东软教育科技集团

项目实训总结报告

项 目 名 称:“识食物者”---美食识别系统

学 校 名 称: 内蒙古大学

项 目 讲 师: 王志勇

姓 名: 武泳桦

学 号: 0191121332

实 训 日 期: 2022年7月11日-8月6日

**1 引言** （一级标题，小三黑体，居中）

1.1项目概述（二级标题，四号黑体，左对齐）

说明项目目的、背景、意义及参考资料（包括企业提供的素材、DEMO）等；

**1.1.1项目目的**

“识食物者”——美食识别程序作为一款美食识别系统，主要可以识别六类食物（汉堡、牛排、甜甜圈、披萨、冰激凌、蛋挞）。用户可以上传美食图片进行识别。该项目不仅可以作为独立的一款识别系统给用户使用，更可以作为一个饮食健康类系统的子系统。希望可以利用人工智能技术为人们的饮食健康贡献一份力量。

**1.1.2项目背景**

“识食物者”——美食识别系统是大三下学期暑期实训活动中我们组做的项目，是本次实训的目标和成果。随着生活和饮食习惯的改变，糖尿病已经成为继心脑血管疾病、恶性肿瘤之后影响人类健康的第三大因素。饮食习惯很多时候成为糖尿病的诱因。控制饮食健康显得就格外重要。我们的美食识别系统将成为一种先进的饮食记录手段，未来结合大数据技术能更好地进行数据分析、数据挖掘。

**1.1.3项目意义**

随着科学技术的不断进步和发展，人工智能的发展是时代的必然，也将对我国经济发展作出重要的贡献。2017年国务院发布了新一代人工智能发展规划。到2021年，这一年被认为是AI的普及之年。中国信息通信研究院发布报告显示，2020年全球人工智能产业规模1565亿美元，增长率是12%，我国的产业规模大概是3100亿元，同比增长了15%。作为大学生的我们，积极拥抱人工智能技术，在本次人工智能企业实训中学习人工智能技术，希望未来能学有所成，解决国家卡脖子的领域——新一代人工智能。本次项目提高了我们专业知识运用、系统分析与开发、项目管理与实践、沟通交流、洞察计算机相关领域快速适应新技术等方面能力，让我们更好地了解、学习了人工智能技术，锻炼了开发软件系统的能力，以及接触了真实的企业氛围。该项目的实际意义在于帮助用户跟踪饮食记录，控制饮食。

**1.1.4参考资料**

[1] Flask Web 开发实战 ：入门、进阶与原理解析／李辉著 ．—北京：机械工业出版社，2018.8 (Web 开发技术丛书）

[2] Flask Web开发 : 基于Python的Web应用开发实战 / (美) 格林布戈 (Grinberg,M.) 著 ; 安道译. -- 北京人民邮电出版社, 2015.1 （图灵程序设计丛书）

[3] Python深度学习/(美)弗朗索瓦·肖莱 著 张亮(hysic) 译 人民邮电出版社出版时间：2018-08（图灵程序设计丛书）

[4] 猫狗识别程序demo、cifar-10 识别程序demo、FLask基础demo

1.2 需求分析 *(课程目标支撑4 ; 毕业要求支撑10.2)*

为什么做这个题目，有什么实际价值

**1.2.1选题原因**

2021年11月14日是“世界糖尿病日”，今年的主题是“人人享有糖尿病健康管理（Access to Diabetes Care）”。过去的10年间（2011年~2021年），我国糖尿病患者人数由9000万增加至1亿4000万，增幅达56%，其中约7283万名患者尚未被确诊，比例高达51.7%。 面对如此惊人的糖尿病患者增幅，作为计算机专业学生，我们希望可以使用人工智能技术帮助人们预防糖尿病。

饮食习惯是导致社会上普遍存在几种与健康有关的疾病的主要原因之一。中国工程院院士、上海交通大学医学院附属瑞金医院副院长**宁光**在2018年中华医学会糖尿病学分会(CDS)第二十二次全国学术会议上，分享了自己带领团队利用大数据分析建立的“国人饮食行为图谱”，发现：煎炸食品、烧烤食品倾向与糖尿病患病率、高血压患病率及BMI（体质指数）正相关；甜食与糖尿病患病率正相关；辣口味食品倾向于糖尿病患病率负相关。因此跟踪食物的消费行为、有效地监控饮食，不仅可以帮助个人预防疾病，而且使患有这些疾病的人更好地管理自己的健康。

**1.2.2实际价值**

控制饮食的传统方法是每天手写一份饮食记录。这样的方法效率比较低，存在重复记录并且不利于进行数据分析。同时，随着社交网络的快速发展,人们通常会上传、分享和记录食物图片,因此食物图像分类的应用价值也越来越大,对营养搭配、食品推荐、餐饮、社交等方面都产生了积极的影响。此外近年来,智能穿戴设备发展迅速,用可穿戴照相机记录一天的饮食,自动识别照片中的食物成为可能。因此,自动识别食物图像并对其进行分类,提高其检测精度,减少模型大小,具有重要现实意义。

传统的图像识别算法在食物识别和分类已有很多研究,但是对于各种各样的食物和复杂的外界环境,传统的图像识别算法表现十分乏力,普遍准确率过低。2009年李飞飞、Jia Deng 等研究员在 CVPR 2009 上发表了一篇名为《ImageNet: A Large-Scale Hierarchical Image Database》的论文。2010年ImageNet大规模视觉识别挑战赛（ILSVRC）上新模型不断刷新准确率分类错误率。 2012 卷积神经网络在计算机视觉上取得极大的成功后，人们开始一直在探索拥有更强大机器视觉的可能性。ILSVRC举办了七年，错误率从 0.28 降到了 0.03；物体识别的平均准确率从 0.23 上升到了 0.66。因此我们使用CNN卷积网络模型对食物进行分类，希望对人们的饮食健康贡献一份力量。

1.3 运行环境

软、硬件环境

（正文，小四号宋体，1.25倍行距）

**1.3.1软件环境：**

|  |  |
| --- | --- |
| 软件/包名 | 版本 |
| Python | 3.8 |
| Keras | 2.9.0 |
| Tensorflow-gpu | 2.9.1 |
| Flask | 2.1.3 |
| Graphviz | 0.20.1 |
| Jinja2 | 3.1.2 |
| Jupyter Notebook | 6.4.12 |
| Matplotlib | 3.5.2 |
| Numpy | 1.23.1 |
| Opencv-python | 4.6.0.66 |
| Pandas | 1.4.3 |
| Pip | 22.2 |
| Requests | 2.28.1 |
| BeautifulSoup4 | 4.11.1 |
| Seaborn | 0.11.2 |
| Sklearn | 0.0 |
| Pycharm | 2022.1.4 |
| Flask-Bootstrap | 3.3.7.1 |
| Flask-SQLAlchemy | 2.5.1 |
| Flask-WTF | 1.0.1 |

**1.3.2硬件环境：**

机械革命极光Air 15.6英寸12代英特尔酷睿i7-12700H 16G 512G RTX3050 144HZ

**2 项目设计**（小三黑体，居中）

*(课程目标支撑3 ; 毕业要求支撑9.2)*

（以个人主要负责内容为主）

2.1 设计思路

我主要负责的是模型训练模块和系统调试整合。系统调试整合主要是和前后端队友交流沟通，将队友们负责的前端页面和后端功能进行匹配，然后对后端功能进行连接调试。下面主要讲解模型训练模块的设计思路。

模型训练部分：

设计之前，我阅读相关文献和博客，了解了国际上发表的优秀的深度学习模型是如何设计的。然后开始设计我自己的深度学习模型。

我从模型的输入、模型本身、模型的输出的角度出发，考虑将大的模块划分成三个子模块：数据模块、模型训练模块、模型预测模块，然后对这三个子模块再进行设计。

数据模块负责给模型训练模块提供训练数据。主要功能有将数据集加载到内存中，对数据集进行统计，输出数据集的信息，然后把图片数据转换成特定的格式。所以这里将数据模块设计成三个小模块组成：数据加载、数据处理、数据统计。

模型训练模块负责训练出能识别六类食物的深度学习模型。设计思路有以下六点。第一，我们在搭建网络结构时会使用到预训练模型，然后在基础上进一步设计网络结构。第二，使用迁移学习的方法训练模型。迁移学习分为特征提取方法和微调模型方法。将两种方法抽象成两个模块。第三，因为数据集较小，所以需要使用数据增强的方法。第四，模型训练过程中，会随着迁移学习的方法不同而调整模型的编译设置。所以可以将模型编译设置抽象出来成为一个模块。第五，训练好的模型要保存下来，根据指定路径保存供模型预测使用。第六，训练过程准确率和损失可视化有助于模型评估和训练。

模型预测模块要直接加载已经训练好的模型，加载要预测的数据并处理成模型要求的格式，然后模型进行推理返回结果。所以该模块可划分成三个子模块：模型加载、数据集加载、模型推理。

2.2 模块功能介绍

**2.2.1 数据模块**

（1）数据加载：将数据集加载到内存

（2）数据统计：返回数据集信息

（3）数据处理：将输入的图片数据转换成模型要求的格式，包括图片修改尺寸格式、数据归一化、标准化、将标签转换成独热编码。

**2.2.2 模型训练模块**

（1）网络搭建：搭建训练模型的网络结构

（2）自定义网络层模块：将预训练模型和自己添加的网络层（比如Dense层）相连接，也可以只定义自己的模型。

（3）模型加载：加载训练好的模型，可加载模型权重或模型结构或模型全部信息（包括配置信息）全部加载。

（3）特征提取：冻结预训练模型全部参数，提取出从预训练模型输出的特征，用于训练顶层分类层。

（4）微调：解冻预训练模型的一部分层，用于后续进行端到端的训练。

（5）数据增强模块：对已有数据进行图像变换，生成新数据扩充训练集。可以进行变换的方式有：随机旋转(rotate)、图像随机偏移(shift)、图像随机推移错切(shear)、图像随机翻转(flip)、Sample-wise图像像素标准化、Feature-wise图像像素标准化、ZCA白化转换、图像张量维度的重排序、存储增强图像数据。

（6）模型编译模块：配置模型编译信息，比如学习率、优化器等

（7）模型保存：保存模型到指定路径

（8）训练过程可视化：将训练过程准确率和损失的变化可视化有助于模型评估和训练。

**2.2.3.模型预测模块**

（1）模型加载模块：加载已训练好的模型

（2）数据集加载模块：加载预测的数据

（3）模型推理：模型对数据集进行预测，给出识别结果

2.3 模块结构图

主程序调用数据模块、数据训练模块、数据预测模块完成总体模块的功能。

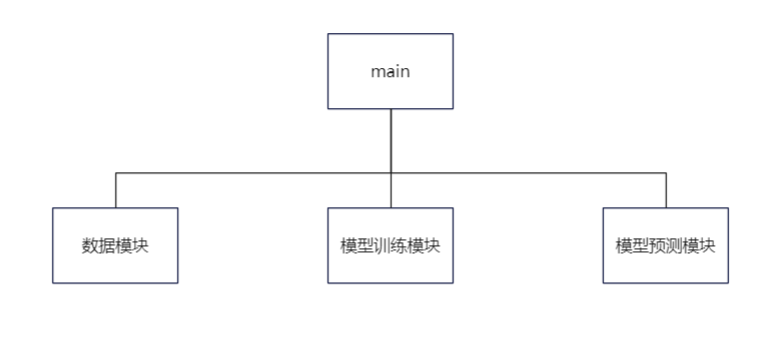


图2.3-1.2总体结构

数据模块包括数据加载、数据统计、数据处理模块。

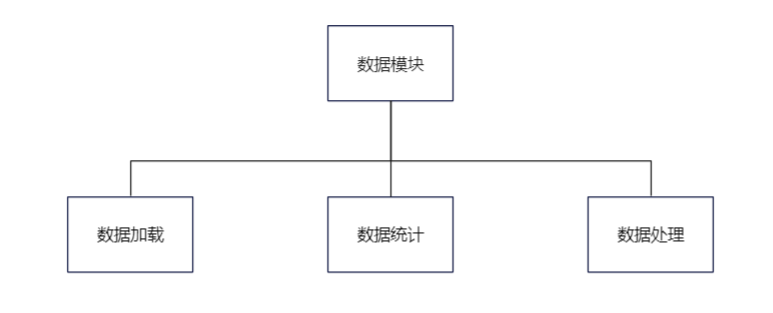


图2.3-2 数据模块结构

模型训练模块包括网络搭建模块、特征提取模块、微调模块、数据增强模块、模型编译模块、模型保存模块、训练可视化模块。

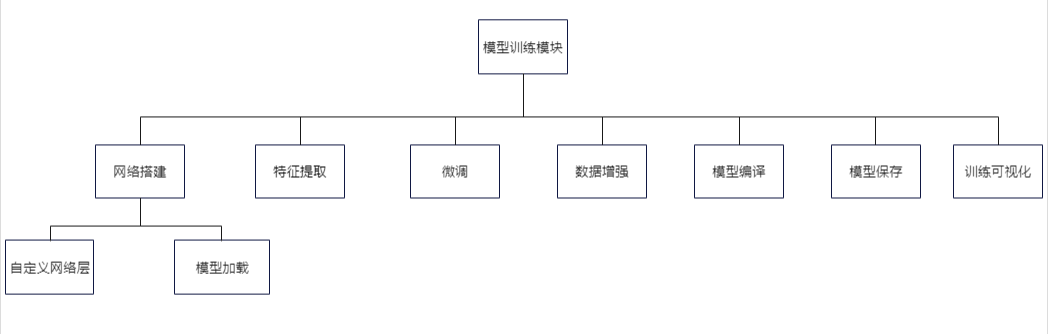


图2.3-3 数据模块结构

模型预测模块包括模型加载模块、数据模块、模型推理模块。

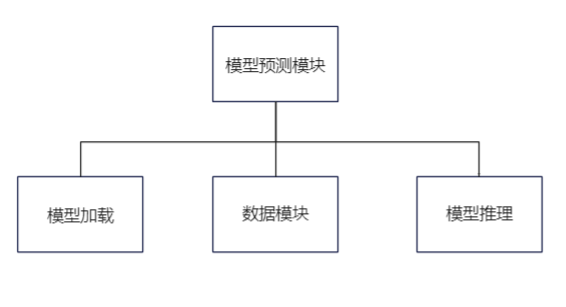


图2.3-4 模型预测模块结构

2.4 程序流程图

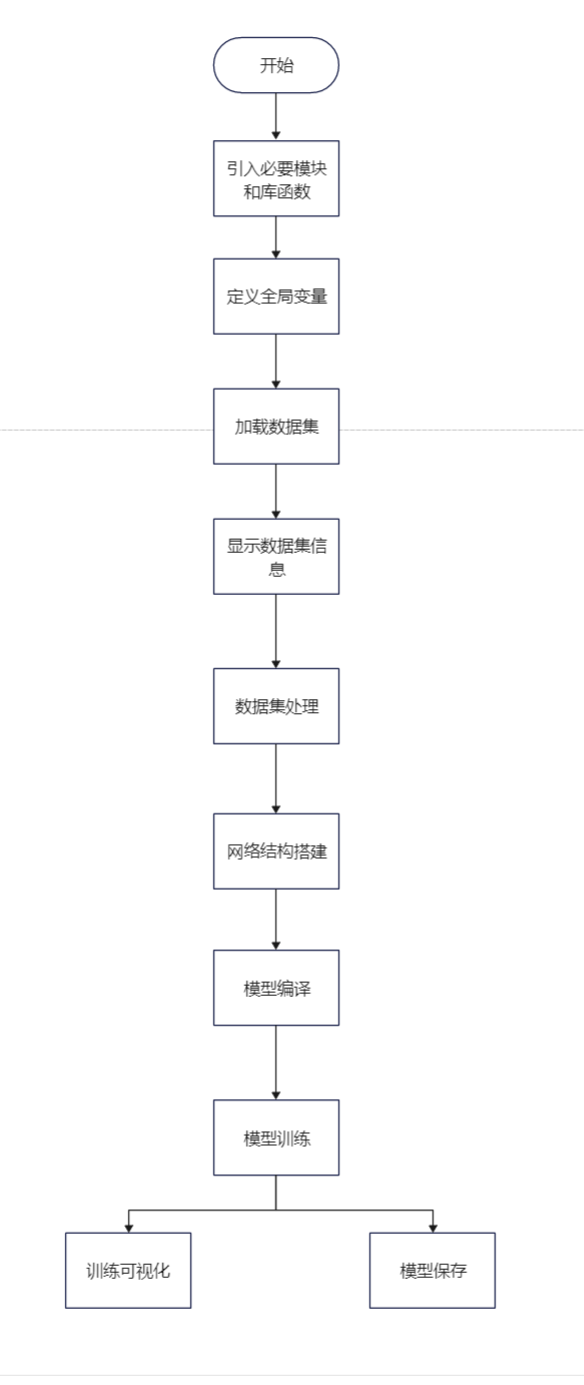
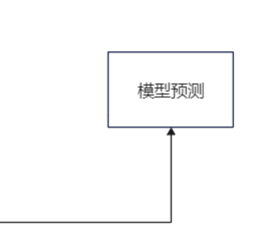


图2.4-1.1总体流程图

引入必要模块和库函数后，定义全局变量如全连接层数、冻结层数等。加载数据集后显示数据集信息：训练集数量、验证集数量、测试集数量。将数据集进行处理：数据增强、数据归一化、标准化、缩放成指定尺寸等。接着构建网络模型结构，进行模型编译，开始训练。训练完成后，展示训练过程可视化的图表，保存训练过程中表现最好的模型。最后可以用保存的模型进行模型预测。

2.5 功能设计分工

需详细标注每部分的内容，具体任务及任务安排等。

|  |  |  |
| --- | --- | --- |
| **序号** | **任务名称** | **工时** |
| 1 | 尝试预训练模型Resnet | 0.5天 |
| 2 | 尝试预训练模型Inception V3 | 0.5天 |
| 3 | 实现数据加载预处理、预训练模型加载、模型网络搭建模块 | 0.25天 |
| 4 | 实现数据增强模、预训练提取特征、微调、模型保存模块 | 0.25天 |
| 5 | 实现模型预测模块、模型结果可视化模块 | 0.15天 |
| 6 | 整理合并大部分代码 | 0.2天 |
| 7 | 整理合并剩余代码 | 0.15天 |
| 8 | InceptionV3调整参数达到更好水平 | 0.5天 |

**3 详细设计**（小三黑体，居中）

*(课程目标支撑3 ; 毕业要求支撑9.2)*

详细写明项目的设计（系统设计、数据库设计、页面等）等，并写明对所涉技术、核心功能等描述；

模型训练部分分成了三个子模块：数据模块、模型训练模块、模型预测模块。首先讲述模块的详细设计，使用哪些技术。然后对模型训练的重中之重——迁移学习进行设计。最后设计出总体程序流程图中的详细流程。

**3.1模块的详细设计**

**3.1.1数据模块：**

（1）数据加载模块

模型训练时的数据集加载使用keras框架提供的ImageDataGenerator类中的flow\_from\_directory方法。模型预测时数据集加载使用keras框架提供的preprocessing库中的Image类中load\_img方法、img\_to\_array方法。

（2）数据处理模块

模型训练和模型预测时的数据处理使用keras框架内置的预训练模型提供的预处理函数进行处理。

**3.1.2模型训练模块**

模型训练模块使用深度学习框架keras进行模型训练。keras提供一致且简单的 API，它将常见用例所需的用户操作数量降至最低，并且在用户错误时提供清晰和可操作的反馈，被工业界和学术界广泛采用。并且它支持多个后端引擎，不会将你锁定到一个生态系统中，可以轻松将模型转化为产品，拥有强大的多 GPU 和分布式训练支持。

（1）网络搭建

使用keras中的applications包提供的内置预训练模型进行预训练模型加载。keras提供的models、layers类进行自定义网络架构的搭建。

（2）特征提取

将训练数据输入预训练模型后得到特征，然后将特征输入分类器中训练分类器。

（3）微调

假设预训练模型共有n层，指定冻结前k层，解冻n-k层后，将该预训练模型和提前训练好的顶层分类器进行连接，开始端到端的训练。

（4）数据增强

使用keras提供的ImageDataGenerator类，设置相应的数据增强参数，然后使用flow\_from\_directory方法生成数据增强后的数据。

（5）模型编译

提供对特征提取、微调和其他三种条件下的编译选项进行分别配置，有利于降低耦合，方便调参。优化器可使用adam、SGD、rmsprop。因为要对六类食物进行识别，所以损失函数使用多分类交叉熵损失函数。

（6）模型保存

保存整个模型：model.save(filepath)将Keras模型和权重保存在一个HDF5文件中，该文件将包含模型的结构、模型的权重训练配置（损失函数，优化器，准确率等）、优化器的状态（以便于从上次训练中断的地方）

保存模型的结构：model.to\_jason()将模型序列化保存为json文件，里面记录了网络的整体结构, 各个层的参数设置等信息。 将json字符串保存到文件。

保存模型权重：model.save\_weights()我们经过调参后网络的输出精度比较满意后,可以将训练好的网络权重参数保存下来。

（7）训练可视化

使用matplotlib库中的pyplot类方法进行可视化。

**3.1.3模型预测模块**

（1）数据加载模块

使用keras中的preprocessing库中的image类中的load\_img方法和img\_to\_array方法。

（2）模型加载模块

使用keras中models类中的load\_model方法。

（3）模型推理模块

使用keras中models类中的predict方法。

**3.2迁移学习设计**

**3.2.1设计原则**

根据图片大小来选择网络和超参数，根据图片类别数和数量来选择batch\_size，根据网络规模和loss下降速率来选择学习率。

**3.2.2预训练模型的选择**

在图像领域，有一个比赛是每位从业者心中的丰碑——ILSVRC。ILSVRC（ImageNet Large Scale Visual Recognition Challenge）是近年来机器视觉领域最受追捧也是最具权威的学术竞赛之一，代表了图像领域的最高水平。

ILSVRC从2010年举办到2017年。在12-15年期间各路公司和实验室在ImageNet比赛上提出了一些经典网络，比如AlexNet，ZFNet，OverFeat，VGG，Inception，ResNet、WideResNet，FractalNet，DenseNet，ResNeXt，DPN，SENet等等。keras将其中很多模型都内嵌到了自身中，非常方便调用。

根据这些预训练模型的准确的、模型大小等因素，我选择ResNet50和InceptionV3进行尝试。这些模型的准确度较高，并且模型参数较少，训练速度快、泛化能力强。

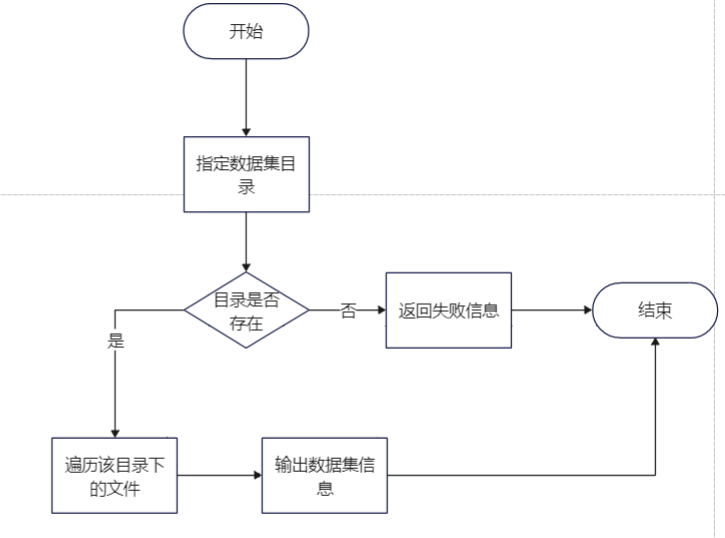
**3.2.3顶层分类器设计**

首先设置一个让模型具有较强拟合能力的分类器，比如有512神经元的Dense层，如果拟合能力交差考虑适当增加层数或神经元数量。然后尝试降低模型的过拟合，使用添加dropout层、使用权重衰减、减小网络容量等方法。接着调整学习率、优化器、epoch等超参数，使用数据增强的方法。最后不断迭代调参，训练出最好的模型。

**3.2.4微调设计**

连接预训练模型和训练好的顶层分类器，解冻预先训练好的网络高层中的某些层，使用较少学习率进行学习。对于ResNet50解冻后五层进行训练，InceptionV3解冻mixed7之后的层。

**3.3程序流程图详细设计**

、图3.3-1显示数据信息流程图

显示数据集信息首先要制定数据集所在根目录，然后制定训练集、测试集、验证集目录。判定该目录是否存在，如果存在，就遍历该目录下文件统计信息，然后输出数据集信息。否则，返回失败信息。

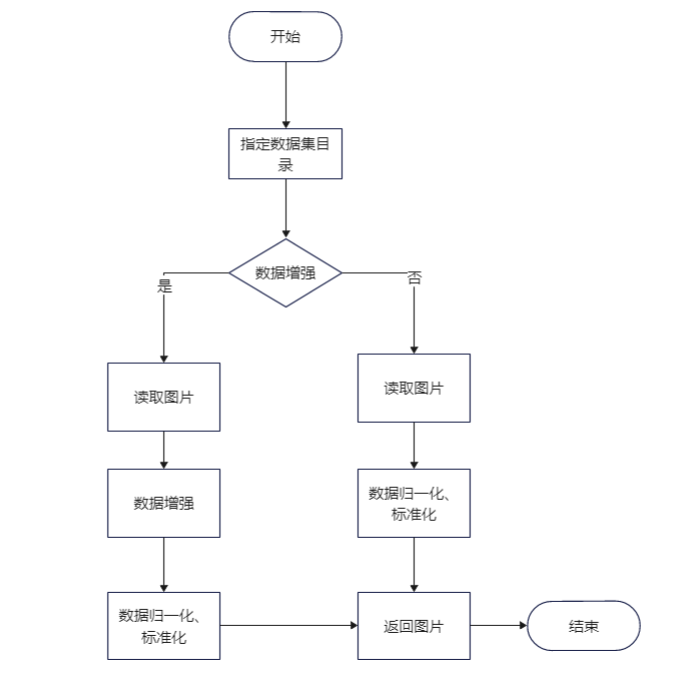


图3.3-2数据集处理流程图

数据集处理首先要制定数据集目录，然后判断是否需要数据增强，如果需要，就读取数据集图片进行数据增强后，进行数据归一化和标准化，最后返回增强后的图片。反之读取图片后只进行归一化和标准化，最后返回图片。

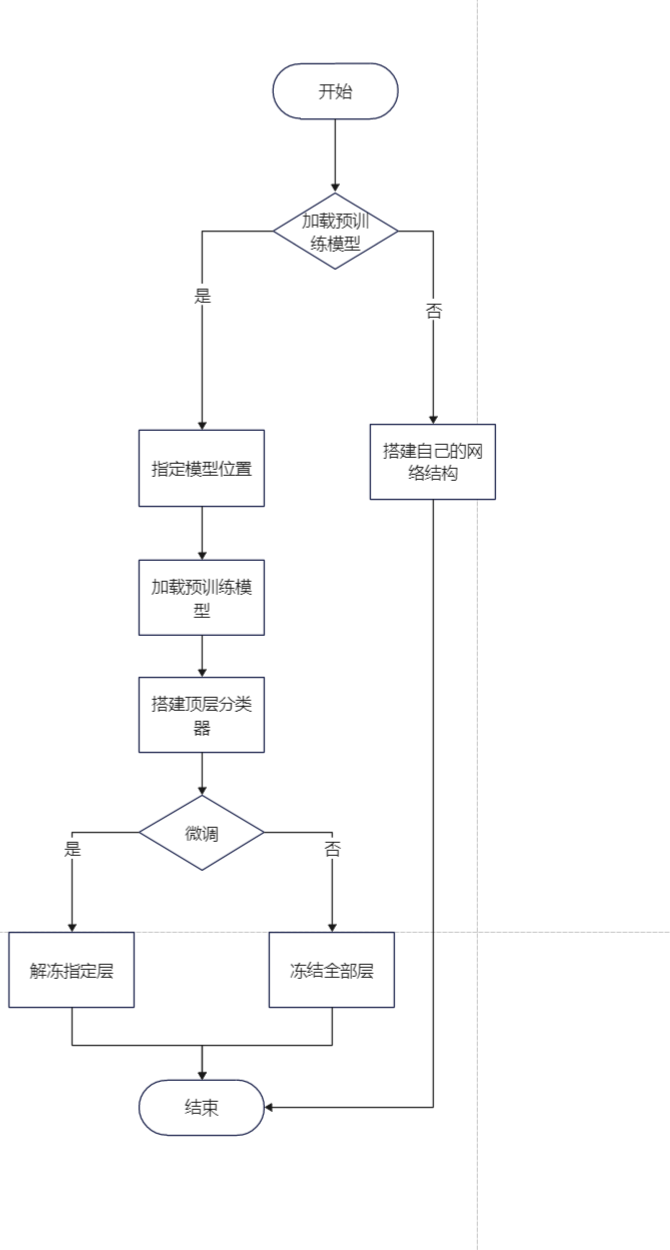


图3.3-3网络结构搭建流程图

网络结构搭建首先判断是否加载预训练模型，如果加载则指定模型位置然后进行加载，接着搭建顶层分类器后如果需要微调，则解冻指定层，否则冻结全部层。如果不加载预训练模型，则搭建自己的网络结构。

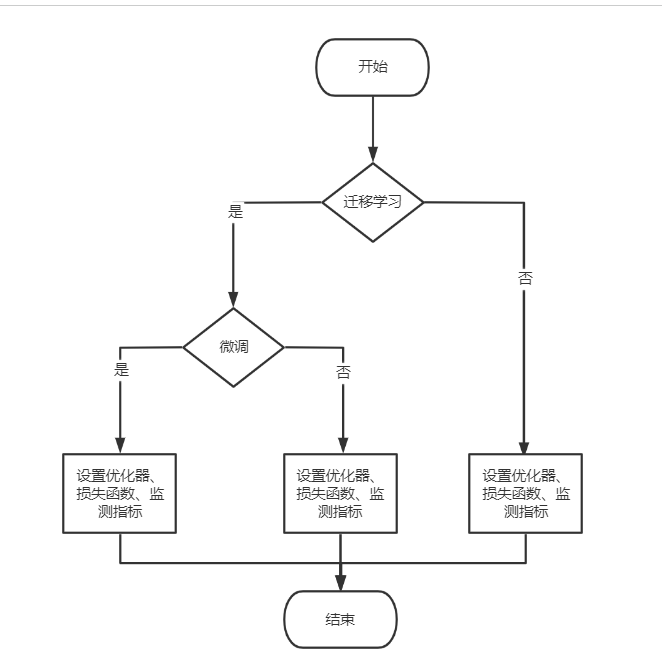


图3.3-4模型编译流程图

模型编译首先判断是否使用迁移学习。这里有三个完全一样的步骤：设置优化器、损失函数、监测指标。这指的是不同情况，会进行不同的编译设置。

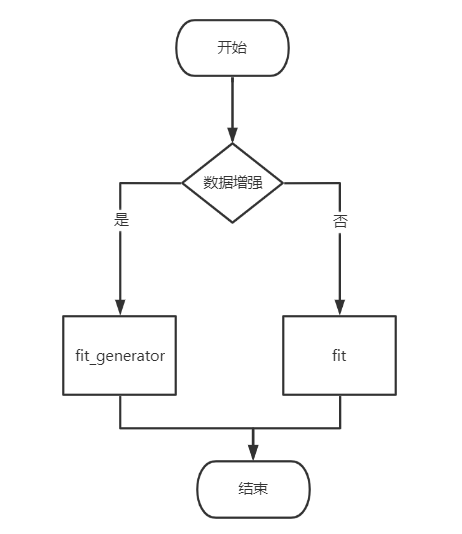


图3.3-5模型训练流程图

模型训练首先判断是否使用了数据增强，如果是，就是用fit\_generator方法，否则使用fit方法即可进行训练。

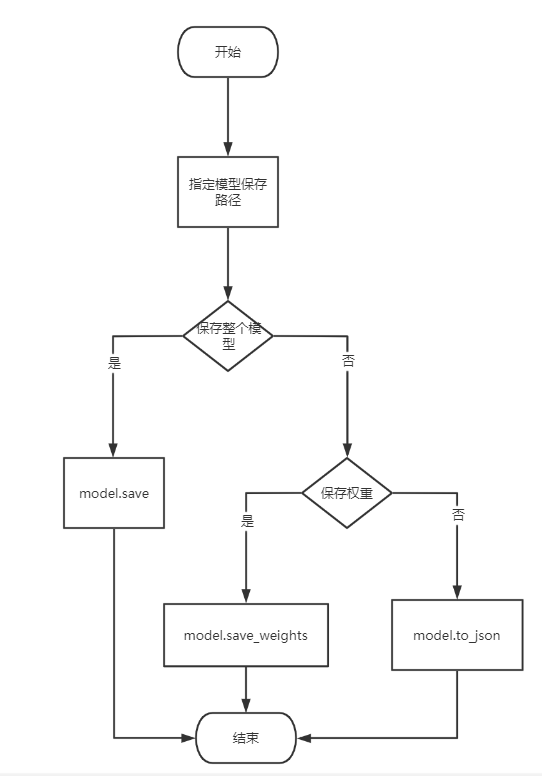


图3.3-6模型保存流程图

模型保存首先要制定模型保存路径，然后判断是否保存整个模型，如果是，调用model.save方法；如果不需要保存整个模型，而想只保存权重，则使用model.save\_weights方法，如果只想保存模型网络结构，则调用model.to\_json方法。

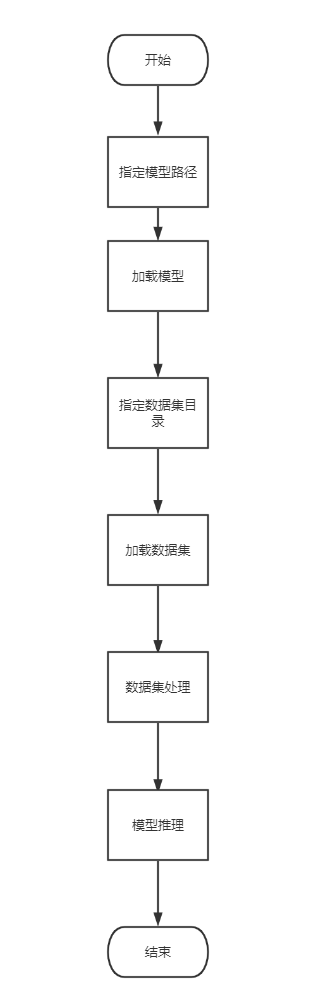


图3.3-7模型预测流程图

模型预测首先要指定模型路径加载模型，然后指定预测数据集路径加载数据集，之后进行数据集处理，将处理好的数据送入模型，模型进行推理然后给出结果。

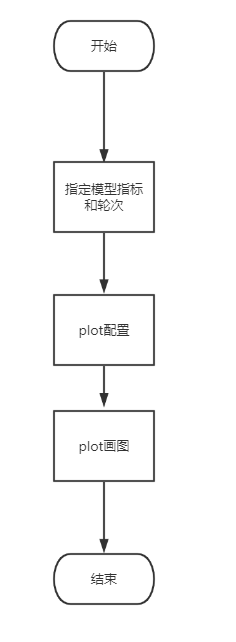


图3.3-8训练可视化流程图

训练可视化首先要制定模型指标和epoch轮次，然后对plot的一些属性进行配置，之后使用plot绘图。

**4 实现方案**（小三黑体，居中）

*(课程目标支撑3 ; 毕业要求支撑9.2)*

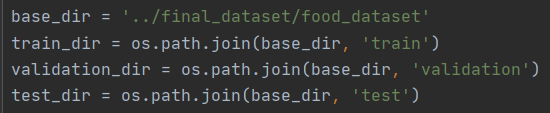
详细写明项目功能实现描述、核心代码及实现结果等；

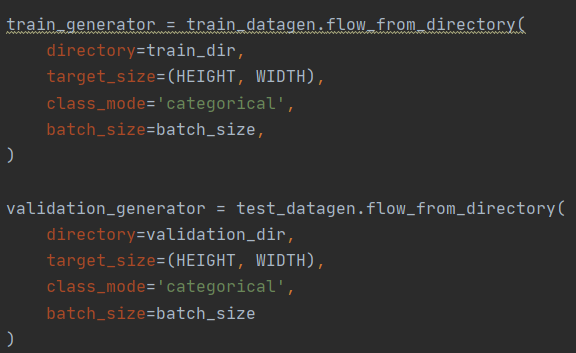
并对调试好的程序，从执行程序弹出界面开始，每一步操作截一个图，并附加说明（图片格式为JPG格式）。

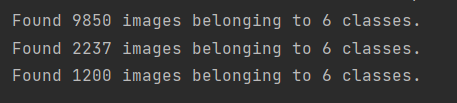
**4.1数据模块**

**4.1.1数据加载**

指定数据集路径，使用flow\_from\_directory方法加载数据集，得到训练数据生成器和验证数据生成器。

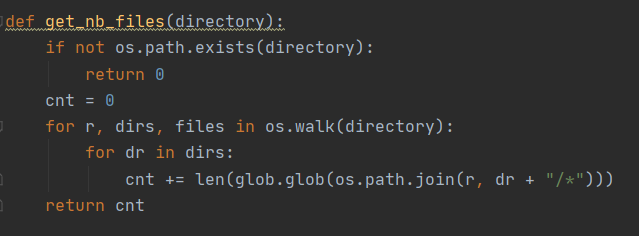


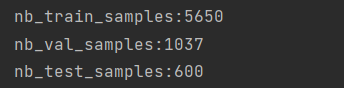




**4.1.2数据统计**

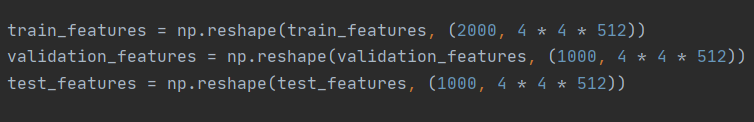
通过递归搜索数据集文件夹进行文件数量统计。



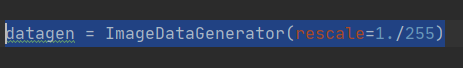


**4.1.3数据处理**

**（**1）将特征抽取后得到的特征形状变为二维形状，才能输入到分类器中。

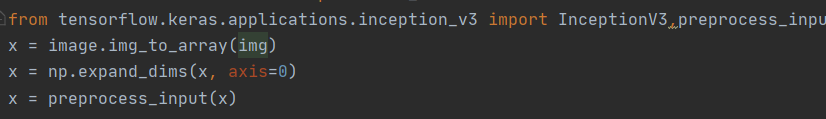


（2）使用ImageDataGenerator类进行归一化

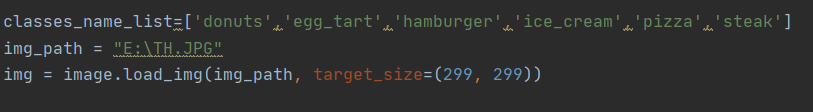


（3）预测单张图片预处理

调用预训练模型提供的预处理函数

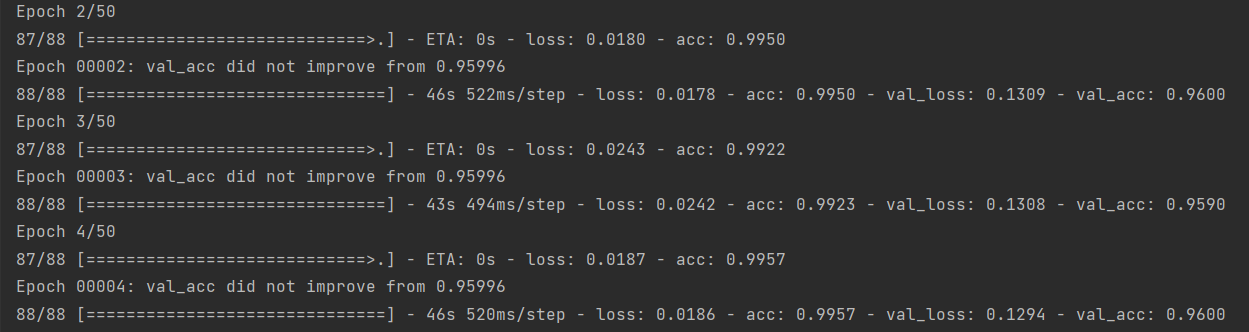


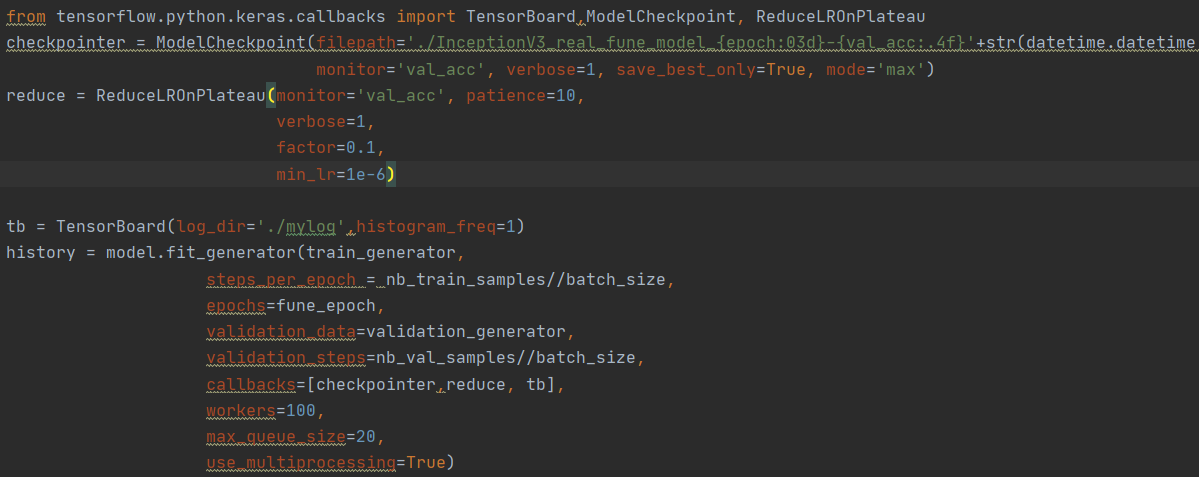
统一输入图片尺寸



**4.2模型训练模块**

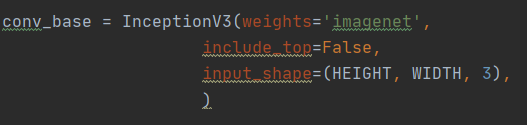
指定回调函数ModelCheckpoint用于保存训练过程中的模型，ReduceLROnPlateau用于学习率自适应改变，TensorBoard记录训练过程可视化。fit\_generator方法开始训练。设定100个线程，最大缓冲队列为20。steps\_per\_epoch和validation\_steps作为生成器停止的判断条件，epochs训练轮次。



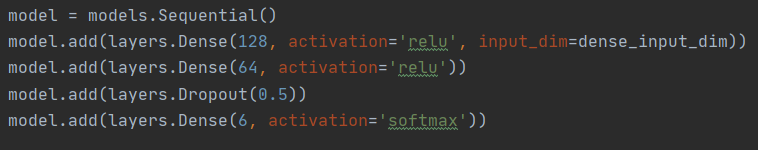


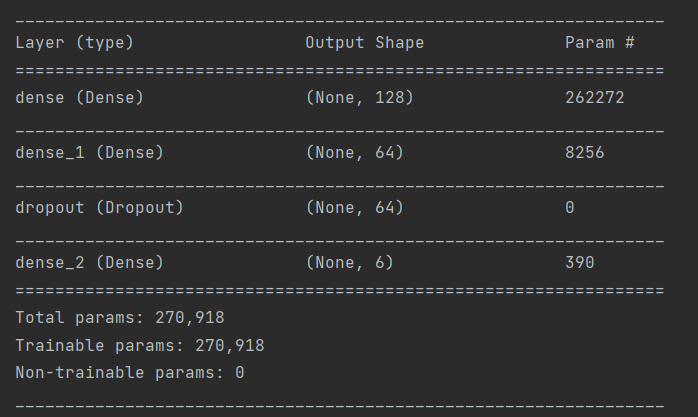
**4.2.1网络搭建**

预处理模型加载

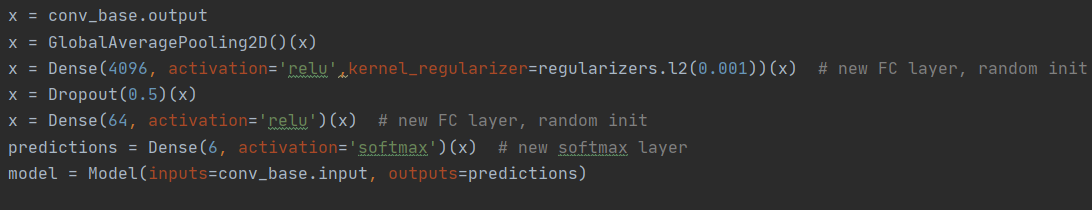


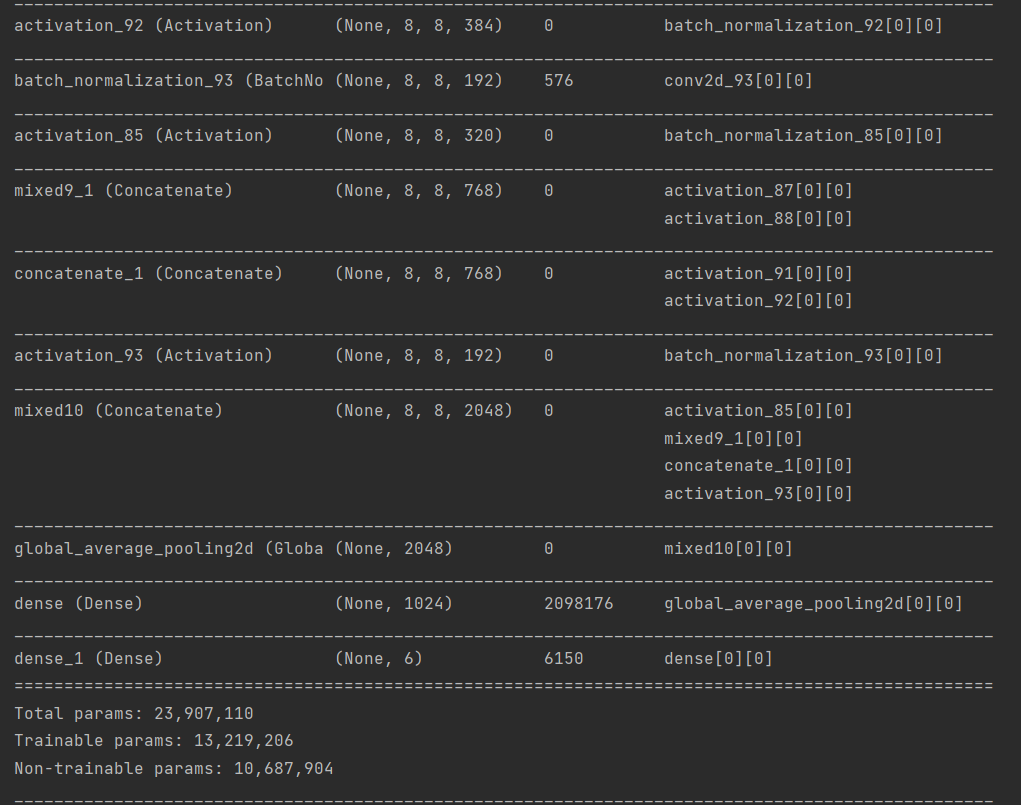
顶层分类器设计





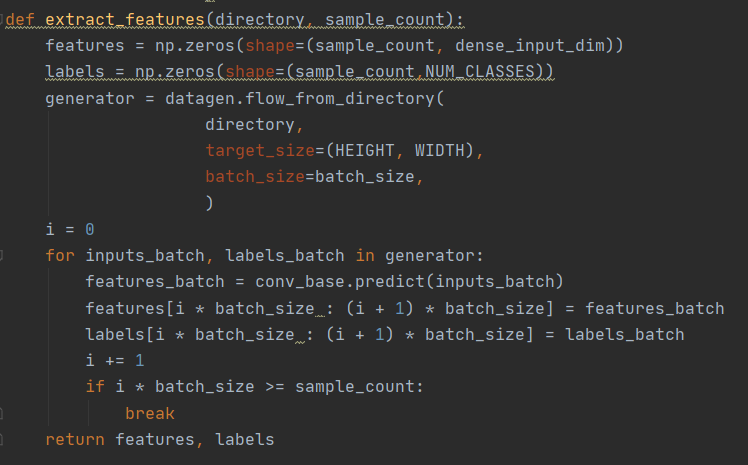
预训练模型和顶层分类器连接





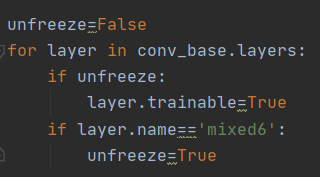
**4.2.2特征提取**

初始化features、labels张量，定义图片数据生成器，启动生成器后将数据输入到预训练模型中得到特征，然后保存到features张量中，对应的标签也进行保存。最后将特征和标签返回。



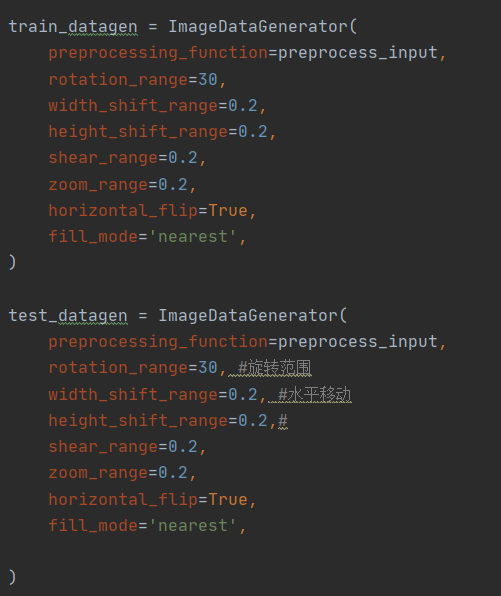
**4.2.3微调**

解冻mixed6后面的层。



**4.2.4数据增强**

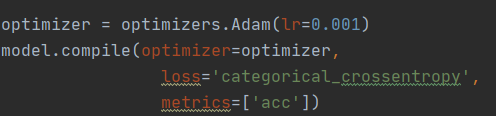
使用ImageDataGenerator类进行数据增强，对原始数据进行旋转30度变换，水平移动20%，垂直移动20%，错切变换因子0.2,缩放变换20%，使用翻转变换，填充模式为最近填充。



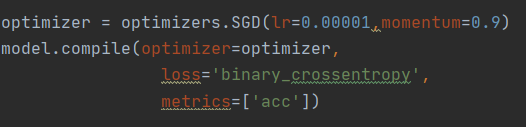


**4.2.5模型编译**

（1）特征提取



（2）微调



**4.2.6模型保存**

（1）保存整个模型，并且加入时间作为文件名的一部分以区分模型。

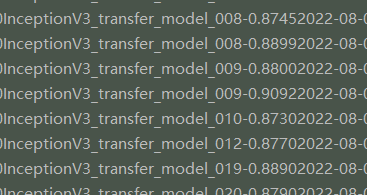


（2）保存权重



（3）保存模型结构

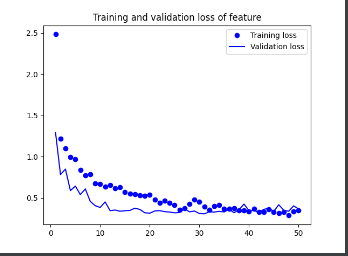
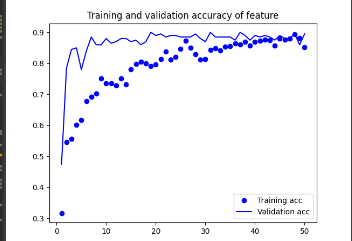




**4.2.7训练可视化**

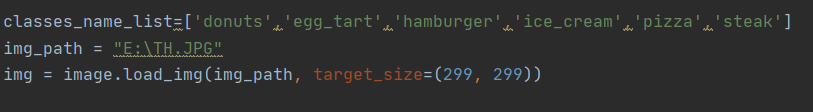
使用matplotlib中的pyplot类，从模型训练返回的history对象中取得训练集准确率、验证集准确率、训练集损失、验证集损失。指定标题和图片说明，展示图片。





**4.3模型预测模块**

**4.3.1数据加载模块**

使用keras提供的image类中load\_img方法。

**4.3.2模型加载模块**

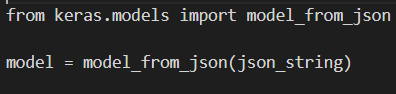
（1）加载整个模型



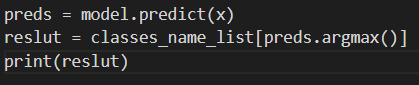
（2）加载权重

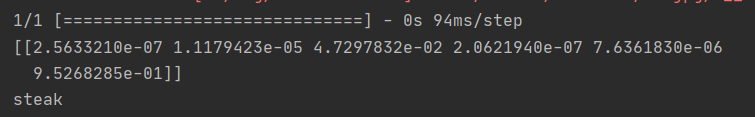


（3）加载模型结构



**4.3.3模型预测模块**





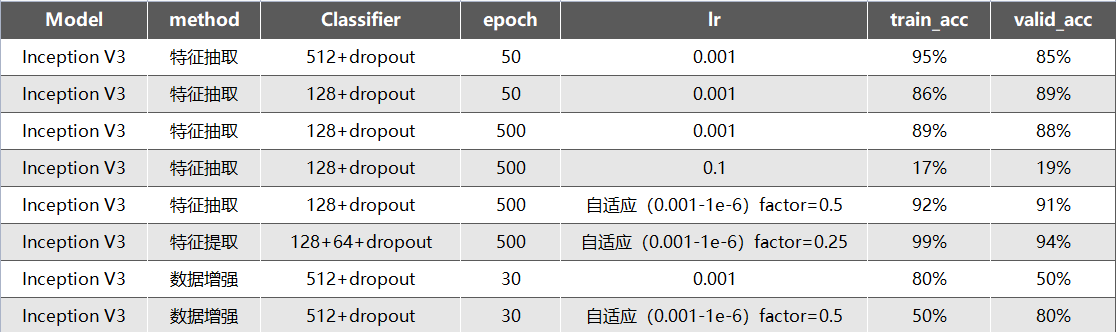
**5 系统测试**（小三黑体，居中）

*(课程目标支撑3 ; 毕业要求支撑9.2)*

详细写明测试方法、测试描述（bug与修正方法等）、预期结果等；

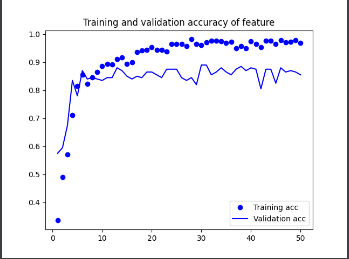
对于模型训练部分而言，系统测试过程是对模型进行调参的过程。下面讲述调参历程。主要选择InceptionV3模型进行实验。

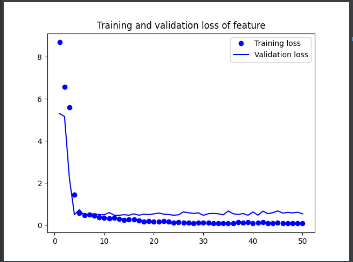
最终Inception模型最好效果：训练集准确率99%、验证集准确率96.3%、测试集准确率93%。



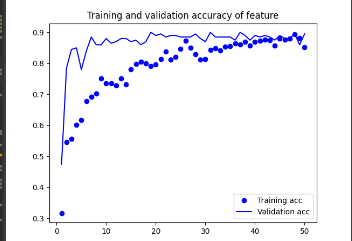
**5.1特征提取**

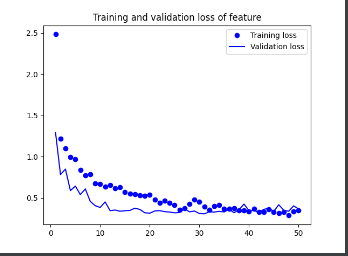
（1）我们先尽可能让模型拟合训练数据，所以这里分类器用的是512个神经元。存在过拟合现象。



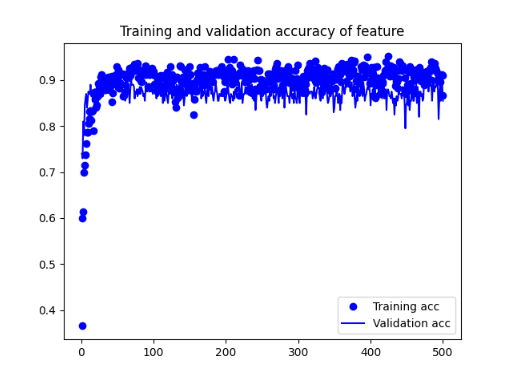


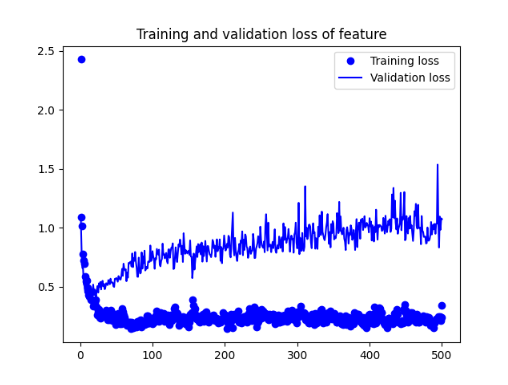
（2）然后尝试减小网络容量的方法，



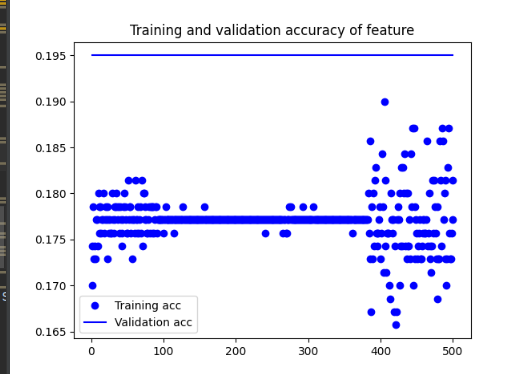


（3）看图像还有上升趋势，于是尝试提高训练时间，结果发现并没有潜力。

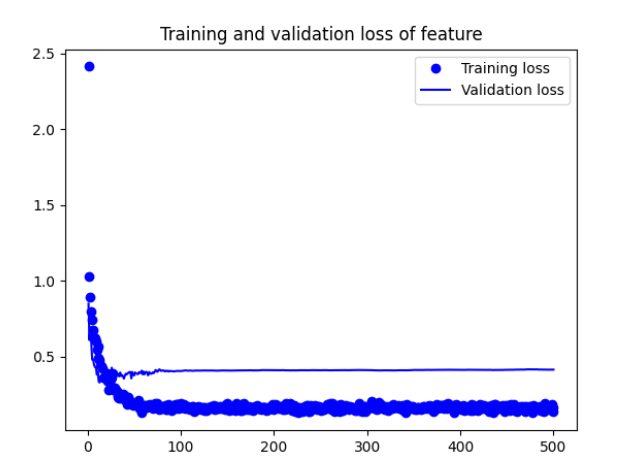


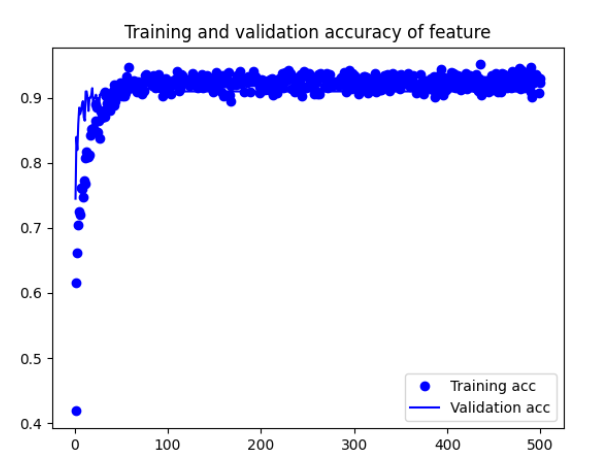


（4）然后更换方向，尝试修改学习率。0．1的学习率实在不好，什么东西都没学上。



（5）准确率自己调很麻烦，也不知道该什么时候去调整。就使用了回调函数ReduceLROnPlateau在训练过程中自己调整。结果还不错。





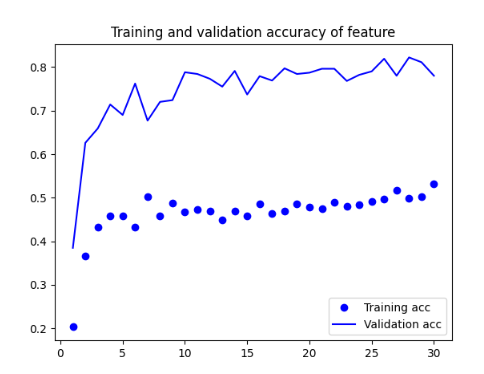
**5.2带数据增强的特征提取**

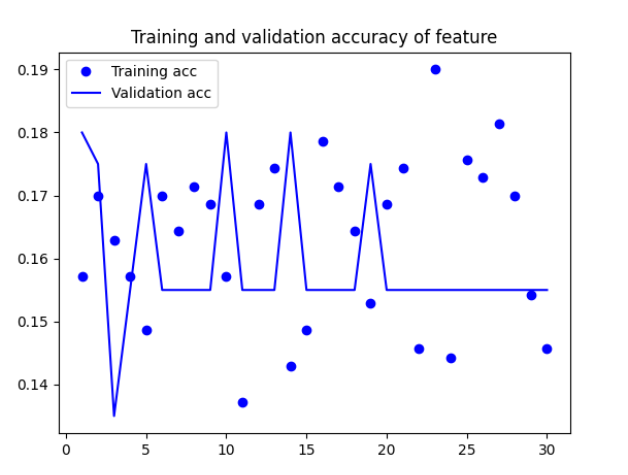
（1）上图发现训练集准确度还不是很高，拟合情况不是很好，尝试增大网络容量提高拟合能力，同时修改学习率。模型验证集准确率到达94%，接下来想提高模型准确率的方向就是解决过拟合问题。

而这里已经基本尝试了特征抽取的方法，似乎要了瓶颈。

考虑到我们数据集比较小，于是使用数据增强技术，进一步调高模型准确率。

但使用带有数据增强的特征提取方法后，由于训练速度减缓，只训练了30轮，但训练速度较慢，于是放弃这种方法。





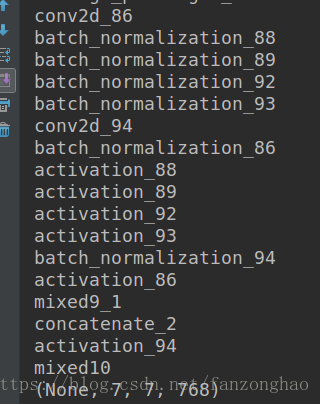
在尝试最终方法微调之前，我研究了为什么添加数据增强之后训练速度大幅降低。原因在于数据增强技术涉及到图片的加载和保存，也就是说涉及到IO。IO又是CPU负责，于是没加数据增强之前，是一次性将批量数据读入内存后，GPU就开始全力运算。而加了数据增强之后，就CPU需要多次从磁盘中读取一部分数据进行数据增强，IO次数增加了。并且由于运算速度的差异，GPU运算完一批数据后，会等待CPU运来下一批数据。这样造成了时间开销。解决方法就是使用多线程技术，让CPU和GPU尽量实现并行。

做了关于多线程数量的实验，发现100个线程是最好的。再多了程序会崩溃。

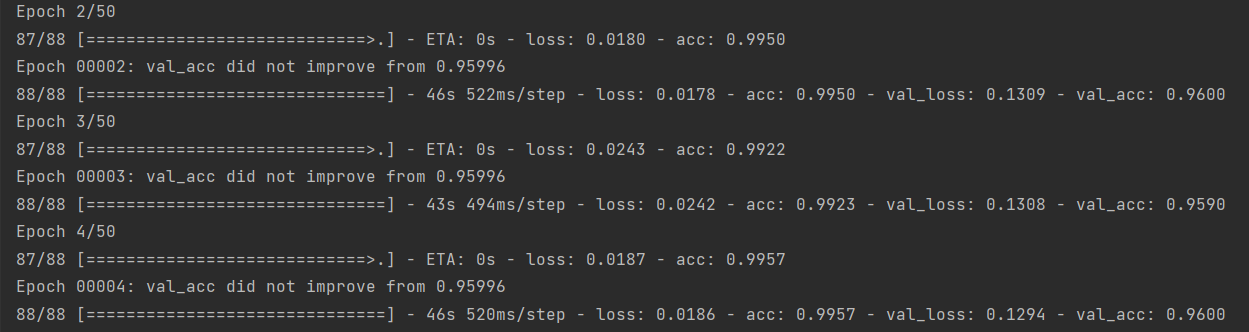
表格

描述已自动生成

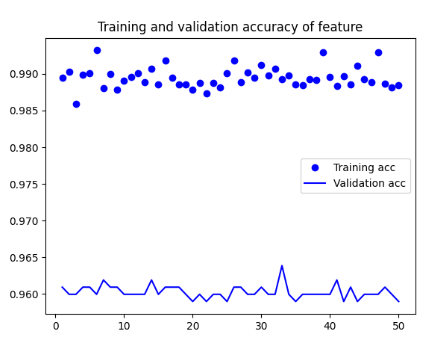
**5.3微调**

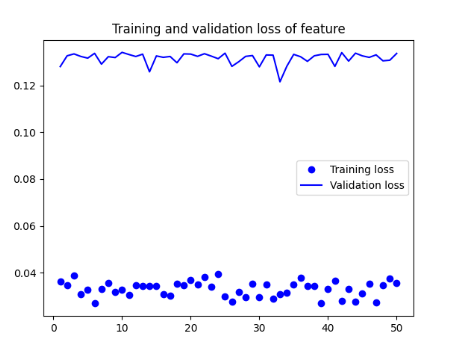
训练速度提高以后我们开始进行微调。解冻mixed7以后的层。

然后我们将特征抽取方法得到的94%准确度的模型加载进来，然后进行微调。达到了96%以上的准确度。



下面是plot画出的图像。因为是使用的加载模型的方式，在训练过程中，会使用直接终止程序运行方法来停止训练，导致不是持续性的图像。下图是加载最好模型后的准确度曲线。





最终InceptionV3模型最好的表现为：

* 训练集准确率：99%
* 验证集准确率：96.3%
* 测试集准确率：93%

**6 前沿分析**（小三黑体，居中）

*(课程目标支撑6 ; 毕业要求支撑12.2)*

2009年李飞飞、Jia Deng 等研究员在 CVPR 2009 上发表了一篇名为《ImageNet: A Large-Scale Hierarchical Image Database》的论文。2010年ImageNet大规模视觉识别挑战赛（ILSVRC）上新模型不断刷新准确率分类错误率。ILSVRC举办了七年，是近年来机器视觉领域最受追捧也是最具权威的学术竞赛之一，代表了图像领域的最高水平。在12-15年期间各路公司和实验室在ImageNet比赛上提出了一些经典网络，比如AlexNet，ZFNet，OverFeat，VGG，Inception，ResNet、WideResNet，FractalNet，DenseNet，ResNeXt，DPN，SENet等等。错误率从 0.28 降到了 0.03；物体识别的平均准确率从 0.23 上升到了 0.66。

2020年Facebook AI 研究团队提出了FixEfficientNet 连续成为计算机视觉领域的一项任务。它是目前最先进的，在 ImageNet 数据集上有最好的结果，参数为 480M，top-1 准确率为 88.5%，top-5 准确率为 98.7%。

详细写实训过程的心得

通过这次项目实训，我不仅在技术方面有所提高，而且更重要的是有机会了解和体验企业里的氛围，真实地感受到了一个完整项目是如何一步步实现的。第一天上课老师就强调：虽然是老师授课，但不再被叫作“上课”，而是叫作“上班”，一切按照企业里的要求来。这让我对于时间观念和自律性有了更强的提高。

上学期我学习了软件过程与管理、软件架构与设计模式课程。学这两门课的时候感觉全是理论知识，真不知道作者是怎么能想出这么多理论，一点也不符合计算机实践性学科的感觉。但这次做项目的时候，我突然感觉自己和曾经没学过这些课时候的自己不一样了。在设计曾经让我最头疼的软件系统架构时，脑子里自动就浮现出了学过的各种软件架构：“4+1”视图模型、3层经典架构等等，写代码时，脑海中又浮现出学过的设计模式：工厂模式、控制器模式等等。这时我发现那些理论知识不仅是用来考试的，而是在实践中起着巨大的作用。这让我对于从书本上学习的各种理论知识有了更加深刻的领悟。这些理论知识无声无息地存在于整个项目过程中。

我在项目中主要负责的是模型优化和系统调试整合。

这是我做的第一个深度学习项目。以前总听人说，做深度学习的人往往自称为：“炼丹师”、“炼金术师”，将训练深度学习模型叫做“炼丹”和“炼金术”。之前我对这些戏谑的昵称没有任何感觉，但这次轮到我真正经历时，才发现这些称呼形容的太贴切了！优化器、学习率、Dense层数、每个Dense层的神经元数、预训练模型的选择、epoch、解冻的层数......这些超参数的选择，就像是在这么多变量的排列组合中来回尝试，每次尝试前都不知道结果，仅靠一些简单的原则确定一个大致方向，完完全全是在黑暗中摸索，用时间去试错。而我所训练的InceptionV3验证集准确度最高达到96.3%，训练集99%，测试集93%。这是怎么达到的呢？首先是一个通过特征提取方法得到的验证集准确度94%的分类器，然后将它加载进来解冻一部分层，然后去训练。大多数时候验证集准确度就是88%，刚上一点90%，又会跌下来。来来回回在90%上下涨跌，仿佛再多的epoch都没有用。训练50轮就需要半小时基本。在一个深夜调参绝望时，我按ctrl+C终止了程序，再重新运行，没想到第一轮就达到了70%%，第二轮就达到了92%。这让我充满希望！瞬间激动！结果一夜过去了，当我醒来发现本该运行200epoch的它在40个epoch程序就自动退出了！！准确率达到了92%！！这可是微调，为什么准确率还没单单一个特征提取高呢?于是再调参，尝试，最后慢慢提升到了94%......95%......然后又调回92%.......我忍住迷茫，然后每次将最高的准确度模型加载进来，重新微调。前三轮如果上不了90%就立刻停止，重新运行。在答辩前的最后一个深夜，那是凌晨两点，我疲惫不堪的调着参。虽然疲惫，但我竟然感觉调参比打游戏还上瘾——就为了一个更高的准确度。当我终止了上一个模型的训练后，又重新运行，没想到——第一轮94%，第二轮95%.....在第8轮就上涨到了96%！！！最后再运行50个epoch内达到了最高验证集96.3%的准确度！！虽然往后再训练就一直在96%左右，但由于时间紧迫，我已经心满意足！事后我想为什么会上升到96%呢？我想到了一个比较无奈的原因——每次模型重新运行时会随机更新参数，准确率差的时候可能因为初始化到了不好的参数，甚至于陷入了局部最优点，而无法挣脱。而准确率好则是因为初始化了好的参数，这样的参数离全局最优点非常靠近。就像所谓的条条大路通罗马，但有的参数就出生在罗马。

除了模型优化让我崩溃，系统整合也薅掉我多头发。我们组是前端负责设计页面，后端只做简单页面，我需要把前后端结合起来。前端页面的变量和后端不一致改起来就很痛苦了，但更痛苦的是看懂别人的程序，好在后端同学一直给我讲，我加快了整合速度。但是这之前一个奇怪的问题出现了——这个程序只有后端同学能运行，其他同学要不就是包总找不动，要么就是页面找不到，路由不对。这个项目的结构也比较奇特。我只好重构了整个项目，使用flask的蓝图将项目重构。然后开始了痛苦的重构过程。功夫不负有心人，最后花了近五个小时完成了重构（中途被各种bug、环境不一致折磨）

虽然项目过程很痛苦，但我在这次实训中感受到了真正的团队合作和一个team的感觉。虽然之前有过很多次小组作业，但很多时候就是自己挑大梁。这次项目中，我的队友们团结一心，全力以赴，互帮互助，一起过了好几天没有午休的日子，熬了好几个夜。最后三天敲代码时每天都十几个小时待在腾讯会议里，谁有问题一说，全组人一起想办法讨论解决。

当模型准确率上不去，发现数据集噪声很大，崔爽同学爬取了近一万张图片，亲眼亲手一张张的进行校对剔除，最后让我们的模型训练有了更好的数据。

前端由王蕊同学负责，她因为基础薄弱，进度稍慢，但她认真学习不懂的地方，向我问完问题后会立刻去实践。在遇到很难解决的问题时，我们全组一起讨论后也没有解决方法，她会熬夜到凌晨1、2点去解决。

后端由郭龙强同学负责，他编程基础薄弱。每天在做后端时总会问我，你怎么敢相信我写的代码能跑起来？我说：你肯定没问题。郭龙强通过努力自学，不懂就问，积极请教同学，为我们组搭建起来了系统的强大后端。

答辩前一天我清晰的记得腾讯会议开了13个小时。原本李纪元同学已经完成了自己的任务，我还有一些任务没完成，他没有去想着早点休息，而是要求给他再分配一些任务，好让我也早点休息。最后他等着我完工后，我们一起才退出的会议。当PPT做完后的那刻，我清晰的记得是凌晨3点半。最后这个项目汇聚了我们全组的努力和智慧，汇聚了我们全组的心血，不论如何，我都觉得这是所有项目里最好的项目！

这次的项目实训对我来说是一个很好的成长过程，我不仅锻炼了动手能力、学习了知识、还感受到了宝贵的团队归属感、锻炼了团队协作能力和组织能力。在以后的学习和工作中我会一直记得这段宝贵的经历。

感谢我的队友们！感谢热心帮助过我们的同学们！

最后感谢王老师的辛苦传授和耐心解答。王老师会牺牲自己午休的时间为同学们解答问题。我清晰的记得我在一次中午下课问了老师几个问题，然后王老师中午13:20回复了我特别特别长的一条消息，里面按条分点地罗列了问题的解答，还为我扩展了很多知识。我当时看到后非常感动，非常感激老师。那么长的消息打字出来，可能都得20分钟。非常感谢王老师！老师您辛苦了！听着您经历三周每天讲课长达6个小时后，说话时难掩喉咙疼痛，我就希望如果是在线下上课就好了，可以为您多倒杯水、带一些金嗓子喉片。可惜是线上网课，只能充满感激和心疼。短短的四周却在我心里留下了深深的师生情谊。希望和王老师在未来有机会见面，在现实里叫您一声老师。祝愿王老师工作顺利，生活愉快！