基于MindSpore的图像识别全流程实验指导书

一、实验环境构建

1、登录华为云官网https://www.huaweicloud.com/?ticket=ST-4217587-ZMyac1H4moJ0pjSOHq4ubtbh-sso&locale=zh-cn，在搜索框输入对象存储服务进入控制台。（若已经创建过对象存储服务则直接跳到第4步项目创建）

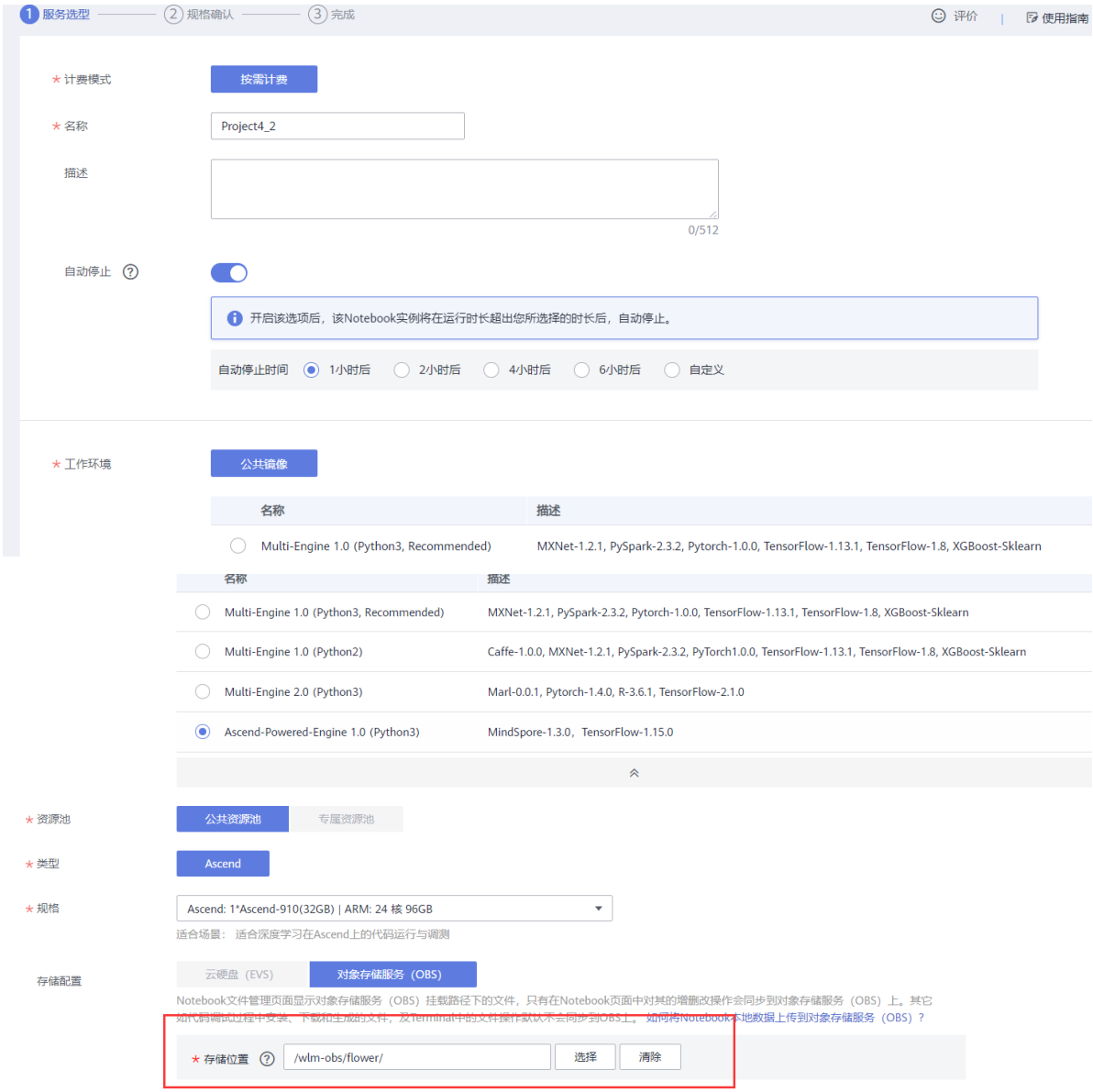
2、选择桶列表中的创建桶操作。

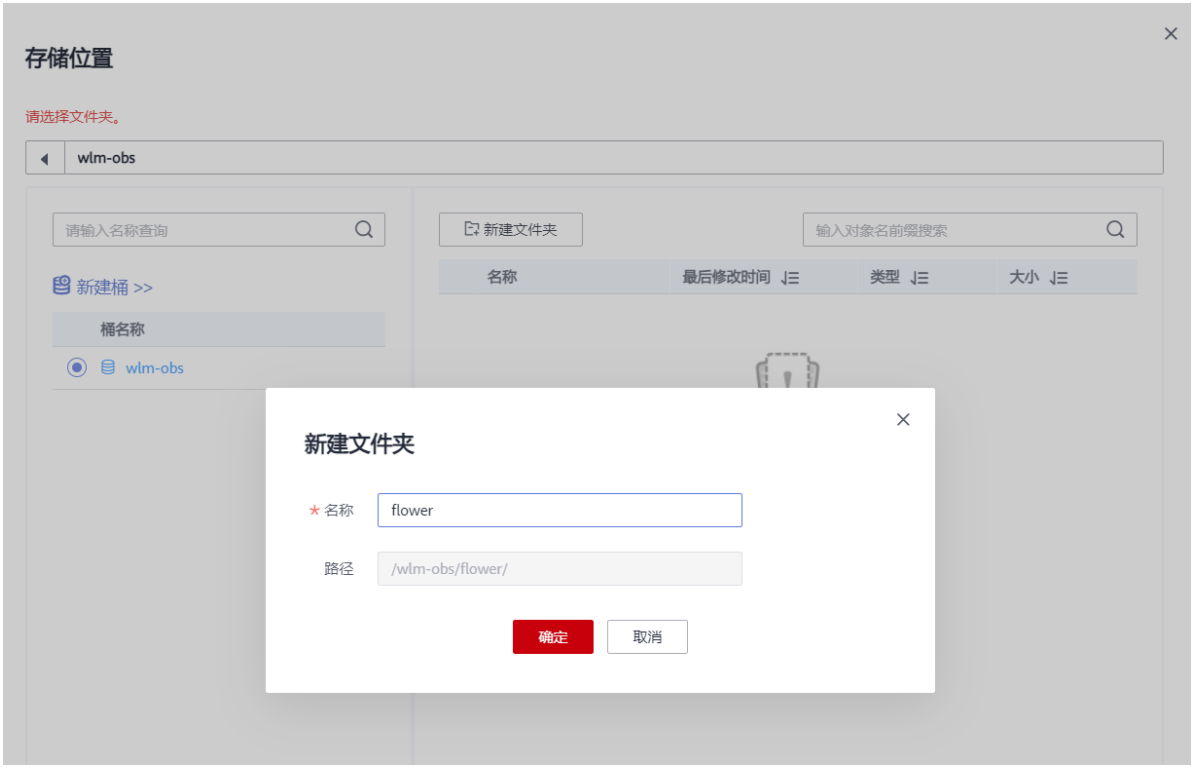
3、资源的相关配置情况如下，区域选择华北-北京四，桶名称需设置为全局唯一，桶默认私有，点击立即创建。

4、项目创建，在搜索框输入ModelArts进入控制台。

5、点击开发环境，创建Notebook实例，注意此处采用旧版Notebook创建实例界面。

6、实例配置情况如下所示，存储位置选择之前创建的桶资源，选择新建一个flower的文件夹用于存储相关资源。

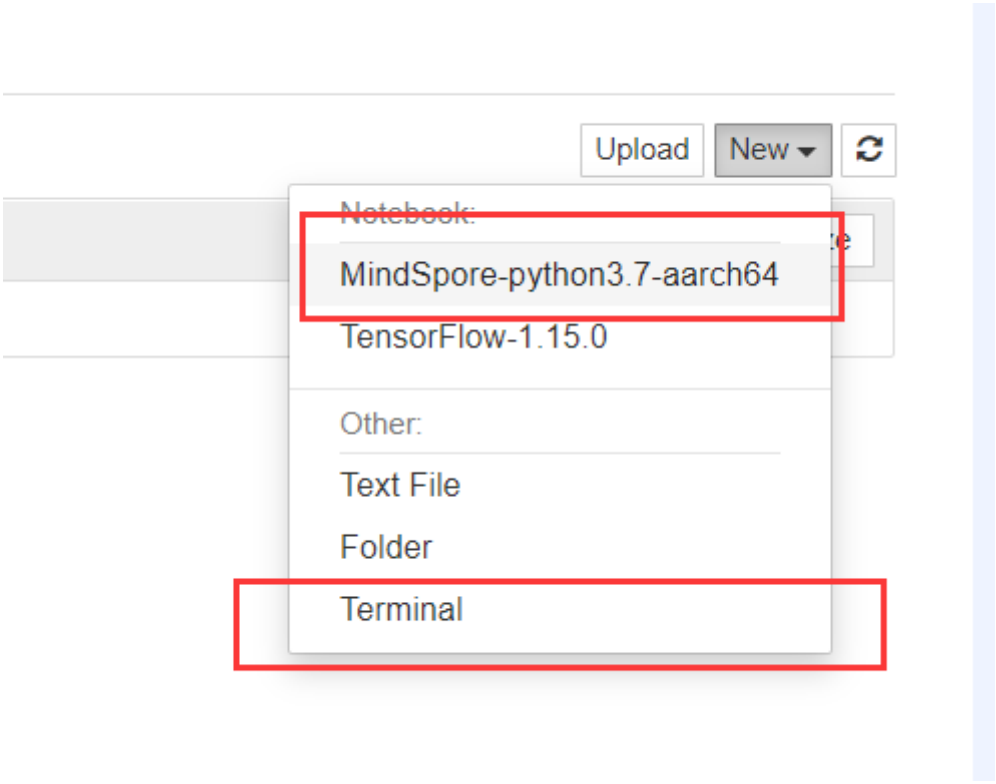


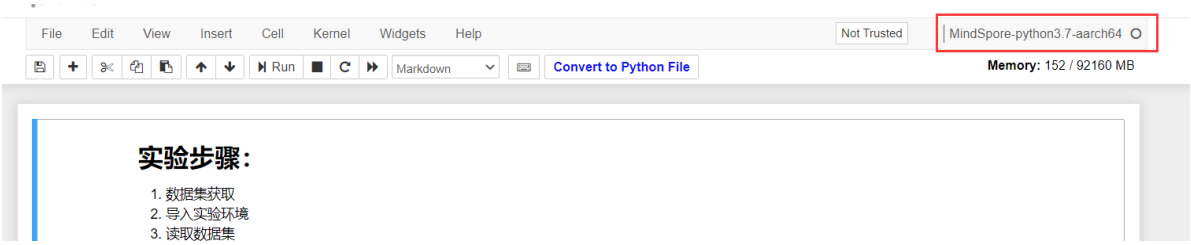


7、各项信息确认无误后，创建新环境。

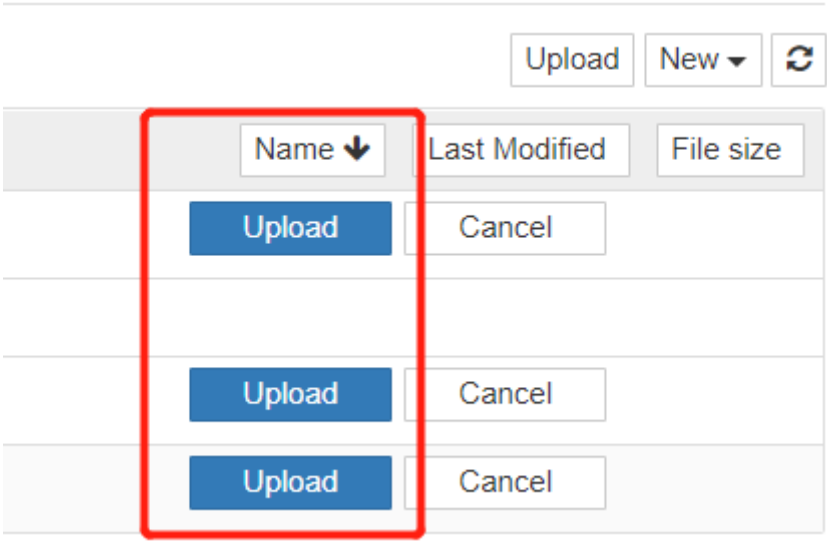
8、打开Notebook实验环境进行实验。

9、在实验中可以创建相关的终端和代码实例，需要注意的是，本实验环境应选择MindSpore-python3.7的内核。





10、若有额外资源可通过Upload上传，需要注意的是，需要点击蓝色的Upload按钮才能正式将资源上传至服务器。



二、实验设计

实验主要步骤如下，其中标注为核心部分的内容是需要在实验报告中着重讨论和研究的部分：

- 数据集获取
- 实验环境导入
- 读取数据集

- 模型构建训练（核心）
- 模型预测（核心）
- 模型保存和转换
- 模型部署上线

三、实验流程

1、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。

```
#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
```

2、实验环境导入

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。

```
# 隐藏警告
import warnings
warnings.filterwarnings('ignore')

#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")
```

3、定义模型的相关变量

此处仅给出一个样例，具体参数可根据实验模型构建需求确定。

```
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,
})
```

4、数据集读取及预处理（此处仅给出一种数据读取及预处理的样例，具体可根据自己模型需求进行调整）

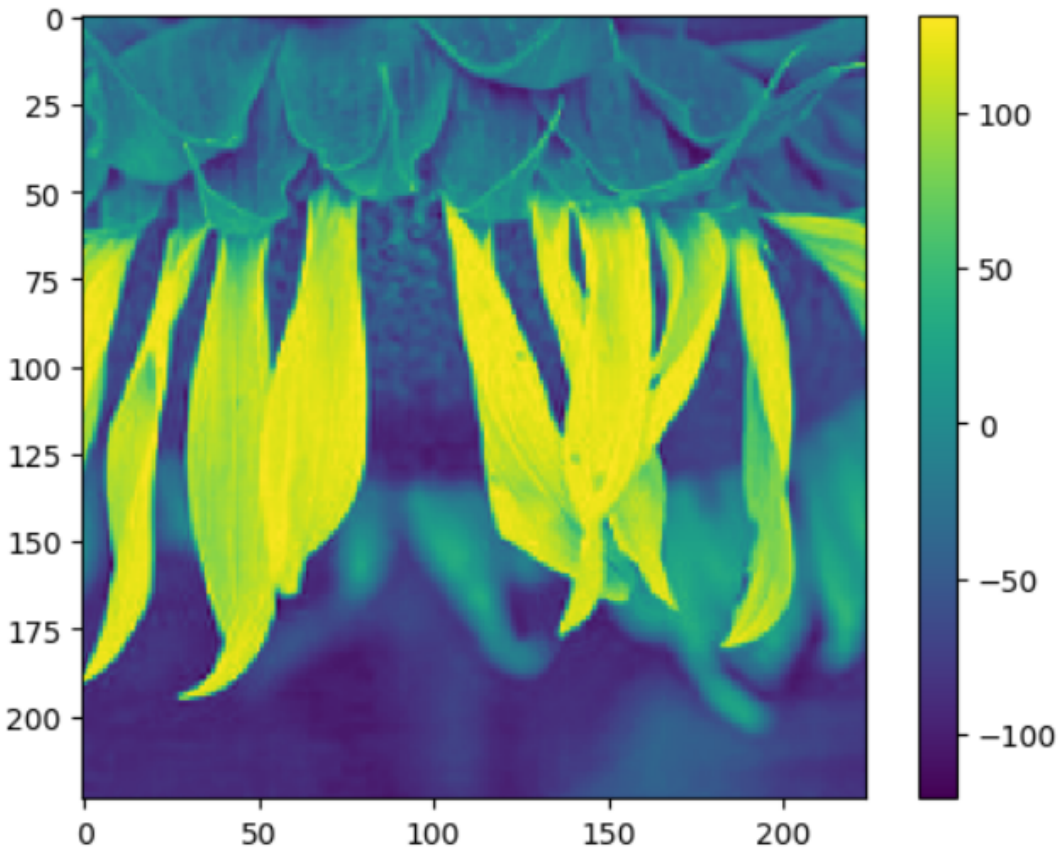
数据读取并处理流程如下：

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。

```
# 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()
```

```
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
```



5、模型构建训练（调用现有模型框架实现即可）

该步骤是实验核心部分，主要考察对图像识别模型的构建以及训练设计。可考虑采用常见的ResNet50等轻量级图像识别模型进行实验图像识别任务的构建。

该步骤总体流程如下：

- 数据预处理（参考第4步给出的样例，根据不同图像识别模型的需求进行预处理调整）
- 神经网络定义（定义基本的网络结果及初始化方式）
- 损失函数定义（选择合适的损失函数进行误差计算）
- 优化器定义（计算和更新训练参数的梯度）
- 模型训练（定义训练迭代的方式）
- 模型评估（准确率等指标进行模型评估）

6、模型预测

导入测试集数据，利用已训练好的数据内容对模型的结果进行预测。

7、模型保存和转换

- 将模型保存为onnx格式文件（此处以resnet50模型生成的模型文件为例进行示范）

```
#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')
```

- 将模型存储至自己创建的obs桶中，注意根据自己创建的桶名修改存储路径

```
import moxing
moxing.file.copy_parallel(src_url='./flowers/best_model.onnx',
dst_url='s3://wlm-obs/flower/onnx/best_model.onnx')
```

- 上传华为云服务的配置文件insert_op_conf.cfg文件至桶内的/flowers/model/onnx目录下，将insert_op_conf.cfg文件放置在best_model.onnx的同目录下。（配置文件的下载链接为https://professional-construction.obs.cn-north-4.myhuaweicloud.com:443/deep-learning/flowermodelfiles/insert_op_conf.cfg）

  进入华为云的对象存储服务，点击自己创建的桶。

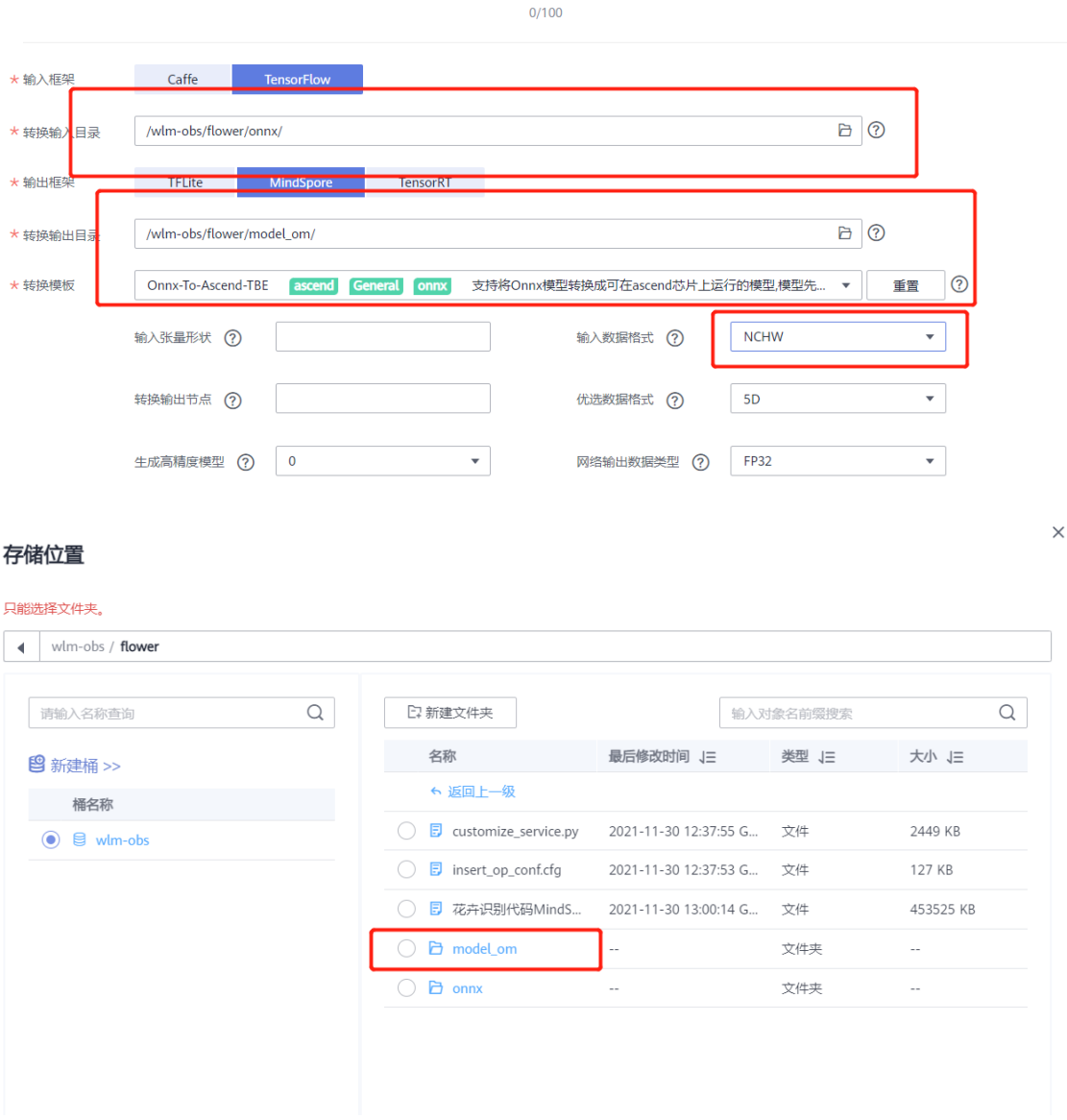
选择对象中的上传对象操作。

上传insert_op_conf.cfg配置文件。

- 将onnx格式模型转换为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的model_om目录(选择新建文件夹进行创建)。



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

9、模型部署上线

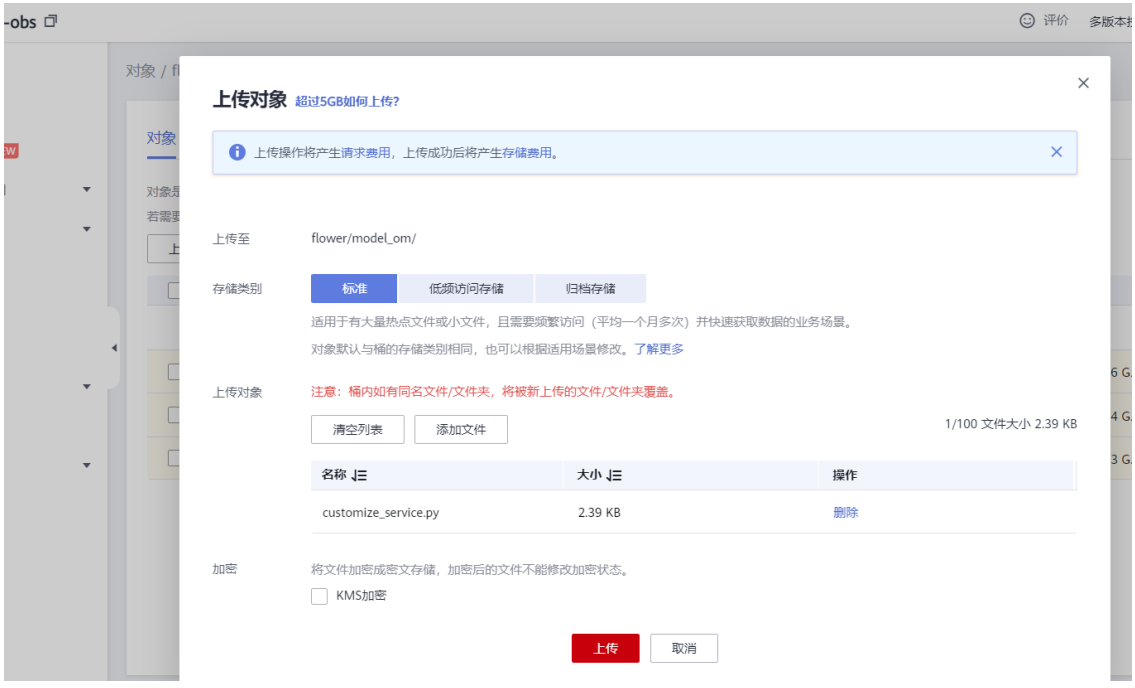
- 编写测试数据读取代码customize_service.py，上传编写测试数据读取代码文件至om模型目录下。（给出一个样例如下所示，其下载地址为：https://chuangxin.obs.cn-north-4.myhuaweicloud.com:443/MS1.0fulldeployment/customize_service.py）

```
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():
```

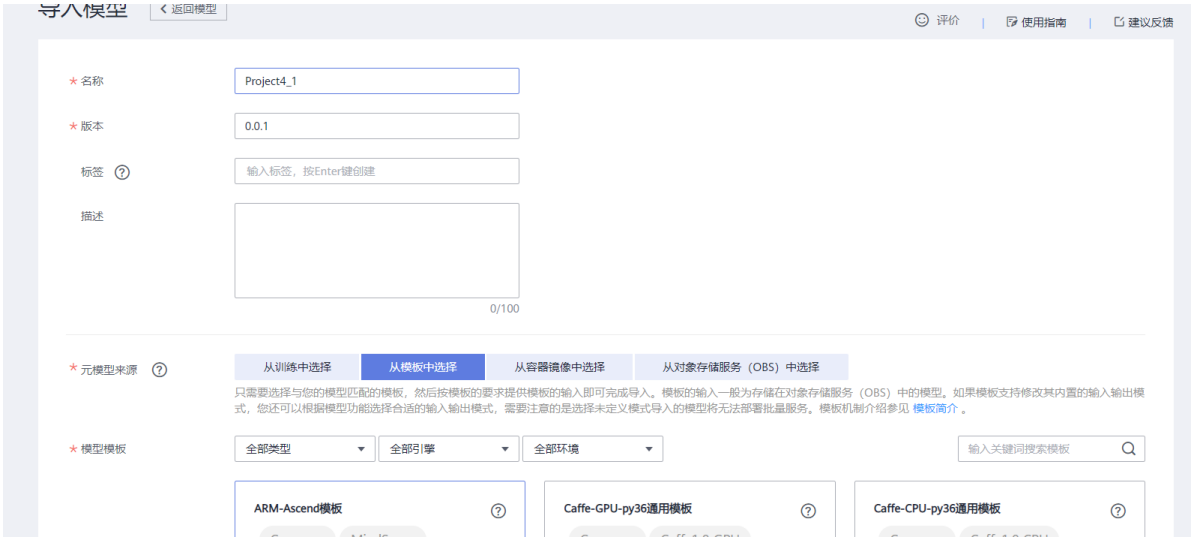
```
      for file_name, file_content in v.items():
        image = Image.open(file_content)
        image = keep_ratio_resize(image, base=256)
        image = central_crop(image, base=224)
        image = np.array(image)  # HWC
        # AIPP should use RGB format.
        # mean reg is applied in AIPP.
        # 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
```

- 在OBS桶内上传customize_service.py 文件到om的目录下



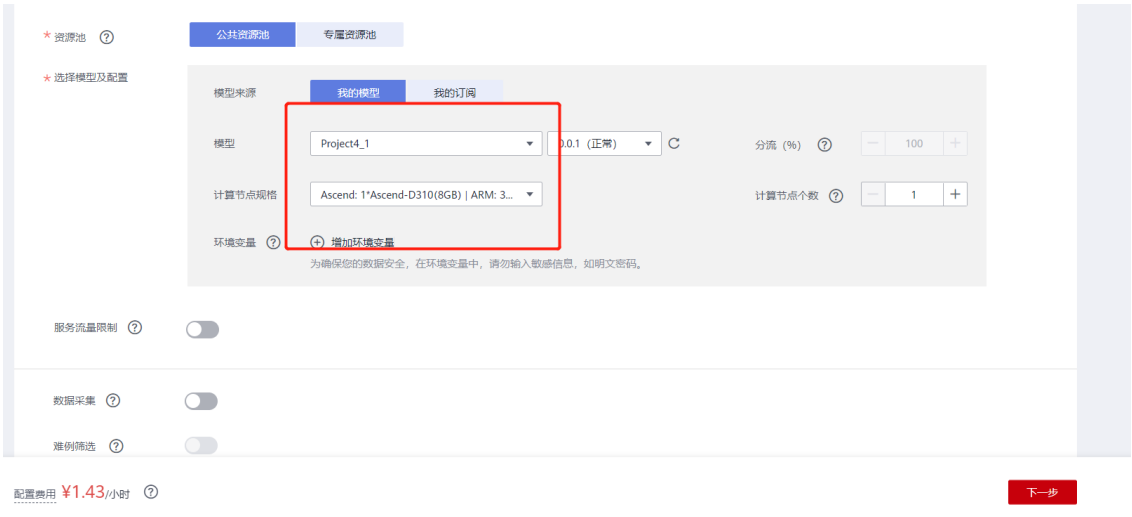
- 导入模型。进入ModelArts控制台，点击模型管理=>模型=>导入

- 相关配置如下所示



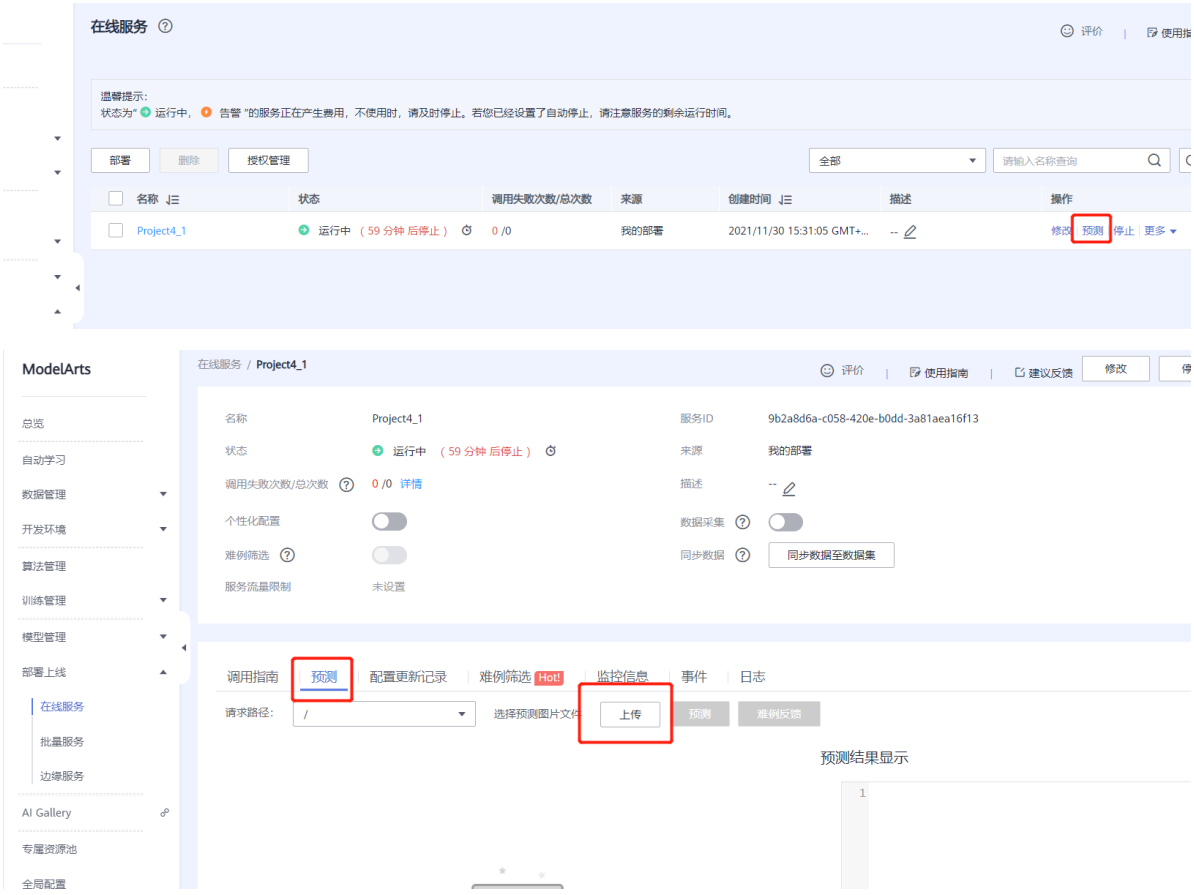
- 等待几分钟，模型导入成功后点击三角箭头=>部署=>在线服务，如下图所示。

- 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

- 点击上图中预测进入预测界面，点击预测=>上传，选取实验要求图像数据进行预测。



- 预测结果如下，实验完成

四、实验提交材料及评分标准

1、实验提交材料

- 图像识别模型的notebook代码
- 实验报告（包括在线部署相关结果和完整实验流程）

2、评分标准

得分	评价依据
0分	抄袭或被抄袭，未提交实验相关材料
1-4分	未完成第二节实验设计的全部内容，根据完成内容量情况进行评分
5分	完成实验全部内容，但实验报告缺少必要的叙述以及中间结果
6分	完成全部实验内容，实验报告内容详实，有自己对模型的探索和尝试