**大数据项目总结报告**

**韩孟卓**

**2020.6.1**

目录

[**一、** **项目综述** 3](#_Toc42205520)

[**1.** **技术应用** 3](#_Toc42205521)

[**2.** **项目摘要** 3](#_Toc42205522)

[**3.** **文件介绍** 4](#_Toc42205523)

[**4.** **环境配置** 4](#_Toc42205524)

[**二、** **研究过程及结果分析** 5](#_Toc42205525)

[**1.** **研究过程** 5](#_Toc42205526)

[**1.1** **Tensorflow模型实现服装分类** 5](#_Toc42205527)

[**1.2** **保存模型** 10](#_Toc42205528)

[**1.3** **读取模型h5文件** 11](#_Toc42205529)

[**1.4** **创建Cassandra数据库** 11](#_Toc42205530)

[**1.5** **部署到Flask里并将结果插入Cassandra数据库** 12](#_Toc42205531)

[**1.6** **存入Docker容器** 14](#_Toc42205532)

[**三、** **学习感受与收获** 16](#_Toc42205533)

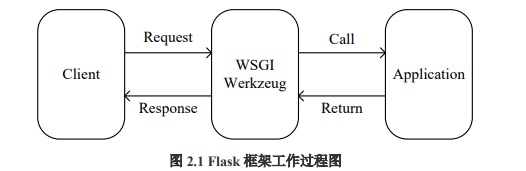
[**1.** **不足之处** 16](#_Toc42205534)

[**2.** **所得收获** 16](#_Toc42205535)

1. **项目综述**
2. **技术应用**

**Docker：**可以让开发者打包他们的应用以及依赖包到一个轻量级、可移植的容器中，然后发布到任何流行的机器上，也可以实现虚拟化。本次项目使用容器技术，实现了环境+应用的集成，方便项目部署和移植。

**Flask:** Python web开发框架，轻量级，采用了MVC开发模式，将前端，后端完美的分开，实现了代码的静态化。Flask的基本模式为在程序里将一个视图函数分配给一个URL，每当用户访问这个URL时，系统就会执行给该URL分配好的视图函数，获取函数的返回值并将其显示到浏览器上，其工作过程见图。



图表 1 Flask工作流程图

**Cassandra：**开源分布式NoSQL数据库系统,底层基于java实现，基于键值对的对数据进行存储。其特点是由一堆数据库节点共同构成的一个分布式网络服务，对Cassandra 的一个写操作，会被复制到其他节点上去，对Cassandra的读操作，也会被路由到某个节点上面去读取。对于一个Cassandra集群来说，扩展性能是比较简单的事情，只管在群集里面添加节点就可以了。

**Tensorflow：**谷歌开发的主流人工智能开发框架。本次以服装图像分类（Classify images of clothing）为模型进行训练。

**GitHub：**面向开源及私有软件项目的托管平台，具有Git代码仓库托管及基本的Web管理界面功能。本次项目利用GitHub获取模型数据和提交最终结题报告。

1. **项目摘要**

本项目利用python编程将前端、后端、图片识别、数据库以及Docker容器串联起来，形成一套完整的操作链条。使用者给出一张数字图片，利用已经训练好的模型文件（model.h5），对图片内容进行识别，从而给出图片中的具体数字是几。之后将其封装成API接口，供Flask应用调用。具体Flask主要实现用户基于网页上传图片，后端服务接收图片，之后将图片送入人工智能算法，得出结果，后端将结果展示给用户。最后将一系列结果（提供图片的当前时间、文件名、预测的数字）写入Cassandra数据库并部署到Docker容器中。

1. **文件介绍**

tensorflow model.py：服饰模型程序

kears\_save：存放人工智能模型文件

model.py：用户解析模型文件

static：存放flask应用的图片,CSS,JS等文件

templates:存放模板文件

cassandra\_op.py：用来实现对cassandra数据库的增删改查

README.md:项目描述文件

upload\_pictures.py:存放和前端交互的后端代码，起着控制器的作用

1. **环境配置**

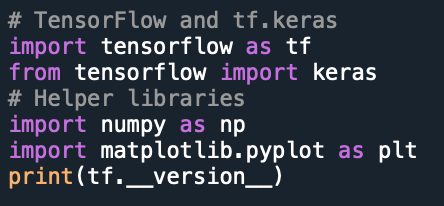
Python库：tensorflow:2.2.0，cassandra-driver

java环境：jdk:1.8.0\_202

数据库要求：Cassandra:3.11.6

1. **研究过程及结果分析**
2. **研究过程**
   1. **Tensorflow模型实现服装分类**

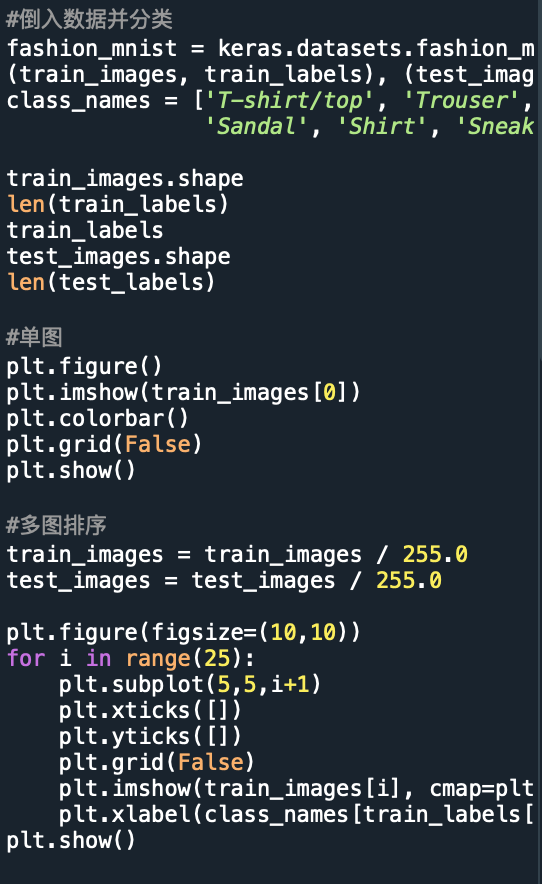
（1）导入需要的所有工具包，使用tf.keras作为API在TensorFlow中构建和训练模型。



程序图 1.1

（2）准备数据

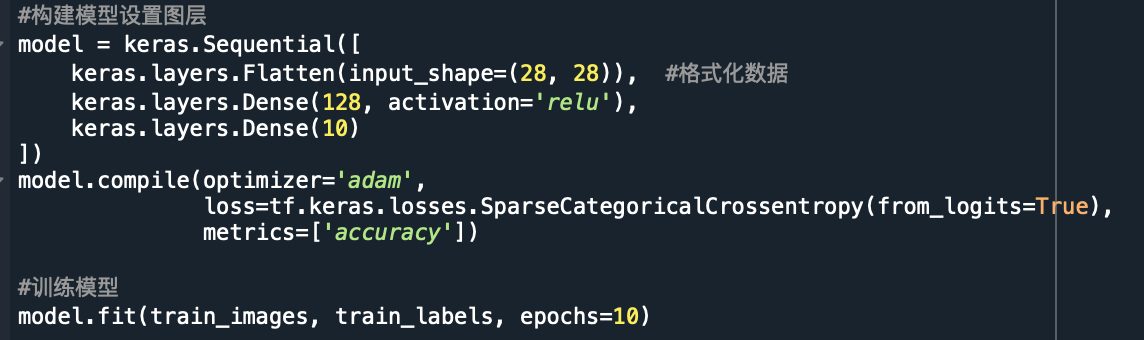
利用fashion\_mnist数据训练一个神经网络模型对衣服的图像进行分类，比如运动鞋和衬衫。数据由60000张（服饰图）训练数据和10000张测试数据组成，每条数据的范围从0到9共10类，图像是28\*28个numpy数组。在对网络进行训练之前，必须对数据进行预处理。



程序图 1.2

（3）创建并训练模型

构建神经网络需要对模型的层进行配置，然后对模型进行编译。要求模型能够将图像和标签联系起来，并对测试集作出预测，最后检验预测结果是否与test\_labels数组中的标签匹配。



程序图 1.3

（4）评估模型

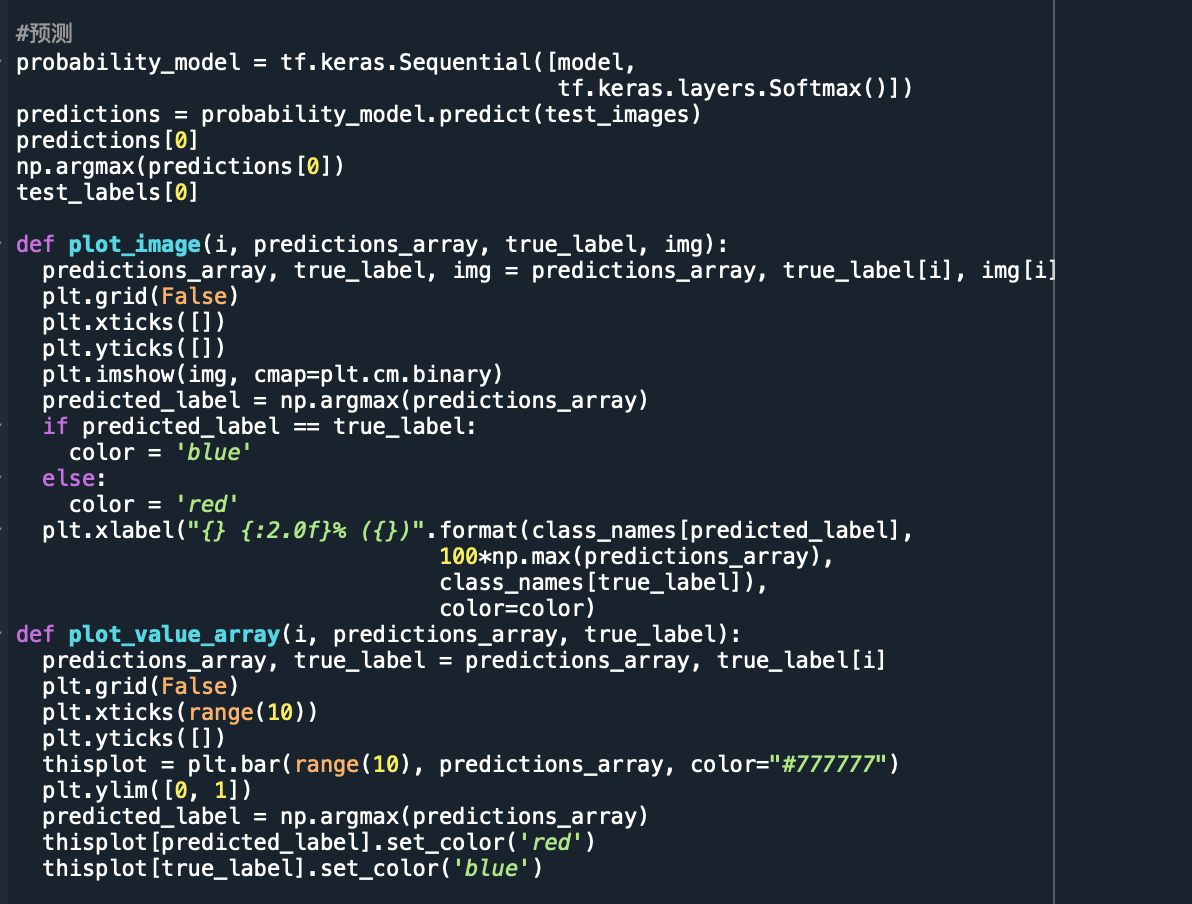


程序图 1.4

（5）对预测结果可视化

作出预测：

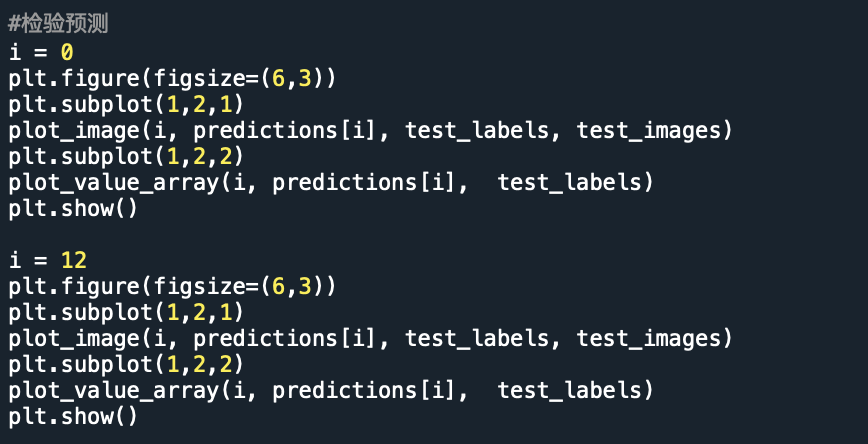
通过训练模型，可以使用它对一些图像进行预测，预测正确显示蓝色，否则显示红色。



程序图 1.5

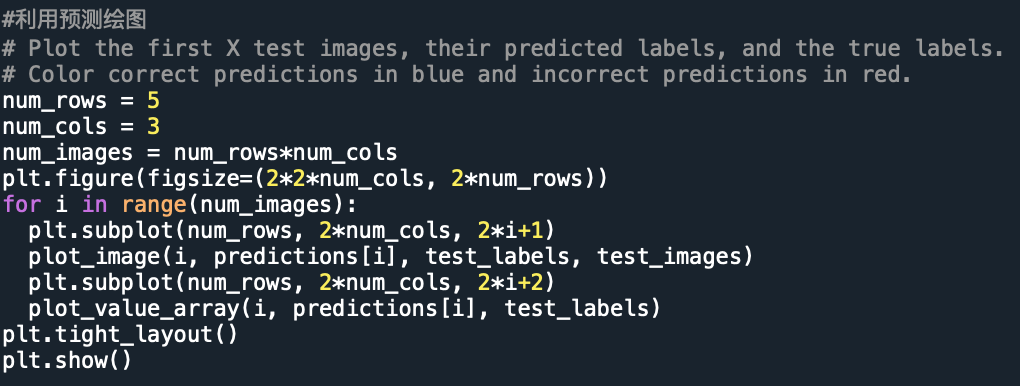
验证预测：

分别显示第0张和第12张图片、预测和预测数组。正确的预测标签是蓝色的，错误的预测标签是红色的。



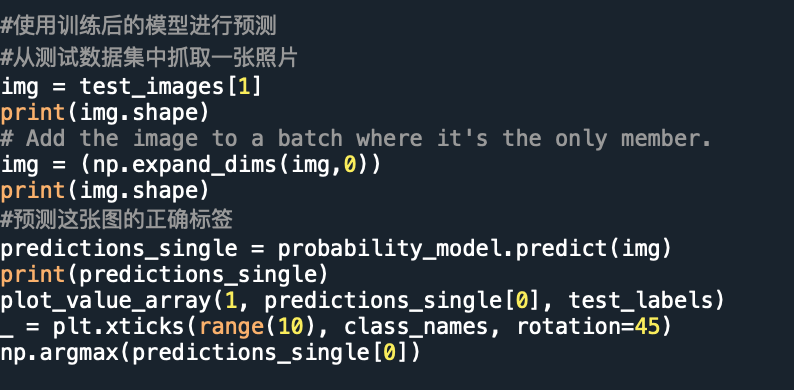
程序图 1.6

预测绘图：



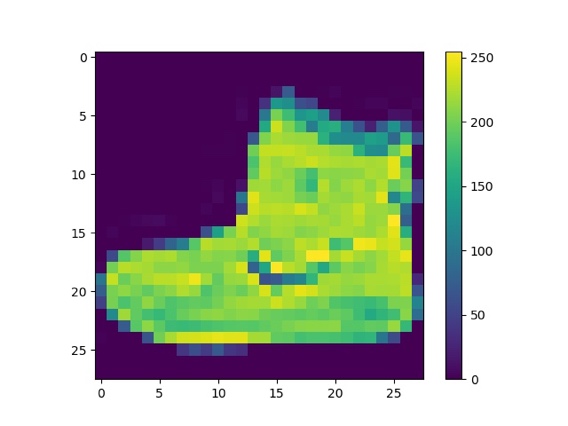
程序图 1.7

使用训练后的模型进行预测：



程序图 1.8

（6）结果分析：



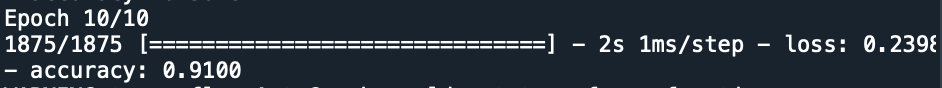
结果图 1.1

结果图1显示检查训练集中的第一个图像，得到的图像会显示像素值落在0到255的范围内。



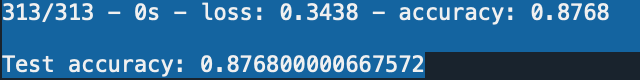
结果图 1.2

结果图2显示来自训练集的前25个图像，并在每个图像下面显示类名，以便检验数据格式是否正确。



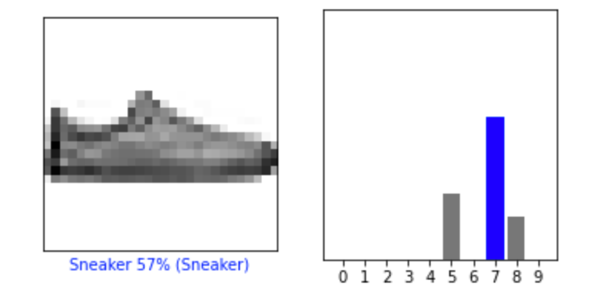
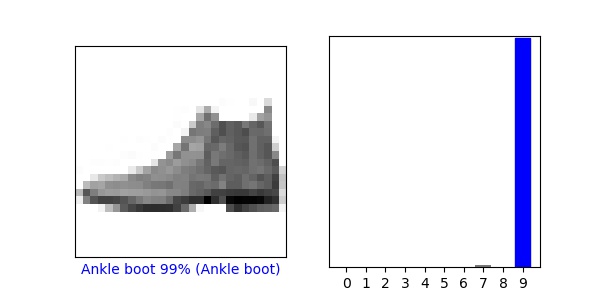
结果图 1.3

根据输出结果可以看出训练的模型准确率为0.91。



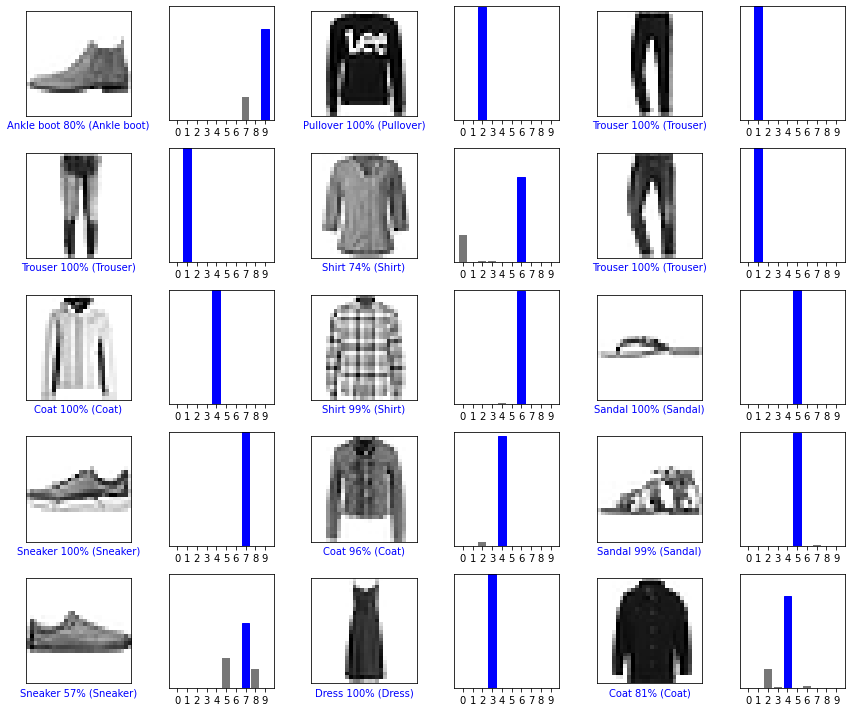
结果图 1.4

测试数据集上的精度比训练数据集上的精度略低一些。这种训练精度和测试精度之间的差距表示过度拟合。当机器学习模型在新的、以前不可见的输入上的表现不如在训练数据上的表现时，就会发生过度拟合。过度拟合的模型会“记住”训练数据集中的噪声和细节，从而对模型在新数据上的性能产生负面影响。



结果图 1.5 结果图 1.6

左边图像99%的概率认为是Ankle boot，预测结果高度符合；右边图像57%的概率认为是Sneaker，准确率稍有下降，可能是由于模型训练次数不够或者图像分辨率过低导致。

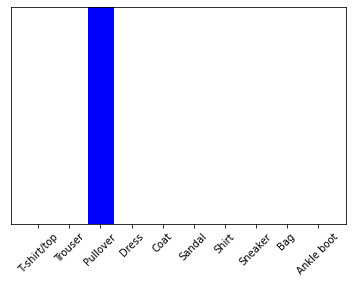


结果图 1.7

上述图像可以看出，每一次的预测结果不会完全相同，即使模型完全成熟也会存在误差。



结果图 1.8

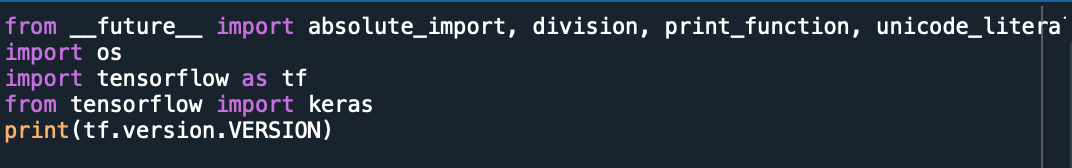


结果图 1.9

通过数据和图表可以看出，预测结果中有两类极其相似，但最终模型预测出的标签是正确的。

* 1. **保存模型**

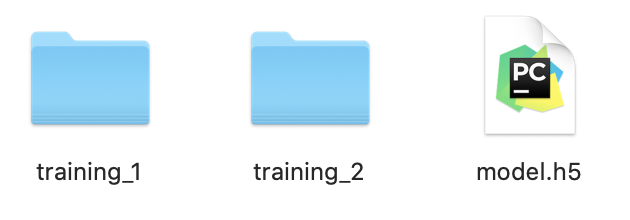
存储模型有多种方式，本次项目使用tf.kears作为API用在tensorflow中构建和训练模型，文件名为model，并保存格为h5文件，可视为二进制的blob。该文件内保存了完整的模型，包括权重、模型配置以及优化器配置。



程序图 2.1



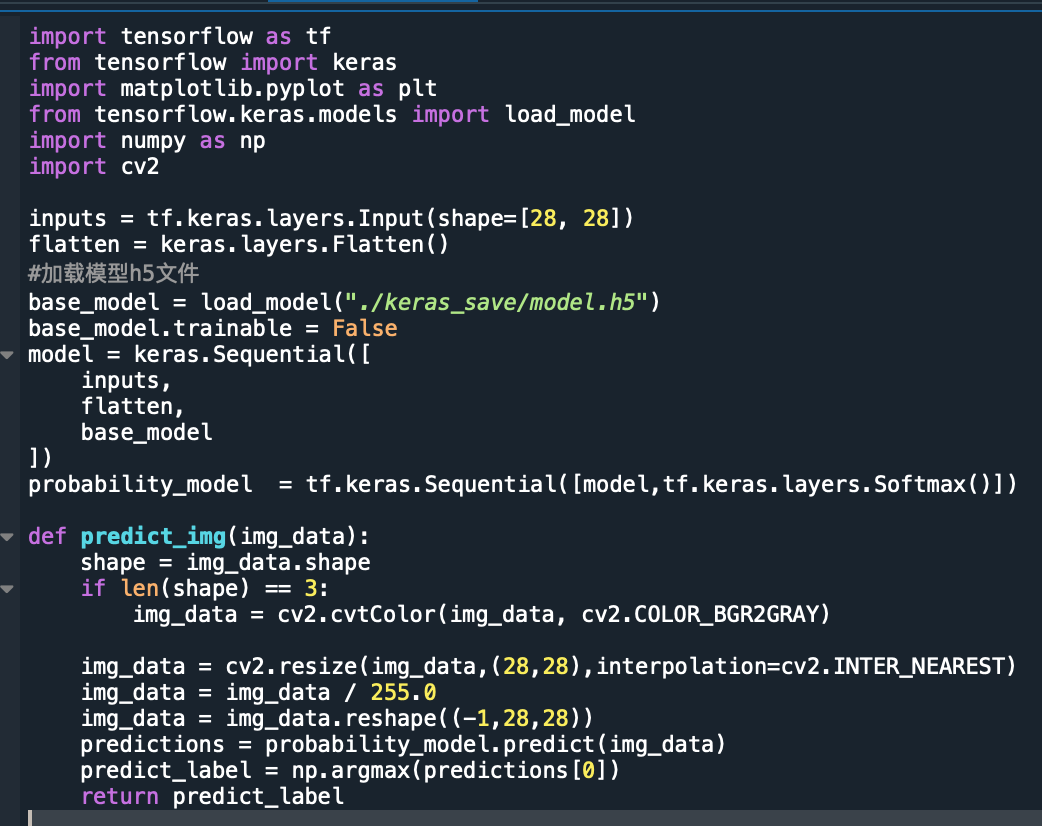
程序图 2.2

****

结果图 2.1

* 1. **读取模型h5文件**

model.h5用于存放训练好的模型和参数。model.py用户解析模型文件，从而实现对图片的预测。

****

程序图 3.1

* 1. **创建Cassandra数据库**

cassandra必须要配置java环境才能正常运行，本次使用的版本为jdk:1.8.0\_202。

# 首先从官网下载cassandra并解压下载好的文件

wget  http://mirrors.cnnic.cn/apache/cassandra/3.11.6/apache-cassandra-3.11.6-bin.tar.gz

tar -zxvf apache-cassandra-3.11.6-bin.tar.gz

# 更改文件目录名字

mv apache-cassandra-3.11.6cassandra

# 设置cassandra数据目录

# 打开cassandra的配置文件cassandra.yaml

　vim~/cassandra/conf/cassandra.yaml

data\_file\_directories: - /var/lib/cassandra/data #为数据文件目录

commitlog\_directory: /var/lib/cassandra/commitlog #为日志文件目录

saved\_caches\_directory: /var/lib/cassandra/saved\_caches #为缓存文件目录

# 新建目录，用于存放cassandra运行过程中数据，缓存，日志文件。

~ sudo mkdir -p /var/lib/cassandra/data

~ sudo mkdir -p /var/lib/cassandra/saved\_caches

~ sudo mkdir -p /var/lib/cassandra/commitlog

~ sudo mkdir -p /var/log/cassandra

# 把目录归属改成操作用户

~ sudo chown -R ubuntu:ubuntu /var/lib/cassandra

~ sudo chown -R ubuntu:ubuntu /var/log/cassandra/

# 配置环境变量

~ sudo vi /etc/environment

CASSANDRA\_HOME=/home/ubuntu/cassandra

# 让环境变量生效

~ . /etc/environment

# 查看环境变量

~ echo $CASSANDRA\_HOME/home/ubuntu/cassandra

# 启动cassandra

~/cassandra/bin$ ./Cassandra

上述为创建cassandra数据库的命令行，以上工作均为后续文件写入做准备。

* 1. **部署到Flask里并将结果插入Cassandra数据库**

第一部分实现界面设计，实现方法div+css。它是用户和后端交互的接口。该部分实现的代码存放在templates目录下，包括两个文件upload.html和upload\_ok.html。

upload.html文件用来实现图片的上传。



结果图 6.1

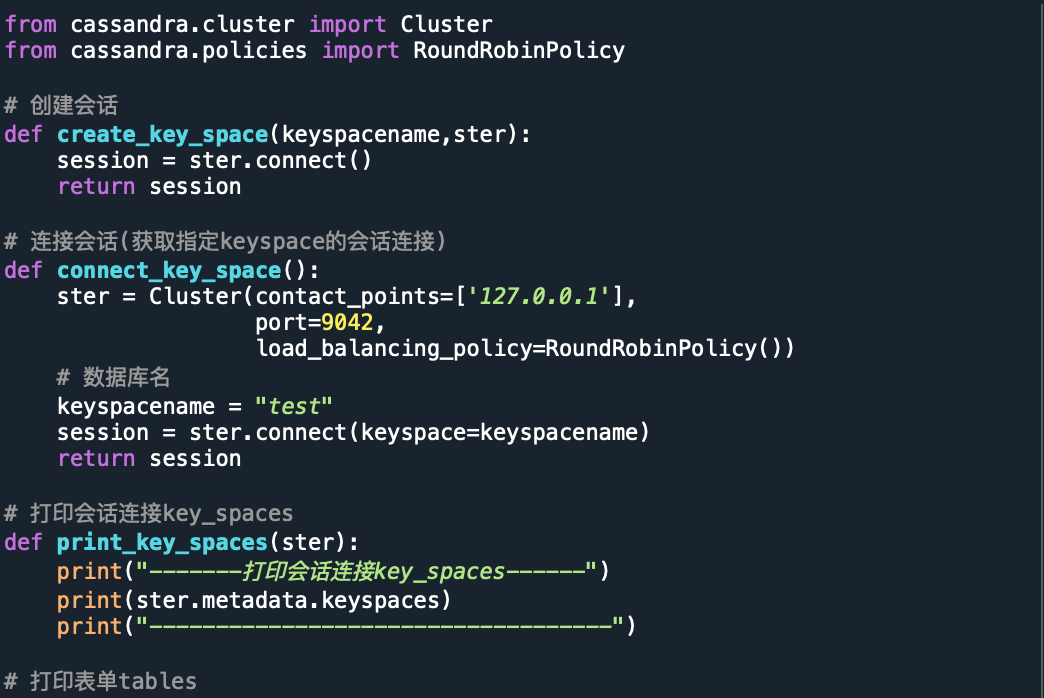
upload\_ok.html文件用来处理后端返回给用户的请求结果。



结果图 6.2

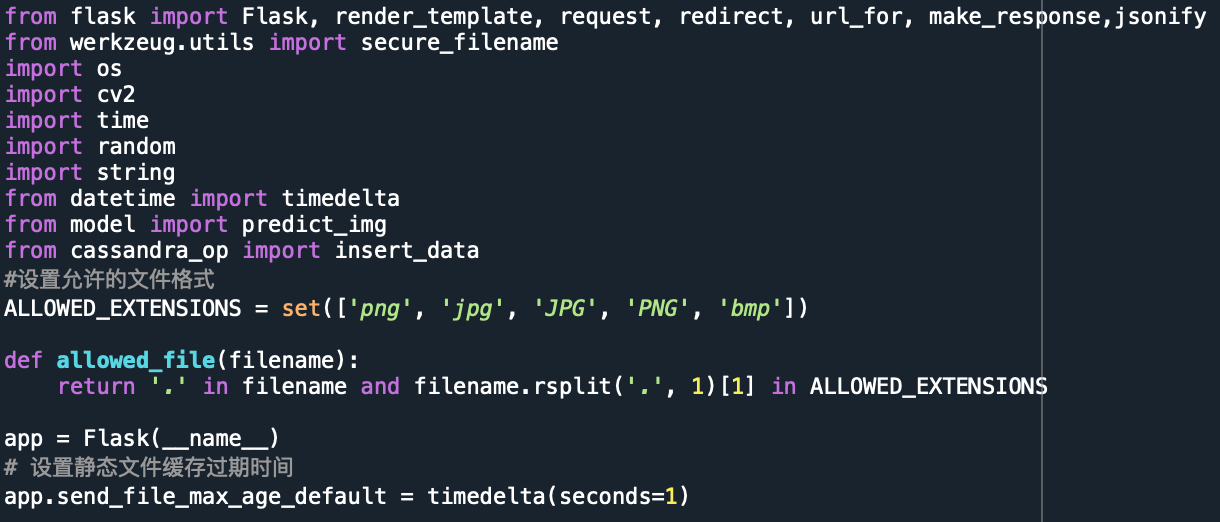
第二部分主要用来实现接收前端请求，返回用户请求结果。文件分为cassandra\_op.py和upload\_pictures.py。

cassandra\_op.py：用于对cassandra数据库的操作，创建并与flask文件链接。

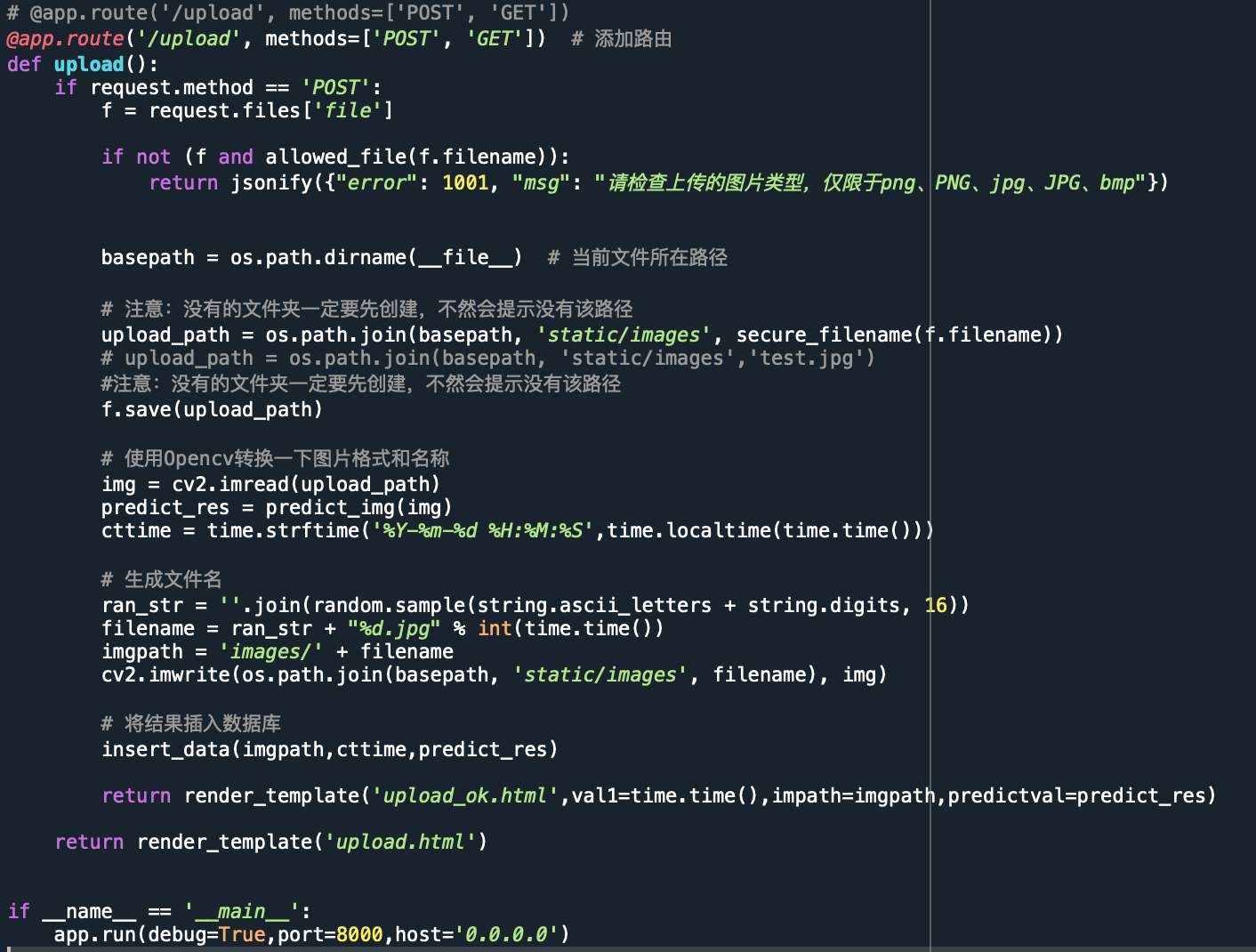


程序图 5.1

upload\_pictures.py：用于接受用户前端的请求，并对请求进行响应。



程序图 5.2

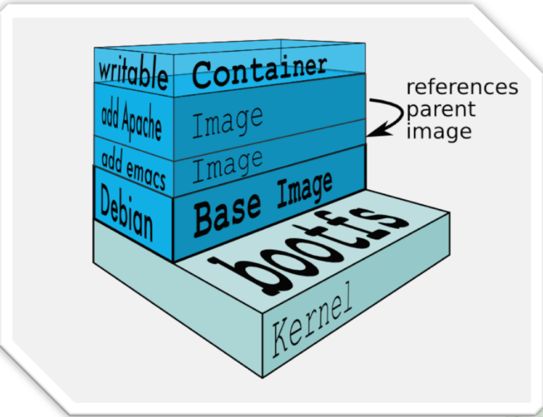


程序图 5.3

* 1. **存入Docker容器**

docker利用容器来运行应用，docker容器是由docker镜像创建的运行实例。docker容器类似虚拟机，可以执行包含启动，停止，删除等。容器中会运行特定应用的代码及所需的依赖文件，可以把容器看作一个简易版的linux环境（包含root用户权限，进程空间，用户空间和网络空间等）和运行在其中的应用程序。

相对于镜像来说容器是动态的，容器在启动的时候创建了一层可写层次作为最上层。一个镜像可创建多个容器，每个容器都有各自的一个可读写层，这些层相互独立共享下面的镜像。



图表 6.1

我们本次搭建的小项目是基于ubuntu镜像进行实现。

push镜像至docker库：

1. 在[hub.docker.com](https://hub.docker.com/)上注册帐户，登录本地的Docker公共注册表。

命令：docker login

1. 将本地映像与注册表上的存储库相关联，username/repository:tag:

命令：docker tag REPOSITORY username/repository:tag

1. 将标记的图像上传到存储库。

命令：docker push username/repository:tag

1. 之后在ubuntu镜像的基础上利用上述命令进行了容器创建，项目环境的部署，镜像的发布流程。



程序图 6.1

程序图6.1具体操作可参看视频mit.wmv，运行成功后便可在网页中输入产生的网址：192.168.10.148:8080/upload，上传一张手写数字图片，得到预测结果。

1. **学习感受与收获**
2. **不足之处**
3. 测试模型时输出的结果每次准确率都会有细微差别，后续查看参考资料得知这个需要经验调参，因为只是个基础模型，许多细节问题还有待完善；另一方面也可能是测试图片包含噪声过多造成的。但本次项目并未对这个问题进行深入研究。
4. 由于电脑内存不足，后续运行docker时在另一台设备上完成的，迁移数据时路径改变导致项目完成时间变长。
5. **所得收获**

本次项目所接触的工具对于我而言基本是全新的，全篇通过Python进行编程，以Tensorflow的服饰模型为例，以Flask为框架将模型与Cassandra数据库连接起来，然后部署到Docker容器中，最终打包上传到GitHub，方便大家从网络上直接拉取文件。

以上这些内容，只有Python是在之前有所了解。起初上课时对于老师所讲的内容也是一知半解，理论听懂了，但是运用到实际中来还是会有很多困难，一开始甚至连这几样之间如何连接都没考虑清楚，更别说程序的编写，Flask与Cassandra对接的接口问题，Cassandra的安装等等许多问题。

整个项目的制作大概花费了两周时间，从网上又查看了许多文献资料来补充对项目的理解。最终完成项目开发的时候整个人都得到了升华，在参加项目前我可能都想不到自己能写出这么多的程序，并将他们都连接起来。总而言之，非常感谢老师能够带我入门，这次马马虎虎的作业也勾起了我对机器学习的兴趣，为我打开了另一扇门供将来发展做选择。我打算在毕设中继续完善这块内容，争取能够利用视频流识别Tensorflow中的模型。