远程科研项目学习报告——大数据方向

2019/9/15

万铠宁

1.学习背景

大数据是指无法在一定时间范围内用常规软件工具进行捕捉、管理和处理的数据集合，是需要新处理模式才能具有更强的决策力、洞察发现力和流程优化能力的海量、高增长率和多样化的信息资产。其被定义为一种规模大到在获取、存储、管理、分析方面大大超出了传统数据库软件工具能力范围的数据集合，具有海量的数据规模、快速的数据流转、多样的数据类型和价值密度低四大特征。同时，大数据需要新处理模式才能具有更强的决策力、洞察发现力和流程优化能力来适应海量、高增长率和多样化的信息资产。

从战略角度来说，大数据技术的意义不在于掌握庞大的数据信息，而在于对这些含有意义的数据进行专业化处理。换而言之，如果把大数据比作一种产业，那么这种产业实现盈利的关键，在于提高对数据的“加工能力”，通过“加工”实现数据的“增值”。

从技术上看，大数据又与云计算存在着必不可分的关系。大数据必然无法用单台的计算机进行处理，必须采用分布式架构。它的特色在于对海量数据进行分布式数据挖掘。但它必须依托云计算的分布式处理、分布式数据库和云存储、虚拟化技术。随着云时代的来临，大数据也吸引了越来越多的关注。分析师团队认为，大数据通常用来形容一个公司创造的大量非结构化数据和半结构化数据,这些数据在下载到关系型数据库用于分析时会花费过多时间和金钱。大数据分析常和云计算联系到一起，因为实时的大型数据集分析需要像 MapReduce 一样的框架来向数十、数百或甚至数千的电脑分配工作。

在本次项目中，我通过 python 为主的编程分析工具，以 Linux 和 Windows 为主要操作系统，主要学习了大数据分析中的以下内容：

（1）.服务部署工具 Docker，服务调用工具 Flask，半结构化数据 Json、XML、CSV、Yaml，发布服务方法 RESTful API 调用方式。

（2）.存储技术中 NoSQL 的 Cassandra 原理以及基本操作，Python 代码和容器 Docker 进

行通信的方法。

（3）.Spark 批处理,Spark 流数据处理;数据可视化,Playground 深度学习可视化;Cloud Native Landscape 之中的其他数据工具

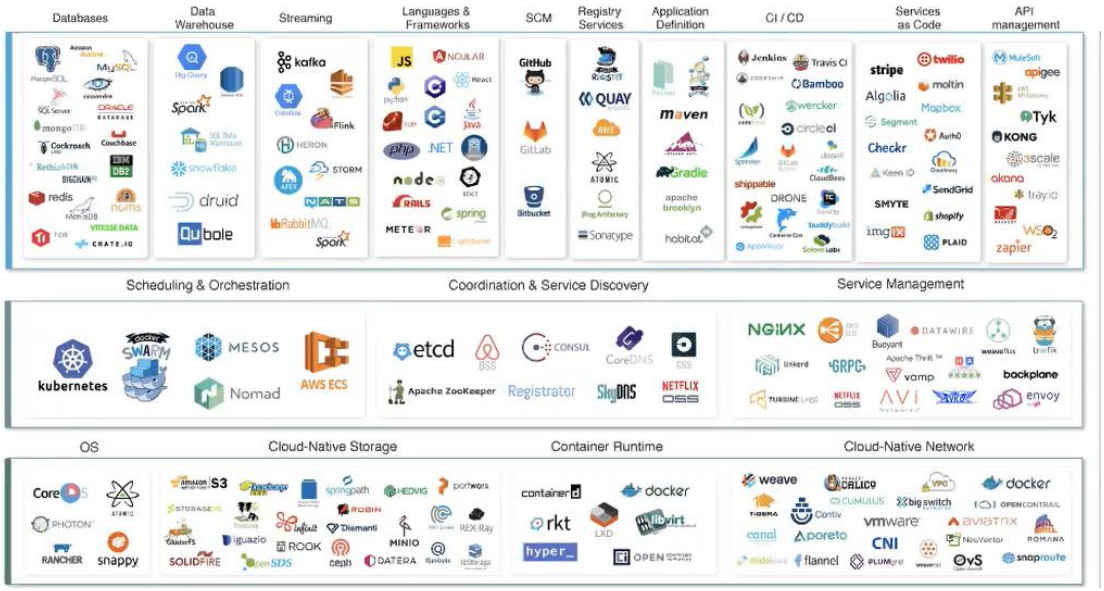

图 1.1 Cloud Native Landscape 之中的工具

现在的社会是一个高速发展的社会，科技发达，信息流通，人们之间的交流越来越密切，生活也越来越方便，大数据就是这个高科技时代的产物。未来的时代将不是 IT 时代，而是 DT（Data Technology 数据科技）的时代。大数据帮助当今的很多小而美模式的中小微企业实现了转型，也帮助大型企业与时俱进。而上述的这些知识作为大数据分析的基础，同时涵盖了前端，存储，服务等不同层面的不同工具，在现在以及未来的技术领域中有着不错的前景。

2. 服务端技术

2.1 Docker 容器技术

Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的镜像中，然后发布到任何流行的 Linux 或 Windows 机器上，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口。作为一个开源的引擎，其可以轻松的

为任何应用创建一个轻量级的、可移植的、自给自足的容器。开发者在笔记本上编译测试通过的容器可以批量地在生产环境中部署，包括 VMs（虚拟机）、 bare metal、OpenStack 集群和其他的基础应用平台。 主要基于 Go 语言开发 并遵从 Apache2.0 协议开源。

Docker 拥有以下优点：更高效的利用系统资源，更快速的启动时间，一致的运行环境，持续交付和部署，更轻松的迁移，更轻松的维护和扩展。因此对比传统虚拟机，其在启动，硬盘使用，性能以及系统支持量上都有巨大的提升。

从构成方面来讲，Docker 作为一个引擎，它的主要构成为 server、REST API、CLI 三部分。Server 是指 Docker 利用更加节省的硬件资源提供给用户更多的计算资源的守护进程。而 REST 风格的 API 封装了 Docker 的内部实现，并提供了对外的接口。同时 Docker 提供了 CLI 命令行工具，方便我们更容易的上手。

Docker 的核心共分为四部分：image（镜像）、container（容器）、network（网络）、datavolumes（数据卷）。

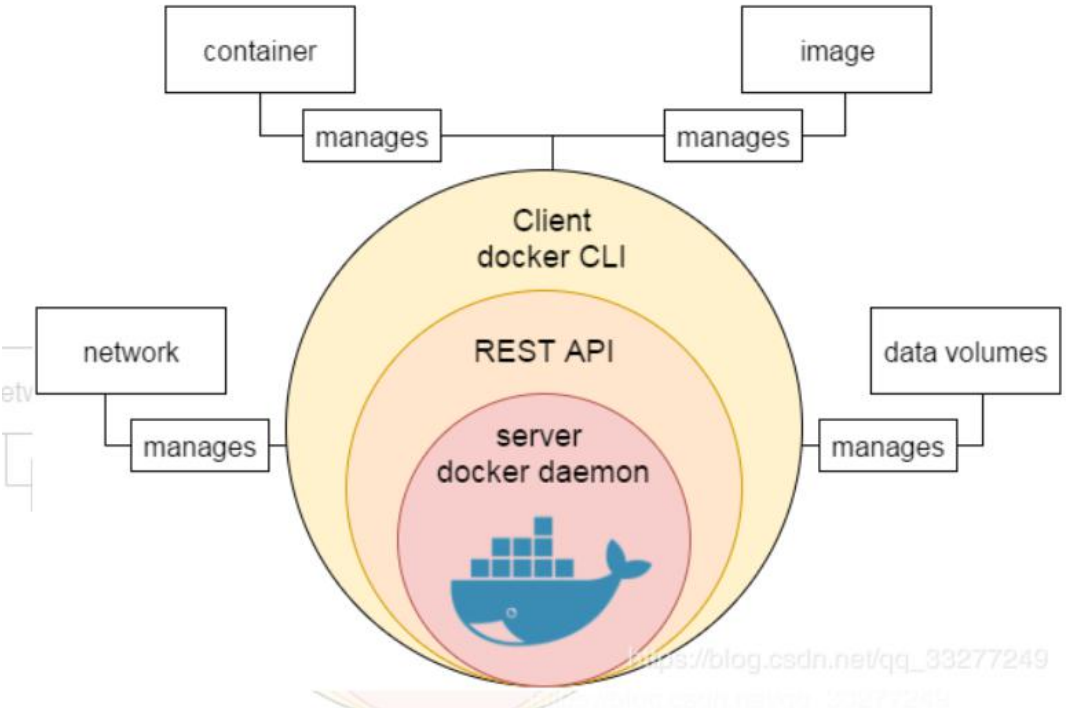

图 2.1 Docker 引擎的构成

2.2 Flask 技术

Flask 是一个使用 Python 编写的轻量级 Web 应用框架。其 WSGI 工具箱采用 Werkzeug ，模板引擎则使用 Jinja2 。Flask 使用 BSD 授权。同时其也被称为“microframework” ，因为它使用简单的核心，用 extension 增加其他功能。Flask 没有默认使用的数据库、窗体验证工具。

Flask 是一个轻量级的可定制框架，使用 Python 语言编写，较其他同类型框架更为灵活、轻便、安全且容易上手。它可以很好地结合 MVC 模式进行开发，开发人员分工合作，小型团队在短时间内就可以完成功能丰富的中小型网站或 Web 服务的实现。另外，Flask 还有很强的定制性，用户可以根据自己的需求来添加相应的功能，在保持核心功能简单的同时实现功能的丰富与扩展，其强大的插件库可以让用户实现个性化的网站定制，开发出功能强大的网站。

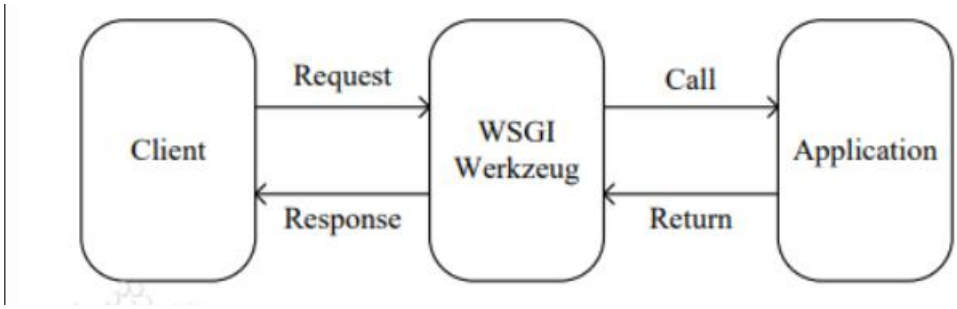

图 2.2 Flask 框架构成图

2.3 RESTful API 调用

RESTful API 是指用 URL 定位资源、用 HTTP 动词（GET、POST、PUT、DELETE)描述操作的 API，采用 HTTP 做传输协议。 其在当今的互联网应用的前端展示媒介很丰富。有手机、有平板电脑还有 PC 以及其他的展示媒介。RESTful API 的应用避免了单独为每个客户端写一个后台系统，而是写一个后台系统提供 rest 风格的 URI。因此，RESTful API 就是通过一

套协议来规范多种形式的前端和同一个后台的交互方式。在 rest 中会通过向服务器提交的请求来表示的操作主要有：（1）.GET（SELECT）：从服务器取出资源（2）.POST（CREATE）：在服务器新建一个资源（3）. PUT（UPDATE）：在服务器更新资源（4）DELETE（DELETE）：从服务器删除资源

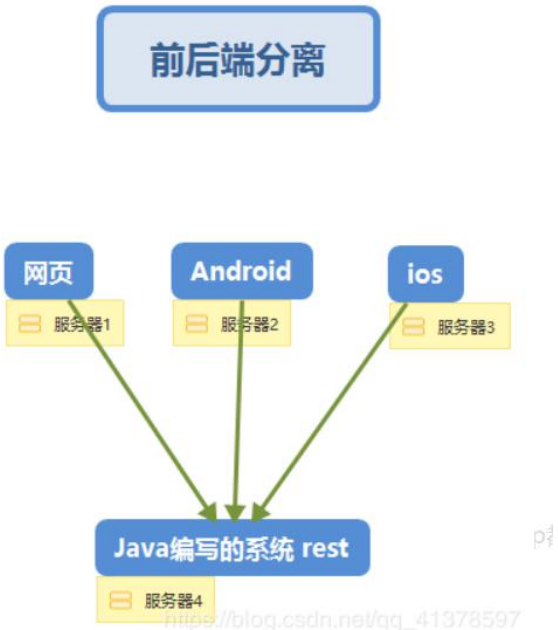

图 2.3 Restful API 框架原理

3. 存储技术

3.1 Cassandra 数据库系统

Cassandra 是一套开源分布式 NoSQL 数据库系统。它最初由 Facebook 开发，用于储存收件箱等简单格式数据，集 GoogleBigTable 的数据模型与 Amazon Dynamo 的完全分布式的架构于一身 Facebook 于 2008 将 Cassandra 开源，此后，由于 Cassandra 良好的可扩展性，被 Digg、Twitter 等知名 Web 2.0 网站所采纳，成为了一种流行的分布式结构化数据存储方案。

Cassandra 是一个混合型的非关系的数据库，类似于 Google 的 BigTable。其主要功能比 Dynamo （分布式的 Key-Value 存储系统）更丰富，但支持度却不如文档存储 MongoDB（介

于关系数据库和非关系数据库之间的开源产品，是非关系数据库当中功能最丰富，最像关系数据库的。支持的数据结构非常松散，是类似 json 的 bjson 格式，因此可以存储比较复杂的数据类型）。它是一个网络社交云计算方面理想的数据库。以 Amazon 专有的完全分布式的 Dynamo 为基础，结合了 Google BigTable 基于列族（Column Family）的数据模型。P2P 去中心化的存储。Cassandra 拥有三个突出特点：模式灵活，可扩展性，多数据中心。

图 3.1 Cassandra 数据库系统

4.其他技术

4.1 SPARK 计算引擎

Apache Spark 是专为大规模数据处理而设计的快速通用的计算引擎。Spark 是 UC Berkeley AMP lab 所开源的类 Hadoop MapReduce 的通用并行框架，拥有 Hadoop MapReduce 所具有的优点；但不同于 MapReduce 的是——Job 中间输出结果可以保存在内存中，从而不

再需要读写 HDFS,因此 Spark 能更好地适用于数据挖掘与机器学习等需要迭代的 MapReduce 的算法。 作为一种与 Hadoop 相似的开源集群计算环境，两者之间还存在一些不同之处，这些有用的不同之处使 Spark 在某些工作负载方面表现得更加优越，换句话说，Spark 启用了内存分布数据集，除了能够提供交互式查询外，它还可以优化迭代工作负载。

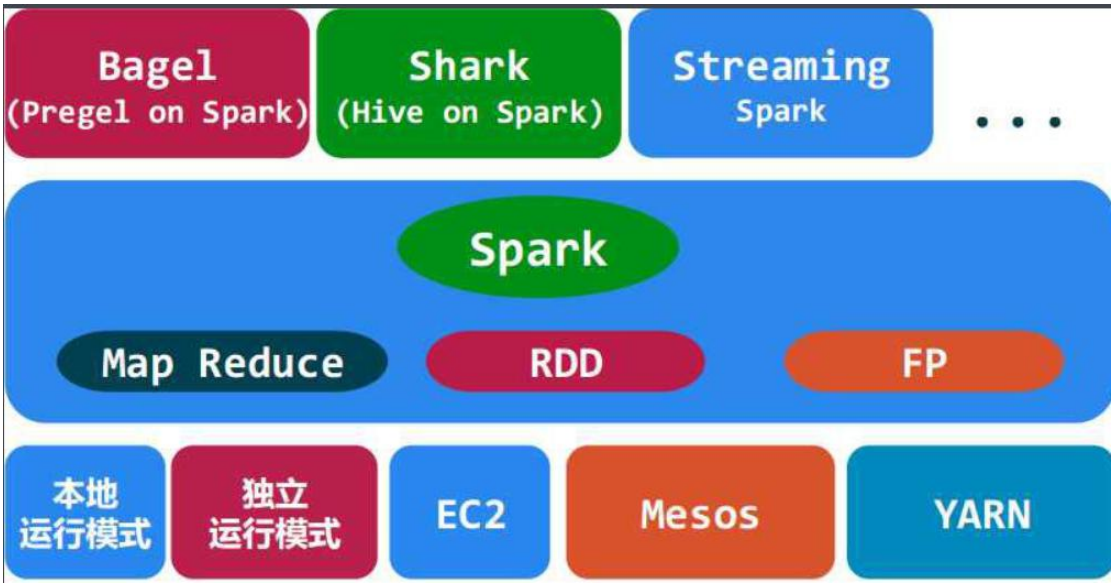

图 4.1 Spark 处理结构

5.项目流程

5.1 项目简介

在本项目中，我们首先需要将 MNIS_SOFTMAXT 或者 MNIS_DEEP(即手写体识别的两种算法)的模型提取出来（作为 moddel.ckpt 模型保存）。其次我们需要读取我们保存的模型使其能够应用到识别手写体图片的数字中，即以模型为基础，我们需要生成一个函数，输入为需要识别的图片，输出为数字。随后我们需要将上述函数部署到 flask 中，同时对 MNIST 中用户每次提交的图片、识别的文字和时间戳信息，都要记录到 Cassandra 以内尽兴存储。

5.2 项目具体流程

5.2.1 提取 model.ckpt

首先我们需要提取 MNIS_SOFTMAX 或者 MNIS_DEEP 算法当中的模型。对于 MNIS_SOFTMAX 模型

而言，我们先创建一个形如 $y = soft\max(Wx + b)$ 的模型，其实现原理可以用类似矩阵乘法和向量相加的线性代数知识解释：对于输入的 xs 加权求和，再分别加上一个偏置量，最后再输入到 softmax 函数中：

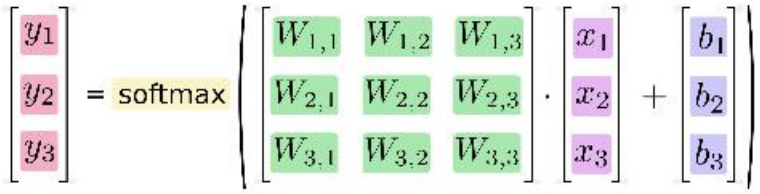

图 4.1 Softmax 原理解释

而其代码实现如下：

创建模型：

```
# Create the model
x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
y = tf.matmul(x, W) + b

# Define loss and optimizer
y_ = tf.placeholder(tf.float32, [None, 10])

# The raw formulation of cross-entropy,
#
#   tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(tf.nn.softmax(y)),
#                                 reduction_indices=[1]))
#
# can be numerically unstable.
#
# So here we use tf.nn.softmax_cross_entropy_with_logits on the raw
# outputs of 'y', and then average across the batch.
cross_entropy = tf.reduce_mean(
    tf.nn.softmax_cross_entropy_with_logits(labels=y_, logits=y))
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)

sess = tf.InteractiveSession()
tf.global_variables_initializer().run()
```

训练模型：

```
# Train
for _ in range(1000):
  batch_xs, batch_ys = mnist.train.next_batch(100)
  sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
```

测试训练结果:

```
# Test trained model
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print(sess.run(accuracy, feed_dict={x: mnist.test.images,
                                    y_: mnist.test.labels}))
```

而 MNIS_DEEP 算法更为复杂，主要原因是应用了卷积神经网络，其结构如下图：

```
W_conv1 = weight_variable([5, 5, 1, 32])
b_conv1 = bias_variable([32])

x_image = tf.reshape(x,[-1,28,28,1])

h_conv1 = tf.nn.relu(conv2d(x_image,W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)

W_conv2 = weight_variable([5, 5, 32, 64])
b_conv2 = bias_variable([64])

h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)

W_fc1 = weight_variable([7 * 7 * 64, 1024])
b_fc1 = bias_variable([1024])

h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)

keep_prob = tf.placeholder("float")
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)

W_fc2 = weight_variable([1024, 10])
b_fc2 = bias_variable([10])

y_conv=tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
```

而我们提取模型的代码如下:

```
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for i in range(20000):
        batch = mnist.train.next_batch(50)
        if i % 100 == 0:
            train_accuracy = accuracy.eval(feed_dict={
                x: batch[0], y_: batch[1], keep_prob: 1.0})
            print('step %d, training accuracy %g' % (i, train_accuracy))
        train_step.run(feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})
    saver.save(sess, 'C:/Users/user/IdeaProjects/deep_experiment/.idea/saver/model.ckpt') #模型储存位置

    print('test accuracy %g' % accuracy.eval(feed_dict={
        x: mnist.test.images, y_: mnist.test.labels, keep_prob: 1.0}))
```

运行程序后，我们发现 model.ckpt 在本地被成功存储。

5.2.2 应用 model.ckpt 识别图片

由于模型的局限性，我们输入的图像仅仅局限于 28*28 单通道灰度图片，因此我们需要安装 opencv 来将用户提交的图片进行转化，其代码实现如下：

```
def imageprepare(save_filepath):
    global img
    img = cv2.imread(save_filepath)  # 手写数字图像所在位置
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  # 转换图像为单通道(灰度图)
    resize_img = cv2.resize(img, (28, 28))  # 调整图像尺寸为28*28
    ret, thresh_img = cv2.threshold(resize_img, 127, 255, cv2.THRESH_BINARY)  # 二值化
    cv2.imwrite('./image/transfer.png', thresh_img)  # 预处理后图像保存位置
    cv2.waitKey(0)
    im = Image.open('./image/transfer.png')  # 读取的图片所在路径，注意是28*28像素
    plt.imshow(im)  # 显示需要识别的图片
    im = im.convert('L')
    tv = list(im.getdata())
    tva = [(255 - x) * 1.0 / 255.0 for x in tv]
    return tva
```

之后我们便可以应用程序识别转化后的图片，注意要与 MNIST_DEEP 算法匹配：

```python
def predict(save_filepath):
    result = imageprepare(save_filepath)
    x = tf.placeholder(tf.float32, [None, 784])
    y_ = tf.placeholder(tf.float32, [None, 10])
    W_conv1 = weight_variable([5, 5, 1, 32])
    b_conv1 = bias_variable([32])
    x_image = tf.reshape(x, [-1, 28, 28, 1])
    h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
    h_pool1 = max_pool_2x2(h_conv1)
    W_conv2 = weight_variable([5, 5, 32, 64])
    b_conv2 = bias_variable([64])
    h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
    h_pool2 = max_pool_2x2(h_conv2)
    W_fc1 = weight_variable([7 * 7 * 64, 1024])
    b_fc1 = bias_variable([1024])
    h_pool2_flat = tf.reshape(h_pool2, [-1, 7 * 7 * 64])
    h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
    keep_prob = tf.placeholder("float")
    h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)
    W_fc2 = weight_variable([1024, 10])
    b_fc2 = bias_variable([10])
    y_conv = tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
    cross_entropy = -tf.reduce_sum(y_ * tf.log(y_conv))
    train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
    correct_prediction = tf.equal(tf.argmax(y_conv, 1), tf.argmax(y_, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
    saver = tf.train.Saver()
    with tf.Session() as sess:
        sess.run(tf.global_variables_initializer())
        saver.restore(sess, "./saver/model.ckpt")  # 使用模型，参数和之前的代码保持一致
        prediction = tf.argmax(y_conv, 1)
        predint = prediction.eval(feed_dict={x: [result], keep_prob: 1.0}, session=sess)
        print('识别结果:')
```

5.2.3 将程序部署到 FLASK 中

该程序需要将前面的两组程序结合起来，应用 FLASK 使其支持用户的‘POST’请求。当我们在 Terminal 中使用 flask run 语句时，即可成功在 127.0.0.1：8000 默认 IP 和端口实现用户上传图片的操作。主要实现语句如下：

```python
@app.route('/mnist', methods=['GET','POST'])
def mnist():
    """
    when users submit pictures to '0.0.0.0:8000/mnist',
    the program returns users predictions and records them on Cassandra
    """
    req_time = datetime.now()

    if request.method == 'POST':
        file = request.files['file']
        if file and allowed_file(file.filename):
            # get the upload time, the filename, the filepath and the prediction
            upload_filename = secure_filename(file.filename)
            save_filename = str(req_time).rsplit('.', 1)[0] + ' ' + upload_filename
            save_filepath = path.join(app.config['UPLOAD_FOLDER'], 'after_recognize.png'
            file.save(save_filepath)
            mnist_result = str(predict(save_filepath))

            insert_filename = '\'' + upload_filename + '\''
            insert_time = '\'' + str(req_time).rsplit('.', 1)[0] + '\''

            # insert data to the Cassandra
            # insertData(insert_filename, insert_time, mnist_result)

            # return the user with the information
            return ("%s%s%s%s%s%s%s%s%s" % ("Upload File Name ", upload_filename, "\n",
                                            "Upload Time: ", req_time, "\n",
                                            "Prediction: ", mnist_result, "\n"))
```

同时需要注意为了网页的模板要求我们还需要写入 html 配置文件：

```
return '''
<!doctype html>
<title>Upload new File</title>
<h1>Upload new File</h1>
<form action="" method=post enctype=multipart/form-data>
  <p><input type=file name=file>
     <input type=submit value=Upload>
</form>
'''
```

6. 可能错误出现与解决办法

（1）docker pull cassandra 过慢导致的无法下载的问题。

由于 docker pull 语句默认采用外网的 library 进行，在不采用 VPN 的情况下很难对其以及相关的包(如 cassandra.driver)进行下载。

解决方法：采用国内云镜像进行加速（如 http://f1361db2.m.daocloud.io）

进入 doceker 服务器命令行中输入

```
docker-machine ssh default
```

并执行如下命令启用镜像

sudo sed -i "s|EXTRA_ARGS='|EXTRA_ARGS='--registry-

mirror=http://f1361db2.m.daocloud.io |g" /var/lib/boot2docker/profile

然后重启 Docker 即可

（2）启用 Flask 时发现有网页错误无法显示。

解决方法：在应用 flask 的程序中应该将 methods 包含‘GET’和‘POST’两种方法，而不应该只有 post，如下图所示

```
@app.route('/mnist', methods=['GET','POST'])
```

（3）文件名称错误无法保存：

解决方法：这是因为文件名中不能包含冒号，而使用 datetime.now()所记录的时间当中又刚好包含有冒号，因此使用 file.save()函数保存用户上传的文件到本地的时候最好不要用时间命名。

7. 项目心得

本次大数据远程科研项目在充实我知识的同时，也拓宽了我在计算机系统与数据统计领域的视野，使得我对于将来自身的职业规划有了更好的展望与更远大的目标。初报名参加该项目时我怀着忐忑不安的性情，因为虽然我大学本科的专业是 ECE（电子与计算机工程），但是我对 Python 语言和大数据分析工具一窍不通，担心老师讲课节奏过快导致我听不懂。但张帆老师的授课完全打消了我的顾虑。即使是作为 MIT 的博士生，即使是大数据领域的大咖，张老师上课时仍完全跟着学生的节奏，积极询问和解答学生们的问题，极力避免大家出现“一知半解，不懂装懂”的情况。这种细致的讲课方式使我对该项目的态度由原先的迷茫胆怯变为了听课时的细致与提问时的大胆，同时从零开始学习的我也逐渐领略到了大数据的趣味性和重要性：大数据与我们的生活息息相关，在信息化时代的当今社会大数据技术几乎可以用在任何领域，但大数据技术所需要的工具又不是一朝一夕能够掌握的，需要我们脚踏实地，努力认真地去学习。

而最后自己独立完成项目的过程，也是一段让我难以遗忘地经历。由于自己之前从未独立完成过科研项目，因此刚开始构思项目大作业时地我是不知所措的，我反反复复观看张老师上课的录像，发现我能获得的只有一个大致的框架，基本上所有的具体信息都没有给到，因此上网搜集资料便耗费了我大量时间和精力。而在这之中又碰到了许多问题，例如：运用网上代码时出现的 bug，将两端程序连接在一起时所遇到的 bug，从网上样板程序中学习算法并应用时遇到的 bug。解决这些 bug 又是十分让人头疼的事情，这导致了我的效率低下。但我自始至终并没有放弃。最终虽然因在 cassandra 上遇到的问题无法解决而导致提交的项目作业并不完整，但对第一次单独进行科研的我来说，我已经给自己交了一份还算不错的答卷，因为我从未停止过思考。而这都要归功于张帆老师的远程授课给我带来的对大数据分析的兴趣。

8. 引用文献

[1] https://github.com/tensorflow/tensorflow/blob/r1.4/tensorflow/examples/tutorials/mnist/mnist_deep.py

[2] https://blog.csdn.net/tjsahwj/article/details/88181779

[3] http://www.tensorfly.cn/tfdoc/tutorials/mnist_pros.html

[4] https://baike.baidu.com/item/Docker/13344470?fr=aladdin

[5] https://blog.csdn.net/xjrhzq2008/article/details/81561117

[6] https://www.cnblogs.com/qiuhlee/p/9866222.html

[7] https://github.com/tensorflow/tensorflow/blob/r1.4/tensorflow/examples/tutorials/mnist/mnist_deep.py