学科实践实验报告

实验题目： **垃圾图片分类**

专　　业： **计算机科学与技术**

姓　　名： **吴雨娟**

学　　号： **22920192204097**

实验日期： 2021.5.26

一、 实验目的

如今国家正大力开展环境保护工作，使环境保护与经济发展相协调。习近平总书记号召：“强化全面推行生活垃圾分类，要求行动自觉、全面动员、全民参与，凝聚全社会的力量共同努力，打赢打好垃圾分类工作的攻坚战、持久战。”在时代背景下，垃圾分类成为亟待解决的问题。而随着人工智能的发展，图片检测技术已经有了一定进展，可以运用到具体生活情况中，为人们的生活提供更多便利。所以我们可以运用目前已有的图像分类预训练模型进行垃圾图片分类训练，实现垃圾图片的分类预测。

二、实验内容

我们需要根据给定的图片，设计出一个对应的模型，检测图片中的垃圾类别。实验中训练和测试图片均来自生活场景。总共四十个类别，图片中垃圾的类别格式是“一级类别/二级类别”，二级类别是具体的垃圾物体类别，也就是训练数据中标注的类别，比如一次性快餐盒、果皮果肉、旧衣服等。训练集的所有图片分别保存在 train 文件夹下面的 0-39 个文件夹中，文件名即类别标签，测试集共有 400 张待分类的垃圾图片在 test 文件夹下，testpath.txt 保存了所有测试集文件的名称，格式为：name+\n。结果文件为 TXT 文件格式，命名为 model_result.txt，文件内的字段需要按照指定格式写入。

三、实验步骤

以下实验步骤以提交的最优版本代码为例，对每一部分的代码进行简要说明。

1.用 pip 指令安装 paddlehub。

```
!pip install paddlehub==1.8.1 -i https://pypi.tuna.tsinghua.edu.cn/simple
```

2.用 linux 命令将压缩文件解压到指定位置。

```
!unzip -q -o ./data/data35095/train.zip -d ./data/rubbish
```

3.将各种可能用到的库全部导入。

```
import os
import zipfile
import random
import json
import cv2
import numpy as np
from PIL import Image
from PIL import ImageEnhance
import paddle
import paddle.fluid as fluid
from paddle.fluid.dygraph import Linear
from paddle.fluid.dygraph import Pool2D, Conv2D
import matplotlib.pyplot as plt
```

4.进行参数配置，和以前差不多。

```
'''
参数配置
'''
train_parameters = {
    "src_path":"/home/aistudio/data/data35095/sisters.zip",        #原始数据集路径
    "target_path":"/home/aistudio/data/rubbish/",                   #要解压的路径
    "label_list_path": "/home/aistudio/data/rubbish/train/label.txt",    #label.txt路径
    "train_list_path": "/home/aistudio/data/rubbish/train/train.txt",    #train.txt路径
    "eval_list_path": "/home/aistudio/data/rubbish/train/eval.txt",      #eval.txt路径
    "readme_path":"/home/aistudio/data/rubbish/readme.json",
    "label_dict":{},
}
```

5.设置获取数据列表的函数，内容也与之前实验里写的差不多。

```
def get_data_list(target_path,train_list_path,eval_list_path):
    '''
    生成数据列表
    '''

    #存放所有类别的信息
    class_detail = []
    #类别信息字典
    class_path = "data/rubbish/garbage_dict.json"
    f = open(class_path, 'r')
    dict_data = json.load(f)
    #获取所有类别保存的文件夹名称
    data_list_path=target_path+"train/"
    class_dirs = os.listdir(data_list_path)
    #总的图像数量
    all_class_images = 0
    #存放类别标签
    class_label=0
    #存放类别数目
```

```
    class_dim = 0
    #存储要写进eval.txt和train.txt中的内容
    label_list=[]
    trainer_list=[]
    eval_list=[]
    #读取每个类别，0-39类
    for class_dir in class_dirs:
        if class_dir != ".DS_Store":
            class_dim += 1
            class_label = int(class_dir)
            #每个类别的信息
            class_detail_list = {}
            eval_sum = 0
            trainer_sum = 0
            #统计每个类别有多少张图片
            class_sum = 0
            #获取类别路径
            path = data_list_path  + class_dir
            # 获取所有图片
            img_paths = os.listdir(path)
```

```
for img_path in img_paths:                                  # 遍历文件夹下的每个图片
    name_path = class_dir + '/' + img_path                      # 每张图片的路径
    if class_sum % 200 == 0:                                # 每100张图片取一个做验证数据
        eval_sum += 1                                       # eval_sum为测试数据的数目
        eval_list.append(name_path + " %d" % class_label + "\n")
    else:
        trainer_sum += 1
        trainer_list.append(name_path + " %d" % class_label + "\n")#trainer_sum测试数据的数目
    class_sum += 1                                          #每类图片的数目
    all_class_images += 1                                   #所有类图片的数目
```

```
            # 说明的json文件的class_detail数据
            class_detail_list['class_name'] = dict_data[class_dir]             #类别名称
            class_detail_list['class_label'] = class_label          #类别标签
            class_detail_list['class_eval_images'] = eval_sum       #该类数据的测试集数目
            class_detail_list['class_trainer_images'] = trainer_sum #该类数据的训练集数目
            class_detail.append(class_detail_list)
            #初始化标签列表
            train_parameters['label_dict'][str(class_label)] = dict_data[class_dir]

    #初始化分类数
    train_parameters['class_dim'] = class_dim

    #乱序
    random.shuffle(eval_list)
    with open(eval_list_path, 'a') as f:
        for eval_image in eval_list:
            f.write(eval_image)

    random.shuffle(trainer_list)
    with open(train_list_path, 'a') as f2:
        for train_image in trainer_list:
            f2.write(train_image)

    for ii in range(40):
        if ii==39:
            label_list.append(str(ii))
        else:
            label_list.append(str(ii) + "\n")
    with open(label_list_path,'a') as f1:
        for label in label_list:
            f1.write(label)

print ('生成数据列表完成！')
```

6.设置目标路径及完成列表的生成。

```
target_path=train_parameters['target_path']
train_list_path=train_parameters['train_list_path']
eval_list_path=train_parameters['eval_list_path']
label_list_path=train_parameters['label_list_path']
```

```
get_data_list(target_path,train_list_path,eval_list_path)
```

7.加载预训练模型，使用 resnet_v2_50_imagenet 作为预训练的模型。经过性能评估，发现这个模型的效果是最好的。

```
import paddle
import paddlehub as hub
module = hub.Module(name="resnet_v2_50_imagenet")
```

8. 加载训练图片数据集。

```
from paddlehub.dataset.base_cv_dataset import BaseCVDataset

class DemoDataset(BaseCVDataset):
    def __init__(self):
        # 数据集存放位置

        self.dataset_dir = "data/rubbish/train"
        super(DemoDataset, self).__init__(
            base_path=self.dataset_dir,
            train_list_file="train.txt",
            validate_list_file="eval.txt",
            test_list_file="eval.txt",
            label_list_file="label.txt",
            )
dataset = DemoDataset()
```

9. 生成 reader，将 dataset 的数据进行预处理，然后以特定格式组织并输入给模型进行训练，例如本实验的 lable map 为 0-39 这 40 个数字，代表垃圾的 40 种分类。

```
data_reader = hub.reader.ImageClassificationReader(
    image_width=module.get_expected_image_width(),
    image_height=module.get_expected_image_height(),
    images_mean=module.get_pretrained_images_mean(),
    images_std=module.get_pretrained_images_std(),
    dataset=dataset
)
```

10. 选择运行时的配置。设置为使用 GPU 版本的 PaddlePaddle，遍历 5 遍训练集，每次训练时给模型输入的每批数据大小为 128，每隔 20step 在验证集上进行一次性能评估，将训练的参数和数据保存到 cv_finetune_turtorial_demo 目录中，使用 DefaultFinetuneStrategy 策略进行 finetune。

```
config = hub.RunConfig(
    use_cuda=True,
    num_epoch =5,
    checkpoint_dir="cv_finetune_turtorial_demo",
    batch_size=128,
    eval_interval=20,
    strategy=hub.finetune.strategy.DefaultFinetuneStrategy()
)
```

11. 有了合适的预训练模型和准备要迁移的数据集后，组建一个 Task。

```
input_dict, output_dict, program = module.context(trainable=True)
img = input_dict["image"]
feature_map = output_dict["feature_map"]
feed_list = [img.name]

task = hub.ImageClassifierTask(
    data_reader=data_reader,
    feed_list=feed_list,
    feature=feature_map,
    num_classes=dataset.num_labels,
    config=config)
```

12. 开始 finetune，会显示每次训练和评估的结果。

```
run_states = task.finetune_and_eval()
```

13. 用 Linux 命令解压测试数据压缩包。

```
!unzip -q -o ./data/data35095/test.zip -d ./data/result
```

14. 将 testpath 中的图片路径保存到 data 中。

```
data = []
for line in open("data/result/testpath.txt","r"): #设置文件对象并读取每一行文件
    line=line.strip("\n")          #删除首尾的"\n"
    data.append("data/result/test/"+line)              #将每一行文件加入到data中

data
```

15. 最后显示预测结果，并把结果写入文件中，这样就得到了结果文件”model_result.txt”。

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as image
#显示matplotlib生成的图形
%matplotlib inline
index = 0
label_map = dataset.label_dict()

run_states = task.predict(data=data)
results = [run_state.run_results for run_state in run_states]
file_handle=open('data/result/model_result.txt',mode='w')
for batch_result in results:
    batch_result = np.argmax(batch_result, axis=2)[0]
    for result in batch_result:
        index += 1
        result = label_map[result]
        print("input %i is %s, and the predict result is %s" %
              (index, data[index - 1], result))
        file_handle.write(result+"\n")
```

四、实验结果及调试过程的分析

1.最优实验结果

吴雨娟	0.9025	2021-05-25 23:09

2.调试过程分析

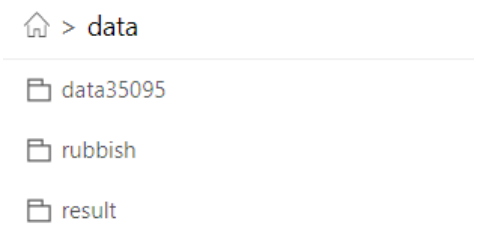
（1）解压

在选择解压方式时，可以不用像之前的实验一样写一大段解压代码，用一行 Linux 命令就可以进行解压。

```
!unzip -q -o ./data/data35095/train.zip -d ./data/rubbish
```

```
!unzip -q -o ./data/data35095/test.zip -d ./data/result
```

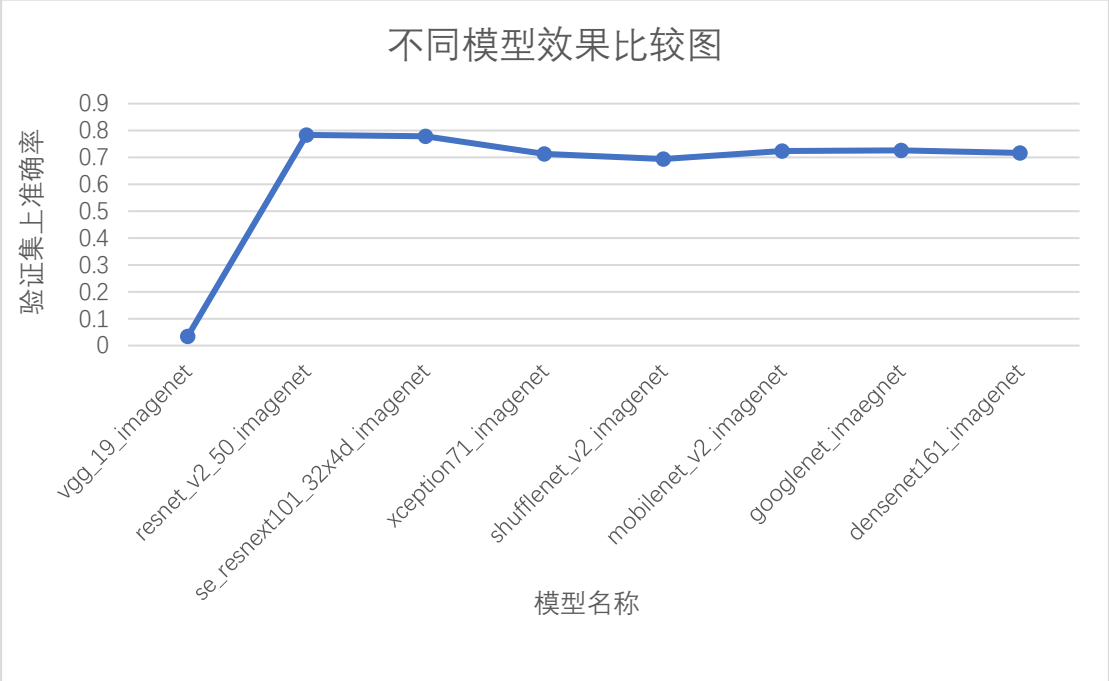
解压结果：



（2）选择模型

在选择模型的时候，有尝试过 CNN、DNN 和 VGG 网络，自己写神经网络的结构。但是由于神经网络设计调参困难，导致准确率一直不高，所以改用 Paddlehub 里迁移学习的模型进行训练。

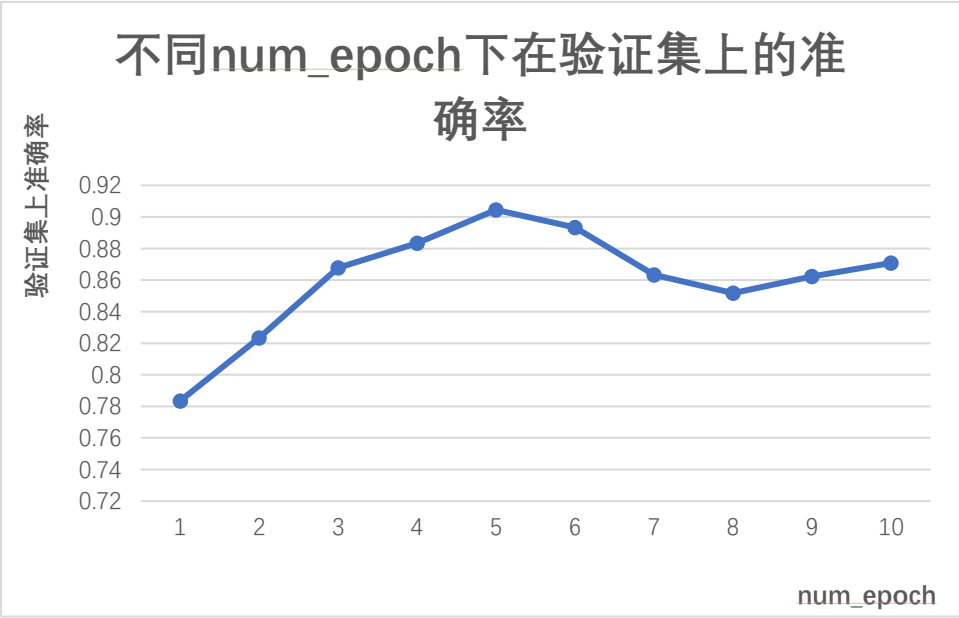
在选择迁移学习的预训练模型时，在适用于图像分类的模型库中找到了评分最高的 9 个网络，排除其中用于动物图片分类的 resnet50_vd_animals 模型，对其他 8 个模型进行了性能评估，最终决定选择 resnet_v2_50_imagenet 模型进行训练。



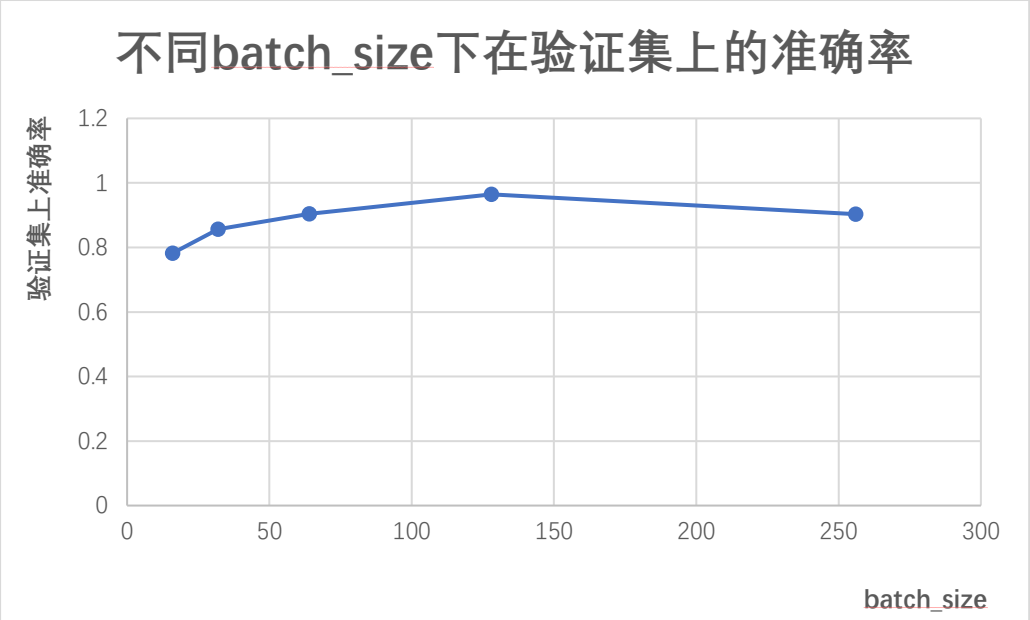
（3）调参

一共有 4 个可以调节的参数：num_epoch,batch_size,eval_interval 和 strategy。其中 eval_interval 是用于观察训练过程，对于训练结果没有什么影响，所以不调节这个参数，对其他三个参数进行调节。

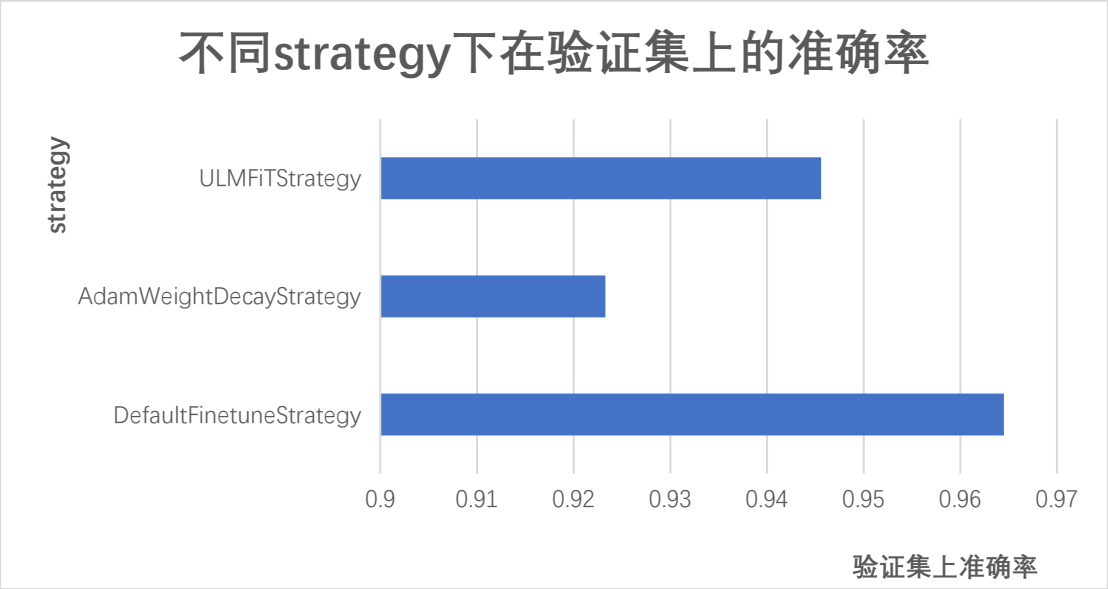
首先调节 num_epoch 参数。在一定范围内,num_epoch 越大，训练效果越好，但是当 num_epoch 过大时，会造成过拟合，训练效果反而不好。所以要进行性能评估测试，在 batch_size=64,strategy 为 DefaultFinetuneStrategy 的条件下调整迭代轮数。经过性能评估测试，选择了 num_epoch=5。



然后调节 batch_size 参数。在不超内存的情况下要尽量增大 batch_size，通过并行化提升内存的利用率，同时把 batch_size 值设置为 2 的幂，以满足 GPU 的要求。同样地，我们要进行性能评估。在 num_epoch=5,strategy 为 DefaultFinetuneStrategy 的条件下调整每次迭代输入的训练图片的个数。经过性能评估测试，选择了 batch_size=128。



最后调节 strategy。尝试三种不同的优化策略：DefaultFinetuneStrategy、AdamWeightDecayStrategy 和 ULMFiTStrategy。在 num_epoch=5,batch_size=128 的条件下调整优化策略，经过性能评估测试，优化策略选择 DefaultFinetuneStrategy。



经过大量的调参测试，在合适的参数下，大部分实验结果都在 0.87-0.9 之间，基本稳定。但是由于迁移学习的模型限制，对于本地的数据不能很完美的兼容，且由于数据加载有一定随机性，相同参数版本每一次的训练结果都不一样。

点开预测图片后，发现有的图片具有高度混淆性，例如图片不完整，特征不清晰，有几类图片之间的相似度高等。这导致准确率一直无法提升到 0.95 以上。

在以上的种种限制下，经过数十次运行、二十几个版本的提交，最高的准确率为 0.9025，已经较好地完成了垃圾图片分类的实验任务。

五、实验总结

通过本次实验，理解了 paddlehub 迁移学习的基本框架，并且能做一些相应的修改，使之更能适配本地的数据。在实践中掌握了调参的基本规律，尝试运用不同的优化策略以及参数，使得到的模型是基本收敛的，但是准确率会受到限制。在今后的学习中，会更深入地研究改进方法，希望能提高迁移学习的准确率。