华为“智能基座”系列课程

基于MindSpore+Orangepi AIpro的MobileNetv2垃圾分类训推全流程

版本：2.0



华为技术有限公司

|  |
| --- |
| 版权所有 © 华为技术有限公司 2024。 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 基于MindSpore+Orangepi AIpro的MobileNetv2垃圾分类训推全流程 2](#_Toc175988987)

[1.1 实验介绍 2](#_Toc175988988)

[1.1.1 实验介绍 2](#_Toc175988989)

[1.1.2 数据集介绍 2](#_Toc175988990)

[1.1.3 模型介绍 3](#_Toc175988991)

[1.1.4 实验环境 4](#_Toc175988992)

[1.1.5 实验目的 5](#_Toc175988993)

[1.2 MindSpore模型训练 5](#_Toc175988994)

[1.2.1 环境准备 5](#_Toc175988995)

[1.2.2 项目下载 5](#_Toc175988996)

[1.2.3 模型训练 6](#_Toc175988997)

[1.3 香橙派 AI Pro端侧部署 22](#_Toc175988998)

[1.3.1 环境准备 22](#_Toc175988999)

[1.3.2 下载项目 23](#_Toc175989000)

[1.3.3 离线模型转换 24](#_Toc175989001)

[1.3.4 执行模型推理 24](#_Toc175989002)

[1.4 实验小结 27](#_Toc175989003)

# 基于MindSpore+Orangepi AIpro的MobileNetv2垃圾分类训推全流程

## 实验介绍

### 实验介绍

垃圾分类有利于资源节约与循环利用，也有利于保护自然生态环境。但在垃圾分类实施过程中，市民往往并不清楚垃圾该如何分类，而且每个城市实施的垃圾分类标准也不一样，所以开发一个垃圾分类的应用有助于普及垃圾分类的知识，精确分类垃圾，解决市民的困惑，促进垃圾分类政策的执行。

我们使用现实生活中常见的垃圾分类数据集，基于Ascend910和Ascend310B算力平台，实现垃圾分类模型训练+推理部署的全流程实验。首先在Ascend910算力平台使用迁移学习训练得到MobileNetv2的垃圾分类模型，接下来将其部署在香橙派AI Pro上，使用Ascend310B算力平台实现端侧推理。

### 数据集介绍

在本实验中，我们将常见生活垃圾分为以下4大类，26个子类：

干垃圾：贝壳、打火机、旧镜子、扫把、陶瓷碗、牙刷、一次性筷子、脏污衣服。

可回收物：报纸、玻璃制品、篮球、塑料瓶、硬纸板、玻璃瓶、金属制品、帽子、易拉罐、纸张。

湿垃圾：菜叶、橙皮、蛋壳、香蕉皮。

有害垃圾：电池、药片胶囊、荧光灯、油漆桶。

其对应的英文标签为：

Seashell, Lighter, Old Mirror, Broom, Ceramic Bowl, Toothbrush, Disposable Chopsticks, Dirty Cloth.

​Newspaper, Glassware, Basketball, Plastic Bottle, Cardboard, Glass Bottle, Metalware, Hats, Cans, Paper.

​Vegetable Leaf, Orange Peel, Eggshell, Banana Peel.

​Battery, Tablet capsules, Fluorescent lamp, Paint bucket.

数据集目录：

data\_en:

├─train:

├─Banana Peel

├─Basketball

├─Battery

…

└─Vegetable Leaf

├─test:

├─Banana Peel

├─Basketball

├─Battery

…

└─Vegetable Leaf

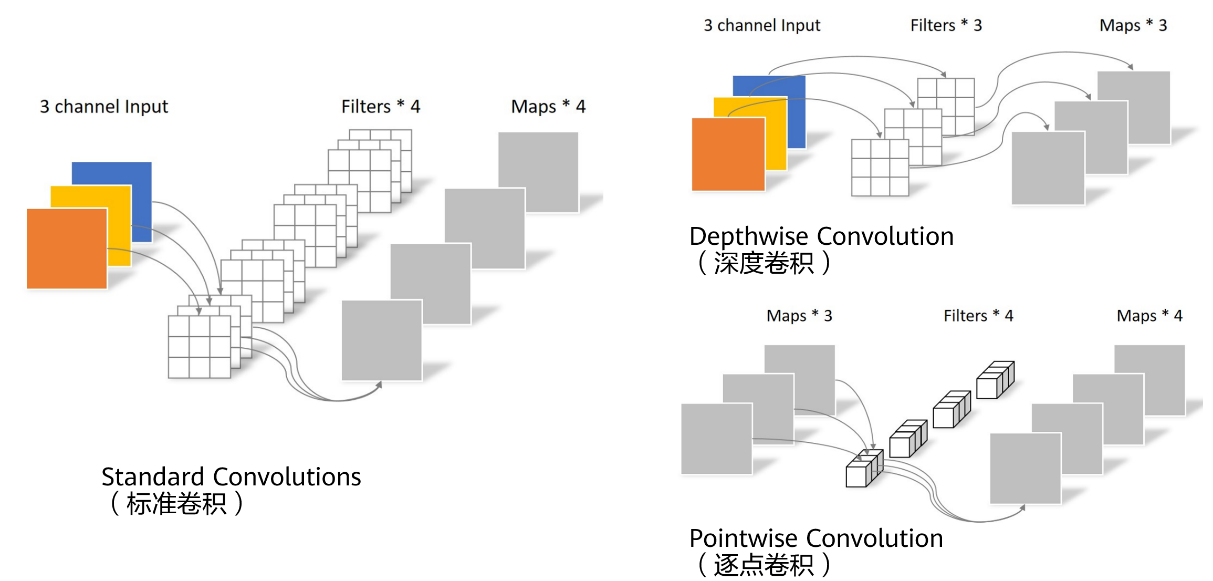
### 模型介绍

本实验我们使用MobileNetv2的预训练模型（即mobilenetV2-200\_1067.ckpt），对模型进行微调（Fine-tuning），只训练最后的FC层，并在训练过程中保存Checkpoint。

MobileNet网络于2017年提出，专注于移动端或者嵌入式设备中的轻量级CNN网络。相比传统卷积神经网络，MobileNet网络使用深度可分离卷积（Depthwise Separable Convolution）的思想在准确率小幅度降低的前提下，大大减小了模型参数与运算量。并引入宽度系数 α和分辨率系数 β使模型满足不同应用场景的需求。

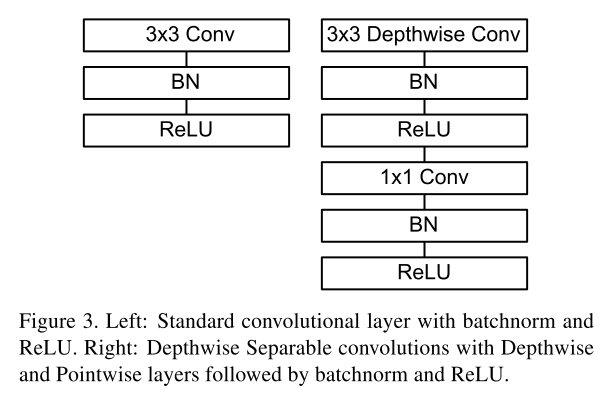
深度可分离卷积（Depthwise Separable Convolution）

将Standard Convolutions（标准卷积）分解为Depthwise Convolution（深度卷积）和Pointwise Convolution（逐点卷积），它默认一种假设，使用分解后的卷积效果和标准卷积效果是近似的。



深度可分离卷积

也就是将跨通道的 3X3 卷积换成单通道的 3X3 卷积+跨通道的 1X1 卷积：



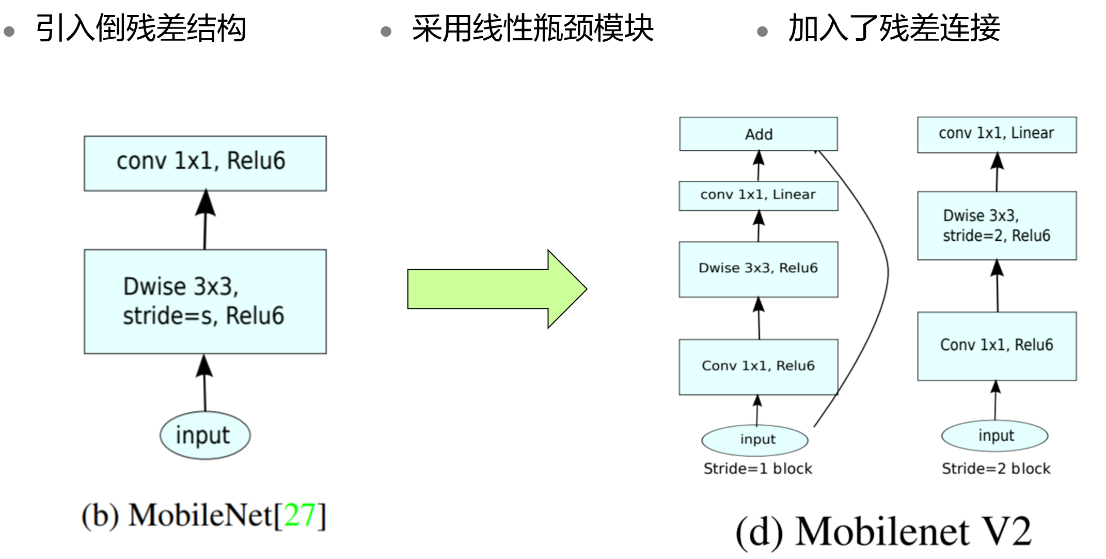
深度可分离卷积

MobileNetv2相比MobileNetv1，有以下改进：

引入倒残差结构

采用线性瓶颈模块

加入了残差连接



MobileNetv2与MobileNetv1的对比

### 实验环境

本实验分为两部分：

MindSpore模型训练：在（华为云ModelArts的）Ascend910开发环境中执行。

香橙派 AI Pro端侧部署：在香橙派 AI Pro上的Ascend310B推理环境中执行。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验环境 | 实验平台 | AI 计算框架 | AI处理器/算力 | 软件 |
| MindSpore模型训练 | 华为云 ModelArts | MindSpore2.2.14 | Ascend 910 | Notebook环境，  Python3.9，  MindSpore2.2 |
| 香橙派 AI Pro端侧部署 | 香橙派 AI Pro | MindSpore2.2.14 | Ascend 310B | Ubuntu 22.04 LTS Arm64，  npu-drive 23.0.RC2， CANN 8.0，  MobaXterm 23 |

环境要求

### 实验目的

熟悉基于MindSpore的 MobileNetv2网络构建；

掌握模型转换工具的基本操作；

掌握AscendCL开发垃圾分类的推理应用；

掌握基于Ascend 910和Ascend310B算力的训练+推理全流程实践。

## MindSpore模型训练

### 环境准备

请参考下方《MindSpore实验环境搭建手册》：



升级MindSpore

本实验需要在MindSpore2.2.14运行，手动升级MindSpore

%env no\_proxy='a.test.com,127.0.0.1,2.2.2.2'

!pip install mindspore==2.2.14

安装download库

!pip install download -i https://pypi.tuna.tsinghua.edu.cn/simple

### 项目下载

下载数据集

from download import download

# 下载data\_en数据集

url = "https://ascend-professional-construction-dataset.obs.cn-north-4.myhuaweicloud.com:443/MindStudio-pc/data\_en.zip"

path = download(url, "./", kind="zip", replace=True)

下载预训练模型

from download import download

# 下载预训练权重文件

url = "https://ascend-professional-construction-dataset.obs.cn-north-4.myhuaweicloud.com:443/ComputerVision/mobilenetV2-200\_1067.zip"

path = download(url, "./", kind="zip", replace=True)

### 模型训练

导入相关模块

import math

import numpy as np

import os

import random

from matplotlib import pyplot as plt

from easydict import EasyDict

from PIL import Image

import numpy as np

import mindspore.nn as nn

from mindspore import ops as P

from mindspore.ops import add

from mindspore import Tensor

import mindspore.common.dtype as mstype

import mindspore.dataset as de

import mindspore.dataset.vision as C

import mindspore.dataset.transforms as C2

import mindspore as ms

from mindspore import set\_context, nn, Tensor, load\_checkpoint, save\_checkpoint, export

from mindspore.train import Model

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

os.environ['GLOG\_v'] = '3' # Log level includes 3(ERROR), 2(WARNING), 1(INFO), 0(DEBUG).

set\_context(mode=ms.GRAPH\_MODE, device\_target="Ascend", device\_id=0) # 设置采用图模式执行，设备为Ascend#

配置训练、验证、推理参数

# 垃圾分类数据集标签，以及用于标签映射的字典。

garbage\_classes = {

'干垃圾': ['贝壳', '打火机', '旧镜子', '扫把', '陶瓷碗', '牙刷', '一次性筷子', '脏污衣服'],

'可回收物': ['报纸', '玻璃制品', '篮球', '塑料瓶', '硬纸板', '玻璃瓶', '金属制品', '帽子', '易拉罐', '纸张'],

'湿垃圾': ['菜叶', '橙皮', '蛋壳', '香蕉皮'],

'有害垃圾': ['电池', '药片胶囊', '荧光灯', '油漆桶']

}

class\_cn = ['贝壳', '打火机', '旧镜子', '扫把', '陶瓷碗', '牙刷', '一次性筷子', '脏污衣服',

'报纸', '玻璃制品', '篮球', '塑料瓶', '硬纸板', '玻璃瓶', '金属制品', '帽子', '易拉罐', '纸张',

'菜叶', '橙皮', '蛋壳', '香蕉皮',

'电池', '药片胶囊', '荧光灯', '油漆桶']

class\_en = ['Seashell', 'Lighter','Old Mirror', 'Broom','Ceramic Bowl', 'Toothbrush','Disposable Chopsticks','Dirty Cloth',

'Newspaper', 'Glassware', 'Basketball', 'Plastic Bottle', 'Cardboard','Glass Bottle', 'Metalware', 'Hats', 'Cans', 'Paper',

'Vegetable Leaf','Orange Peel', 'Eggshell','Banana Peel',

'Battery', 'Tablet capsules','Fluorescent lamp', 'Paint bucket']

index\_en = {'Seashell': 0, 'Lighter': 1, 'Old Mirror': 2, 'Broom': 3, 'Ceramic Bowl': 4, 'Toothbrush': 5, 'Disposable Chopsticks': 6, 'Dirty Cloth': 7,

'Newspaper': 8, 'Glassware': 9, 'Basketball': 10, 'Plastic Bottle': 11, 'Cardboard': 12, 'Glass Bottle': 13, 'Metalware': 14, 'Hats': 15, 'Cans': 16, 'Paper': 17,

'Vegetable Leaf': 18, 'Orange Peel': 19, 'Eggshell': 20, 'Banana Peel': 21,

'Battery': 22, 'Tablet capsules': 23, 'Fluorescent lamp': 24, 'Paint bucket': 25}

# 训练超参

config = EasyDict({

"num\_classes": 26,

"image\_height": 224,

"image\_width": 224,

#"data\_split": [0.9, 0.1],

"backbone\_out\_channels":1280,

"batch\_size": 16,

"eval\_batch\_size": 8,

"epochs": 10,

"lr\_max": 0.05,

"momentum": 0.9,

"weight\_decay": 1e-4,

"save\_ckpt\_epochs": 1,

"save\_ckpt\_path": "./ckpt",

"dataset\_path": "./data\_en",

"class\_index": index\_en,

"pretrained\_ckpt": "./mobilenetV2-200\_1067.ckpt" # mobilenetV2-200\_1067.ckpt mobilenetv2\_ascend.ckpt

})

数据集预处理

利用ImageFolderDataset方法读取垃圾分类数据集，并对数据集进行预处理。

读取数据集时指定训练集和测试集，首先对整个数据集进行归一化、修改图像频道等预处理操作；然后对训练集的数据依次进行RandomCropDecodeResize、RandomHorizontalFlip、RandomColorAdjust、shuffle操作，以增加训练数据的丰富度；接着对测试集进行Decode、Resize、CenterCrop等预处理操作；最后返回处理后的数据集。

def create\_dataset(dataset\_path, config, training=True, buffer\_size=1000):

"""

create a train or eval dataset

Args:

dataset\_path(string): the path of dataset.

config(struct): the config of train and eval in diffirent platform.

Returns:

train\_dataset, val\_dataset

"""

data\_path = os.path.join(dataset\_path, 'train' if training else 'test')

ds = de.ImageFolderDataset(data\_path, num\_parallel\_workers=4, class\_indexing=config.class\_index)

resize\_height = config.image\_height

resize\_width = config.image\_width

normalize\_op = C.Normalize(mean=[0.485\*255, 0.456\*255, 0.406\*255], std=[0.229\*255, 0.224\*255, 0.225\*255])

change\_swap\_op = C.HWC2CHW()

type\_cast\_op = C2.TypeCast(mstype.int32)

if training:

crop\_decode\_resize = C.RandomCropDecodeResize(resize\_height, scale=(0.08, 1.0), ratio=(0.75, 1.333))

horizontal\_flip\_op = C.RandomHorizontalFlip(prob=0.5)

color\_adjust = C.RandomColorAdjust(brightness=0.4, contrast=0.4, saturation=0.4)

train\_trans = [crop\_decode\_resize, horizontal\_flip\_op, color\_adjust, normalize\_op, change\_swap\_op]

train\_ds = ds.map(input\_columns="image", operations=train\_trans, num\_parallel\_workers=4)

train\_ds = train\_ds.map(input\_columns="label", operations=type\_cast\_op, num\_parallel\_workers=4)

train\_ds = train\_ds.shuffle(buffer\_size=buffer\_size)

ds = train\_ds.batch(config.batch\_size, drop\_remainder=True)

else:

decode\_op = C.Decode()

resize\_op = C.Resize((int(resize\_width/0.875), int(resize\_width/0.875)))

center\_crop = C.CenterCrop(resize\_width)

eval\_trans = [decode\_op, resize\_op, center\_crop, normalize\_op, change\_swap\_op]

eval\_ds = ds.map(input\_columns="image", operations=eval\_trans, num\_parallel\_workers=4)

eval\_ds = eval\_ds.map(input\_columns="label", operations=type\_cast\_op, num\_parallel\_workers=4)

ds = eval\_ds.batch(config.eval\_batch\_size, drop\_remainder=True)

return ds

展示部分处理后的数据：

ds = create\_dataset(dataset\_path=config.dataset\_path, config=config, training=False)

print(ds.get\_dataset\_size())

data = ds.create\_dict\_iterator(output\_numpy=True).\_get\_next()

images = data['image']

labels = data['label']

for i in range(1, 5):

plt.subplot(2, 2, i)

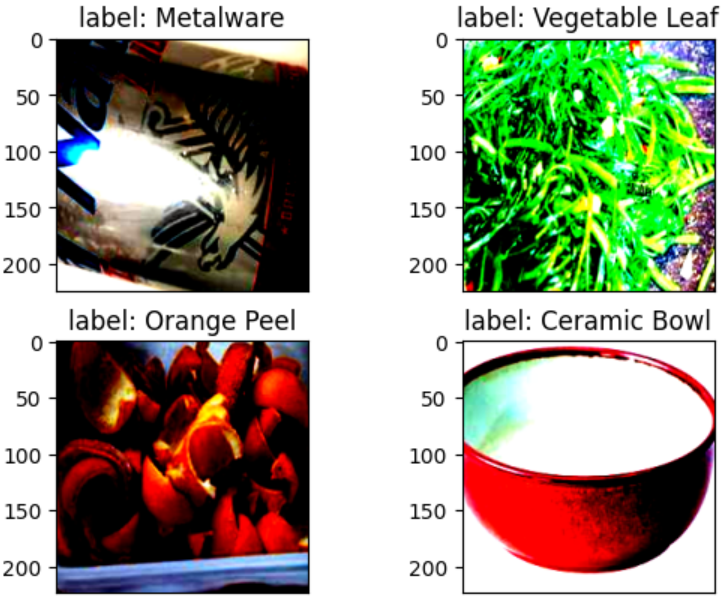
plt.imshow(np.transpose(images[i], (1,2,0)))

plt.title('label: %s' % class\_en[labels[i]])

plt.xticks([])

plt.show()

展示结果如下图所示：



测试图片

模型构建

使用MindSpore定义MobileNetV2网络的各模块时需要继承mindspore.nn.Cell。Cell是所有神经网络（Conv2d等）的基类。

神经网络的各层需要预先在\_\_init\_\_方法中定义，然后通过定义construct方法来完成神经网络的前向构造。原始模型激活函数为ReLU6，池化模块采用是全局平均池化层。

\_\_all\_\_ = ['MobileNetV2', 'MobileNetV2Backbone', 'MobileNetV2Head', 'mobilenet\_v2']

def \_make\_divisible(v, divisor, min\_value=None):

if min\_value is None:

min\_value = divisor

new\_v = max(min\_value, int(v + divisor / 2) // divisor \* divisor)

if new\_v < 0.9 \* v:

new\_v += divisor

return new\_v

class GlobalAvgPooling(nn.Cell):

"""

Global avg pooling definition.

Args:

Returns:

Tensor, output tensor.

Examples:

>>> GlobalAvgPooling()

"""

def \_\_init\_\_(self):

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

def construct(self, x):

x = P.mean(x, (2, 3))

return x

class ConvBNReLU(nn.Cell):

"""

Convolution/Depthwise fused with Batchnorm and ReLU block definition.

Args:

in\_planes (int): Input channel.

out\_planes (int): Output channel.

kernel\_size (int): Input kernel size.

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

groups (int): channel group. Convolution is 1 while Depthiwse is input channel. Default: 1.

Returns:

Tensor, output tensor.

Examples:

>>> ConvBNReLU(16, 256, kernel\_size=1, stride=1, groups=1)

"""

def \_\_init\_\_(self, in\_planes, out\_planes, kernel\_size=3, stride=1, groups=1):

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

padding = (kernel\_size - 1) // 2

in\_channels = in\_planes

out\_channels = out\_planes

if groups == 1:

conv = nn.Conv2d(in\_channels, out\_channels, kernel\_size, stride, pad\_mode='pad', padding=padding)

else:

out\_channels = in\_planes

conv = nn.Conv2d(in\_channels, out\_channels, kernel\_size, stride, pad\_mode='pad',

padding=padding, group=in\_channels)

layers = [conv, nn.BatchNorm2d(out\_planes), nn.ReLU6()]

self.features = nn.SequentialCell(layers)

def construct(self, x):

output = self.features(x)

return output

class InvertedResidual(nn.Cell):

"""

Mobilenetv2 residual block definition.

Args:

inp (int): Input channel.

oup (int): Output channel.

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

expand\_ratio (int): expand ration of input channel

Returns:

Tensor, output tensor.

Examples:

>>> ResidualBlock(3, 256, 1, 1)

"""

def \_\_init\_\_(self, inp, oup, stride, expand\_ratio):

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

assert stride in [1, 2]

hidden\_dim = int(round(inp \* expand\_ratio))

self.use\_res\_connect = stride == 1 and inp == oup

layers = []

if expand\_ratio != 1:

layers.append(ConvBNReLU(inp, hidden\_dim, kernel\_size=1))

layers.extend([

ConvBNReLU(hidden\_dim, hidden\_dim,

stride=stride, groups=hidden\_dim),

nn.Conv2d(hidden\_dim, oup, kernel\_size=1,

stride=1, has\_bias=False),

nn.BatchNorm2d(oup),

])

self.conv = nn.SequentialCell(layers)

self.cast = P.Cast()

def construct(self, x):

identity = x

x = self.conv(x)

if self.use\_res\_connect:

return P.add(identity, x)

return x

class MobileNetV2Backbone(nn.Cell):

"""

MobileNetV2 architecture.

Args:

class\_num (int): number of classes.

width\_mult (int): Channels multiplier for round to 8/16 and others. Default is 1.

has\_dropout (bool): Is dropout used. Default is false

inverted\_residual\_setting (list): Inverted residual settings. Default is None

round\_nearest (list): Channel round to . Default is 8

Returns:

Tensor, output tensor.

Examples:

>>> MobileNetV2(num\_classes=1000)

"""

def \_\_init\_\_(self, width\_mult=1., inverted\_residual\_setting=None, round\_nearest=8,

input\_channel=32, last\_channel=1280):

super(MobileNetV2Backbone, self).\_\_init\_\_()

block = InvertedResidual

# setting of inverted residual blocks

self.cfgs = inverted\_residual\_setting

if inverted\_residual\_setting is None:

self.cfgs = [

# t, c, n, s

[1, 16, 1, 1],

[6, 24, 2, 2],

[6, 32, 3, 2],

[6, 64, 4, 2],

[6, 96, 3, 1],

[6, 160, 3, 2],

[6, 320, 1, 1],

]

# building first layer

input\_channel = \_make\_divisible(input\_channel \* width\_mult, round\_nearest)

self.out\_channels = \_make\_divisible(last\_channel \* max(1.0, width\_mult), round\_nearest)

features = [ConvBNReLU(3, input\_channel, stride=2)]

# building inverted residual blocks

for t, c, n, s in self.cfgs:

output\_channel = \_make\_divisible(c \* width\_mult, round\_nearest)

for i in range(n):

stride = s if i == 0 else 1

features.append(block(input\_channel, output\_channel, stride, expand\_ratio=t))

input\_channel = output\_channel

features.append(ConvBNReLU(input\_channel, self.out\_channels, kernel\_size=1))

self.features = nn.SequentialCell(features)

self.\_initialize\_weights()

def construct(self, x):

x = self.features(x)

return x

def \_initialize\_weights(self):

"""

Initialize weights.

Args:

Returns:

None.

Examples:

>>> \_initialize\_weights()

"""

self.init\_parameters\_data()

for \_, m in self.cells\_and\_names():

if isinstance(m, nn.Conv2d):

n = m.kernel\_size[0] \* m.kernel\_size[1] \* m.out\_channels

m.weight.set\_data(Tensor(np.random.normal(0, np.sqrt(2. / n),

m.weight.data.shape).astype("float32")))

if m.bias is not None:

m.bias.set\_data(

Tensor(np.zeros(m.bias.data.shape, dtype="float32")))

elif isinstance(m, nn.BatchNorm2d):

m.gamma.set\_data(

Tensor(np.ones(m.gamma.data.shape, dtype="float32")))

m.beta.set\_data(

Tensor(np.zeros(m.beta.data.shape, dtype="float32")))

@property

def get\_features(self):

return self.features

class MobileNetV2Head(nn.Cell):

"""

MobileNetV2 architecture.

Args:

class\_num (int): Number of classes. Default is 1000.

has\_dropout (bool): Is dropout used. Default is false

Returns:

Tensor, output tensor.

Examples:

>>> MobileNetV2(num\_classes=1000)

"""

def \_\_init\_\_(self, input\_channel=1280, num\_classes=1000, has\_dropout=False, activation="None"):

super(MobileNetV2Head, self).\_\_init\_\_()

# mobilenet head

head = ([GlobalAvgPooling(), nn.Dense(input\_channel, num\_classes, has\_bias=True)] if not has\_dropout else

[GlobalAvgPooling(), nn.Dropout(0.2), nn.Dense(input\_channel, num\_classes, has\_bias=True)])

self.head = nn.SequentialCell(head)

self.need\_activation = True

if activation == "Sigmoid":

self.activation = nn.Sigmoid()

elif activation == "Softmax":

self.activation = nn.Softmax()

else:

self.need\_activation = False

self.\_initialize\_weights()

def construct(self, x):

x = self.head(x)

if self.need\_activation:

x = self.activation(x)

return x

def \_initialize\_weights(self):

"""

Initialize weights.

Args:

Returns:

None.

Examples:

>>> \_initialize\_weights()

"""

self.init\_parameters\_data()

for \_, m in self.cells\_and\_names():

if isinstance(m, nn.Dense):

m.weight.set\_data(Tensor(np.random.normal(

0, 0.01, m.weight.data.shape).astype("float32")))

if m.bias is not None:

m.bias.set\_data(

Tensor(np.zeros(m.bias.data.shape, dtype="float32")))

@property

def get\_head(self):

return self.head

class MobileNetV2(nn.Cell):

"""

MobileNetV2 architecture.

Args:

class\_num (int): number of classes.

width\_mult (int): Channels multiplier for round to 8/16 and others. Default is 1.

has\_dropout (bool): Is dropout used. Default is false

inverted\_residual\_setting (list): Inverted residual settings. Default is None

round\_nearest (list): Channel round to . Default is 8

Returns:

Tensor, output tensor.

Examples:

>>> MobileNetV2(backbone, head)

"""

def \_\_init\_\_(self, num\_classes=1000, width\_mult=1., has\_dropout=False, inverted\_residual\_setting=None, \

round\_nearest=8, input\_channel=32, last\_channel=1280):

super(MobileNetV2, self).\_\_init\_\_()

self.backbone = MobileNetV2Backbone(width\_mult=width\_mult, \

inverted\_residual\_setting=inverted\_residual\_setting, \

round\_nearest=round\_nearest, input\_channel=input\_channel, last\_channel=last\_channel).get\_features

self.head = MobileNetV2Head(input\_channel=self.backbone.out\_channel, num\_classes=num\_classes, \

has\_dropout=has\_dropout).get\_head

def construct(self, x):

x = self.backbone(x)

x = self.head(x)

return x

class MobileNetV2Combine(nn.Cell):

"""

MobileNetV2Combine architecture.

Args:

backbone (Cell): the features extract layers.

head (Cell): the fully connected layers.

Returns:

Tensor, output tensor.

Examples:

>>> MobileNetV2(num\_classes=1000)

"""

def \_\_init\_\_(self, backbone, head):

super(MobileNetV2Combine, self).\_\_init\_\_(auto\_prefix=False)

self.backbone = backbone

self.head = head

def construct(self, x):

x = self.backbone(x)

x = self.head(x)

return x

def mobilenet\_v2(backbone, head):

return MobileNetV2Combine(backbone, head)

动态学习率

一般情况下，模型训练时采用静态学习率，如0.01。随着训练步数的增加，模型逐渐趋于收敛，对权重参数的更新幅度应该逐渐降低，以减小模型训练后期的抖动。所以模型训练时可以采用动态下降的学习率， 这里使用cosine decay下降策略。

def cosine\_decay(total\_steps, lr\_init=0.0, lr\_end=0.0, lr\_max=0.1, warmup\_steps=0):

"""

Applies cosine decay to generate learning rate array.

Args:

total\_steps(int): all steps in training.

lr\_init(float): init learning rate.

lr\_end(float): end learning rate

lr\_max(float): max learning rate.

warmup\_steps(int): all steps in warmup epochs.

Returns:

list, learning rate array.

"""

lr\_init, lr\_end, lr\_max = float(lr\_init), float(lr\_end), float(lr\_max)

decay\_steps = total\_steps - warmup\_steps

lr\_all\_steps = []

inc\_per\_step = (lr\_max - lr\_init) / warmup\_steps if warmup\_steps else 0

for i in range(total\_steps):

if i < warmup\_steps:

lr = lr\_init + inc\_per\_step \* (i + 1)

else:

cosine\_decay = 0.5 \* (1 + math.cos(math.pi \* (i - warmup\_steps) / decay\_steps))

lr = (lr\_max - lr\_end) \* cosine\_decay + lr\_end

lr\_all\_steps.append(lr)

return lr\_all\_steps

模型训练

在进行正式的训练之前，定义训练函数，读取数据并对模型进行实例化，定义优化器和损失函数。

首先简单介绍损失函数及优化器的概念：

损失函数：又叫目标函数，用于衡量预测值与实际值差异的程度。深度学习通过不停地迭代来缩小损失函数的值。定义一个好的损失函数，可以有效提高模型的性能。

优化器：用于最小化损失函数，从而在训练过程中改进模型。

定义了损失函数后，可以得到损失函数关于权重的梯度。梯度用于指示优化器优化权重的方向，以提高模型性能。

在训练MobileNetV2之前对MobileNetV2Backbone层的参数进行了固定，使其在训练过程中对该模块的权重参数不进行更新；只对MobileNetV2Head模块的参数进行更新。

MindSpore支持的损失函数有SoftmaxCrossEntropyWithLogits、L1Loss、MSELoss等。这里使用SoftmaxCrossEntropyWithLogits损失函数。

训练测试过程中会打印loss值，loss值会波动，但总体来说loss值会逐步减小，精度逐步提高。每个人运行的loss值有一定随机性，不一定完全相同。

每打印一个epoch后模型都会在测试集上的计算测试精度，从打印的精度值分析MobileNetV2模型的预测能力在不断提升。

from mindspore.amp import FixedLossScaleManager

LOSS\_SCALE = 1024

train\_dataset = create\_dataset(dataset\_path=config.dataset\_path, config=config)

eval\_dataset = create\_dataset(dataset\_path=config.dataset\_path, config=config)

step\_size = train\_dataset.get\_dataset\_size()

backbone = MobileNetV2Backbone() #last\_channel=config.backbone\_out\_channels

# Freeze parameters of backbone. You can comment these two lines.

for param in backbone.get\_parameters():

param.requires\_grad = False

# load parameters from pretrained model

load\_checkpoint(config.pretrained\_ckpt, backbone)

head = MobileNetV2Head(input\_channel=backbone.out\_channels, num\_classes=config.num\_classes)

network = mobilenet\_v2(backbone, head)

# define loss, optimizer, and model

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

loss\_scale = FixedLossScaleManager(LOSS\_SCALE, drop\_overflow\_update=False)

lrs = cosine\_decay(config.epochs \* step\_size, lr\_max=config.lr\_max)

opt = nn.Momentum(network.trainable\_params(), lrs, config.momentum, config.weight\_decay, loss\_scale=LOSS\_SCALE)

# 定义用于训练的train\_loop函数。

def train\_loop(model, dataset, loss\_fn, optimizer):

# 定义正向计算函数

def forward\_fn(data, label):

logits = model(data)

loss = loss\_fn(logits, label)

return loss

# 定义微分函数，使用mindspore.value\_and\_grad获得微分函数grad\_fn,输出loss和梯度。

# 由于是对模型参数求导,grad\_position 配置为None，传入可训练参数。

grad\_fn = ms.value\_and\_grad(forward\_fn, None, optimizer.parameters)

# 定义 one-step training函数

def train\_step(data, label):

loss, grads = grad\_fn(data, label)

optimizer(grads)

return loss

size = dataset.get\_dataset\_size()

model.set\_train()

for batch, (data, label) in enumerate(dataset.create\_tuple\_iterator()):

loss = train\_step(data, label)

if batch % 10 == 0:

loss, current = loss.asnumpy(), batch

print(f"loss: {loss:>7f} [{current:>3d}/{size:>3d}]")

# 定义用于测试的test\_loop函数。

def test\_loop(model, dataset, loss\_fn):

num\_batches = dataset.get\_dataset\_size()

model.set\_train(False)

total, test\_loss, correct = 0, 0, 0

for data, label in dataset.create\_tuple\_iterator():

pred = model(data)

total += len(data)

test\_loss += loss\_fn(pred, label).asnumpy()

correct += (pred.argmax(1) == label).asnumpy().sum()

test\_loss /= num\_batches

correct /= total

print(f"Test: \n Accuracy: {(100\*correct):>0.1f}%, Avg loss: {test\_loss:>8f} \n")

import os

# 定义文件夹路径

folder\_path = './ckpt/'

# 检查文件夹是否存在，如果不存在则创建

if not os.path.exists(folder\_path):

os.makedirs(folder\_path)

print(f"Folder '{folder\_path}' created successfully.")

else:

print(f"Folder '{folder\_path}' already exists.")

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

epochs = 10

for t in range(epochs):

print(f"Epoch {t+1}\n-------------------------------")

train\_loop(network, train\_dataset, loss, opt)

ms.save\_checkpoint(network, "./ckpt/save\_mobilenetV2\_model.ckpt")

test\_loop(network, eval\_dataset, loss)

print("Done!")

训练输出如下，模型保存在mobilenetv2\_garbage/ckpt/save\_mobilenetV2\_model.ckpt：

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

Epoch 1

-------------------------------

loss: 3.365921 [ 0/162]

loss: 3.278985 [ 10/162]

……

Epoch 10

-------------------------------

……

loss: 2.831076 [140/162]

loss: 2.920157 [150/162]

loss: 2.913762 [160/162]

Test:

Accuracy: 51.7%, Avg loss: 2.881466

Done!

模型推理

加载save\_mobilenetV2\_model.ckpt对data\_en/test/Cardboard内的图片进行推理，使用load\_checkpoint接口加载数据时，需要把数据传入给原始网络，而不能传递给带有优化器和损失函数的训练网络。

CKPT="save\_mobilenetV2\_model.ckpt"

def image\_process(image):

"""Precess one image per time.

Args:

image: shape (H, W, C)

"""

mean=[0.485\*255, 0.456\*255, 0.406\*255]

std=[0.229\*255, 0.224\*255, 0.225\*255]

image = (np.array(image) - mean) / std

image = image.transpose((2,0,1))

img\_tensor = Tensor(np.array([image], np.float32))

return img\_tensor

def infer\_one(network, image\_path):

image = Image.open(image\_path).resize((config.image\_height, config.image\_width))

logits = network(image\_process(image))

pred = np.argmax(logits.asnumpy(), axis=1)[0]

print(image\_path, class\_en[pred])

def infer():

backbone = MobileNetV2Backbone(last\_channel=config.backbone\_out\_channels)

head = MobileNetV2Head(input\_channel=backbone.out\_channels, num\_classes=config.num\_classes)

network = mobilenet\_v2(backbone, head)

load\_checkpoint(os.path.join(config.save\_ckpt\_path, CKPT), network)

for i in range(91, 100):

infer\_one(network, f'data\_en/test/Cardboard/000{i}.jpg')

infer()

输出：

data\_en/test/Cardboard/00091.jpg Paper

data\_en/test/Cardboard/00092.jpg Plastic Bottle

data\_en/test/Cardboard/00093.jpg Basketball

data\_en/test/Cardboard/00094.jpg Fluorescent lamp

data\_en/test/Cardboard/00095.jpg Fluorescent lamp

data\_en/test/Cardboard/00096.jpg Dirty Cloth

data\_en/test/Cardboard/00097.jpg Tablet capsules

data\_en/test/Cardboard/00098.jpg Cardboard

data\_en/test/Cardboard/00099.jpg Basketball

导出MINDIR模型文件

导出AIR模型文件，用于后续在香橙派 AIPro上的模型转换与推理。

backbone = MobileNetV2Backbone(last\_channel=config.backbone\_out\_channels)

head = MobileNetV2Head(input\_channel=backbone.out\_channels, num\_classes=config.num\_classes)

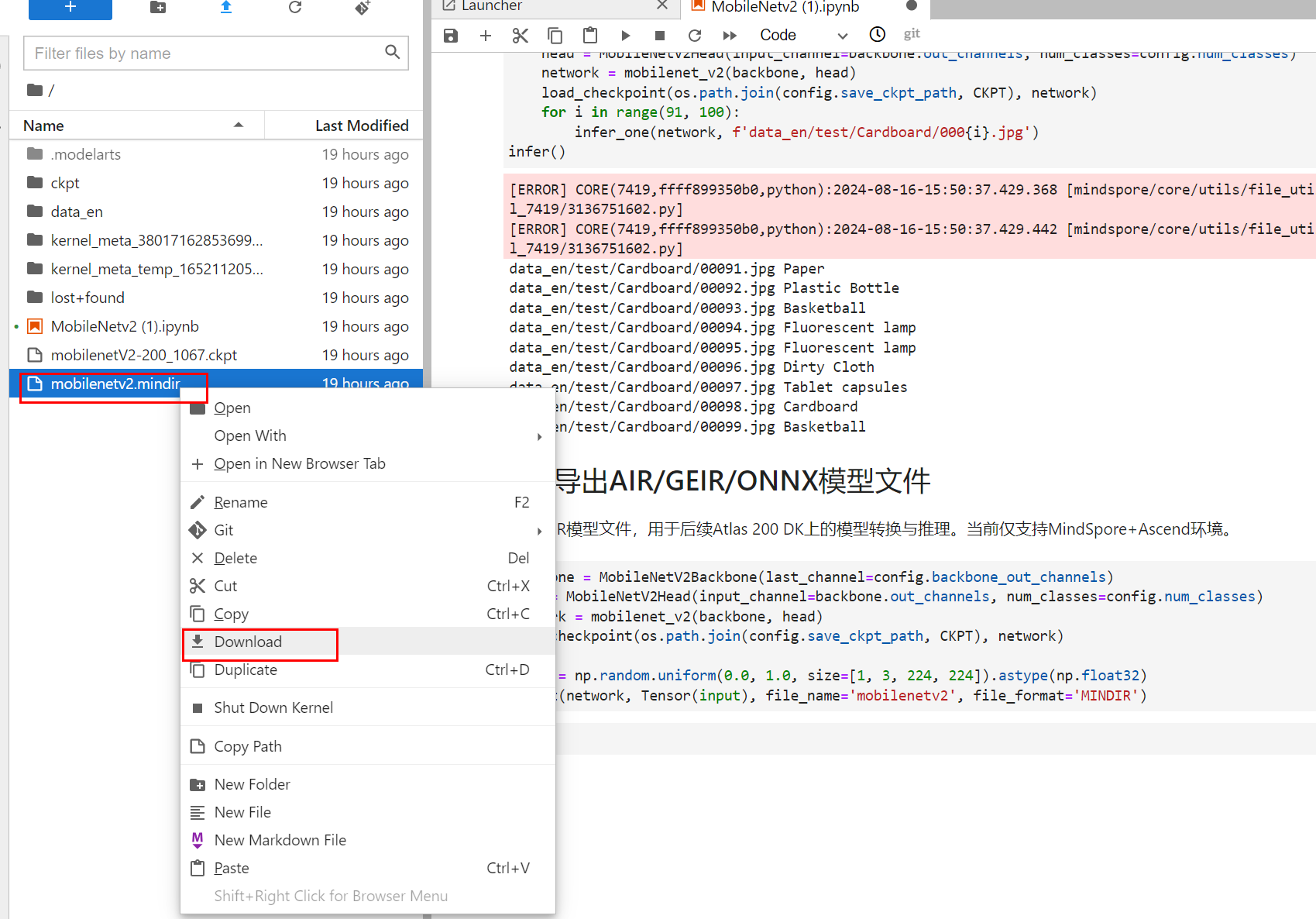
network = mobilenet\_v2(backbone, head)

load\_checkpoint(os.path.join(config.save\_ckpt\_path, CKPT), network)

input = np.random.uniform(0.0, 1.0, size=[1, 3, 224, 224]).astype(np.float32)

export(network, Tensor(input), file\_name='mobilenetv2', file\_format='MINDIR')

下载MINDIR

成的mobilenetv2.mindir文件下载到本机

下载mobilenetv2.mindir模型

如使用华为云环境，请及时关闭云环境

实验完成之后，请及时关闭华为云ModelArts的Notebook开发环境，避免资源浪费。

关闭方式：登录[华为云ModelArts控制台](https://console.huaweicloud.com/modelarts/?region=cn-southwest-2#/dev-container)，在“操作”栏选择“停止”或“更多—>删除”操作。

如下图所示：



及时关闭/删除云环境

## 香橙派 AI Pro端侧部署

本实验使用昇腾CANN在香橙派 AI Pro上开发垃圾分类推理应用，首先使用ATC工具对mobilenetv2.mindir模型进行离线模型转换，接下来使用AscendCL开发推理代码，对图像中的物体进行分类推理，最后使用OpenCV写到本地文件中。输入样例是待推理的jpg图片，输出样例是推理后的jpg图片。



1. 运行管理资源申请：用于初始化系统内部资源，固定的调用流程。
2. 加载模型文件并构建输出内存：从文件加载离线模型moblienetv2.om数据，需要由用户自行管理模型运行的内存，根据内存中加载的模型获取模型的基本信息包含模型输入、输出数据的数据buffer大小；由模型的基本信息构建模型输出内存，为接下来的模型推理做好准备。
3. 数据预处理：对读入的图像数据进行预处理，然后构建模型的输入数据。
4. 模型推理：根据构建好的模型输入数据进行模型推理。
5. 解析推理结果：根据模型输出，解析模型的推理结果。使用OpenCV将垃圾分类的结果写进图片。

### 环境准备

本实验是在新一代开发者套件香橙派 AI Pro上执行，因此需要部署香橙派 AI Pro的实验环境，请参考《手把手教你搭建Orange Pi AI Pro开发环境》实验手册。



按上面手册完成：登录开发板、开发板网络连接、MindSpore版本升级为2.2.14。

### 下载项目

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/contrib/garbage\_picture/model

项目目录如下：

root/samples/python/contrib/garbage\_picture/model

├—data # 需创建

│ # 测试图片

└─model

│ # 推理模型

└─src

│ classify\_test.py # 推理代码

上传mobilenetv2.mindir模型

将导出mindir模型文件传至/root/samples/python/contrib/garbage\_picture/model目录。

创建data文件夹

mkdir /root/samples/python/contrib/garbage\_picture/data

在data目录下载测试图片

cd $HOME/samples/python/contrib/garbage\_picture/data

wget https://obs-9be7.obs.cn-east-2.myhuaweicloud.com/models/garbage\_picture/newspaper.jpg

wget https://obs-9be7.obs.cn-east-2.myhuaweicloud.com/models/garbage\_picture/bottle.jpg

wget https://obs-9be7.obs.cn-east-2.myhuaweicloud.com/models/garbage\_picture/dirtycloth.jpg



测试图片样例

### 离线模型转换

离线模型转换

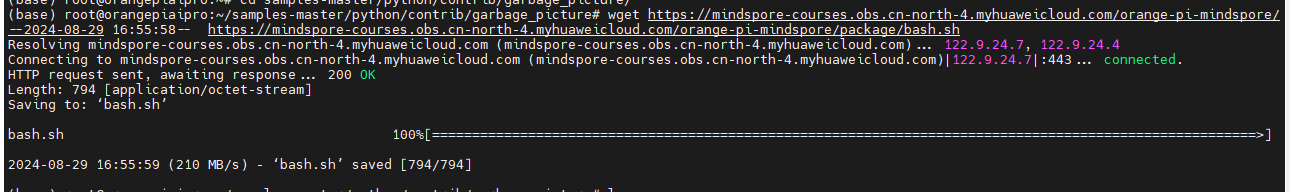
#获取bash.sh文件

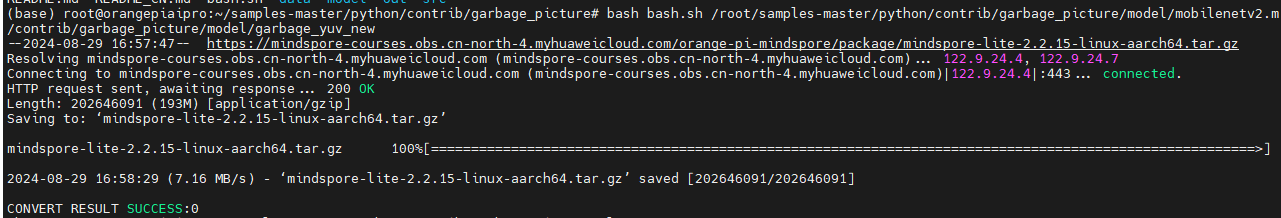
wget https://mindspore-courses.obs.cn-north-4.myhuaweicloud.com/orange-pi-mindspore/package/bash.sh

#执行bash.sh文件

source bash.sh /root/samples/python/contrib/garbage\_picture/model/mobilenetv2.mindir /root/samples/python/contrib/garbage\_picture/model/mobilenetv2

输出“ SUCCESS”代表模型转换成功





模型转换成功

### 执行模型推理

上传classify\_test.py覆盖原脚本文件

#classify\_test.py文件内容如下

#!/usr/bin/env python

# encoding: utf-8

import sys

import os

path = os.path.dirname(os.path.abspath(\_\_file\_\_))

sys.path.append(os.path.join(path, ".."))

sys.path.append(os.path.join(path, "../../../common/"))

sys.path.append(os.path.join(path, "../../../common/acllite"))

import numpy as np

import acl

import base64

import acllite\_utils as utils

from PIL import Image, ImageDraw, ImageFont

import constants as const

from acllite\_model import AclLiteModel

from acllite\_resource import AclLiteResource

import mindspore as ms

from mindspore import Tensor

SRC\_PATH = os.path.realpath(\_\_file\_\_).rsplit("/", 1)[0]

MODEL\_PATH = os.path.join(SRC\_PATH, "../model/garbage\_yuv.om")

MODEL\_WIDTH = 224

MODEL\_HEIGHT = 224

image\_net\_classes = [

"Seashel", "Lighter","Old Mirror", "Broom","Ceramic Bowl", "Toothbrush","Disposable Chopsticks","Dirty Cloth",

"Newspaper", "Glassware", "Basketball", "Plastic Bottle", "Cardboard","Glass Bottle", "Metalware", "Hats", "Cans", "Paper",

"Vegetable Leaf","Orange Peel", "Eggshell","Banana Peel",

"Battery", "Tablet capsules","Fluorescent lamp", "Paint bucket"]

def get\_image\_net\_class(class\_id):

if class\_id >= len(image\_net\_classes):

return "unknown"

else:

return image\_net\_classes[class\_id]

def pre\_process(image\_path, MODEL\_HEIGHT, MODEL\_WIDTH):

"""preprocess"""

"""Precess one image per time.

Args:

image: shape (H, W, C)

"""

image\_input = Image.open(image\_path)

image\_input = image\_input.resize((MODEL\_HEIGHT, MODEL\_WIDTH))

mean=[0.485\*255, 0.456\*255, 0.406\*255]

std=[0.229\*255, 0.224\*255, 0.225\*255]

image = (np.array(image\_input) - mean) / std

image = image.transpose((2,0,1))

img\_tensor = Tensor(np.array([image], np.float32))

return img\_tensor

def post\_process(infer\_output, image\_file):

print("post process")

data = infer\_output[0]

vals = data.flatten()

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

object\_class = get\_image\_net\_class(top\_k[0])

output\_path = os.path.join(os.path.join(SRC\_PATH, "../out"), os.path.basename(image\_file))

origin\_image = Image.open(image\_file)

draw = ImageDraw.Draw(origin\_image)

font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans-Bold.ttf", size=20)

font.size =50

draw.text((10, 50), object\_class, font=font, fill=255)

origin\_image.save(output\_path)

object\_class = get\_image\_net\_class(top\_k[0])

return

def main():

if (len(sys.argv) != 2):

print("The App arg is invalid")

exit(1)

acl\_resource = AclLiteResource()

acl\_resource.init()

model = AclLiteModel(MODEL\_PATH)

image\_dir = sys.argv[1]

images\_list = [os.path.join(image\_dir, img)

for img in os.listdir(image\_dir)

if os.path.splitext(img)[1] in const.IMG\_EXT]

# Create a directory to store the inference results

if not os.path.isdir(os.path.join(SRC\_PATH, "../out")):

os.mkdir(os.path.join(SRC\_PATH, "../out"))

for image\_file in images\_list:

resized\_image = pre\_process(image\_file, MODEL\_HEIGHT, MODEL\_WIDTH)

print("pre process end")

result = model.execute([resized\_image.asnumpy(), ])

post\_process(result, image\_file)

print("process "+image\_file+" end")

if \_\_name\_\_ == '\_\_main\_\_':

main()

执行模型推理

classify\_test.py为模型推理代码，对data文件夹内的图片进行推理。

cd ${HOME}/samples/python/contrib/garbage\_picture/src

python classify\_test.py ../data/



模型推理结果

查看推理结果

推理完成后在out文件夹内得到推理结果图片：



推理结果图片

## 实验小结

本实验是基于Ascend910和Ascend310B算力平台，实现垃圾分类模型训练+推理部署的全流程实验。在Ascend910算力平台使用迁移学习训练得到MobileNetv2的垃圾分类模型，将其部署在香橙派 AI Pro上，使用Ascend310B算力平台实现端侧推理。通过本实验使学员掌握MindSpore的算法开发，昇腾CANN的推理应用开发，Ascend910和Ascend310B算力平台的使用。