信息与软件工程学院

基于 Python 的图像分类项目开发

实践总结报告 PBLF2021

小组成员： 姓名：陈霆润 学号：2020091202003
姓名：幸锴文 学号：2020091201022
姓名：赵之航 学号：2020091202028

第一章 人工智能概述

1.1 人工智能的发展与现状

自 2016 年谷歌的人工智能 AlphaGo 在围棋比赛中先后打败李世石和柯洁，人工自智能成为了社会的热点。人工智能的技术成为了科技的热点，人工智能公司成为. 了资本的追逐对象。各个行业也开始发展人工智能，而后，熟悉的手机语音助手, 人脸识别，购买推荐，自动驾驶都纷纷应用了人工智能。业界将人工智能分为三个阶段，分别是弱人工智能、强人工智能、超人工智能。弱人工智能只能在某一方面超越人类，只能替代某一方面的工作；强人工智能拥有和人类一样的智商，能够在日常生活中代替人类的绝大部分工作；而超人工智能能够像人类一样通过各种采集器、网络进行学习，每天进行多次升级迭代。而在超人工智能阶段，人工智能的水平将远超人类。当今我们的人工智能发展处在弱人工智能阶段，尽管 AlphaGo 在围棋比赛上打败了世界冠军，但也仅仅限于围棋一方面，并不能在多方面取代人类，甚至在其他领域与人类相差甚远。而现在科学家和各大企业想要实现的目标便是使人工智能达到强人工智能阶段。超人工智能更加接近我们幻想中的神和上帝，离我们现阶段的发展还有较远的距离。

1.2 人工智能的应用

现阶段人工智能的几个主要应用有如下几个方面：

i. 无人驾驶汽车

无人驾驶汽车是智能汽车的一种，主要依靠车内以计算机系统为主的智能驾驶控制器来实现无人驾驶。而人工智能主要用于无人驾驶汽车对外界环境的感知、深度学习、共享信息三个方面。

ii. 人脸识别

人脸识别是基于人的面部特征进行身份识别的一种生物识别技术。而其中的算机技术和图像处理技术都离不开人工智能。现阶段，人脸识别已经应用于司法、安保、医疗、教育等多个行业。

iii. 个性化推荐

个性化推荐是一种基于聚类与协同过滤技术的人工智能应用，它的基础便是海量的数据挖掘，通过对用户多次的输入分析来建立数据模型。在用户下次搜索的过程中为用户的兴趣提供相似的推荐。如淘宝、京东上的相似推荐，每日的新闻推荐等。

1.3 人工智能技术与分类

从研究方向上来看，人工智能技术领域包括深度学习技术、、自然语言处理技术、计算机视角技术、智能机器人技术等。

深度学习基于已有的数据进行学习操作，它模拟人脑的机制来解释数据、声音、文本。

自然语言处理技术是用计算机技术进行通讯的一种技术，研究用电子计算机来模拟人的语言的交流过程、目的是使计算机能够了解人类的各种语言，并能够成功的和人类对话。

计算机视角技术是用电脑代替人脑，而用摄像头来代替人眼来对目标进行测量和识别各种图像，并对图像做进一步的处理，使之成为更适合人眼所能识别的图像。

智能机器人技术具备各种各样基于人工智能的传感器和效应器，并配备有“大脑芯片”使其拥有更强的智能性。

第二章　图像分类任务分析

2.1 图像分类应用与技术概要

图像分类项目开发应用分为了三个部分，分别是：

a） 使用预训练模型图像分类

b） 使用 Finetune 训练模型

c） 数据集增强

在第一部分，使用预训练模型图像分类中，我们通过百度的 paddlepaddle 来调用 CNN 预训练模型，分别调用了 resnet，alexnet，vgg 和 googlenet 四个模型，将图片上传后，对返回的结果进行处理，并对比有效的小狗名称，得出相关的统计量。通过程序运行时间、正确分类的小狗所占百分比、正确分类的非小狗图像所占百分比、正确分类的小狗品种所占百分比以及标签匹配书所占百分比这 5 个结果进行横向对比。调用 seaborn 等库来绘制统计图并保存统计图图片。

在第二部分，使用 finetune 训练模型，通过 paddlehub 调用 googlenet 预训练模型，对数据集进行训练。数据集包含五种种类的中药材，分别为百合、党参、枸杞、槐花、金银花。训练后对我们寻找的新的 50 张图片进行训练，成功识别后将 50 张图片复制到对应的子文件夹下。

在第三部分，数据集增强板块，引入了 numpy、PIL、matplotlib 以及 math 库。以矩阵的方式存储一张图像的每个像素点。通过对矩阵操作，实现了旋转（90°，180°，270°）、平移、镜像（垂直、水平）、放大操作，并将进行操作后的图片保存。

2.2 图像分类技术

CNN 模型调用：通过 paddlepaddle 调用 CNN 模型，有四种：resnet，alexnet，vgg 和 googlenet。最后选出 googlenet 的识别效果最好和识别效率最高

调用的第三方库：

a） Math：用于矩阵中角度的计算

b） Seaborn：绘制条形统计图

c） Pandas：绘制条形统计图

d） Numpy：用于矩阵的计算

e） Mathplotlib：保存图片，绘制图像

f） Paddlehub：调用 CNN 模型，调用 finetune_and_eval 接口，创建自定义数据集，生成图像的 reader。

调用标准库：

a） Os：获取图片的路径，创建路径

b） random：打乱所有图片的路径

c） shutil：复制文件

Finetune：调用 finetune_and_eval 接口进行模型的训练

2.3 图像分类实现环境

操作系统：ai studio

编程环境：

i. 基础版：

01) CPU：2 Cores

02) RAM：8GB

03) Disk：100GB

ii. 高级版：（用于 finetune）

01) GPU：Tesla V100. Video

02) Mem：100GB

03) CPU：4 Cores

04) RAM：32GB

05) Disk：100GB

编程语言：Python

第三章　项目实现分析

3.1 任务架构

第一部分 使用预训练模型图像分类：

图 3. 1

第二部分 使用 finetune 训练模型

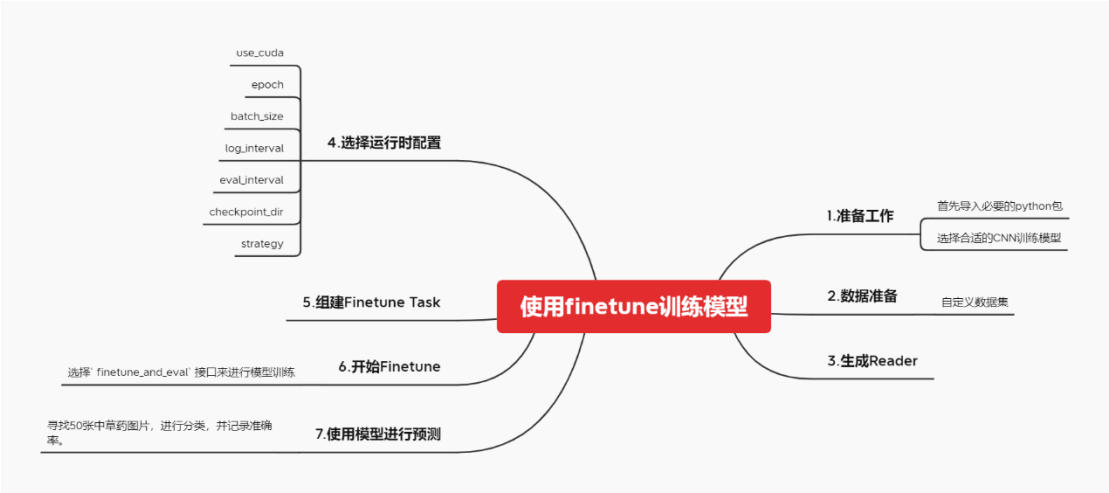

图 3. 2

第三部分 数据集增强

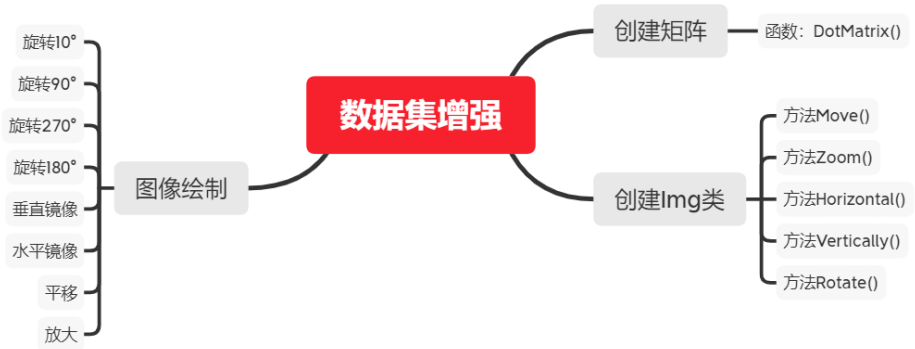

图 3. 3

3.2 任务模块具体分析及实现

3.2.1 使用预训练模型图像分类模块

TODO1 模块

```
# TODO: 1
start = time()
end = time()
total = end - start
seconds = int((total % 3600)%60)
minutes = int((total % 3600) / 60)
hours = int(total / 3600)
print(f"程序运行时间：{hours}:{minutes}:{seconds}")
```

在程序刚开始运行时，使用 start=time()记录开始时间，在程序运行结束后，使用 end=time()记录结束时间，两者之差即为程序运行的时间，再将运行的时间转换为时分秒的形式

程序运行截图：

```
            Cat_07.Jpg
            Cat_01.Jpg
            Great_Horned_Owl_02.Jpg
            Gecko_02.Jpg

        狗狗品种：
             german shorthaired pointer
             boston terrier
             german shepherd dog
             great pyrenees
    程序运行时间：0:0:21
```

图 3. 4

TODO2 模块

```
def get_pet_labels():
    filename_list = listdir("pet_image/")
    petlabels_dic = {}
    for filename in filename_list:
        lower_filename = filename.lower()
        word_list_filename = lower_filename.split('_')
        pet_name = ''
        for index in word_list_filename:
```

```
            if index.isalpha():
                pet_name += (index+" ")
        pet_name = pet_name.strip()
        petlabels_dic[filename] = pet_name
    return petlabels_dic
```

调用 listdir 函数，获得 pet_image 文件夹中的所有图像的名称，并以列表的格式返回。

在 filename_list 的 for 循环中，对于每一个 filename，即图像的名称（形如：fox_squirrel_01.jpg），首先使用函数 split 将分隔符“_”分隔，返回一个列表。

使用 lower()方法，将字符串全部变为小写。

使用 alpha()方法判断字符串是否只有字母而不是数字。

将全是字母的字符串用“+”和空格连接起来，得到最后需要的字符串

将最后得到的字符串作为值，最初的图像名称作为键，放入 petlabels_dic 字典中存储，最后返回该数组

TODO3 模块

```
image_dir = "/home/aistudio/pet_image/"
models ={'resnet':'resnet_v2_50_imagenet', 'alexnet': 'alexnet_imagenet',
'vgg':'vgg19_imagenet','googlenet':'googlenet_imagenet','mobilenet':"mo
bilenet_v2_imagenet"}
import paddlehub as hub

def classifier(image_dir,image_path,model):
  module = hub.Module(name=models[model])
  test_img_path =image_dir+image_path
  input_dict = {"image": [test_img_path]}
  result = module.classification(data=input_dict)
  temp_dict =  result[0][0]
  temp_list = list(temp_dict.keys())
  return temp_list[0]
def classify_images(answers_dic,model):
   results_dic={}
   petlabel_dic = {}
   for key,value in answers_dic.items():
```

```
      label = classifier(image_dir,key,model).lower()
      petlabel_dic[key]=label.strip()
      if_same = 0
      found_idx = petlabel_dic[key].find(value)
      if found_idx ==0 and len(value)==len(petlabel_dic[key]):
         if_same = 1
      if ((found_idx == 0) or (petlabel_dic[key][found_idx-1]==" ")) and
((found_idx+len(value)==len(petlabel_dic[key]))or
(petlabel_dic[key][found_idx+len(value):found_idx+len(value)+1]in     ('
',','))):
         if_same = 1
      found_idx_2 = value.find(petlabel_dic[key])
      if found_idx_2 == 0 and len(value)==len(petlabel_dic[key]):
         if_same = 1
      if  ((found_idx_2  ==  0)  or  (value[found_idx_2-1]==""  "))  and
((found_idx_2 + len(petlabel_dic[key])== len(value)) or
(value[found_idx_2+len(petlabel_dic[key]):found_idx_2+len(petlabel_dic[
key])+1]in (' ',',','))):
         if_same = 1
      index_of_results_dic = []
      index_of_results_dic.append(value)
      index_of_results_dic.append(label)
      index_of_results_dic.append(if_same)
      results_dic[key] = index_of_results_dic
   return results_dic
```

TODO3 分为两个函数：classifier 函数和 classify_images 函数

在 classifier 函数外部，首先定义好模型的名称，在此处我们使用了 5 个模型（在后面的结果中发现 alexnet 准确率极低）。在 classfier 函数中，依照模型的使用方法，将图片路径放入字典 input_dict 中，利用模型识别网络，返回一个结果 result。其中，result[0][0]是一个键为图像识别返回的标签的字符串的字典，且字典中只有一个元素。temp_list = list(temp_dict.keys())，将键值全部提取出来并转化为列表形式，列表（只有一个元素）中的元素即为图像识别的标签。最后返回这个值。

在 classify_images 函数中，考虑到 classifier 函数一次只能返回一个标签值，所以将 classifier 函数运用在 for 循环中。首先创建两个字典，一个为 results_dic，键是

宠物图像文件名，值是关于宠物图像的列表，索引 0 为宠物图像标签，索引 1 为分类器标签，索引 2 为标签比较结果，最后作为返回值返回，另一个是 petlabel_dic，键为宠物文件名，值为宠物标签。classify_images 函数的形参有两个，一个是模型名 model，另一个是 TODO2 模块中的 answers_dic。在 for 循环中，首先传入图像文件夹地址，图像名称，模型名，返回的值进行小写操作，去掉两侧空格操作得到图片识别标签，将图像名作为键，图片识别标签作为值。

考虑到指导 PDF 中的字符串不匹配的情况有两种，一种原标签为 cat，分类器标签为 skunk，polecat， wood pussy and lynx catamount，并不匹配。还有一种是原标签为 german shepherd dog，分类器标签为 german shepherd，应该匹配，但是用指导 PDF 中的方法并不匹配，所以，不仅在原标签中使用 find 函数寻找分类器标签，还在分类器标签中使用 find 函数寻找原标签，满足任意一种即可。最后将宠物图像标签，分类器标签，标签比较结果放入列表 index_of_results_dic，再将 index_of_results_dic 作为值，宠物文件名作为键，放入 results_dic 字典中，最后将其返回。

TODO4 模块：

```
#TODO 4
def adjust_results4_isadog(result_dic,dogsfile):
      dognames_dict = {}
      with open(file=dogsfile) as file_object:
            for line in file_object:
                  dognames_dict[line.strip()]=1
      # print(dognames_dict)
      for key,value in result_dic.items():
            index_3 = index_4 = 0
      # value[0]是图片的 label，value[1]是分类器给出的标签,key_2 是 dognames 字典的键名
            for key_2 in dognames_dict.keys():
                  # 首先用 value[0]和 key_2 比对，然后是 value[1]和 key_2 比对
                  found_idx_1 = key_2.find(value[0])
                  if found_idx_1 ==0 and len(value[0])==len(key_2):
                        index_3 = 1
                  if ((found_idx_1 == 0) or (key_2[found_idx_1-1]==" "))
```

```
and((found_idx_1+len(value[0])==len(key_2))or
(key_2[found_idx_1+len(value[0]):found_idx_1+len(value[0])+1]in       ('
',','))):
                        index_3 = 1
                found_idx_2 = value[0].find(key_2)
                if found_idx_2 ==0 and len(value[0])==len(key_2):
                        index_3 = 1
                if ((found_idx_2 == 0) or (value[0][found_idx_2-1]=="
")) and ((found_idx_2 + len(key_2)== len(value[0])) or

(value[0][found_idx_2+len(key_2):found_idx_2+len(key_2)+1]in (' ',',','))):
                        index_3 = 1
                # 现在是 value[1]和 key_2 比对
                found_idx_3 = key_2.find(value[1])
                if found_idx_3 ==0 and len(value[1])==len(key_2):
                        index_4 = 1
                if ((found_idx_3 == 0) or (key_2[found_idx_3-1]==" "))
and((found_idx_3+len(value[1])==len(key_2))or
(key_2[found_idx_3+len(value[1]):found_idx_3+len(value[1])+1]in       ('
',','))):
                        index_4 = 1
                found_idx_4 = value[1].find(key_2)
                if found_idx_4 ==0 and len(value[1])==len(key_2):
                        index_4 = 1
                if ((found_idx_4 == 0) or (value[1][found_idx_4-1]=="
")) and ((found_idx_4 + len(key_2)== len(value[1])) or

(value[1][found_idx_4+len(key_2):found_idx_4+len(key_2)+1]in (' ',',','))):
                        index_4 = 1
                if index_3==1 and index_4==1:
                        break
            result_dic[key].append(index_3)
            result_dic[key].append(index_4)
```

此模块编写的是 adjust_results4_isadog 函数，首先打开文件 dognames.txt，获取有

效的小狗名称，去掉两侧的空格，以小狗名称作为键，1 为值，放入 dognames_dict 字典中。

与 TODO3 类似，将 result_dic 中的值列表中的索引 0（图片的原标签）与 dognames 字典中的键名作对比，使用两次 find 函数，将比对结果（0 或 1）放入列表的索引 3 中。同理，将索引 1（分类器标签）与 dognames 字典中的键名作对比，使用两次 find 函数，将比对结果（0 或 1）放入列表的索引 4 中。即修改完成 results_dic

TODO5 模块

```
def calculates_results_stats(result_dic):
    result_stats = {}
    n_picture = len(result_dic)  #Z 图像数量
    n_correct_dogs = 0   #A 小狗匹配正确的数量
    n_picture_of_dogs = 0   #B 小狗图像的数量
    n_incorrect_not_dogs = 0 #C 正确非小狗匹配项的数量，两个标签都不是小狗
    n_not_dogs = 0 #D 非小狗图像的数量
    n_correct_breed_and_dogs = 0 #E 正确品种匹配
    n_correct_breed = 0 #Y 标签匹配项的数量
    for key,value in result_dic.items():
      if value[3] == 1:
        n_picture_of_dogs += 1
      if value[3] == 1 and value[4] == 1:
        n_correct_dogs += 1
      if value[3] == 0 and value[4] == 0:
        n_incorrect_not_dogs += 1
      if value[3] == 0:
        n_not_dogs += 1
      if value[2] == 1 and value[3] == 1:
        n_correct_breed_and_dogs += 1
      if value[2] == 1:
        n_correct_breed += 1
    if n_picture_of_dogs == 0 or n_not_dogs == 0 or n_picture ==0:
      print("除数为 0!!!!!")
      return None
```

```
    else:
        #正确分类的小狗图像所占百分比
        pct_correct_dogs = n_correct_dogs / n_picture_of_dogs * 100
        #正确分类的非小狗图像所占百分比
        pct_correct_not_dogs = n_incorrect_not_dogs / n_not_dogs * 100
        #正确分类的小狗品种所占百分比
        pct_correct_dogs_sorted      =      n_correct_breed_and_dogs      /
n_picture_of_dogs * 100
        #百分比标签匹配数（无论是否为小狗）
        pct_correct_breed = n_correct_breed / n_picture * 100

        result_stats['n_picture'] = n_picture
        result_stats['n_correct_dogs'] = n_correct_dogs
        result_stats['n_picture_of_dogs'] = n_picture_of_dogs
        result_stats['n_incorrect_not_dogs'] = n_incorrect_not_dogs
        result_stats['n_not_dogs'] = n_not_dogs
        result_stats['n_correct_breed_and_dogs']                          =
n_correct_breed_and_dogs
        result_stats['n_correct_breed'] = n_correct_breed
        result_stats['pct_correct_dogs'] = pct_correct_dogs
        result_stats['pct_correct_not_dogs'] = pct_correct_not_dogs
        result_stats['pct_correct_dogs_sorted'] = pct_correct_dogs_sorted
        result_stats['pct_correct_breed'] = pct_correct_breed
        return result_stats
```

该模块编写 calculates_results_stats 函数，在此函数中，有多个参数：

1) n_picture = len(result_dic) #Z 图像数量
2) n_correct_dogs = 0 #A 小狗匹配正确的数量
3) n_picture_of_dogs = 0 #B 小狗图像的数量
4) n_incorrect_not_dogs = 0 #C 正确非小狗匹配项的数量，两个标签都不是小狗
5) n_not_dogs = 0 #D 非小狗图像的数量
6) n_correct_breed_and_dogs = 0 #E 正确品种匹配
7) n_correct_breed = 0 #Y 标签匹配项的数量

依据每个参数进行相应的计算，将值存入 result_stats 字典，键即为变量名：

1) 正确分类的小狗图像所占百分比 pct_correct_dogs = n_correct_dogs / n_picture_of_dogs * 100
2) 正确分类的非小狗图像所占百分比 pct_correct_not_dogs = n_incorrect_not_dogs / n_not_dogs * 100
3) 正确分类的小狗品种所占百分比 pct_correct_dogs_sorted = n_correct_breed_and_dogs / n_picture_of_dogs * 100
4) 百分比标签匹配数（无论是否为小狗） pct_correct_breed = n_correct_breed / n_picture * 100

TODO6 模块

```
def print_results(result_dic,result_stats,model_name,
print_incorroect_dogs=False,print_incorrect_breed = False):
    print(f"使用 CNN 模型架构中的{models[model_name]}架构，传入的参数是{model_name}")
    print(f"图像数量：{result_stats['n_picture']}")
    print(f"小狗图像数量：{result_stats['n_picture_of_dogs']}")
    print(f"非小狗图像数量：{result_stats['n_not_dogs']}")
    print("\n结果:")
    print(f"\t 正确的小狗图像所占百分比：{result_stats['pct_correct_dogs']}%")
    print(f"\t 正确的品种图像所占百分比：{result_stats['pct_correct_dogs_sorted']}%")
    print(f"\t 正确的非小狗图像所占百分比：{result_stats['pct_correct_not_dogs']}%")
    print(f"\t 匹配项所占百分比（包含小狗图像和非小狗图像）：{result_stats['pct_correct_breed']}%")
    if print_incorroect_dogs:
       print(f"\n分类错误的项：")
       print("\n    图片名：")
    for key,value in result_dic.items():
       if print_incorroect_dogs:
             if value[2] == 0 :
                   print(f"\t{key.title()}")
    if print_incorrect_breed:
```

```
    temp = []
    print("\n    狗狗品种：")
    for key,value in result_dic.items():
        if value[3] == 1 and value[2] == 0 :
            temp.append(value[0])
    # 由于狗狗的品种会有重复，使用 set 集合去除重复项
    unique = set(temp)
    for dogname in unique:
        print('\t',dogname)
```

TODO6 编写 print_results 函数。参数为 result_dic 字典，result_stats（TODO5 的返回值），模型名 model_name，以及两个默认参数

该函数主要打印结果。在分类错误项中，考虑到狗狗的品种可能会有重复的，所以使用 set 集合去除重复项。

实现截图：

```
图像数量：40
小狗图像数量：30
非小狗图像数量：10

结果：
        正确的小狗图像所占百分比：100.0%
        正确的品种图像所占百分比：80.0%
        正确的非小狗图像所占百分比：90.0%
        匹配项所占百分比（包含小狗图像和非小狗图像）：70.0%

分类错误的项：

    图片名：
        German_Shepherd_Dog_04890.Jpg
        Great_Pyrenees_05367.Jpg
        Boston_Terrier_02259.Jpg
        Gecko_80.Jpg
        Polar_Bear_04.Jpg
        Boston_Terrier_02285.Jpg
        German_Shorthaired_Pointer_04986.Jpg
        Boston_Terrier_02303.Jpg
        Cat_07.Jpg
        Cat_01.Jpg
        Great_Horned_Owl_02.Jpg
        Gecko_02.Jpg

    狗狗品种：
         german shorthaired pointer
         boston terrier
         german shepherd dog
         great pyrenees
程序运行时间：0:0:21
```

图 3. 5

比较模型的效果

```
import seaborn as sns
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#构建一个字典 total_result，将模型名称作为键，得到的相关结果作为值
total_result = {}
# 构建一个运行时间的字典，键为 CNN 模型名称，值为消耗的时间
total_time = {}
for key, value in models.items():
    start = time()
    result_dic_1 = classify_images(answers_dic,key)
    adjust_results4_isadog(result_dic_1,dogfile)
    total_result[key] = calculates_results_stats(result_dic_1)
    end = time()
    total = end - start
    total_time[key] = total

#一：展示程序运行消耗的时间
list_x = []
list_y = []
for key, value in total_time.items():
    list_x.append(key)
    list_y.append(value)
x = np.array(list_x)
y = np.array(list_y)
df = pd.DataFrame({"Runnig Time": x,"Time:(s)": y})
sns.barplot("Runnig Time","Time:(s)",palette="RdBu_r",data=df)
plt.savefig("result/running_time.png")
plt.show()

#二：正确分类的小狗图像所占百分比对比
pct_correct_dogs_x_list = []
pct_correct_dogs_y_list = []
for key, value in total_result.items():
```

```
    pct_correct_dogs_x_list.append(key)
    pct_correct_dogs_y_list.append(value['pct_correct_dogs'])
pct_correct_dogs_x = np.array(pct_correct_dogs_x_list)
pct_correct_dogs_y = np.array(pct_correct_dogs_y_list)
df = pd.DataFrame({"Percentage of correctly classified dogs images":pct_correct_dogs_x,"Percentage:%":pct_correct_dogs_y})
sns.barplot("Percentage of correctly classified dogs images","Percentage:%",palette="pastel",data=df)
plt.savefig("result/pecentage_of_correctly_classfied_dogs_images.png")
plt.show()

#三：正确分类的非小狗图像所占百分比对比
pct_correct_not_dogs_x_list = []
pct_correct_not_dogs_y_list = []
for key, value in total_result.items():
    pct_correct_not_dogs_x_list.append(key)
    pct_correct_not_dogs_y_list.append(value['pct_correct_not_dogs'])
pct_correct_not_dogs_x = np.array(pct_correct_not_dogs_x_list)
pct_correct_not_dogs_y = np.array(pct_correct_not_dogs_y_list)
df = pd.DataFrame({"Percentage of correctly classified non-dogs images":pct_correct_not_dogs_x,"Percentage:%":pct_correct_not_dogs_y})
sns.barplot("Percentage of correctly classified non-dogs images","Percentage:%",palette="Set2",data=df)
plt.savefig("result/pecentage_of_correctly_classfied_not_dogs_images.png")
plt.show()

#四：正确分类的小狗品种所占百分比对比
pct_correct_dogs_sorted_x_list = []
pct_correct_dogs_sorted_y_list = []
for key, value in total_result.items():
    pct_correct_dogs_sorted_x_list.append(key)

pct_correct_dogs_sorted_y_list.append(value['pct_correct_dogs_sorted'])
```

```
pct_correct_dogs_sorted_x = np.array(pct_correct_dogs_sorted_x_list)
pct_correct_dogs_sorted_y = np.array(pct_correct_dogs_sorted_y_list)
df = pd.DataFrame({'Percentage of correctly classified dogs':pct_correct_dogs_sorted_x,"Percentage:%":pct_correct_dogs_sorted_y})
sns.barplot("Percentage of correctly classified dogs","Percentage:%",palette="Set2",data=df)
plt.savefig("result/pecentage_of_correctly_classfied_dogs.png")
plt.show()

#五：标签匹配数所占百分比对比
pct_correct_breed_x_list = []
pct_correct_breed_y_list = []
for key, value in total_result.items():
    pct_correct_breed_x_list.append(key)
    pct_correct_breed_y_list.append(value['pct_correct_breed'])
pct_correct_breed_x = np.array(pct_correct_breed_x_list)
pct_correct_breed_y = np.array(pct_correct_breed_y_list)
df = pd.DataFrame({"Percentage of the number of tag matches":pct_correct_breed_x,"Percentage:%":pct_correct_breed_y})
sns.barplot("Percentage of the number of tag matches","Percentage:%",palette="ch:s=.25,rot=-.25",data=df)
plt.savefig("result/pecentage_of_the_number_of_tag_matches.png")
plt.show()
```

在该模块中，调用了 seaborn，numpy，pandas 和 matplotlib 库，绘制了以下五幅图：

1) 程序运行消耗的时间
2) 正确分类的小狗图像所占百分比对比
3) 正确分类的非小狗图像所占百分比对比
4) 正确分类的小狗品种所占百分比对比
5) 标签匹配数所占百分比对比

3.2.2 使用 Finetune 训练模型模块

由于第二部分和第三部分除了自己找的 50 张图片不同，其他都是一样的，故合并在此处一起写。

准备工作：

```
!pip install paddlehub==1.7.0 -i https://pypi.tuna.tsinghua.edu.cn/simple
# -*- coding: utf8 -*-
import paddlehub as hub
%set_env CPU_NUM=1
import paddle
paddle.enable_static()
# 选择模型
# 此处代码为加载 Hub 提供的图像分类的预训练模型
module = hub.Module(name="googlenet_imagenet")
```

首先导入 paddlehub 库，然后选择模块一中准确率最高的模型：googlenet_imagenet

数据准备

```
# 此处演示的是直接用 PaddleHub 提供的数据集
from paddlehub.dataset.base_cv_dataset import BaseCVDataset
class DemoDataset(BaseCVDataset):
    def __init__(self):
        # 数据集存放位置
        self.dataset_dir = "work/zhongyaocai"
        super(DemoDataset, self).__init__(
            base_path=self.dataset_dir,
            train_list_file="train_list.txt",
            validate_list_file="validate_list.txt",
            test_list_file="test_list.txt",
            # predict_file="predict_list.txt",
            label_list_file="label_list.txt",
            # label_list=["数据集所有类别"]
            )
dataset = DemoDataset()
```

制作数据集：编写了两个程序，代码如下

程序一：

```
# -*- coding: utf-8 -*-
```

```
import os
labels = []
with open("label_list.txt") as label_object:
    label_list = label_object.readlines()
    for str_label in label_list:
        temp_list = str_label.split(' ')
        labels.append(temp_list[0])

# 设定文件路径
for label in labels:
    path = 'classification/'+label
    i = 1
    # 对目录下的文件进行遍历
    for file in os.listdir(path):
        # 判断是否是文件
        if os.path.isfile(os.path.join(path, file)) == True:
            # 设置新文件名
            new_name=file.replace(file,
f"{label}_used_for_classification_{i}.jpg")
            # 重命名
            os.rename(os.path.join(path,    file),    os.path.join(path,
new_name))
            i += 1
# 结束
print("End")
```

程序一针对于自己获取的 50 张图片命名不规范的问题做出了修改，首先程序读取 label_list.txt 中的五个标签，分别对应到文件夹中五种中药材。将五种中药材的名字统一改为“药材名”+_used_for_classification_+“数字”+.jpg 的格式。并且移动到对应的文件夹中。

这 50 张图片命名规范对主程序进行图像分类没有任何影响，只是为了在分类后方便比对是否分类错误。

程序二：

```
from os import listdir
```

```
import random

def get_item_labels(file):
    # 返回的是一个列表，获取文件夹里的文件名。
    filename_list = listdir(file)
    itemlabels_list = []
    for filename in filename_list:
        itemlabels_list.append(filename)
    return itemlabels_list



folder_names = ['baihe','dangshen', 'gouqi',
                'huaihua', 'jinyinhua']

# 读取所有的图片
total_lists = {}
for folder in folder_names:
    temp_list = get_item_labels(folder)
    random.shuffle(temp_list)
    total_lists[folder] = temp_list

label = 0
with open('train_list.txt', 'w') as file_object_1:
    with open('test_list.txt', 'w') as file_object_2:
        with open('validate_list.txt', 'w') as file_object_3:
            for key, value in total_lists.items():
                num = 0
                for name in value:
                    num += 1
                    if num <= 120:
                        file_object_1.write(key + "/" + name + ' ' +
str(label) + '\n')
                    if 120 < num <= 150:
                        file_object_2.write(key + "/" + name + ' ' +
```

```
str(label) + '\n')
                if num > 150:
                    file_object_3.write(key + "/" + name + ' ' +
str(label) + '\n')
                label += 1
```

程序二是为了将老师提供的数据集图片标签写入到 train_list.txt、test_list.txt、validate_list.txt 三个 txt 文件中。

通过以上两个程序，修改好了自定义数据集以及自己要找的 50 张中草药图片

中间步骤（包括：生成 Reader、选择运行时配置、组建 Finetune Task、开始 Finetune）

```
data_reader = hub.reader.ImageClassificationReader(
    image_width=module.get_expected_image_width(), #预期图片经过 reader 处理后的图像宽度
    image_height=module.get_expected_image_height(),#预期图片经过 reader 处理后的图像高度
    images_mean=module.get_pretrained_images_mean(),#进行图片标准化处理时所减均值。默认为 None
    images_std=module.get_pretrained_images_std(), #进行图片标准化处理时所除标准差。默认为 None
dataset=dataset)

config = hub.RunConfig(
    use_cuda=True,     #是否使用 GPU 训练，默认为 False;
    num_epoch=20,      #Fine-tune 的轮数；使用 4 轮，直到训练准确率达到 90%多
    checkpoint_dir="cv_finetune_turtorial_demo",        #模型 checkpoint 保存路径，若用户没有指定，程序会自动生成;
    batch_size=32,                                      #训练的批大小，如果使用 GPU，请根据实际情况调整 batch_size;
    eval_interval=50,                                   #模型评估的间隔，默认每 100 个 step 评估一次验证集;
strategy=hub.finetune.strategy.DefaultFinetuneStrategy()) #Fine-tune 优化策略;
```

```
#获取 module 的上下文信息包括输入、输出变量以及 paddle program
input_dict, output_dict, program = module.context(trainable=True)

#待传入图片格式
img = input_dict["image"]

#从预训练模型的输出变量中找到最后一层特征图，提取最后一层的 feature_map
feature_map = output_dict["feature_map"]

#待传入的变量名字列表
feed_list = [img.name]

task = hub.ImageClassifierTask(
    data_reader=data_reader,        #提供数据的 Reader
    feed_list=feed_list,            #待 feed 变量的名字列表
    feature=feature_map,            #输入的特征矩阵
    num_classes=dataset.num_labels, #分类任务的类别数量，此处来自于数据集的
num_labels
config=config)                  #运行配置

run_states = task.finetune_and_eval() # 通过众多 finetune API 中的
finetune_and_eval 接口，可以一边训练网络，一边打印结果
```

这部分步骤大部分没有进行修改，保持原样

use_cuda=True 使用 GPU 训练，num_epoch=20 设置 finetune 的轮数为 20 轮。

使用模型进行预测

```
import numpy as np
import random
import os
import shutil

# 获取所有图片的路径
def get_file_names():
    filenames                                                    =
```

```
os.listdir("work/zhongyaocai/classification/total_images/")
    random.shuffle(filenames)
    new_filenames = []
    for filename in filenames:
        new_filename                                                    =
"work/zhongyaocai/classification/total_images/"+filename
        new_filenames.append(new_filename)
    return new_filenames

# 复制文件到指定的文件夹中
def copyfile(srcfile, dstfile):
    if not os.path.isfile(srcfile):
        print("%s not exist!" % (srcfile))
    else:
        fpath, fname = os.path.split(dstfile)  # 分离文件名和路径
        if not os.path.exists(fpath):
            os.makedirs(fpath)  # 创建路径
        shutil.copyfile(srcfile, dstfile)  # 复制文件

data = get_file_names()       #此处传入需要识别的照片地址
# print(data)  #测试使用
label_map = dataset.label_dict()  # 结果为{0: 'baihe 0', 1: 'dangshen 1',
2: 'gouqi 2', 3: 'huaihua 3', 4: 'jinyinhua 4'}
index = 0
label_folders = {}   # 创建一个字典，键为标签（0 到 4），值为对应的文件夹的名字
for label,value in label_map.items():
    str_list = value.split(' ')
    label_folders[label] = str_list[0]

classify_images_dir = "work/zhongyaocai/classification/"

# get classification result
run_states = task.predict(data=data) #进行预测
results = [run_state.run_results for run_state in run_states] #得到用新模
```

```
型预测 test 照片的结果


for batch_result in results:
    # get predict index
    batch_result = np.argmax(batch_result, axis=2)[0]
    for result in batch_result:
        index += 1
        # 将图片复制到识别后对应的文件夹中,首先获取文件名，然后创建复制后的路径
        str_list = data[index-1].split('/')
        destFile = classify_images_dir + label_folders[result] + '/' +
str_list[-1]
        sourceFile = data[index -1]
        copyfile(sourceFile, destFile)
        result = label_map[result]
        print("input %i is %s, and the predict result is %s" %
              (index, data[index - 1], result))
```

首先导入模块 numpy、random、os、shutil 库

自定义函数：

- get_file_names 函数：获取所有图片的路径，即自己寻找的 50 张中药材图片，该文件夹路径为：work/zhongyaocai/classification/total_images。50 张中药材图片都在其中。使用方法 os.listdir 获取所有的图片名称，再用 random.shuffle 打乱顺序，体现随机性。将 50 张随机排序的图片名加上路径，并放在列表 new_filenames 中并返回。
- copyfile 函数：接受两个参数 srcfile 和 dstfile，srcfile 是原来的文件，dstfile 是要复制到某处的文件。首先判断文件 srcfile 是否存在，若存在则将目标文件分离文件名和路径。若路径存在，则将文件 srcfile 复制到 dstfile 处。注：该函数是用于复制，而不是移动文件。

首先调用 get_file_names 函数，得到含有 50 张随机排序的图片名（含路径）的列表 data。读取数据集中文件 label_list，返回一个字典 label_map（键值对为：{0: 'baihe 0', 1: 'dangshen 1', 2: 'gouqi 2', 3: 'huaihua 3', 4: 'jinyinhua 4'}）。创建变量 index，初始值设置为 0，用于确定一共识别了多少张图片。然后创建一个字典：键为标签（0 到 4），值为对应的文件夹的名字。即：{0: 'baihe', 1: 'dangshen', 2: 'gouqi', 3: 'huaihua', 4: 'jinyinhua'}，创建变量 classify_images_dir，用于存储用于分类的文件夹的路径

接着进行预测，将 data 数据传入方法 task.predict 中，通过解析列表得到结果

results。对 results 进行处理，最后图像识别的结果存储到 result（即为数字 0~4）中，首先找到源文件路径，即为 sourceFile = data[index -1]，然后找到目标路径，即为 destFile = classify_images_dir + label_folders[result] + '/' + str_list[-1]。调用函数 copyfile，将源文件复制到目标路径。result = label_map[result]，将原来的 result 中数字转化为更清晰易懂的 str 类型，例："jinyinhua 4"．并打印出结果

3.2.3 数据集增强代码展示模块

创建 Img 类

```
import math
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image

def DotMatrix(A, B):
    return np.matmul(A, B)  # 返回矩阵

class Img:
    def __init__(self, image, rows, cols, center=[0, 0]):
        self.src = image  # 原始图像,是一个二维数组，每一个元素都是一个 1*3 矩阵
        self.rows = rows  # 原始图像的行
        self.cols = cols  # 原始图像的列
        self.center = center  # 旋转中心，默认是[0,0]
        self.change_src_pos = False  # 是否更改旋转中心，用于旋转函数中，其他函数均为 False
        self.angle_270 = False # 在旋转函数中，判断旋转角度是否为 270°
        self.angle_90 = False  # 在旋转函数中，判断旋转角度是否为 90°

    def Move(self, delta_x, delta_y):
        '''
        本函数处理生成做图像平移的矩阵
        '''
        self.transform = np.array([[1, 0, delta_x],
                                   [0, 1, delta_y],
                                   [0, 0, 1]])
```

```
        self.change_src_pos = False

    def Zoom(self, factor):  # 缩放
        # factor>1 表示缩小；factor<1 表示放大
        self.transform = np.array([[factor, 0, 0],
                                   [0, factor, 0],
                                   [0, 0, 1]])
        self.change_src_pos = False

    def Horizontal(self):
        # 水平镜像
        self.transform = np.array([[-1, 0, self.rows],
                                   [0, 1, 0],
                                   [0, 0, 1]])
        self.change_src_pos = False

    def Vertically(self):
        # 垂直镜像
        self.transform = np.array([[1, 0, 0],
                                   [0, -1, self.cols],
                                   [0, 0, 1]])
        self.change_src_pos = False

    def Rotate(self, beta):  # 旋转，传入的值是数字，不是弧度
        # beta>0 表示逆时针旋转；beta<0 表示顺时针旋转
        if (beta + 90) % 360 == 0:
            self.angle_270 = True
        elif (beta - 90) % 360 == 0:
            self.angle_90 = True
        else:
            self.angle_270 = False
            self.angle_90 = False
        beta = math.radians(beta) #将数字转化为弧度
        self.transform = np.array([[math.cos(beta), math.sin(beta), 0],
```

```
                                [-math.sin(beta), math.cos(beta), 0],
                                [0, 0, 1]])
        self.change_src_pos = True # 改变旋转的中心

    def Process(self):
        if self.change_src_pos:                 # 用于旋转
            # 初始化定义目标图像，具有 3 通道 RBG 值
            if self.angle_270 is False and self.angle_90 is False:
                self.dst    =    np.zeros((self.rows,    self.cols,    3),
dtype=np.uint8)
            else:  #如果是旋转 270°或 90°，则需要修改图片的大小，将图片的像素高
和宽互换
                self.dst    =    np.zeros((self.cols,    self.rows,    3),
dtype=np.uint8)
            #  提供 for 循环，遍历图像中的每个像素点，然后使用矩阵乘法，找到变换
后的坐标位置
            for i in range(self.rows):
                for j in range(self.cols):
                    src_pos = np.array([i - self.rows / 2, j - self.cols /
2, 1])  # 设置原始坐标点矩阵
                    [x, y, z] = DotMatrix(self.transform, src_pos)  # 和对
应变换做矩阵乘法
                    x = int(x) + int(self.rows / 2)
                    y = int(y) + int(self.cols / 2)

                    if self.angle_270 is False and self.angle_90 is False:
                        # 旋转角度不是 270°或 90°，采用一般的旋转方法，生成的图
片会有部分空白（180°和 360°除外）
                        if x >= self.rows or y >= self.cols or x < 0 or y
< 0:
                            self.dst[i][j] = 255  # 处理未落在原图像中的点的
情况
                        else:
                            self.dst[i][j] = self.src[x][y]  # 使用变换后的
```

```
位置
                elif self.angle_90:
                    # 旋转 90°，生成的图片不会存在空白
                    self.dst[j][i] = self.src[i][self.cols - 1 - j]
                elif self.angle_270:
                    # 旋转 270，生成的图片不会存在空白
                    self.dst[j][i] = self.src[i][j]
        else:  # 其他的操作：镜像，放缩，平移
            # 初始化定义目标图像，具有 3 通道 RBG 值
            self.dst = np.zeros((self.rows, self.cols, 3), dtype=np.uint8)
            #  提供 for 循环，遍历图像中的每个像素点，然后使用矩阵乘法，找到变换
后的坐标位置
            for i in range(self.rows):
                for j in range(self.cols):
                    src_pos = np.array([i - self.center[0], j -
self.center[1], 1])  # 设置原始坐标点矩阵
                    [x, y, z] = DotMatrix(self.transform, src_pos)  # 和对
应变换做矩阵乘法
                    x = int(x) + self.center[0]
                    y = int(y) + self.center[1]
                    if x >= self.rows or y >= self.cols or x < 0 or y < 0:
                        self.dst[i][j] = 255  # 处理未落在原图像中的点的情况
                    else:
                        self.dst[i][j] = self.src[x][y]  # 使用变换后的位置
```

首先导入模块 math、matplotlib、numpy、PIL 中的 Image，接着定义函数 DotMatrix，用于矩阵的乘法，返回的是结果。创建类 Img，属性有

1) self.src：用于存储原始图像，是一个二维的数组，每一个元素都是一个 1*3 的矩阵
2) self.rows：用于存储原始图像的行
3) self.cols：用于存储原始图像的列
4) self.center：旋转中心的坐标，默认是[0,0]
5) self.change_src_pos：是否更改旋转中心，默认为 False
6) self.angle_270：在旋转函数中，判断旋转角度是否为 270°
7) self.angle_90：在旋转函数中，判断旋转角度是否为 90°

定义的方法有：

1） Move：用于处理图像平移

2） Zoom：用于处理图像缩放

3） Horizontal：用于处理图像水平镜像

4） Vertically：用于处理图像垂直镜像

5） Rotate：用于处理图像旋转

6） Process：用于图像的处理

主程序：

```
if __name__ == '__main__':
    infer_path = r'sample_picture.png'  # 示例图片
    imgv = Image.open(infer_path)  # 打开图片
    plt.imshow(imgv)  # 根据数组绘制图像
    print("原图像")
    plt.show()  # 显示图像
    rows = imgv.size[1]
    cols = imgv.size[0]
    print(rows, cols)  # 注意此处 rows 和 cols 的取值方式，获得的是图片的像素高和宽
    imgv = np.array(imgv)  # 从图像生成数组
    img = Img(imgv, rows, cols, [0, 0])  # 生成一个自定 Img 类对象[0,0]代表处理的中心点
    #以下操作为图像的绘制
    # 1.旋转 10°
    img.Rotate(10)   # 旋转 10°
    img.Process()  # 进行矩阵变换
    img1 = Image.fromarray(img.dst)  # 从处理后的数组生成图像
    plt.imshow(img1)
    plt.savefig("pictures/rotate_10_angle.png") # 保存图片
    print("逆时针旋转图片 10°")
    plt.show()

    # 2.旋转 90°
    img.Rotate(90) # 旋转 90°
```

```
img.Process()  # 进行矩阵变换
img2 = Image.fromarray(img.dst)  # 从处理后的数组生成图像
plt.imshow(img2)
plt.savefig("pictures/rotate_90_angle.png") # 保存图片
print("逆时针旋转图片 90°")
plt.show()

# 3.旋转 270°
img.Rotate(270) # 旋转 270°
img.Process()  # 进行矩阵变换
img3 = Image.fromarray(img.dst)  # 从处理后的数组生成图像
plt.imshow(img3)
plt.savefig("pictures/rotate_270_angle.png") # 保存图片
print("逆时针旋转图片 270°")
plt.show()

# 4.旋转 180°
img.Rotate(180) # 旋转 180°
img.Process()  # 进行矩阵变换
img4 = Image.fromarray(img.dst)  # 从处理后的数组生成图像
plt.imshow(img4)
plt.savefig("pictures/rotate_180_angle.png") # 保存图片
print("逆时针旋转图片 180°")
plt.show()

# 5.镜像
img.Vertically() #垂直镜像(0,0)
img.Process()  # 进行矩阵变换
img5 = Image.fromarray(img.dst)  # 从处理后的数组生成图像
plt.imshow(img5)
plt.savefig("pictures/vertical.png") # 保存图片
print("垂直镜像")
plt.show()
```

```
# 6.镜像
img.Horizontal() #水平镜像(0, 0)
img.Process()  # 进行矩阵变换
img6 = Image.fromarray(img.dst)  # 从处理后的数组生成图像
plt.imshow(img6)
plt.savefig("pictures/horizontal.png") # 保存图片
print("水平镜像")
plt.show()

# 7.平移
img.Move(-500,-500) #平移(-500,-500)
img.Process()  # 进行矩阵变换
img7 = Image.fromarray(img.dst)  # 从处理后的数组生成图像
plt.imshow(img7)
plt.savefig("pictures/move.png") # 保存图片
print("平移图片")
plt.show()

# 8.平移
img.Zoom(0.5) #缩放
img.Process()  # 进行矩阵变换
img8 = Image.fromarray(img.dst)  # 从处理后的数组生成图像
plt.imshow(img8)
plt.savefig("pictures/zoom.png") # 保存图片
print("放大图片")
plt.show()
```

a) 首先传入示例图片路径，通过 Image 处理，再用 matplotlib 库中函数 imshow 和 show 显示原图像

b) 通过处理 imgv 生成数组 imgv。创建 Img 类的实例 img

c) 图像操作：

1) 旋转 10°
2) 旋转 90°
3) 旋转 270°
4) 旋转 180°

5） 垂直镜像

6） 水平镜像

7） 平移

8） 缩放

第四章　复杂工程问题分析

4.1 第一部分预训练模型问题及解决方案

1) 模型使用

a） 问题描述：调用模型时不知道怎么使用模型

b） 解决方法：查阅官方文档资料（网址为：https://aistudio.baidu.com/aistudio/projectdetail/361892），获得了示例代码：

```
import paddlehub as hub
module = hub.Module(name="alexnet_imagenet")
test_img_path = "/PATH/TO/IMAGE/"
# set input dict
input_dict = {"image": [test_img_path]}
# execute predict and print the result
results = module.classification(data=input_dict)
for result in results:
    print(result)
```

2) resnet 模型

a） 问题描述：resnet 模型准确率为 0，图片识别的结果与图片的标签完全匹配不上

b） 解决方法：查找其他的模型，最后使用新的模型：googlenet_imagenet 和 mobilenet_v2_imagenet

3) find 函数

a） 问题描述：考虑到指导 PDF 中的字符串不匹配的情况有两种，一种原标签为 cat，分类器标签为 skunk，polecat，wood pussy and lynx catamount，并不匹配。还有一种是原标签为 german shepherd dog，分类器标签为 german shepherd，应该匹配，但是用指导 PDF 中的方法并不匹配

b） 解决方法：不仅在原标签中使用 find 函数寻找分类器标签，还在分类器标签中使用 find 函数寻找原标签，满足任意一种即可。

4) 输出结果

a） 问题描述：在主程序输出错误的项时，输出错误的狗狗品种，对于某些

模型，可能某一类的狗狗都会识别错误，此时输出的结果就会有重复的项

b) 解决方法：为了消除重复的项，考虑到集合的唯一性，对识别错误狗狗的品种的列表 temp 使用 set 集合去除重复项：

```
# 由于狗狗的品种会有重复，使用 set 集合去除重复项
unique = set(temp)
for dogname in unique:
   print('\t',dogname)
```

4.2 第二部分使用 Finetune 问题及解决方案

1) 数据准备
   a) 问题描述：在 baihe 文件夹中的文件并没有 200 张，但是图片的编号达到了 200 多
   b) 解决方案：重新命名 baihe 文件夹中的所有文件。使用 for 循环。
2) predict_file
   a) 问题描述：在数据准备中，添加指导 word 中的代码 predict_file="predict_list.txt"时，程序报错。
   b) 解决方案：询问老师，应该是继承的类中删去了这个属性。将这一行代码注释掉，不使用 predict_list.txt
3) 将识别的文件移动到指定处
   a) 问题描述：不知道怎么将文件移动
   b) 解决方案：在网上查找资料，自己定义了一个函数 copyfile()，将文件复制到指定的路径
4) 文件数量
   a) 问题描述：文件的总数超过了 1000，即超过了上传的上限
   b) 解决方案：与老师沟通后，先将程序运行，然后将数据集压缩，与其他的文件一起上传。

4.3 数据集增强板块问题及解决方案

1) 绘制图像
   a) 问题描述：绘制图像时有时会无法显示图像，或多张图像显示部分
   b) 解决方案：切换网络，不使用学校的 uestc-wifi 连接，使用手机流量开热点连接
2) 旋转问题
   a) 问题描述：在旋转的过程中，当旋转到 90° 的倍数时，考虑到图像是正

的，周围没有空白（不像旋转 10° 时图像是斜的，图像旁边有部分空白），故想要修改原有代码，考虑到特殊的情况

b） 解决方案：将原来传递的参数（弧度）更改为新的参数（数字。例：90 代表 90°）。使用 if-elif-else 判断角度是否为 90 的倍数，是 90 的多少倍。在不同的情况下修改图片大小的格式

第五章　团队协作和人员实现展示

5.1 团队协作方式

1) 我们小组采取课堂上相互讨论指出项目 bug，课堂下整合资料的方式协作。在课堂上，我们负责写自己的代码，出现问题时会相互交流讨论，在网上查找解决方案，如果网上没有，就会去请教老师。在课堂外，我们会聚在一起，分享自己代码中的易错点。小组分工在下面人员完成情况展示中。

2) 心得：小组合作减轻了每一个人的负担，让我们可以专注于负责自己的模块，而不用去关心多余的事情。此外，小组分工中，与小组成员的合作也使我更进一步意识到了团队合作的重要性，也让我们明白了在与小组成员共同编写程序时，要注意格式的规范，增强程序的可读性。

3) 小组得分：（按比例：陈霆润：2，赵之航：1，幸锴文：1）

5.2 人员完成情况展示

TODO5+TODO6 的运行结果内容：（因为 TODO6 就是打印 TODO5 的内容，故以下的截图只有 TODO6）

陈霆润：

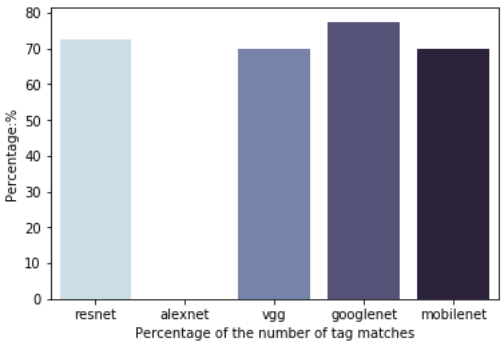

图 5.2.1(a)

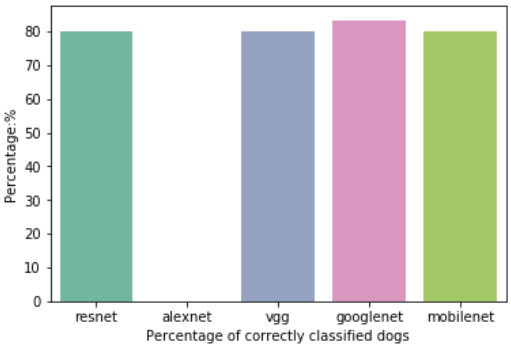

图 5.2.1(b)

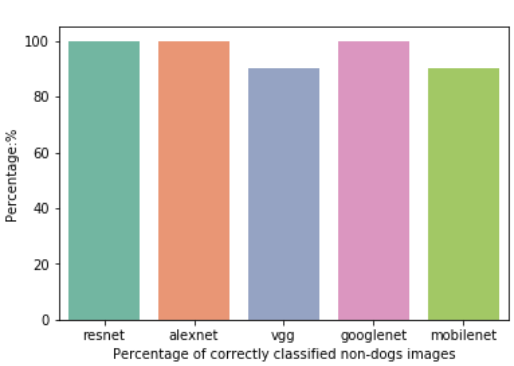

图 5.2.1（c）

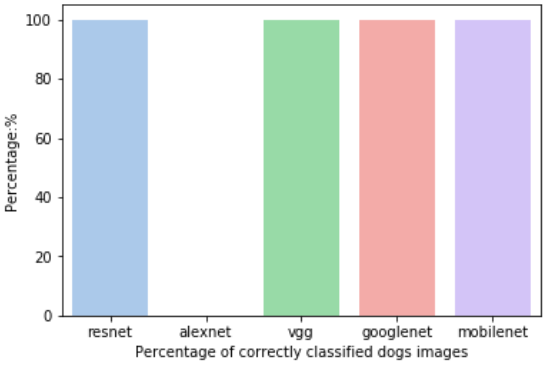

图 5.2.1（d）

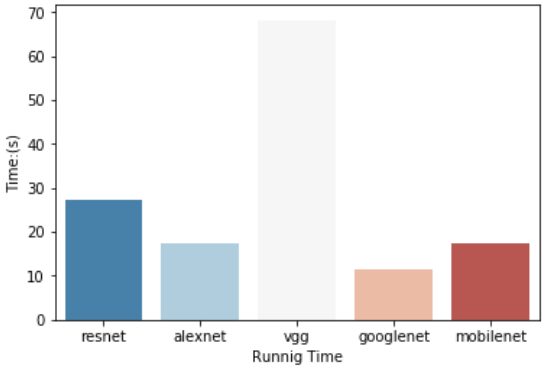

图 5.2.1（e）

```
图像数量：40
小狗图像数量：30
非小狗图像数量：10

结果：
        正确的小狗图像所占百分比：100.0%
        正确的品种图像所占百分比：80.0%
        正确的非小狗图像所占百分比：90.0%
        匹配项所占百分比（包含小狗图像和非小狗图像）：70.0%
```

```
分类错误的项：

    图片名：
        German_Shepherd_Dog_04890.jpg
        Great_Pyrenees_05367.jpg
        Boston_Terrier_02259.jpg
        Gecko_80.jpg
        Polar_Bear_04.jpg
        Boston_Terrier_02285.jpg
        German_Shorthaired_Pointer_04986.jpg
        Boston_Terrier_02303.jpg
        Cat_07.jpg
        Cat_01.jpg
        Great_Horned_Owl_02.jpg
        Gecko_02.jpg

    狗狗品种：
         german shorthaired pointer
         boston terrier
         german shepherd dog
         great pyrenees
程序运行时间：0:0:21
```

图 5.2.2

赵之航：

```
Key: Dalmatian_04017.jpg  Value: ['Dalmatian_04017.jpg'
Key: Beagle_01141.jpg  Value: ['Beagle_01141.jpg', 'bea
Key: skunk_029.jpg  Value: ['skunk_029.jpg', 'skunk', 1
Key: Golden_retriever_05195.jpg  Value: ['Golden_retrie

Key: Great_pyrenees_05367.jpg  Value: ['Great_pyrenees_
Key: cat_01.jpg  Value: ['cat_01.jpg', 'dhole', 0, 0, 0
Key: German_shepherd_dog_04890.jpg  Value: ['German_shepherd_dog_04890.jpg', 'malino
Key: gecko_80.jpg  Value: ['gecko_80.jpg', 'African chameleon', 0, 0, 0]
Key: cat_02.jpg  Value: ['cat_02.jpg', 'tabby', 0, 0, 0]
Key: great_horned_owl_02.jpg  Value: ['great_horned_owl_02.jpg', 'hare', 0, 0, 0]
TODO 5
  小狗图像数量     30
  图像是小狗的比例     75.0
  正确分类的小狗图像数量     30
  正确分类为小狗图像所占的比例     100.0
  正确分类为非小狗图像所占的比例     100.0
  正确分类的_小狗_品种图像所占百分比     93.3
  正确匹配的图像所占百分比（无论是否为小狗）     85.0
```


图 5.2.3

幸锴文：

运行时长: 34毫秒　结束时间: 2021-05-16 19:57:59

```
使用CNN模型架构中的resnet_v2_50_imagenet架构，传入的参数为resnet
图像数量： 40
小狗图像数量： 30
非小狗图像数量： 10
正确的小狗图像所占百分比： 100.0
正确的品种图像所占百分比： 80.0
正确的非小狗图像所占百分比： 100.0
匹配项所占百分比： 72.5
错误的项有：
Rabbit_002.jpg
Boston_terrier_02303.jpg
Boston_terrier_02259.jpg
Boston_terrier_02285.jpg
Great_pyrenees_05367.jpg
fox_squirrel_01.jpg
polar_bear_04.jpg
great_horned_owl_02.jpg
German_shorthaired_pointer_04986.jpg
gecko_02.jpg
German_shepherd_dog_04890.jpg
gecko_80.jpg
cat_07.jpg
skunk_029.jpg
cat_01.jpg
cat_02.jpg
```

图 5.2.4

小组分工：（针对于任务使用 Finetune 和任务数据集增强）

陈霆润：

1) 编写两个小程序，一是为了修改 baihe 文件夹中的文件命名不规范，二是为了将 50 张图片重命名，便于主程序运行后对照结果。
2) 负责数据集增强中的类的旋转操作和处理操作方法
3) 主函数中的旋转 10、90、270、180 四种图像的绘制
4) 负责最终报告

赵之航：

1) 查找 2 组不同的 50 张中药材图片，以及数据集的相关制作
2) 负责数据集增强的矩阵乘法、类的平移操作和缩放操作方法
3) 主函数中对原图像的处理、绘制，以及平移图像绘制、缩放图像绘制

幸锴文：

1) 编写函数 get_file_names，获取 50 张图片的路径，并存储在 data 列表中；
2) 编写函数 copyfile，将文件复制到指定处
3) 负责数据集增强中的类的水平镜像和垂直镜像方法
4) 主函数中的水平镜像图像绘制、垂直镜像图像绘制

任务二的准确率：

```
识别准确率为：94.0%

识别错误的图片：
        图片jinyinhua_used_for_classification_1.jpg，被识别为huaihua类
        图片dangshen_used_for_classification_2.jpg，被识别为huaihua类
        图片jinyinhua_used_for_classification_9.jpg，被识别为huaihua类
```

图 5.2.5(a)

条形统计图如下：

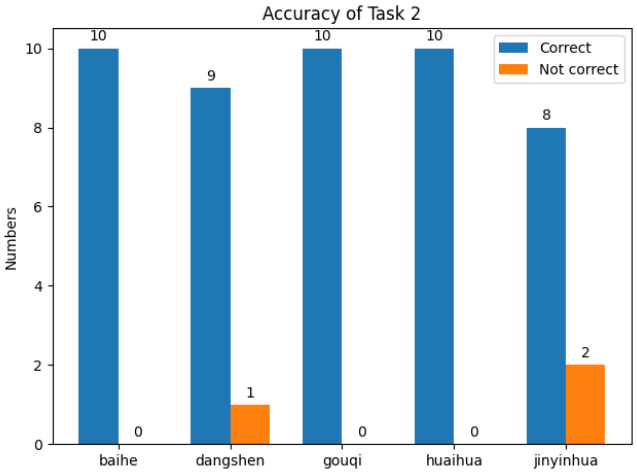

图 5.2.5(b)

任务三准确率：

```
识别准确率为：96.0%

识别错误的图片：
        图片baihe_used_for_classification_7.jpg，被识别为gouqi类
        图片baihe_used_for_classification_8.jpg，被识别为gouqi类
```

图 5.2.6(a)

条形统计图如下：

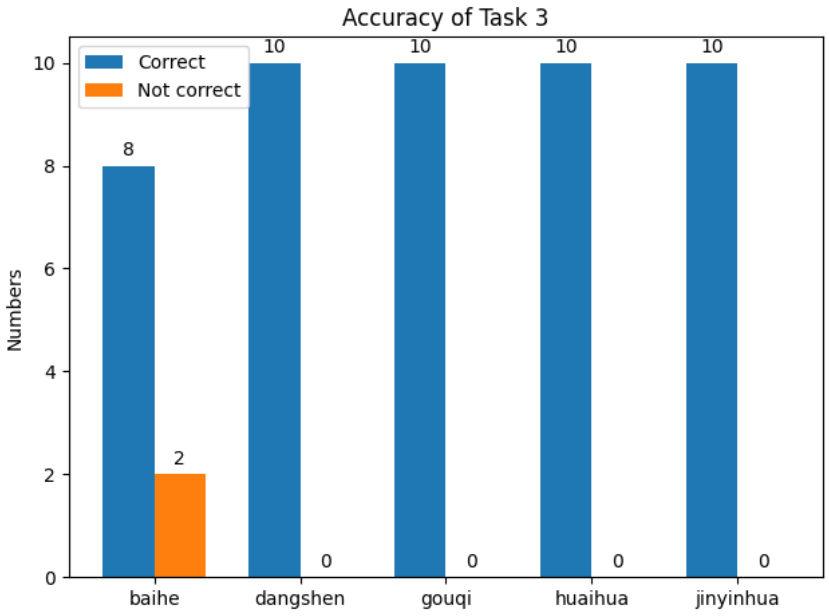

图 5.2.6(b)

任务二实现效果截图：

```
[2021-04-06 17:39:04,247] [    INFO] - PaddleHub predict start
[2021-04-06 17:39:04,250] [    INFO] - The best model has been loaded
[2021-04-06 17:39:05,042] [    INFO] - PaddleHub predict finished.
input 1 is work/zhongyaocai/classification/total_images/gouqi_used_for_classification_7.jpg, and the predict result is gouqi 2
input 2 is work/zhongyaocai/classification/total_images/gouqi_used_for_classification_4.jpg, and the predict result is gouqi 2
input 3 is work/zhongyaocai/classification/total_images/dangshen_used_for_classification_10.jpg, and the predict result is dangshen 1
input 4 is work/zhongyaocai/classification/total_images/huaihua_used_for_classification_10.jpg, and the predict result is huaihua 3
input 5 is work/zhongyaocai/classification/total_images/jinyinhua_used_for_classification_4.jpg, and the predict result is jingyinhua 4
input 6 is work/zhongyaocai/classification/total_images/huaihua_used_for_classification_2.jpg, and the predict result is huaihua 3
input 7 is work/zhongyaocai/classification/total_images/jinyinhua_used_for_classification_9.jpg, and the predict result is huaihua 3
input 8 is work/zhongyaocai/classification/total_images/dangshen_used_for_classification_3.jpg, and the predict result is dangshen 1
input 9 is work/zhongyaocai/classification/total_images/gouqi_used_for_classification_8.jpg, and the predict result is gouqi 2
input 10 is work/zhongyaocai/classification/total_images/baihe_used_for_classification_10.jpg, and the predict result is baihe 0
input 11 is work/zhongyaocai/classification/total_images/huaihua_used_for_classification_5.jpg, and the predict result is huaihua 3
input 12 is work/zhongyaocai/classification/total_images/baihe_used_for_classification_3.jpg, and the predict result is baihe 0
input 13 is work/zhongyaocai/classification/total_images/gouqi_used_for_classification_2.jpg, and the predict result is gouqi 2
input 14 is work/zhongyaocai/classification/total_images/gouqi_used_for_classification_1.jpg, and the predict result is gouqi 2
input 15 is work/zhongyaocai/classification/total_images/huaihua_used_for_classification_4.jpg, and the predict result is huaihua 3
input 16 is work/zhongyaocai/classification/total_images/dangshen_used_for_classification_2.jpg, and the predict result is huaihua 3
input 17 is work/zhongyaocai/classification/total_images/dangshen_used_for_classification_6.jpg, and the predict result is dangshen 1
input 18 is work/zhongyaocai/classification/total_images/jinyinhua_used_for_classification_3.jpg, and the predict result is jingyinhua 4
input 19 is work/zhongyaocai/classification/total_images/huaihua_used_for_classification_8.jpg, and the predict result is huaihua 3
input 20 is work/zhongyaocai/classification/total_images/baihe_used_for_classification_6.jpg, and the predict result is baihe 0
input 21 is work/zhongyaocai/classification/total_images/baihe_used_for_classification_7.jpg, and the predict result is baihe 0
input 22 is work/zhongyaocai/classification/total_images/baihe_used_for_classification_9.jpg, and the predict result is baihe 0
input 23 is work/zhongyaocai/classification/total_images/dangshen_used_for_classification_1.jpg, and the predict result is dangshen 1
input 24 is work/zhongyaocai/classification/total_images/huaihua_used_for_classification_6.jpg, and the predict result is huaihua 3
input 25 is work/zhongyaocai/classification/total_images/dangshen_used_for_classification_9.jpg, and the predict result is dangshen 1
input 26 is work/zhongyaocai/classification/total_images/huaihua_used_for_classification_1.jpg, and the predict result is huaihua 3
input 27 is work/zhongyaocai/classification/total_images/dangshen_used_for_classification_4.jpg, and the predict result is dangshen 1
input 28 is work/zhongyaocai/classification/total_images/huaihua_used_for_classification_9.jpg, and the predict result is huaihua 3
input 29 is work/zhongyaocai/classification/total_images/jinyinhua_used_for_classification_7.jpg, and the predict result is jingyinhua 4
```

图 5.2.7(a)

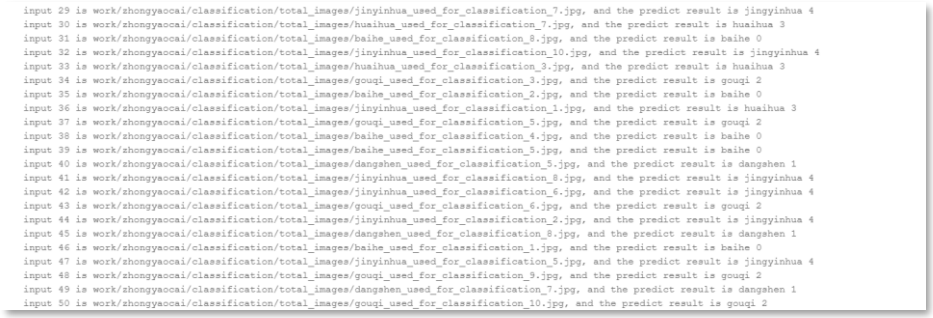

图 5.2.7(b)

任务四实现效果截图：

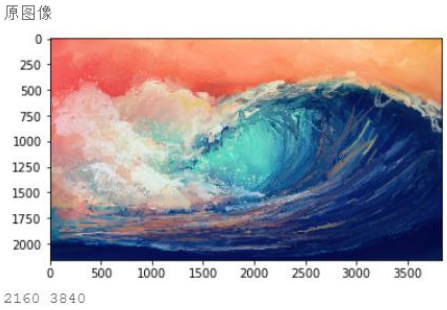

图 5.2.8(a)

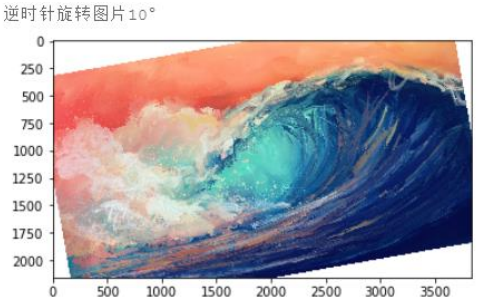

图 5.2.8(b)

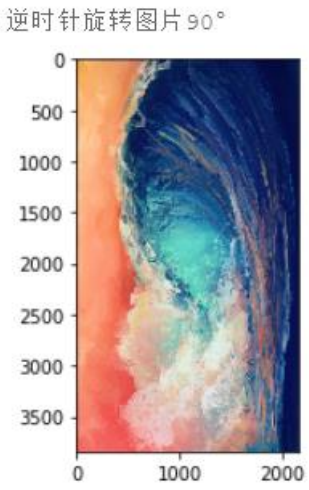

图 5.2.8(c)

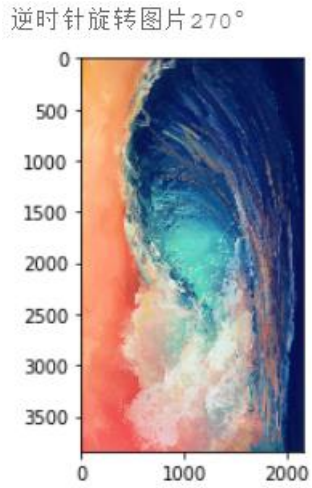

图 5.2.8(d)

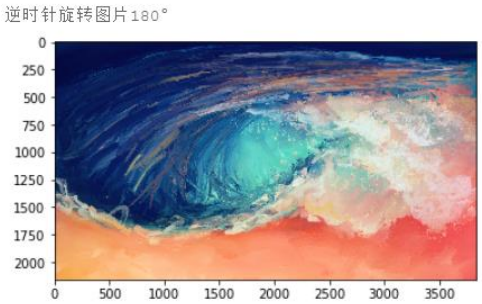

图 5.2.8(e)

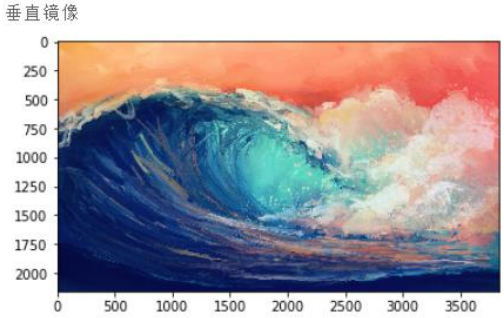

图 5.2.8(f)

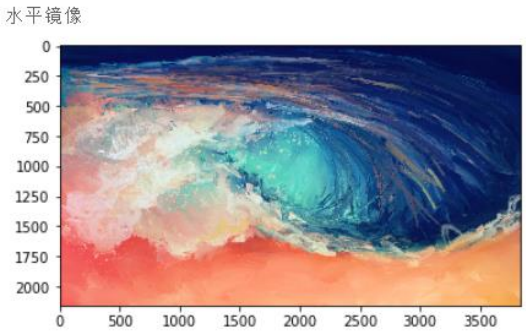

图 5.2.8(g)

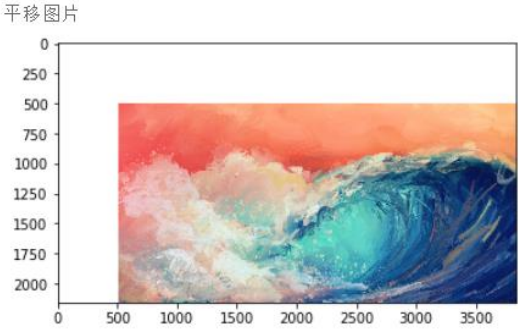

图 5.2.8(h)

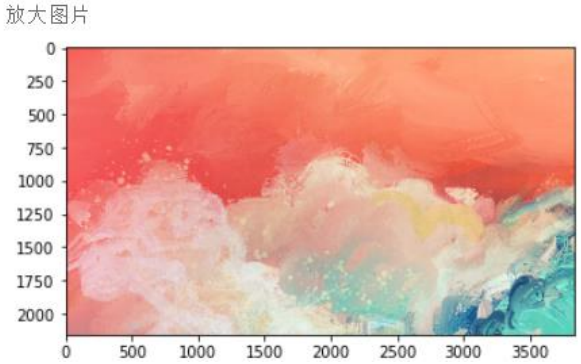

图 5.2.8(i)

第六章　总结

参加这门课程，对于刚开始学习 Python 的我来说，具有很大的挑战。在刚开始的时候面对老师讲的人工智能的相关知识感到十分迷茫，更不用说刚开始上机写项目时的手足无措。从零到一的突破往往是最难的，不会使用 jupyter notebook，不明白各个板块之间的联系，往下一翻，看到的代码令自己头大。第一个任务 TODO1 的代码就让我感到迷茫，不知道从哪里开始，要调用哪些函数，对此一无所知。没有办法，只好看着指导 PDF 来写，照着 PDF 一步一步地完成自己的代码。然而，当写完 TODO1 时，心里有一种说不出的喜悦，就像一道许久没有做出来的数学题突然被解出来，大脑突然顿悟的豁然开朗。然后后面就沉浸到这种状态中，一口气做到 TODO3。面对从未使用过的模型的调用，我开始阅读示例代码，理清每一步的步骤，又一次的感受到了这种喜悦。在这之后，在实现零的突破后，后面就很快速了。

对于这个课程中的几项作业中的代码，在做的过程中有或多或少的困惑的地方，遇到不懂，会上网查，自己动手解决问题，若是网上也没有，也会找老师请教。就是在这样的磕磕绊绊的过程中，我一步一步完成了曾经让我毫无头绪的代码。不仅学会了调用多种第三方库和标准库，还学会了如何在网上快速找到问题的解答

我很喜欢这门课程，不同于我在网上看的网课教学全部都讲，也不同于以前

学习的囿于定式的课程，这门课程将我现阶段理解不了的代码写了出来，而那些相对容易的代码空着让我自己补充，旁边又有 markdown 解释该步骤，是一种循循诱导的过程，一步一步地带我完成这块拼图。每当完成一个子程序并运行成功时，都是一种难以名状的喜悦。此外，课程中的非唯一答案也让我有更大发挥空间。我可以在第一部分中调用 seaborn 库画指定的条形图。不像 icoding 作业中的固定步骤和固定答案，这门课程中每个项目都让我的思维发散，不拘泥于形式，让我思考如何将我的输出美化（修改输出格式），让我思考怎样节省代码，甚至当我认为原来的代码不够好时（有些情况没有考虑到），我可以修改原来的代码，改成我想要实现的代码。这门不拘泥于形式的课程真正的感受到了做项目的感觉，没有固定的答案，没有最好的实现，只有更好的实现，一步一步将自己的实现做的更加完美。

最后，我十分感谢杨老师。谢谢杨老师不厌其烦地为我解答各种各样的问题，为我指出一些细碎的错误。当然，也少不了我的组员的帮助，他们为我指出了我编写程序中的错误，谢谢他们的包容。