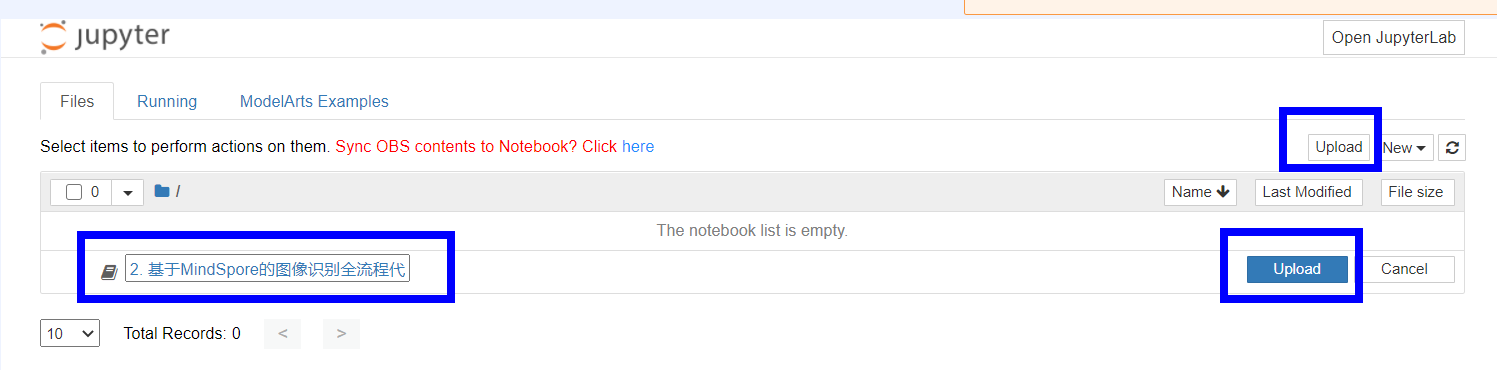


点击new可以新建代码，文件夹，还有终端等；

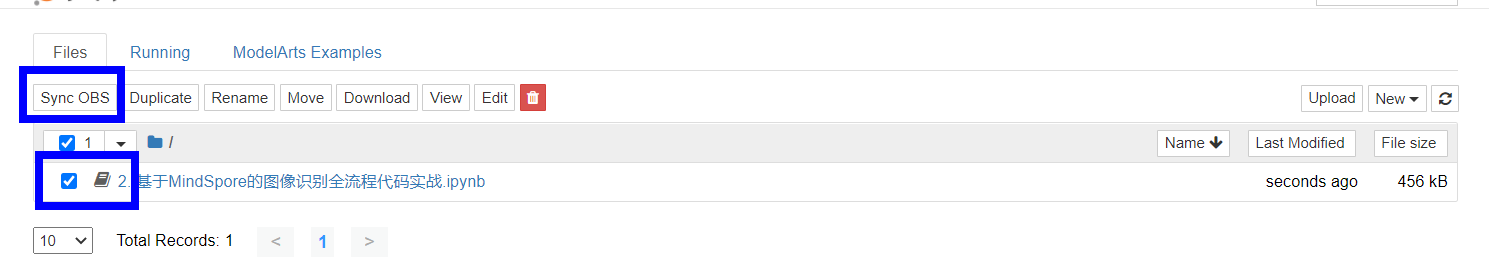


### 上传实验数据，同步实验数据

点击upload，上传本次实验的代码，注意需要点击蓝色的upload才能上传；

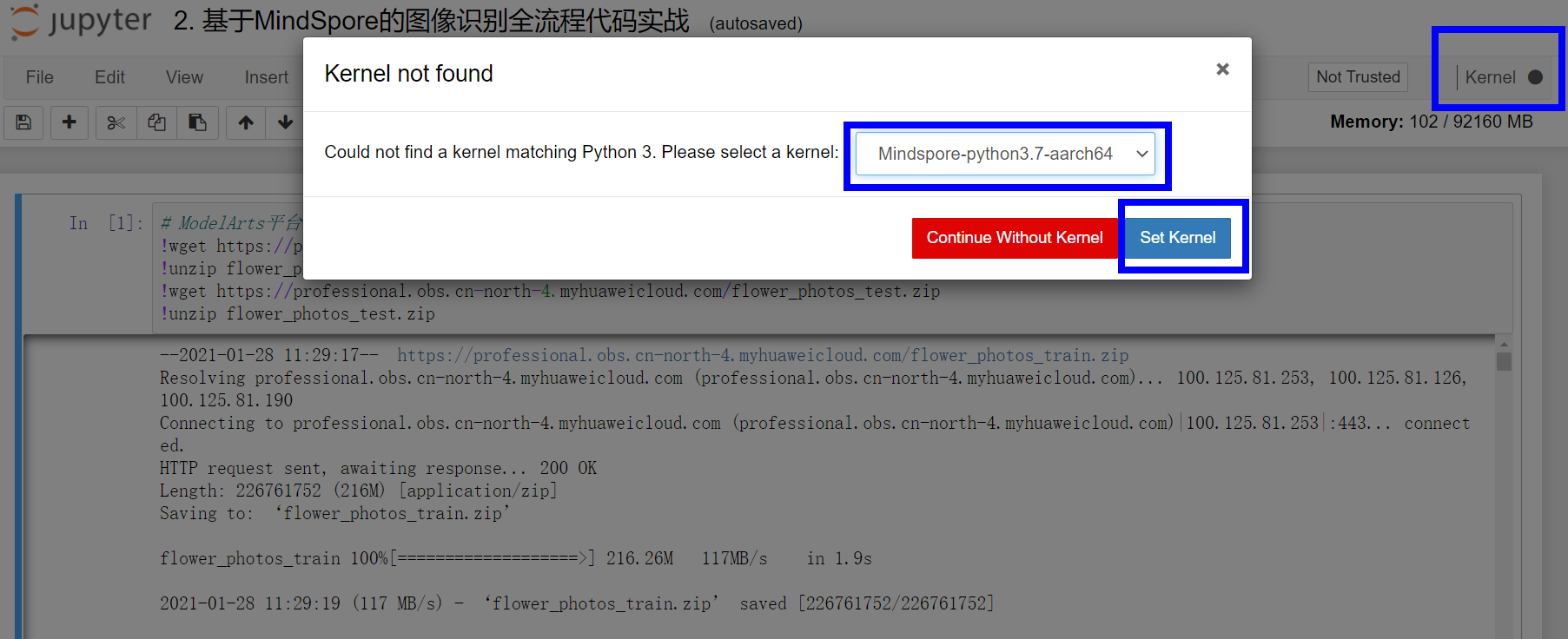


选中实验代码及数据集，之后会出现Sync OBS，点击Sync OBS进行同步数据，将代码和数据同步至OBS桶内；



### 设置代码运行Kernel

打开实验代码，选择实验所需Kernel，本次实验用到MindSpore1.1.1，python3.7的实验环境，之后点击set kernel；



可以在右上角查看当前实验kernel



## 实验总体设计

该实验主要步骤包括：

1. 数据集获取
2. 导入实验环境
3. 读取数据集
4. 模型构建训练
5. 模型预测
6. 模型保存和转换
7. 编辑模型推理代码和配置文件
8. 模型部署上线

## 实验过程

本节将详细介绍实验的设计与实现。

2.4.1数据集获取；

2.4.2导入实验环境；

2.4.3读取数据集；

2.4.4 模型构建训练；

2.4.5模型预测；

2.4.6 模型保存和转换；

2.4.7 模型部署上线。

### 数据集获取

该数据集是开源数据集，总共包括5种花的类型：分别是daisy（雏菊，633张），dandelion（蒲公英，898张），roses（玫瑰，641张），sunflowers（向日葵，699张），tulips（郁金香，799张），保存在5个文件夹当中，总共3670张，大小大概在230M左右。为了在模型部署上线之后进行测试，数据集在这里分成了flower\_train和flower\_test两部分。

在ModelArts平台输入代码自动获取数据，获取后的数据会存在当前项目的work目录下，不会同步到OBS，如果想要查看下载的文件，可以调用终端，输入cd work命令，切换到work目录下之后，输入ls命令进行查看。

代码：

#ModelArts平台输入代码会自动下载数据，下载完成之后不需要二次运行，不然会报错。

!wget https://professional.obs.cn-north-4.myhuaweicloud.com/flower\_photos\_train.zip

!unzip flower\_photos\_train.zip

!wget https://professional.obs.cn-north-4.myhuaweicloud.com/flower\_photos\_test.zip

!unzip flower\_photos\_test.zip

### 导入实验环境

导入相应的模块

os模块主要用于处理文件和目录，比如：获取当前目录下文件，删除制定文件，改变目录，查看文件大小等；MindSpore是目前业界最流行的深度学习框架，在图像，语音，文本，目标检测等领域都有深入的应用，也是该实验的核心，主要用于定义占位符，定义变量，创建卷积神经网络模型；numpy是一个基于python的科学计算包，在该实验中主要用来处理数值运算。

代码：

#easydict模块用于以属性的方式访问字典的值

from easydict import EasyDict as edict

#os模块主要用于处理文件和目录

import os

import numpy as np

import matplotlib.pyplot as plt

import mindspore

#导入mindspore框架数据集

import mindspore.dataset as ds

#vision.c\_transforms模块是处理图像增强的高性能模块，用于数据增强图像数据改进训练模型。

from mindspore.dataset.vision import c\_transforms as vision

from mindspore import context

import mindspore.nn as nn

from mindspore.train import Model

from mindspore.nn.optim.momentum import Momentum

from mindspore.train.callback import ModelCheckpoint, CheckpointConfig, LossMonitor

from mindspore import Tensor

from mindspore.train.serialization import export

from mindspore.train.loss\_scale\_manager import FixedLossScaleManager

from mindspore.train.serialization import load\_checkpoint, load\_param\_into\_net

import mindspore.ops as ops

# 设置MindSpore的执行模式和设备

context.set\_context(mode=context.GRAPH\_MODE, device\_target="Ascend")

定义变量

代码：

cfg = edict({

'data\_path': 'flower\_photos\_train', #训练数据集，如果是zip文件需要解压

'test\_path':'flower\_photos\_test', #测试数据集，如果是zip文件需要解压

'data\_size': 3616,

'HEIGHT': 224, # 图片高度

'WIDTH': 224, # 图片宽度

'\_R\_MEAN': 123.68,

'\_G\_MEAN': 116.78,

'\_B\_MEAN': 103.94,

'\_R\_STD': 1,

'\_G\_STD': 1,

'\_B\_STD':1,

'\_RESIZE\_SIDE\_MIN': 256,

'\_RESIZE\_SIDE\_MAX': 512,

'batch\_size': 32,

'num\_class': 5, # 分类类别

'epoch\_size': 150, # 训练次数

'loss\_scale\_num':1024,

'prefix': 'resnet-ai',

'directory': './model\_resnet',

'save\_checkpoint\_steps': 10,

})

### 读取数据集

数据读取并处理流程如下：

MindSpore的mindspore.dataset提供了ImageFolderDataset函数，可以直接读取文件夹图片数据并映射文件夹名字为其标签(label)。这里我们使用ImageFolderDataset函数 读取'daisy','dandelion','roses','sunflowers','tulips'数据。并将这五类标签映射为： {'daisy':0,'dandelion':1,'roses':2,'sunflowers':3,'tulips':4}

使用RandomCropDecodeResize、HWC2CHW、shuffle进行数据预处理

代码：

# 数据处理

def read\_data(path,config,usage="train"):

#从目录中读取图像的源数据集。

dataset = ds.ImageFolderDataset(path,

class\_indexing={'daisy':0,'dandelion':1,'roses':2,'sunflowers':3,'tulips':4})

# define map operations

decode\_op = vision.Decode()

normalize\_op = vision.Normalize(mean=[cfg.\_R\_MEAN, cfg.\_G\_MEAN, cfg.\_B\_MEAN], std=[cfg.\_R\_STD, cfg.\_G\_STD, cfg.\_B\_STD])

resize\_op = vision.Resize(cfg.\_RESIZE\_SIDE\_MIN)

center\_crop\_op = vision.CenterCrop((cfg.HEIGHT, cfg.WIDTH))

horizontal\_flip\_op = vision.RandomHorizontalFlip()

channelswap\_op = vision.HWC2CHW()

random\_crop\_decode\_resize\_op = vision.RandomCropDecodeResize((cfg.HEIGHT, cfg.WIDTH), (0.5, 1.0), (1.0, 1.0), max\_attempts=100)

if usage == 'train':

dataset = dataset.map(input\_columns="image", operations=random\_crop\_decode\_resize\_op)

dataset = dataset.map(input\_columns="image", operations=horizontal\_flip\_op)

else:

dataset = dataset.map(input\_columns="image", operations=decode\_op)

dataset = dataset.map(input\_columns="image", operations=resize\_op)

dataset = dataset.map(input\_columns="image", operations=center\_crop\_op)

dataset = dataset.map(input\_columns="image", operations=normalize\_op)

dataset = dataset.map(input\_columns="image", operations=channelswap\_op)

if usage == 'train':

dataset = dataset.shuffle(buffer\_size=10000) # 10000 as in imageNet train script

dataset = dataset.batch(cfg.batch\_size, drop\_remainder=True)

else:

dataset = dataset.batch(1, drop\_remainder=True)

dataset = dataset.repeat(1)

dataset.map\_model = 4

return dataset

de\_train = read\_data(cfg.data\_path,cfg,usage="train")

de\_test = read\_data(cfg.test\_path,cfg,usage="test")

print('训练数据集数量：',de\_train.get\_dataset\_size()\*cfg.batch\_size)#get\_dataset\_size()获取批处理的大小。

print('测试数据集数量：',de\_test.get\_dataset\_size())

de\_dataset = de\_train

data\_next = de\_dataset.create\_dict\_iterator(output\_numpy=True).\_\_next\_\_()

print('通道数/图像长/宽：', data\_next['image'][0,...].shape)

print('一张图像的标签样式：', data\_next['label'][0]) # 一共5类，用0-4的数字表达类别。

plt.figure()

plt.imshow(data\_next['image'][0,0,...])

plt.colorbar()

plt.grid(False)

plt.show()

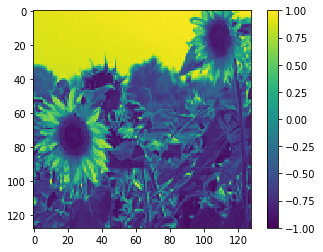
输出：

训练数据集数量： 3616

测试数据集数量： 52

通道数/图像长/宽： (3, 224, 224)

一张图像的标签样式： 3



### 模型构建训练

本节主要介绍了如何构建一个图片识别模型。在章节的最后，我们又介绍了如何保存一个模型的计算图和模型结构，为后续的模型部署上线做准备。

定义模型

"""ResNet."""

def \_weight\_variable(shape, factor=0.01):

init\_value = np.random.randn(\*shape).astype(np.float32) \* factor

return Tensor(init\_value)

def \_conv3x3(in\_channel, out\_channel, stride=1):

weight\_shape = (out\_channel, in\_channel, 3, 3)

weight = \_weight\_variable(weight\_shape)

return nn.Conv2d(in\_channel, out\_channel,

kernel\_size=3, stride=stride, padding=0, pad\_mode='same', weight\_init=weight)

def \_conv1x1(in\_channel, out\_channel, stride=1):

weight\_shape = (out\_channel, in\_channel, 1, 1)

weight = \_weight\_variable(weight\_shape)

return nn.Conv2d(in\_channel, out\_channel,

kernel\_size=1, stride=stride, padding=0, pad\_mode='same', weight\_init=weight)

def \_conv7x7(in\_channel, out\_channel, stride=1):

weight\_shape = (out\_channel, in\_channel, 7, 7)

weight = \_weight\_variable(weight\_shape)

return nn.Conv2d(in\_channel, out\_channel,

kernel\_size=7, stride=stride, padding=0, pad\_mode='same', weight\_init=weight)

def \_bn(channel):

return nn.BatchNorm2d(channel, eps=1e-4, momentum=0.9,

gamma\_init=1, beta\_init=0, moving\_mean\_init=0, moving\_var\_init=1)

def \_bn\_last(channel):

return nn.BatchNorm2d(channel, eps=1e-4, momentum=0.9,

gamma\_init=0, beta\_init=0, moving\_mean\_init=0, moving\_var\_init=1)

def \_fc(in\_channel, out\_channel):

weight\_shape = (out\_channel, in\_channel)

weight = \_weight\_variable(weight\_shape)

return nn.Dense(in\_channel, out\_channel, has\_bias=True, weight\_init=weight, bias\_init=0)

class ResidualBlock(nn.Cell):

"""

ResNet V1 residual block definition.

Args:

in\_channel (int): Input channel.

out\_channel (int): Output channel.

stride (int): Stride size for the first convolutional layer. Default: 1.

Returns:

Tensor, output tensor.

Examples:

>>> ResidualBlock(3, 256, stride=2)

"""

expansion = 4

def \_\_init\_\_(self,

in\_channel,

out\_channel,

stride=1):

super(ResidualBlock, self).\_\_init\_\_()

channel = out\_channel // self.expansion

self.conv1 = \_conv1x1(in\_channel, channel, stride=1)

self.bn1 = \_bn(channel)

self.conv2 = \_conv3x3(channel, channel, stride=stride)

self.bn2 = \_bn(channel)

self.conv3 = \_conv1x1(channel, out\_channel, stride=1)

self.bn3 = \_bn\_last(out\_channel)

self.relu = nn.ReLU()

self.down\_sample = False

if stride != 1 or in\_channel != out\_channel:

self.down\_sample = True

self.down\_sample\_layer = None

if self.down\_sample:

self.down\_sample\_layer = nn.SequentialCell([\_conv1x1(in\_channel, out\_channel, stride),

\_bn(out\_channel)])

self.add = ops.Add()

def construct(self, x): # pylint: disable=missing-docstring

identity = x

out = self.conv1(x)

out = self.bn1(out)

out = self.relu(out)

out = self.conv2(out)

out = self.bn2(out)

out = self.relu(out)

out = self.conv3(out)

out = self.bn3(out)

if self.down\_sample:

identity = self.down\_sample\_layer(identity)

out = self.add(out, identity)

out = self.relu(out)

return out

class ResNet(nn.Cell):

"""

ResNet architecture.

Args:

block (Cell): Block for network.

layer\_nums (list): Numbers of block in different layers.

in\_channels (list): Input channel in each layer.

out\_channels (list): Output channel in each layer.

strides (list): Stride size in each layer.

num\_classes (int): The number of classes that the training images are belonging to.

Returns:

Tensor, output tensor.

Examples:

>>> ResNet(ResidualBlock,

>>> [3, 4, 6, 3],

>>> [64, 256, 512, 1024],

>>> [256, 512, 1024, 2048],

>>> [1, 2, 2, 2],

>>> 10)

"""

def \_\_init\_\_(self,

block,

layer\_nums,

in\_channels,

out\_channels,

strides,

num\_classes):

super(ResNet, self).\_\_init\_\_()

if not len(layer\_nums) == len(in\_channels) == len(out\_channels) == 4:

raise ValueError("the length of layer\_num, in\_channels, out\_channels list must be 4!")

self.conv1 = \_conv7x7(3, 64, stride=2)

self.bn1 = \_bn(64)

self.relu = ops.ReLU()

self.maxpool = nn.MaxPool2d(kernel\_size=3, stride=2, pad\_mode="same")

self.layer1 = self.\_make\_layer(block,

layer\_nums[0],

in\_channel=in\_channels[0],

out\_channel=out\_channels[0],

stride=strides[0])

self.layer2 = self.\_make\_layer(block,

layer\_nums[1],

in\_channel=in\_channels[1],

out\_channel=out\_channels[1],

stride=strides[1])

self.layer3 = self.\_make\_layer(block,

layer\_nums[2],

in\_channel=in\_channels[2],

out\_channel=out\_channels[2],

stride=strides[2])

self.layer4 = self.\_make\_layer(block,

layer\_nums[3],

in\_channel=in\_channels[3],

out\_channel=out\_channels[3],

stride=strides[3])

self.mean = ops.ReduceMean(keep\_dims=True)

self.flatten = nn.Flatten()

self.end\_point = \_fc(out\_channels[3], num\_classes)

def \_make\_layer(self, block, layer\_num, in\_channel, out\_channel, stride):

"""

Make stage network of ResNet.

Args:

block (Cell): Resnet block.

layer\_num (int): Layer number.

in\_channel (int): Input channel.

out\_channel (int): Output channel.

stride (int): Stride size for the first convolutional layer.

Returns:

SequentialCell, the output layer.

Examples:

>>> \_make\_layer(ResidualBlock, 3, 128, 256, 2)

"""

layers = []

resnet\_block = block(in\_channel, out\_channel, stride=stride)

layers.append(resnet\_block)

for \_ in range(1, layer\_num):

resnet\_block = block(out\_channel, out\_channel, stride=1)

layers.append(resnet\_block)

return nn.SequentialCell(layers)

def construct(self, x): # pylint: disable=missing-docstring

x = self.conv1(x)

x = self.bn1(x)

x = self.relu(x)

c1 = self.maxpool(x)

c2 = self.layer1(c1)

c3 = self.layer2(c2)

c4 = self.layer3(c3)

c5 = self.layer4(c4)

out = self.mean(c5, (2, 3))

out = self.flatten(out)

out = self.end\_point(out)

return out

def resnet50(class\_num=10):

"""

Get ResNet50 neural network.

Args:

class\_num (int): Class number.

Returns:

Cell, cell instance of ResNet50 neural network.

Examples:

>>> net = resnet50(10)

"""

return ResNet(ResidualBlock,

[3, 4, 6, 3],

[64, 256, 512, 1024],

[256, 512, 1024, 2048],

[1, 2, 2, 2],

class\_num)

def resnet101(class\_num=1001):

"""

Get ResNet101 neural network.

Args:

class\_num (int): Class number.

Returns:

Cell, cell instance of ResNet101 neural network.

Examples:

>>> net = resnet101(1001)

"""

return ResNet(ResidualBlock,

[3, 4, 23, 3],

[64, 256, 512, 1024],

[256, 512, 1024, 2048],

[1, 2, 2, 2],

class\_num)

自定义动态学习率

def get\_lr(global\_step,

total\_epochs,

steps\_per\_epoch,

lr\_init=0.01,

lr\_max=0.1,

warmup\_epochs=5):

"""

Generate learning rate array.

Args:

global\_step (int): Initial step of training.

total\_epochs (int): Total epoch of training.

steps\_per\_epoch (float): Steps of one epoch.

lr\_init (float): Initial learning rate. Default: 0.01.

lr\_max (float): Maximum learning rate. Default: 0.1.

warmup\_epochs (int): The number of warming up epochs. Default: 5.

Returns:

np.array, learning rate array.

"""

lr\_each\_step = []

total\_steps = steps\_per\_epoch \* total\_epochs

warmup\_steps = steps\_per\_epoch \* warmup\_epochs

if warmup\_steps != 0:

inc\_each\_step = (float(lr\_max) - float(lr\_init)) / float(warmup\_steps)

else:

inc\_each\_step = 0

for i in range(int(total\_steps)):

if i < warmup\_steps:

lr = float(lr\_init) + inc\_each\_step \* float(i)

else:

base = ( 1.0 - (float(i) - float(warmup\_steps)) / (float(total\_steps) - float(warmup\_steps)) )

lr = float(lr\_max) \* base \* base

if lr < 0.0:

lr = 0.0

lr\_each\_step.append(lr)

current\_step = global\_step

lr\_each\_step = np.array(lr\_each\_step).astype(np.float32)

learning\_rate = lr\_each\_step[current\_step:]

return learning\_rate

开始训练

完成数据预处理、网络定义、损失函数和优化器定义之后，开始模型训练。模型训练包含2层迭代，数据集的多伦迭代epoch和一轮数据集内按分组从数据集中抽取数据，输入网络计算得到损失函数，然后通过优化器计算和更新训练参数的梯度。

net=resnet50(class\_num=cfg.num\_class)

#计算softmax交叉熵。

loss = nn.SoftmaxCrossEntropyWithLogits(sparse=True, reduction="mean")

#设置Adam优化器

train\_step\_size = de\_train.get\_dataset\_size()

lr = Tensor(get\_lr(global\_step=0, total\_epochs=cfg.epoch\_size, steps\_per\_epoch=train\_step\_size))

opt = Momentum(net.trainable\_params(), lr, momentum=0.9, weight\_decay=1e-4, loss\_scale=cfg.loss\_scale\_num)

#opt = Momentum(filter(lambda x: x.requires\_grad, net.get\_parameters()), 0.002,

# 0.9, 0.00004, loss\_scale=1024.0)

loss\_scale = FixedLossScaleManager(cfg.loss\_scale\_num, False)

model = Model(net, loss\_fn=loss, optimizer=opt, loss\_scale\_manager=loss\_scale, metrics={'acc'})

loss\_cb = LossMonitor(per\_print\_times=train\_step\_size)

ckpt\_config = CheckpointConfig(save\_checkpoint\_steps=cfg.save\_checkpoint\_steps, keep\_checkpoint\_max=1)

ckpoint\_cb = ModelCheckpoint(prefix=cfg.prefix, directory=cfg.directory, config=ckpt\_config)

print("============== Starting Training ==============")

model.train(cfg.epoch\_size, de\_train, callbacks=[loss\_cb,ckpoint\_cb], dataset\_sink\_mode=True)

输出：

============== Starting Training ==============

epoch: 1 step: 113, loss is 1.2005742

epoch: 2 step: 113, loss is 5.448204

epoch: 3 step: 113, loss is 1.4671483

epoch: 4 step: 113, loss is 1.3227137

epoch: 5 step: 113, loss is 1.3022066

epoch: 6 step: 113, loss is 1.182671

…

epoch: 146 step: 113, loss is 0.0083019305

epoch: 147 step: 113, loss is 0.0132089155

epoch: 148 step: 113, loss is 0.0026898778

epoch: 149 step: 113, loss is 0.013294414

epoch: 150 step: 113, loss is 0.07428124

模型评估

# 使用测试集评估模型，打印总体准确率

metric = model.eval(de\_test)

print(metric)

输出：

{'acc': 0.8846153846153846}

### 模型预测

# 预测

class\_names = {0:'daisy',1:'dandelion',2:'roses',3:'sunflowers',4:'tulips'}

for i in range(10):

test\_ = de\_test.create\_dict\_iterator().\_\_next\_\_()

test = Tensor(test\_['image'], mindspore.float32)

predictions = model.predict(test)

predictions = predictions.asnumpy()

true\_label = test\_['label'].asnumpy()

#显示预测结果

p\_np = predictions[0, :]

pre\_label = np.argmax(p\_np)

print('第' + str(i) + '个sample预测结果：', class\_names[pre\_label], ' 真实结果：', class\_names[true\_label[0]])

输出：

第0个sample预测结果： dandelion 真实结果： dandelion

第1个sample预测结果： sunflowers 真实结果： sunflowers

第2个sample预测结果： daisy 真实结果： dandelion

第3个sample预测结果： daisy 真实结果： daisy

第4个sample预测结果： daisy 真实结果： daisy

第5个sample预测结果： sunflowers 真实结果： sunflowers

第6个sample预测结果： tulips 真实结果： tulips

第7个sample预测结果： daisy 真实结果： daisy

第8个sample预测结果： roses 真实结果： roses

第9个sample预测结果： tulips 真实结果： daisy

### 模型保存和转换

保存模型为onnx格式

#创建文件夹

if not os.path.exists('./flowers/'):

os.mkdir('./flowers/')

param\_dict = load\_checkpoint(os.path.join(cfg.directory,cfg.prefix+'-'+str(cfg.epoch\_size)+'\_'+str(train\_step\_size)+'.ckpt'))

# load the parameter into net

resnet=resnet50(class\_num=cfg.num\_class)

load\_param\_into\_net(resnet, param\_dict)

x = np.random.uniform(-1.0, 1.0, size = [1, 3, cfg.HEIGHT, cfg.WIDTH]).astype(np.float32)

export(resnet, Tensor(x), file\_name = './flowers/best\_model.onnx', file\_format = 'ONNX')

将模型保存到obs桶里面

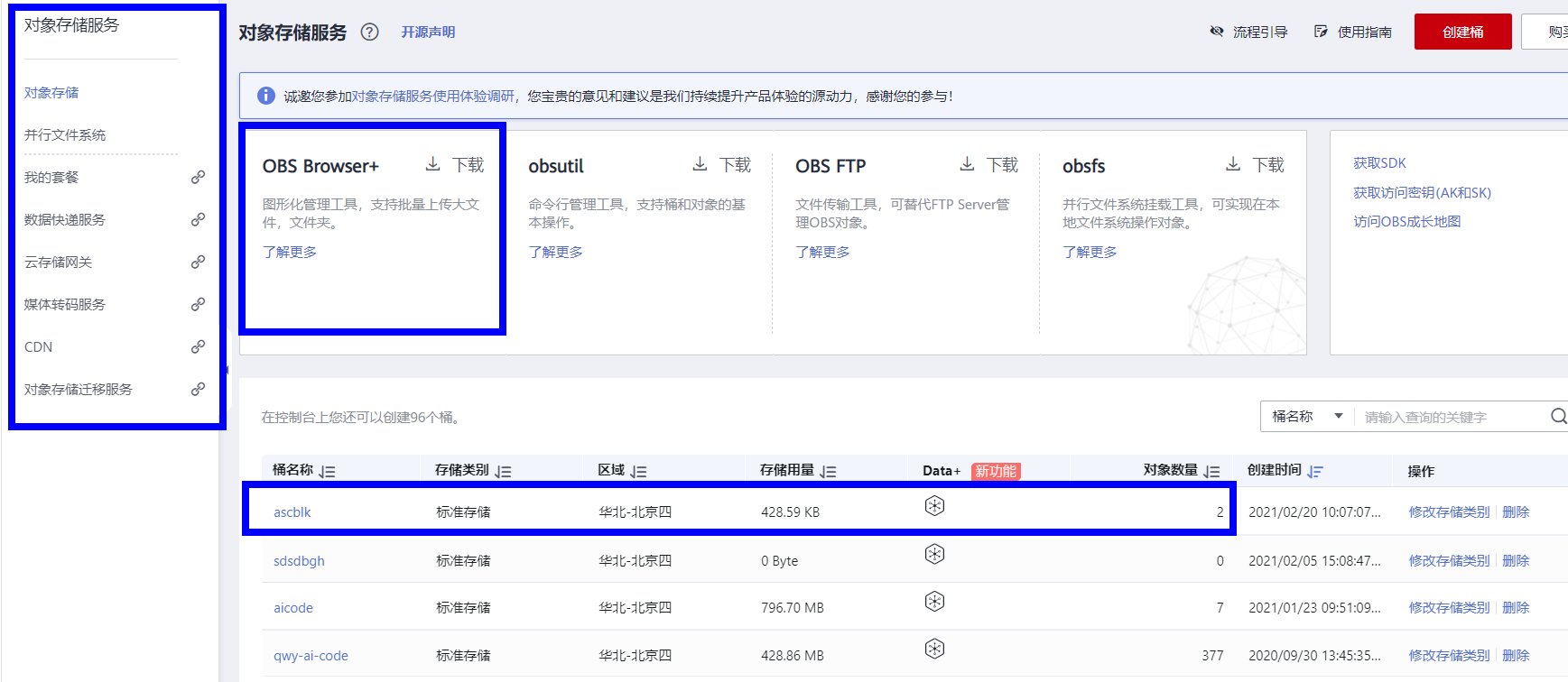
将项目中训练好的模型保存到自己的OBS桶中，下方代码中标黄部分为桶名称，需要替换成自己的桶名称，如何查看自己的桶名称：

<https://www.huaweicloud.com/s/JW9icyU>

在华为云官网上搜索OBS，找到OBS页面，点击进入控制台：



进入OBS控制台之后，可以看到自己创建的所有桶：



选中桶，进入桶内，点击左侧的对象，就能看到桶内保存的文件：



把ascblk修改成自己的桶名，之后运行代码，在桶内查看模型是否保存至桶内的/flowers/model/onnx目录下

#创建桶，根据自己情况修改路径

import moxing

moxing.file.copy\_parallel(src\_url='./flowers/best\_model.onnx', dst\_url='s3://ascblk/flower/onnx/best\_model.onnx')

上传insert\_op\_conf.cfg文件至桶内onnx模型目录下

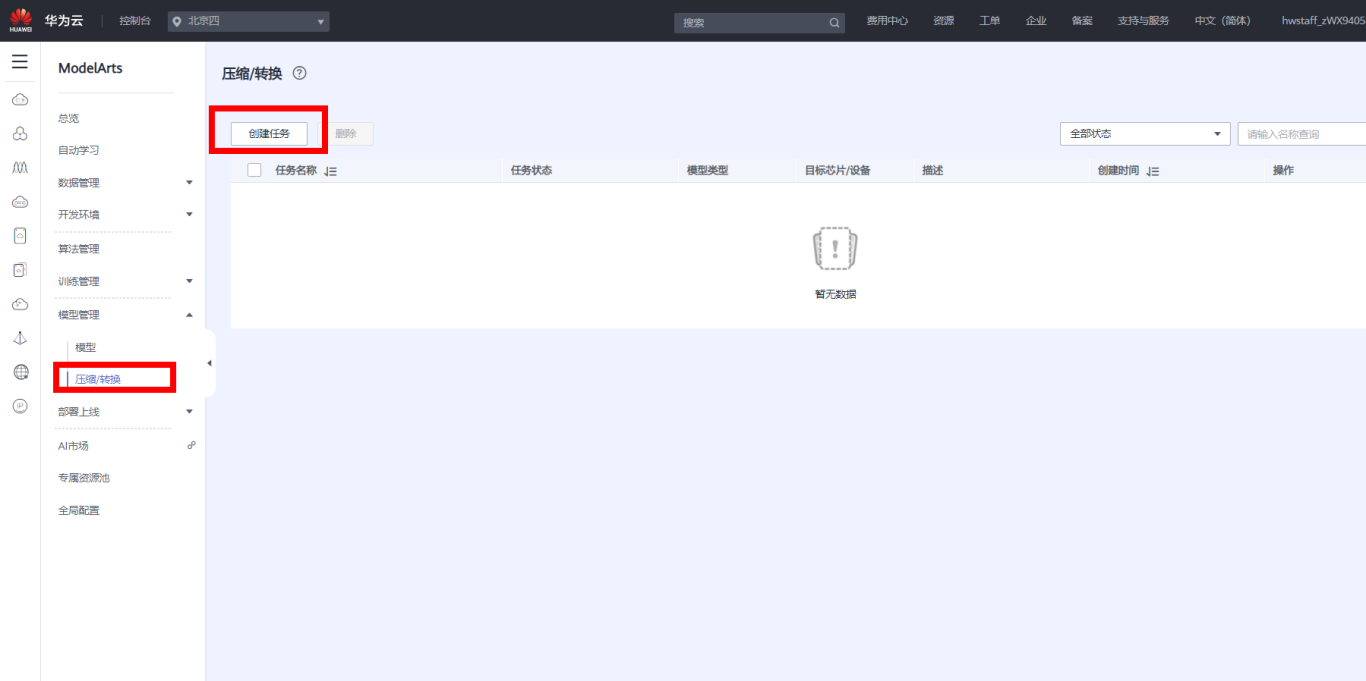
上传insert\_op\_conf.cfg文件至桶内/flowers/model/onnx目录下，将insert\_op\_conf.cfg文件放置在best\_model.onnx的同目录下，如图所示：

（如需下载insert\_op\_conf.cfg文件，请点击<https://professional-construction.obs.cn-north-4.myhuaweicloud.com:443/deep-learning/flowermodelfiles/insert_op_conf.cfg>）



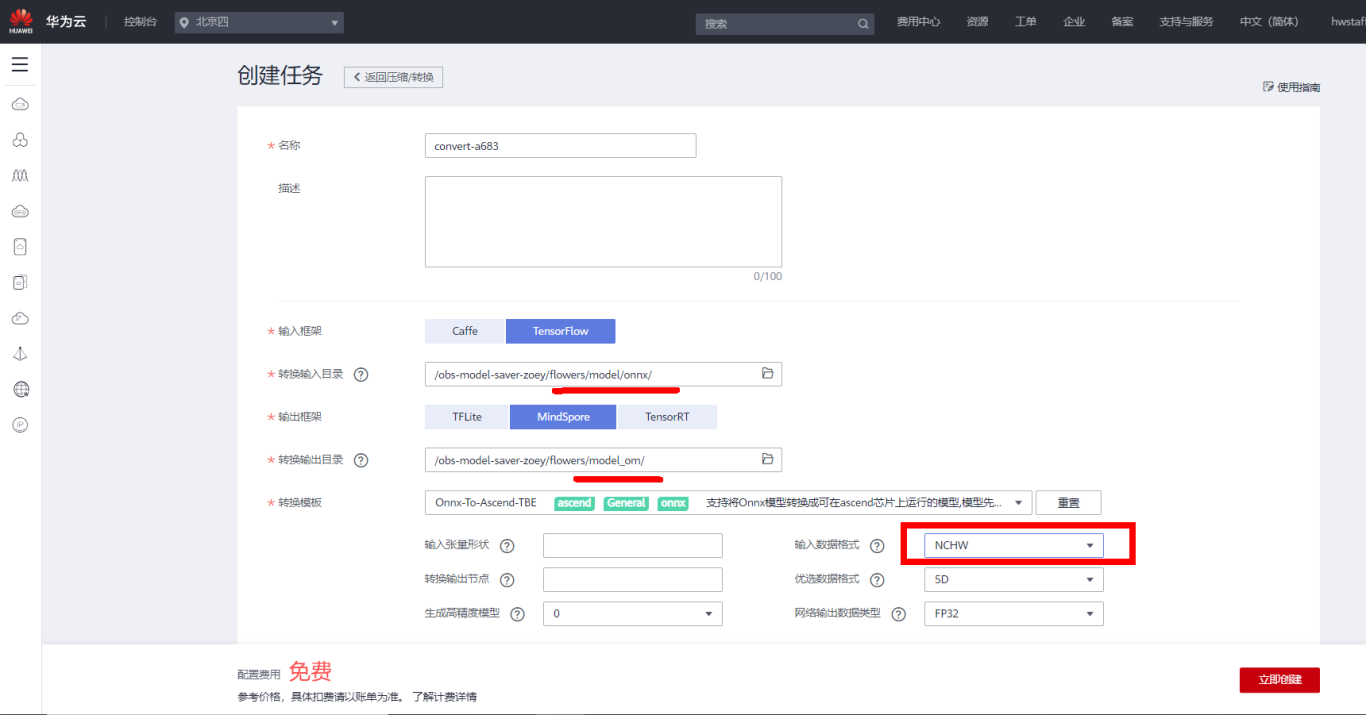
将onnx格式的模型转换为om格式

进入ModelArts控制台，点击模型管理>压缩/转换>创建任务



按照下图填写。其中转换输入目录是前面代码生成的.onnx文件目录。转换输出路径为空白目录(提前在obs桶中建好空白目录)，创建model\_om空白文件夹。





点击立即创建，等待几分钟，运行成功，可查看obs输出目录下是否有生成的.om文件。

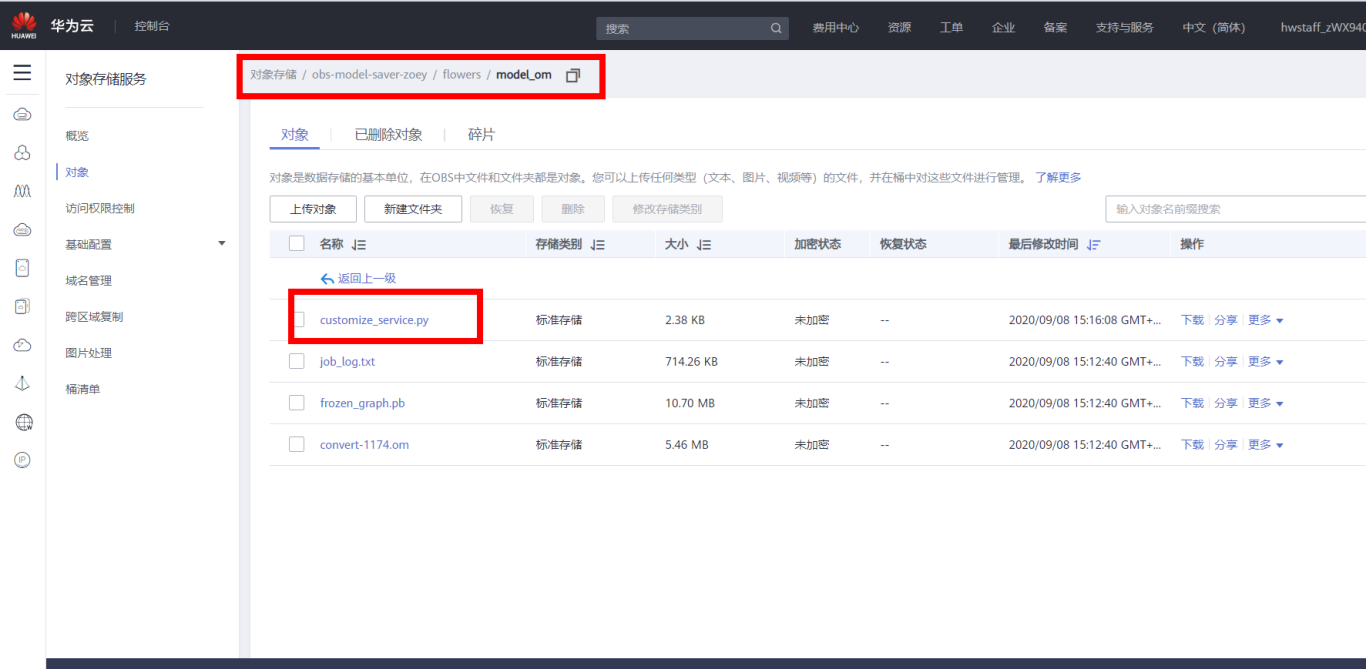
### 模型部署上线

上传customize\_service.py（编写测试数据读取代码）文件至om模型目录下

编写测试数据读取代码如下所示，并将写好的度测试数据代码文件customize\_service.py上传到obs桶内的om模型路径下，和上一步生成的om模型在一个文件夹里面，见下图。

（如需下载customize\_service.py，请点击：<https://chuangxin.obs.cn-north-4.myhuaweicloud.com:443/MS1.0fulldeployment/customize_service.py>）

在OBS桶内上传customize\_service.py 文件到om的目录下，如图所示：



以下是customize\_service.py的代码内容：

from \_\_future\_\_ import absolute\_import

from \_\_future\_\_ import division

from \_\_future\_\_ import print\_function

import os

import numpy as np

from PIL import Image

from hiai.nn\_tensor\_lib import NNTensor

from hiai.nntensor\_list import NNTensorList

from model\_service.hiai\_model\_service import HiaiBaseService

"""AIPP example

aipp\_op {

aipp\_mode: static

input\_format : RGB888\_U8

mean\_chn\_0 : 123

mean\_chn\_1 : 117

mean\_chn\_2 : 104

}

"""

labels\_list = []

label\_txt\_path = os.path.join(os.path.dirname(os.path.realpath(\_\_file\_\_)), 'labels.txt')

if os.path.exists(label\_txt\_path):

with open(label\_txt\_path, 'r') as f:

for line in f:

if line.strip():

labels\_list.append(line.strip())

def keep\_ratio\_resize(im, base=256):

short\_side = min(float(im.size[0]), float(im.size[1]))

resize\_ratio = base / short\_side

resize\_sides = int(round(resize\_ratio \* im.size[0])), int(round(resize\_ratio \* im.size[1]))

im = im.resize(resize\_sides)

return im

def central\_crop(im, base=224):

width, height = im.size

left = (width - base) / 2

top = (height - base) / 2

right = (width + base) / 2

bottom = (height + base) / 2

# Crop the center of the image

im = im.crop((left, top, right, bottom))

return im

class DemoService(HiaiBaseService):

def \_preprocess(self, data):

preprocessed\_data = {}

images = []

for k, v in data.items():

for file\_name, file\_content in v.items():

image = Image.open(file\_content)

image = keep\_ratio\_resize(image, base=256)

image = central\_crop(image, base=224)

image = np.array(image) # HWC

# AIPP should use RGB format.

# mean reg is applied in AIPP.

# Transpose is applied in AIPP

tensor = NNTensor(image)

images.append(tensor)

tensor\_list = NNTensorList(images)

preprocessed\_data['images'] = tensor\_list

return preprocessed\_data

def \_inference(self, data, image\_info=None):

result = {}

for k, v in data.items():

result[k] = self.model.proc(v)

return result

def \_postprocess(self, data):

outputs = {}

prob = data['images'][0][0][0][0].tolist()

outputs['scores'] = prob

labels\_list = {0:'daisy',1:'dandelion',2:'roses',3:'sunflowers',4:'tulips'}

if labels\_list:

outputs['predicted\_label'] = labels\_list[int(np.argmax(prob))]

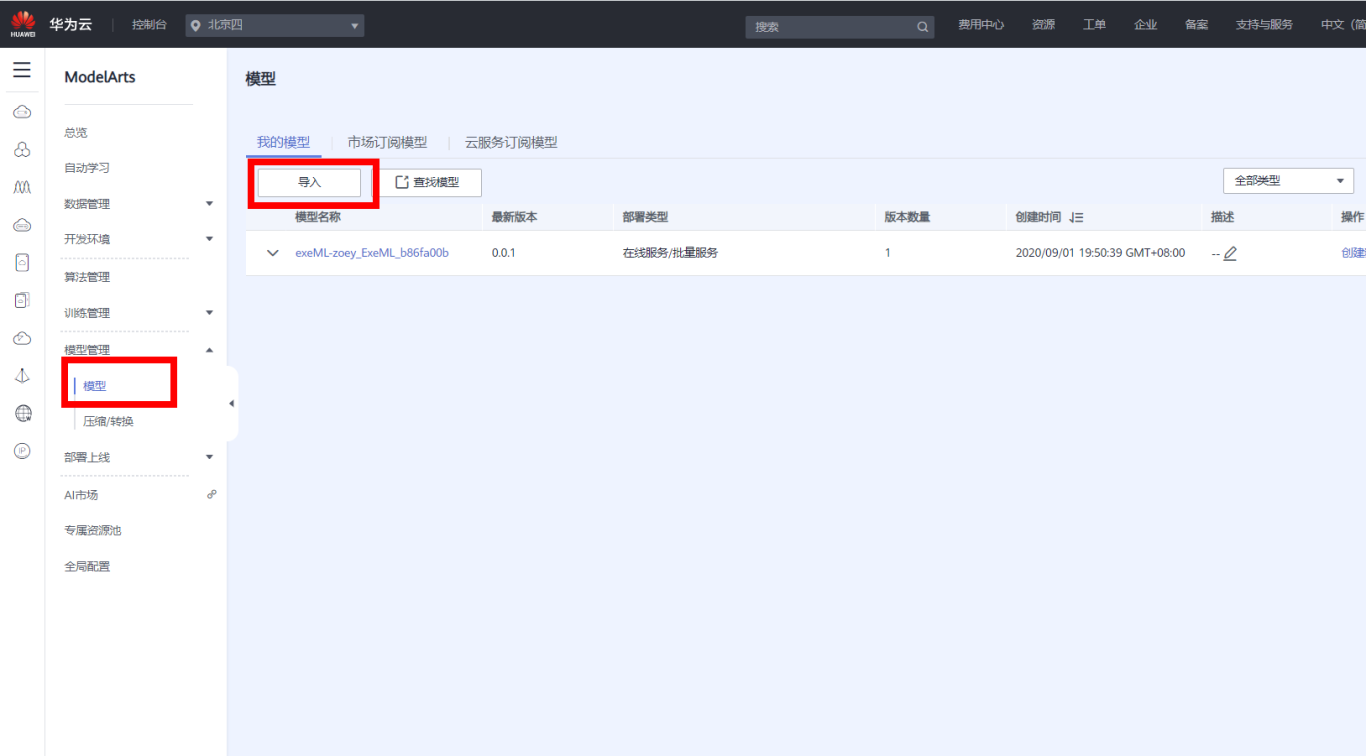
else:

outputs['predicted\_label'] = str(int(np.argmax(prob)))

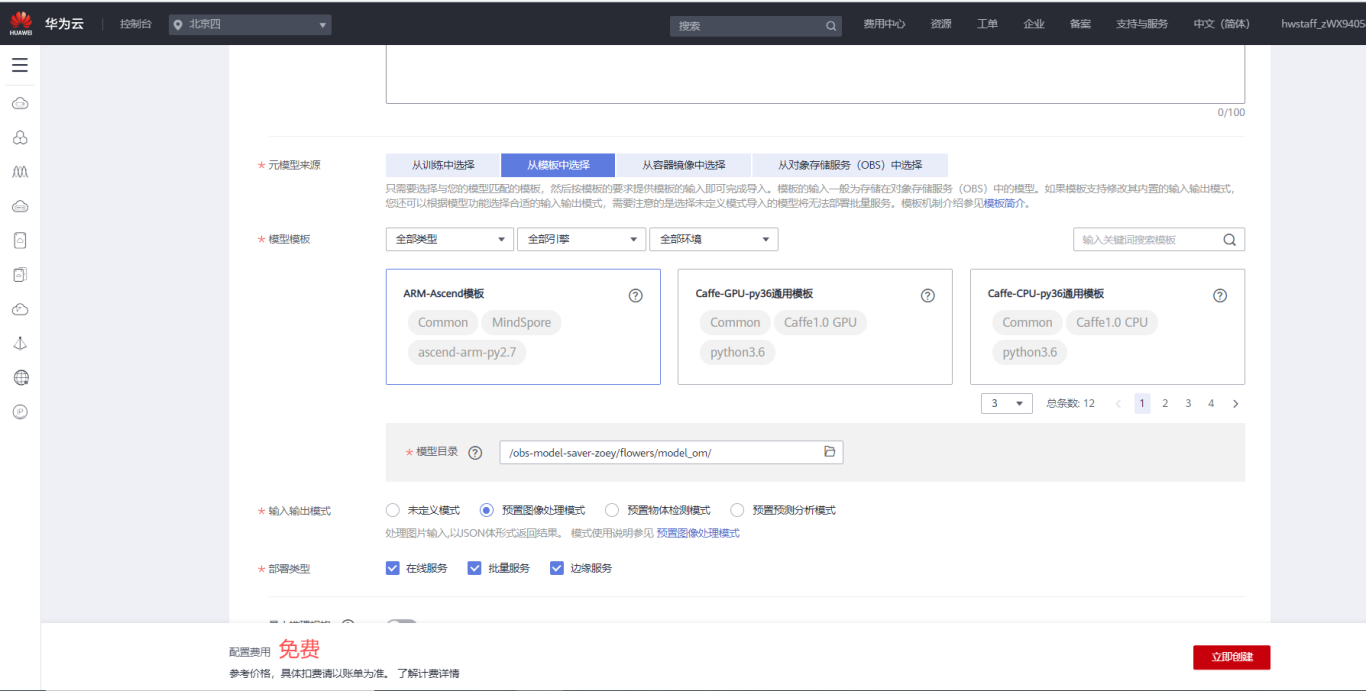
return outputs

导入模型

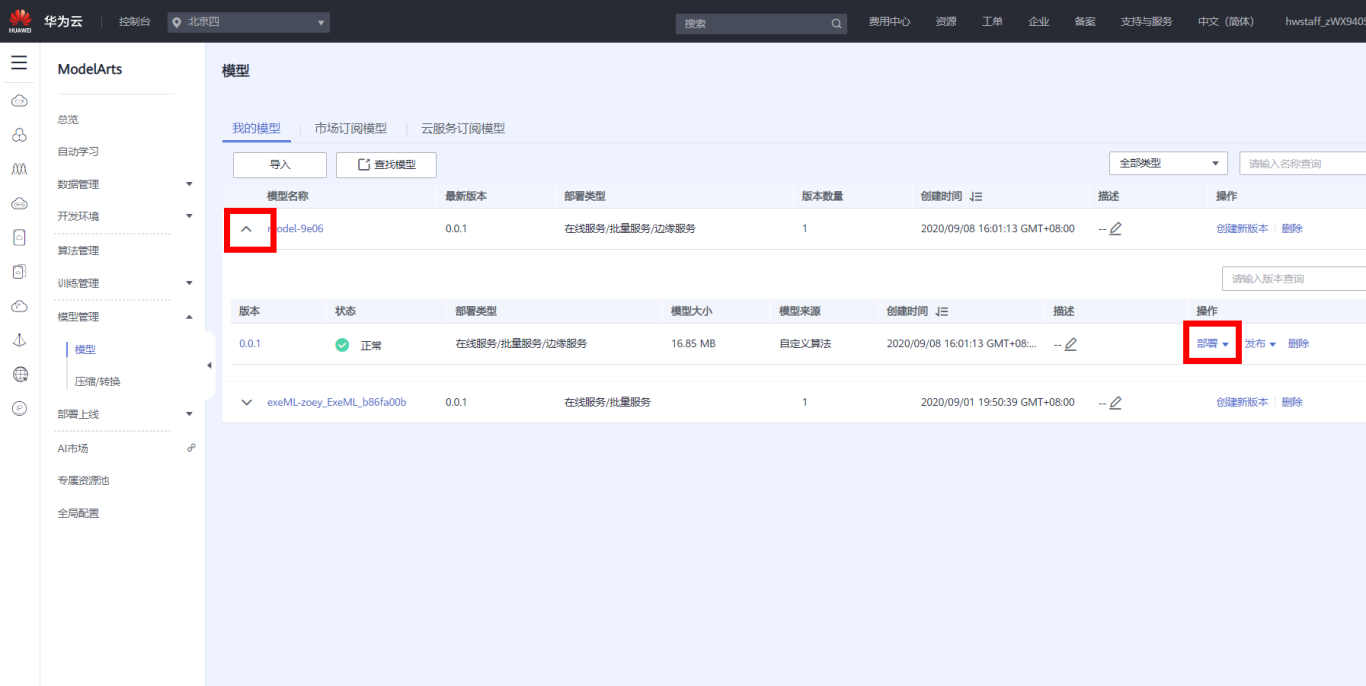
进入ModelArts控制台，点击模型管理>模型>导入，见下图。



按照下图填写，其中

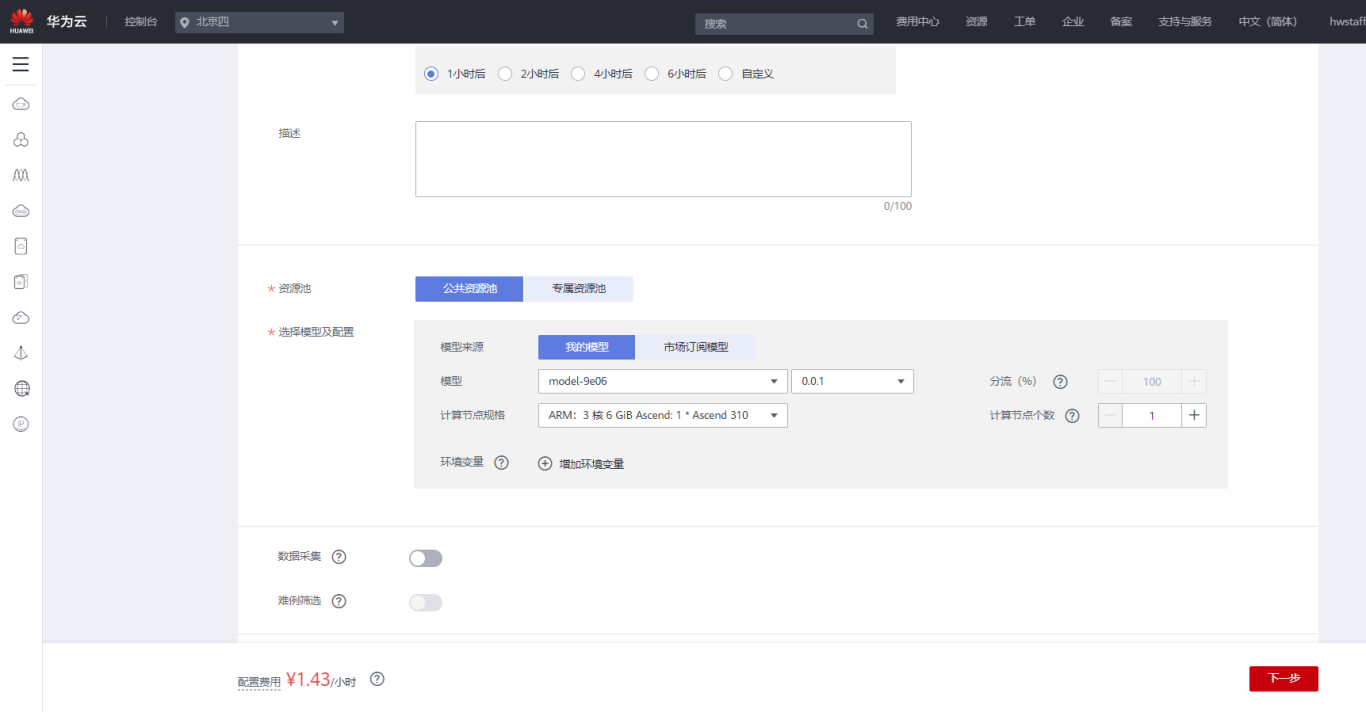


等待几分钟，模型导入成功后点击三角箭头>部署>在线服务，如下图所示：

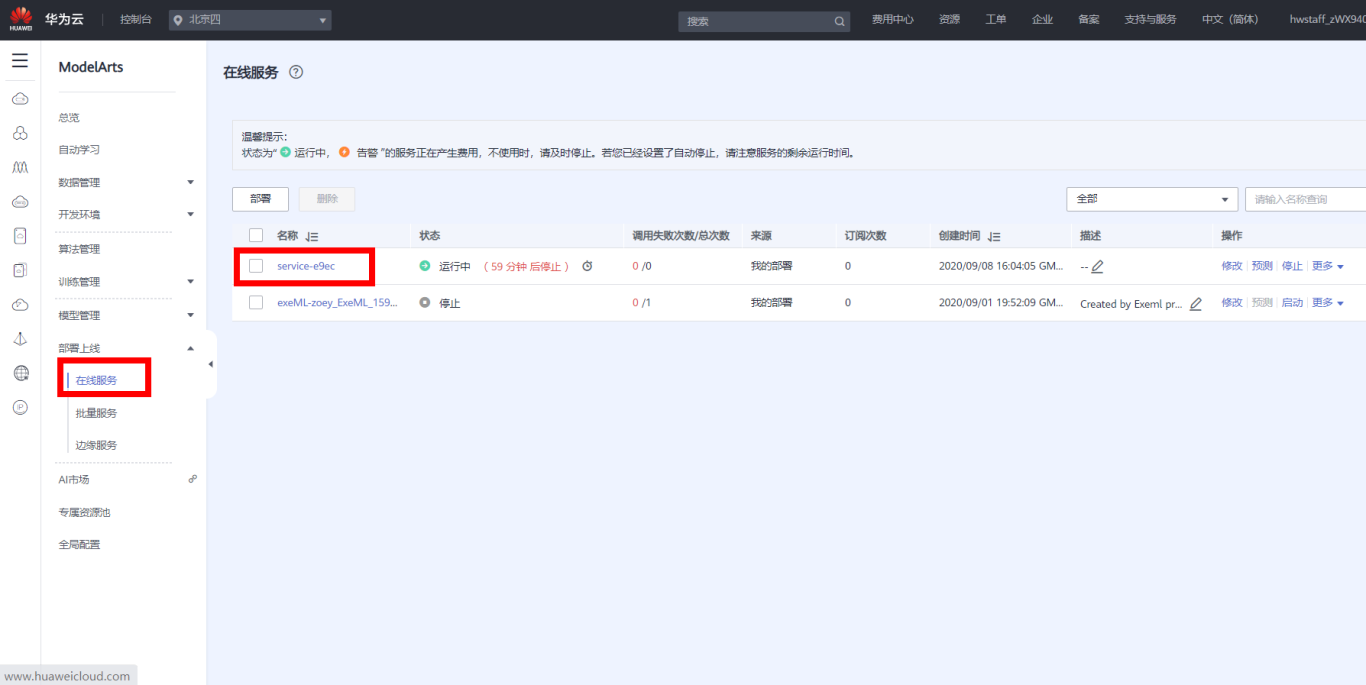


界面会自动跳转到部署上线>在线服务界面。按下图所示填写

其中模型名字必须与前面导入的模型名字相同，计算节点规格选择Ascend310



点击下一步并提交。进入部署在线>在线服务,点击所对应的模型。如下图所示：



预测

点击上图中预测进入预测界面，点击预测>上传。如下图所示：



上传在本地目录flower\_photos\_test中选择图片，点击预测。结果如下图所示。



## 实验总结

本章提供了一个基于华为ModelArts平台的花卉图像识别实验。该实验演示了如何利用华为云ModelArts完成图像识别任务。通过MindSpore来构建图像识别模型，然后将模型部署到ModelArts上提供在线预测服务。