《机器学习基础》实验报告

年级、专业、班级	2019 级计算机科学与技术（卓越）02 班	姓名	李燕琴

实验题目	基于 MindSpore 的图像识别		
实验时间	2021/12/05	实验地点	DS1422
实验成绩		实验性质	□验证性 □设计性 □综合性

教师评价：

□算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；

□实验结果正确； □语法、语义正确； □报告规范；

其他：

评价教师签名：

一、实验目的

掌握 MindSpore 的使用，并掌握在华为云上部署模型的基本流程

二、实验项目内容

1.理解卷积神经网络，掌握 MindSpore 的使用

2.利用云平台部署算法

三、实验过程或算法（源程序）

（一）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
!rm flower_photos_train.zip

!wget https://professional.obs.cn-north-4.myhuaweicloud.com/flower_photos_test.zip
!unzip flower_photos_test.zip
!rm flower_photos_test.zip
```

（二）导入实验环境

1、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。

```
# 导入相应模块
from easydict import EasyDict as edict
import glob
import os
import numpy as np
import matplotlib.pyplot as plt
import mindspore
import mindspore.dataset as ds
import mindspore.dataset.vision.c_transforms as CV
import mindspore.dataset.transforms.c_transforms as C
from mindspore.dataset.vision import Inter
from mindspore.common import dtype as mstype
from mindspore import context
from mindspore.common.initializer import TruncatedNormal
from mindspore import nn
from mindspore.train import Model
from mindspore.train.callback import ModelCheckpoint, CheckpointConfig, LossMonitor, TimeMonitor
from mindspore import Tensor
from mindspore.train.serialization import export
from mindspore.ops import operations as P

```

```
context.set_context(mode=context.GRAPH_MODE,device_target="Ascend")
```

2、定义变量

```
cfg = edict({
    'data_path': './flower_photos_train',
    'test_path': './flower_photos_test',
    'data_size': 3618,
    'image_width': 128,  # 图片宽度
    'image_height': 128,  # 图片高度
    'batch_size': 48,
    'channel': 3,  # 图片通道数
    'num_class': 5,  # 分类类别
    'weight_decay': 0.0,
    'lr':0.001,  # 学习率
    'dropout_ratio': 0.9,
    'epoch_size': 20,  # 训练次数
    'sigma':0.01,

    'save_checkpoint_steps': 1,  # 多少步保存一次模型
    'keep_checkpoint_max': 3,  # 最多保存多少个模型
    'output_directory': './flowers/checkpoint',  # 保存模型路径
    'output_prefix': "CKP"  # 保存模型文件名字
})
```

3、读取数据集

数据读取并处理流程如下：

（1）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}

（2）使用 RandomCropDecodeResize、HWC2CHW、shuffle 进行数据预处理。

代码如下：

```
def read_data(path,config):
    de_dataset = ds.ImageFolderDataset(path, class_indexing={'daisy':0,'dandelion':1,'roses':2,'sunflowers':3,'tulips':4},num_parallel_workers=8)

    rescale = 1.0 / 127.5
    shift = -1.0
```

```
    transform_img=CV.RandomCropDecodeResize([config.image_width,config.image_height], scale=(0.08, 1.0), ratio=(0.75, 1.333))  #改变尺寸
    de_dataset = de_dataset.map(input_columns="image", operations=transform_img, num_parallel_workers=8)
    de_dataset = de_dataset.map(input_columns="image", operations=CV.Rescale(rescale, shift), num_parallel_workers=8)
    de_dataset = de_dataset.map(input_columns="image", operations=CV.HWC2CHW(), num_parallel_workers=8)
    # de_dataset = de_dataset.map(input_columns="image", operations=C.TypeCast(mstype.float32), num_parallel_workers=8)
    de_dataset = de_dataset.shuffle(buffer_size=cfg.data_size)

    de_dataset = de_dataset.batch(config.batch_size, drop_remainder=True)

    de_dataset = de_dataset.repeat(config.epoch_size)
    return de_dataset

de_train = read_data(cfg.data_path,cfg)
de_test = read_data(cfg.test_path,cfg)
print('训练数据集数量：',de_train.get_dataset_size()*cfg.batch_size)
print('测试数据集数量：',de_test.get_dataset_size()*cfg.batch_size)
de_dataset = de_train
data_next = de_dataset.create_dict_iterator()._get_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])

# 图像可视化
plt.figure()
plt.imshow(data_next['image'][0,...].asnumpy().T)
plt.colorbar()
plt.grid(False)
plt.show()
```

4、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。

（1）定义模型

```
# 参考官网示例实现
class ImgClfNet(nn.Cell):
    def __init__(self, num_class=5,channel=3,dropout_ratio=0.5,trun_sigma=0.01):
        super(ImgClfNet, self).__init__()
```

```
5.
6.          self.num_class = num_class  # 一共分五类
7.          self.channel = channel      # 图片通道数是 3
8.          self.dropout_ratio=dropout_ratio  # dropput 概率
9.
10.         self.relu = nn.ReLU()  # 激活函数
11.         self.max_pool2d = nn.MaxPool2d(kernel_size=2, stride=2,pad_mode="
    valid") # 最大池化
12.         self.dropout = nn.Dropout(self.dropout_ratio)  # drop_our 层
13.         self.batch_norm2d = nn.BatchNorm2d(num_features=32)
14.         # 四层卷积
15.         self.conv1 = nn.Conv2d(self.channel, 32,
16.                                kernel_size=5, stride=1, padding=0,
17.                                has_bias=True, pad_mode="same",
18.                                weight_init=TruncatedNormal(sigma=trun_sig
    ma),bias_init='zeros')
19.         self.conv2 = nn.Conv2d(32, 64,
20.                                kernel_size=5, stride=1, padding=0,
21.                                has_bias=True, pad_mode="same",
22.                                weight_init=TruncatedNormal(sigma=trun_sig
    ma),bias_init='zeros')
23.         self.conv3 = nn.Conv2d(64, 128,
24.                                kernel_size=3, stride=1, padding=0,
25.                                has_bias=True, pad_mode="same",
26.                                weight_init=TruncatedNormal(sigma=trun_sig
    ma),bias_init='zeros')
27.         self.conv4 = nn.Conv2d(128, 128,
28.                                kernel_size=3, stride=1, padding=0,
29.                                has_bias=True, pad_mode="same",
30.                                weight_init=TruncatedNormal(sigma=trun_sig
    ma), bias_init='zeros')
31.         # 张量展开
32.         self.flatten = nn.Flatten()
33.
34.         # 全连接层
35.         self.fc1 = nn.Dense(8*8*128, 1024,weight_init =TruncatedNormal(si
    gma=trun_sigma),bias_init = 0.1)
36.         self.fc2 = nn.Dense(1024, 512, weight_init=TruncatedNormal(sigma=
    trun_sigma), bias_init=0.1)
37.         self.fc3 = nn.Dense(512, self.num_class, weight_init=TruncatedNor
    mal(sigma=trun_sigma), bias_init=0.1)
38.
39.     def construct(self, x):
40.         # print(x.shape)
```

```
        x = self.conv1(x)
        # print(x.shape)
        x = self.batch_norm2d(x)
        x = self.relu(x)
        x = self.max_pool2d(x)

        # print(x.shape)
        x = self.conv2(x)
        # print(x.shape)
        x = self.relu(x)
        x = self.max_pool2d(x)

        # print(x.shape)
        x = self.conv3(x)
        # print(x.shape)
        x = self.max_pool2d(x)

        # print(x.shape)
        x = self.conv4(x)
        x = self.max_pool2d(x)

        # print(x.shape)
        x = self.flatten(x)
        # print(x.shape)
        x = self.fc1(x)
        x = self.relu(x)
        x = self.dropout(x)
        x = self.fc2(x)
        x = self.relu(x)
        x = self.dropout(x)
        x = self.fc3(x)
        return x

# 创建模型
net=ImgClfNet(num_class=cfg.num_class, channel=cfg.channel, dropout_ratio=cfg.dropout_ratio)
# 设置损失函数，交叉熵
net_loss = nn.SoftmaxCrossEntropyWithLogits(sparse=True, reduction="mean")
```

（2）自定义动态学习率

```
# 余弦学习率设置
```

```
cosine_decay_lr = nn.CosineDecayLR(min_lr=0.001, max_lr=0.01, decay_steps=4)
# 动态学习率配置
net_opt = nn.Momentum(net.trainable_params(), learning_rate=cosine_decay_lr, momentum=0.9)
```

（3）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。

```
model = Model(net, loss_fn=net_loss, optimizer=net_opt, metrics={"acc"})

loss_cb = LossMonitor(per_print_times=img_train.get_dataset_size())
cfg_ck = CheckpointConfig(save_checkpoint_steps=cfg.save_checkpoint_steps,keep_checkpoint_max=cfg.keep_checkpoint_max)
ckpt_cb = ModelCheckpoint(prefix=cfg.prefix, directory=cfg.directory, config=cfg_ck)
print("="*20+"模型开始训练"+"="*20)
model.train(cfg.epoch_size, img_train, callbacks=ckpt_cb, dataset_sink_mode=True)
```

（4）模型评估

```
# 使用测试集评估模型
img_test = get_img_data(cfg.test_path,cfg)
print('测试数据集数量：',img_test.get_dataset_size()*cfg.batch_size)
print("测试集合准确率",model.eval(img_test))
```

5、模型保存和转换

（1）保存模型为 onnx 格式，并上传到桶里

```
# 导出 onnx 模型
import numpy as np
from mindspore import export
from mindspore import Tensor
input_tensor = Tensor(np.ones([1, 3,128,128]).astype(np.float32))
export(net, Tensor(input_tensor), file_name='ImgClfNet', file_format='ONNX')

# 导出到 obs 桶
import moxing as mox
```

```
10. mox.file.copy_parallel(src_url='./ImgClfNet.onnx', dst_url='obs://course-lyq/flower/onnx/best_model.onnx')
```

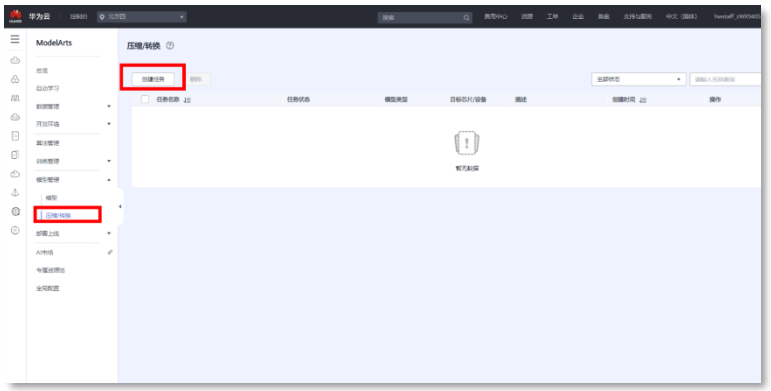
（2）上传 insert_op_conf.cfg 文件至桶内 onnx 模型目录下

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：

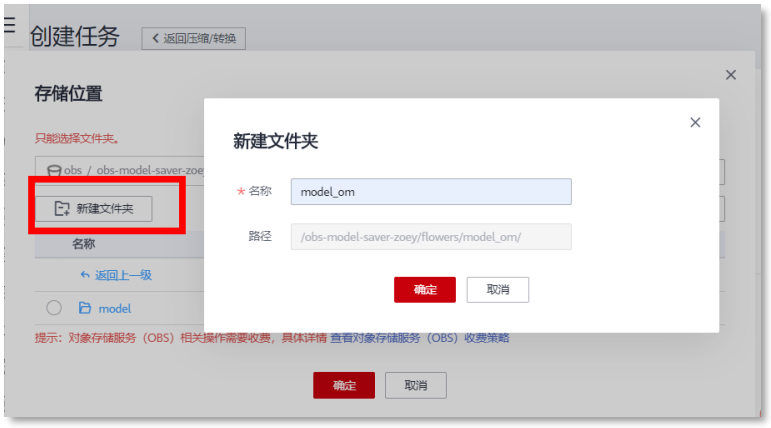
（如需下载 insert_op_conf.cfg 文件，请点击 https://professional-construction.obs.cn-north-4.myhuaweicloud.com:443/deep-learning/flowermodelfiles/insert_op_conf.cfg）

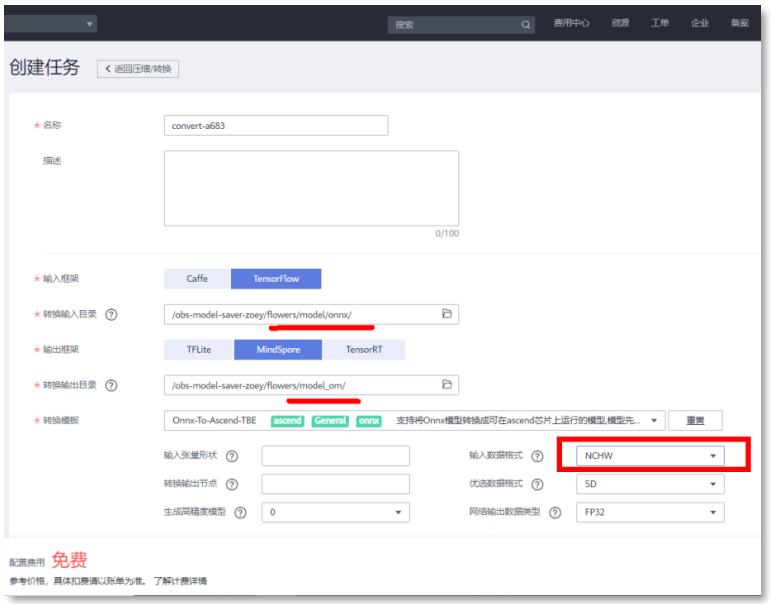
（3）将 onnx 格式的模型转换为 om 格式

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。

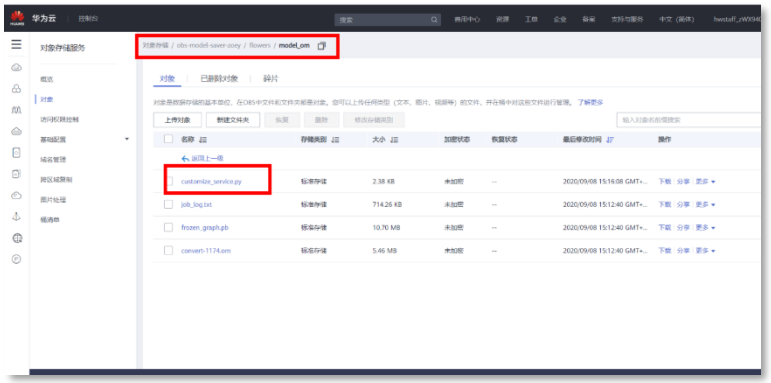
（4）模型部署上线

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：

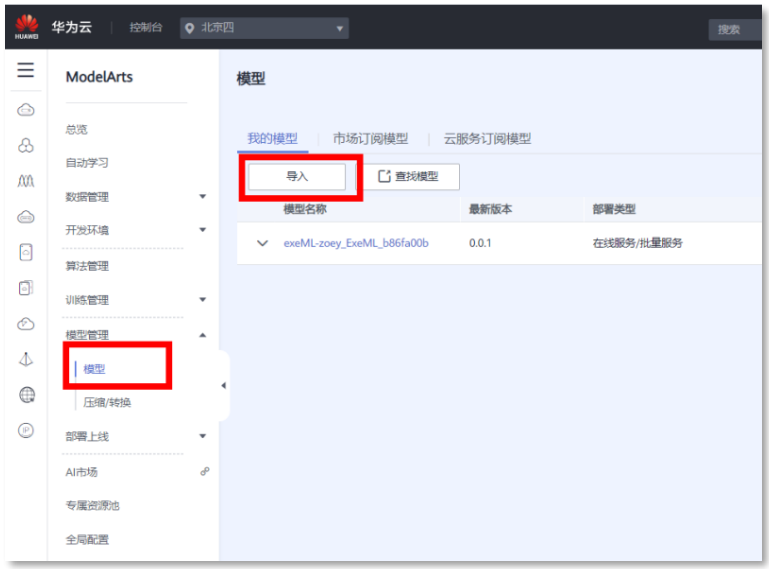
```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

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())

```

```
32.
33. def keep_ratio_resize(im, base=256):
34.   short_side = min(float(im.size[0]), float(im.size[1]))
35.   resize_ratio = base / short_side
36.   resize_sides = int(round(resize_ratio * im.size[0])), int(round(resize_
    ratio * im.size[1]))
37.   im = im.resize(resize_sides)
38.   return im
39.
40.
41. def central_crop(im, base=224):
42.   width, height = im.size
43.   left = (width - base) / 2
44.   top = (height - base) / 2
45.   right = (width + base) / 2
46.   bottom = (height + base) / 2
47.   # Crop the center of the image
48.   im = im.crop((left, top, right, bottom))
49.   return im
50.
51.
52. class DemoService(HiaiBaseService):
53.
54.   def _preprocess(self, data):
55.
56.     preprocessed_data = {}
57.     images = []
58.     for k, v in data.items():
59.       for file_name, file_content in v.items():
60.         image = Image.open(file_content)
61.         image = keep_ratio_resize(image, base=128)
62.         image = central_crop(image, base=128)
63.         image = np.array(image)  # HWC
64.         # AIPP should use RGB format.
65.         # mean reg is applied in AIPP.
66.         # Transpose is applied in AIPP
67.         tensor = NNTensor(image)
68.         images.append(tensor)
69.     tensor_list = NNTensorList(images)
70.     preprocessed_data['images'] = tensor_list
71.     return preprocessed_data
72.
73.   def _inference(self, data, image_info=None):
74.     result = {}
```

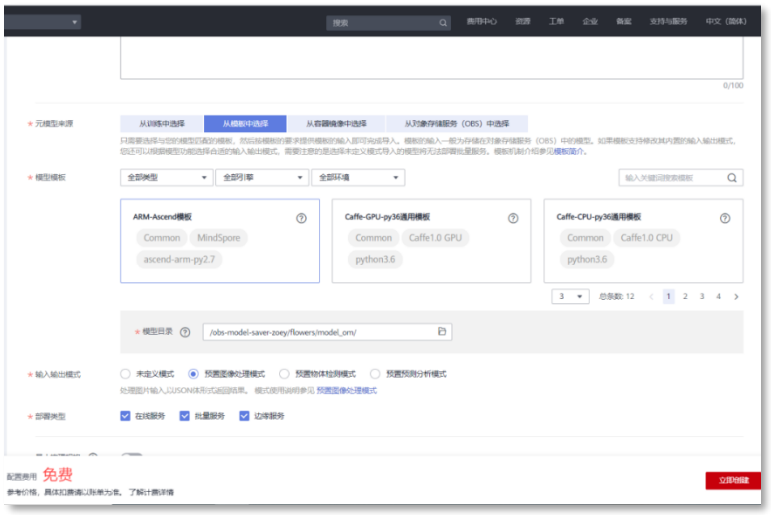
```
    for k, v in data.items():
      result[k] = self.model.proc(v)

    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
```

（5）导入模型

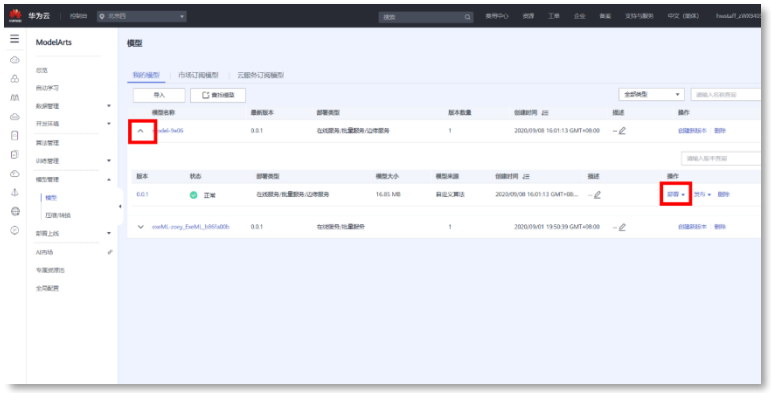
进入 ModelArts 控制台，点击模型管理>模型>导入，见下图。



按照下图填写，其中

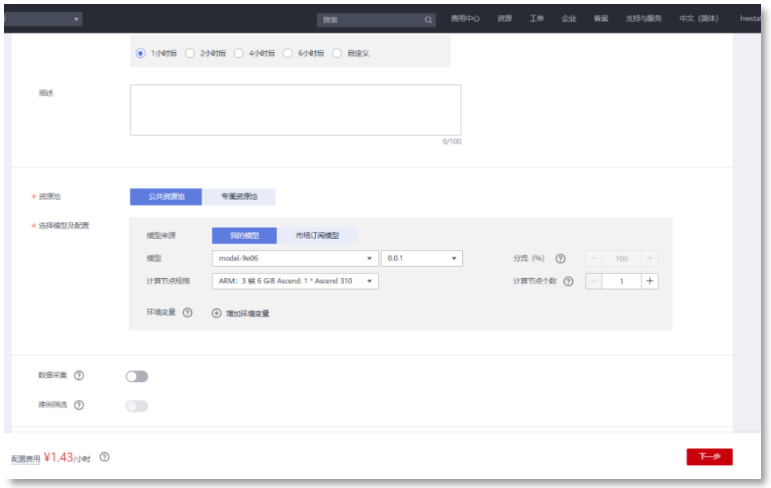


等待几分钟，模型导入成功后点击三角箭头>部署>在线服务，如下图所示：

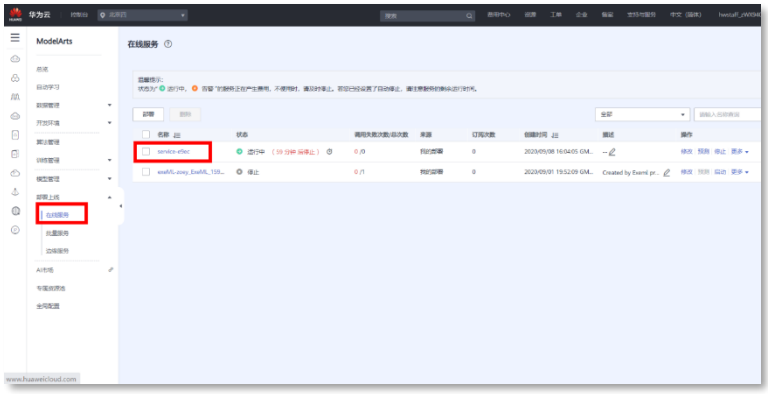


界面会自动跳转到部署上线>在线服务界面。按下图所示填写

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

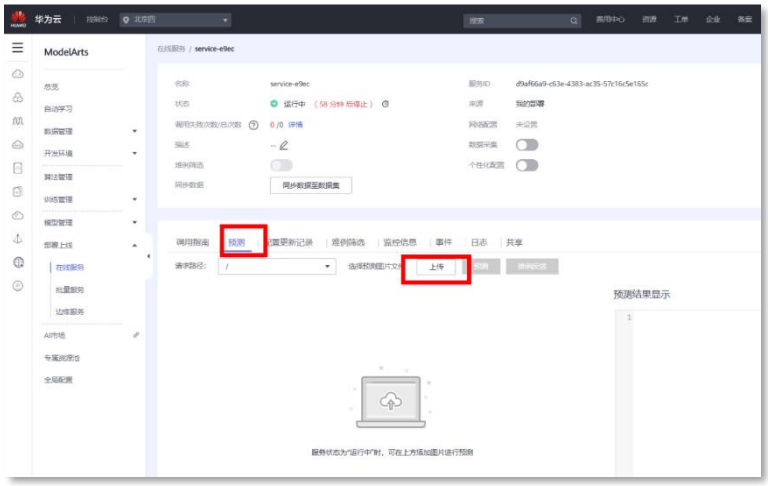


点击下一步并提交。进入部署在线>在线服务，点击所对应的模型。如下图所示：

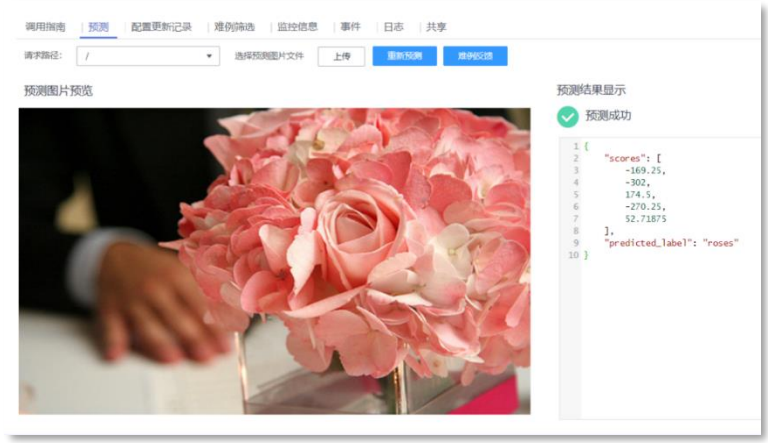


6、预测

点击上图中预测进入预测界面，点击预测>上传。如下图所示：



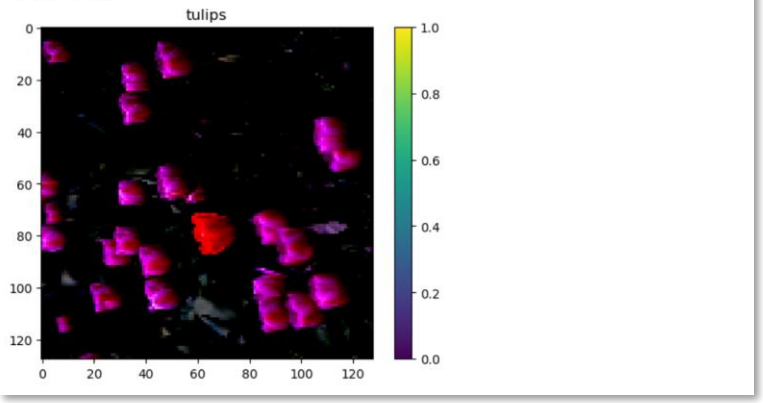
上传在本地目录 `flower_photos_test` 中选择图片，点击预测。结果如下图所示。



四、实验结果及分析

1、读取数据集，并初步可视化

```
训练数据集数量： 72320
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integer
s).
(C, H, W)= (3, 128, 128)
class = "tulips"
```



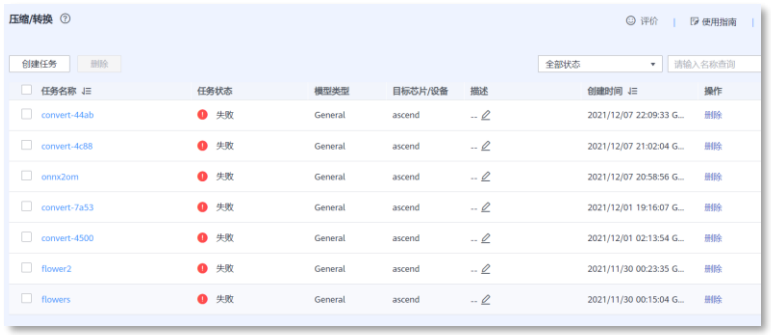
（2）模型训练过程图示

```
====================模型开始训练====================
epoch: 1 step: 2260, loss is 1.5966454
epoch: 2 step: 2260, loss is 1.5602721
epoch: 3 step: 2260, loss is 1.5815554
epoch: 4 step: 2260, loss is 1.5791204
epoch: 5 step: 2260, loss is 1.6485419
epoch: 6 step: 2260, loss is 1.6124294
epoch: 7 step: 2260, loss is 1.6330886
epoch: 8 step: 2260, loss is 1.5917475
epoch: 9 step: 2260, loss is 1.3166935
epoch: 10 step: 2260, loss is 1.2683109
epoch: 11 step: 2260, loss is 0.8609748
epoch: 12 step: 2260, loss is 0.98688114
epoch: 13 step: 2260, loss is 0.9312388
epoch: 14 step: 2260, loss is 0.49182147
epoch: 15 step: 2260, loss is 0.703841
epoch: 16 step: 2260, loss is 0.78158236
epoch: 17 step: 2260, loss is 0.8852223
epoch: 18 step: 2260, loss is 0.63129336
epoch: 19 step: 2260, loss is 0.7782753
epoch: 20 step: 2260, loss is 0.51701295
```

（3）模型评估

```
测试数据集数量： 640
测试集合准确率 {'acc': 0.7703125}
```

（4）压缩转换结果图



前期失败截图

后期成功截图

（5）部署预测图