远程科研总结报告

——基于 TensorFlow 的手写体识别

罗皓

2019.5.31

目录

一、背景介绍

1. 大数据

大数据（Big Data），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。

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。阿里巴巴创办人马云来台演讲中就提到，未来的时代将不是 IT 时代，而是 DT 的时代，DT 就是 Data Technology 数据科技，显示大数据对于阿里巴巴集团来说举足轻重。

有人把数据比喻为蕴藏能量的煤矿。煤炭按照性质有焦煤、无烟煤、肥煤、贫煤等分类，而露天煤矿、深山煤矿的挖掘成本又不一样。与此类似，大数据并不在“大”，而在于“有用”。价值含量、挖掘成本比数量更为重要。对于很多行业而言，如何利用这些大规模数据是赢得竞争的关键

2. 机器学习

机器学习(Machine Learning, ML)是一门多领域交叉学科，涉及概率论、统计学、逼近论、凸分析、算法复杂度理论等多门学科。专门研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构使之不断改善自身的性能。

它是人工智能的核心，是使计算机具有智能的根本途径，其应用遍及人工智能的各个领域，它主要使用归纳、综合而不是演绎。

3. TensorFlow

TensorFlow™是一个基于数据流编程（dataflow programming）的符号数学系统，被广泛应用于各类机器学习（machine learning）算法的编程实现，其前身是谷歌的神经网络算法库 DistBelief。

谷歌大脑自 2011 年成立起开展了面向科学研究和谷歌产品开发的大规模深度学习应用研究，其早期工作即是 TensorFlow 的前身 DistBelief。DistBelief 的功能是构建各尺度下的神经网络分布式学习和交互系统，也被称为“第一代机器学习系统”。DistBelief 在谷歌和 Alphabet 旗下其它公司的产品开发中被改进和广泛使用。2015 年 11 月，在 DistBelief 的基础上，谷歌大脑完成了对“第二代机器学习系统”TensorFlow 的开发并对代码开源。相比于前作，TensorFlow 在性能上有显著改进、构架灵活性和可移植性也得到增强。此后 TensorFlow 快速发展，截至稳定 API 版本 1.12，已拥有包含各类开发和研究项目的完整生态系统。

4. Docker

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。在本次项目中，我们将应用打包，部

署进容器中对其操作。

5. Python

Python 是一个高层次的结合了解释性、编译性、互动性和面向对象的脚本语言。它的设计具有很强的可读性，相比其他语言经常使用英文关键字，其他语言的一些标点符号，它具有比其他语言更有特色语法结构。

Python 社区提供了大量的第三方模块，使用方式与标准库类似。它们的功能无所不包，覆盖科学计算、Web 开发、数据库接口、图形系统多个领域，并且大多成熟而稳定。第三方模块可以使用 Python 或者 C 语言编写。SWIG,SIP 常用于将 C 语言编写的程序库转化为 Python 模块。Boost C++ Libraries 包含了一组库，Boost.Python，使得以 Python 或 C++ 编写的程序能互相调用。借助于拥有基于标准库的大量工具、能够使用低级语言如 C 和可以作为其他库接口的 C++，Python 已成为一种强大的应用于其他语言与工具之间的胶水语言。

6. Cassandra

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。Cassandra 最初由 Facebook 开发，后转变成了开源项目。Cassandra 的主要特点就是它不是一个数据库，而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。

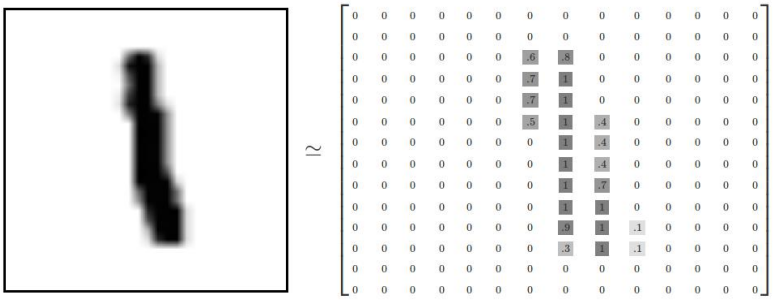
二、项目过程

项目目标：将 MNIST 应用部署到容器里面，用户通过 curl -X POST 命令提交带有手写体的数字图片，你的程序先将本图片识别出来，然后将识别的数字返回给用户。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。

1. Minst 数据集的使用

基于 MNIST 数据集实现手写字识别可谓是深度学习经典入门必会的技能，下载下来的数据集被分成两部分：60000 行的训练数据集（mnist.train）和 10000 行的测试数据集（mnist.test），在机器学习模型设计时必须有一个单独的测试数据集不用于训练而是用来评估这个模型的性能，从而更加容易把设计的模型推广到其他数据集上（泛化）。

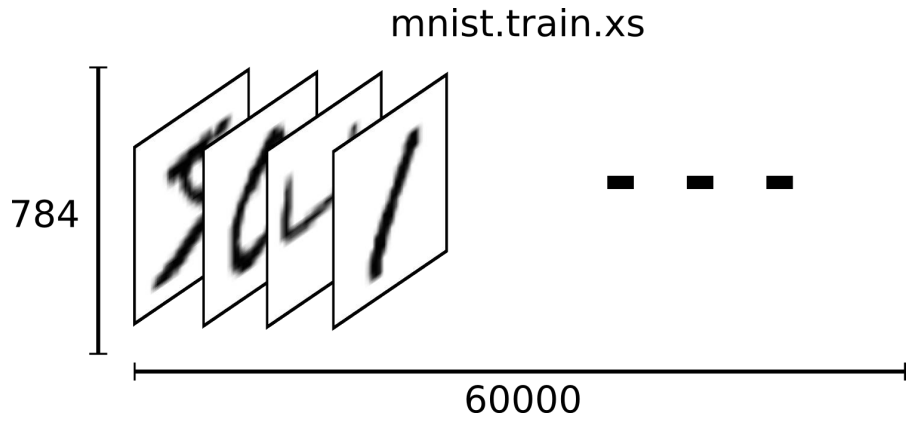
正如前面提到的一样，每一个 MNIST 数据单元有两部分组成：一张包含手写数字的图片和一个对应的标签。我们把这些图片设为“xs”，把这些标签设为“ys”。训练数据集和测试数据集都包含 xs 和 ys，比如训练数据集的图片是 mnist.train.images ，训练数据集的标签是 mnist.train.labels。每张均为 28*28 像素的黑白图片。



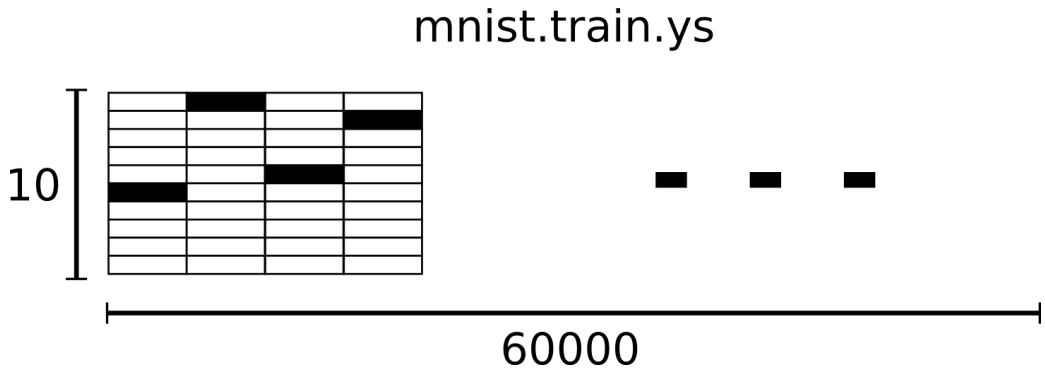
我们把这个数组展开成一个向量，长度是 28x28 = 784。如何展开这个数组（数字间的顺序）不重要，只要保持各个图片采用相同的方式展开。从这个角度来看，MNIST 数据集的图片就是在 784 维向量空间里面的点, 并且拥有比较复杂的结构 (提醒: 此类数据的可视化是计算密集型的)。

展平图片的数字数组会丢失图片的二维结构信息。这显然是不理想的，最优秀的计算机视觉方法会挖掘并利用这些结构信息，我们会在后续教程中介绍。但是在这个教程中我们忽略这些结构，所介绍的简单数学模型，softmax 回归(softmax regression)，不会利用这些结构信息。

因此，在 MNIST 训练数据集中，mnist.train.images 是一个形状为 [60000, 784] 的张量，第一个维度数字用来索引图片，第二个维度数字用来索引每张图片中的像素点。在此张量里的每一个元素，都表示某张图片里的某个像素的强度值，值介于 0 和 1 之间



相对应的 MNIST 数据集的标签是介于 0 到 9 的数字，用来描述给定图片里表示的数字。为了用于这个教程，我们使标签数据是"one-hot vectors"。 一个 one-hot 向量除了某一位的数字是 1 以外其余各维度数字都是 0。所以在此教程中，数字 n 将表示成一个只有在第 n 维度（从 0 开始）数字为 1 的 10 维向量。比如，标签 0 将表示成([1,0,0,0,0,0,0,0,0,0])。因此， mnist.train.labels 是一个 [60000, 10] 的数字矩阵。



2. Softmax 回归模型的实现

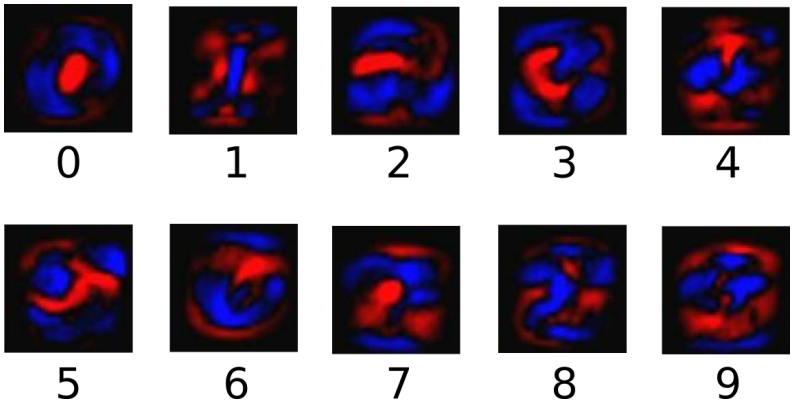
MNIST 的每一张图片都表示一个数字，从 0 到 9。我们希望得到给定图片代表每个数字的概率。比如说，我们的模型可能推测一张包含 9 的图片代表数字 9 的概率是 80%但是判断它是 8 的概率是 5%（因为 8 和 9 都有上半部分的小圆），然后给予它代表其他数字的概率更小的值。

这是一个使用 softmax 回归（softmax regression）模型的经典案例。softmax 模型可以用来给不同的对象分配概率。即使在之后，我们训练更加精细的模型时，最后一步也需要用 softmax 来分配概率。

softmax 回归（softmax regression）分两步：第一步

为了得到一张给定图片属于某个特定数字类的证据（evidence），我们对图片像素值进行加权求和。如果这个像素具有很强的证据说明这张图片不属于该类，那么相应的权值为负数，相反如果这个像素拥有有利的证据支持这张图片属于这个类，那么权值是正数。

下面的图片显示了一个模型学习到的图片上每个像素对于特定数字类的权值。红色代表负数权值，蓝色代表正数权值。



我们也需要加入一个额外的偏置量（bias），因为输入往往会带有一些无关的干扰量。因此对于给定的输入图片 x 它代表的是数字 i 的证据可以表示为

$$\text{evidence}_i = \sum_j W_{i,\,j} x_j + b_i$$

其中 W_i 代表权重，b_i 代表数字 i 类的偏置量，j 代表给定图片 x 的像素索引用于像素求和。然后用 softmax 函数可以把这些证据转换成概率 y：

$$y = \text{softmax}(\text{evidence})$$

这里的 softmax 可以看成是一个激励（activation）函数或者链接（link）函数，把我们定义的线性函数的输出转换成我们想要的格式，也就是关于 10 个数字类的概率分布。因此，给定一张图片，它对于每一个数字的吻合度可以被 softmax 函数转换成为一个概率值。softmax 函数可以定义为：

$$\text{softmax}(x) = \text{normalize}(\exp(x))$$

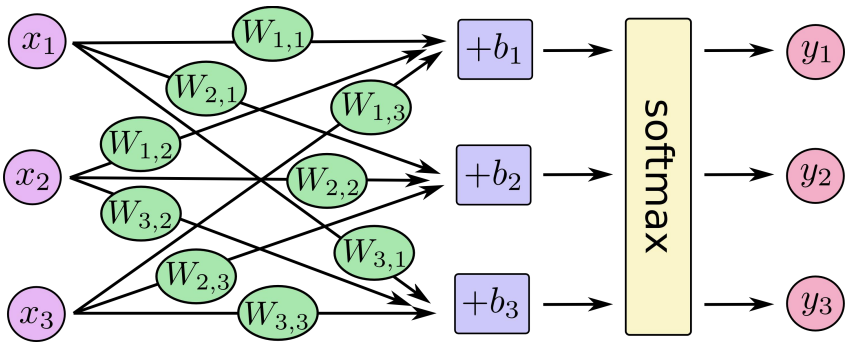
展开等式右边的子式，可以得到：

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

但是更多的时候把 softmax 模型函数定义为前一种形式：把输入值当成幂指数求值，再正则化这些结果值。这个幂运算表示，更大的证据对应更大的假设模型（hypothesis）里面

的乘数权重值。反之，拥有更少的证据意味着在假设模型里面拥有更小的乘数系数。假设模型里的权值不可以是 0 值或者负值。Softmax 然后会正则化这些权重值，使它们的总和等于 1，以此构造一个有效的概率分布。（更多的关于 Softmax 函数的信息，可以参考 Michael Nieslen 的书里面的这个部分，其中有关于 softmax 的可交互式的可视化解释。）

对于 softmax 回归模型可以用下面的图解释，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：



如果把它写成一个等式，我们可以得到：

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{matrix} W_{1,1}x_1 + W_{1,2}x_2 + W_{1,3}x_3 + b_1 \\ W_{2,1}x_1 + W_{2,2}x_2 + W_{2,3}x_3 + b_2 \\ W_{3,1}x_1 + W_{3,2}x_2 + W_{3,3}x_3 + b_3 \end{matrix} \right)$$

我们也可以用向量表示这个计算过程：用矩阵乘法和向量相加。这有助于提高计算效率。（也是一种更有效的思考方式）

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

更进一步，可以写成更加紧凑的方式：

$$y = \text{softmax}(Wx + b)$$

在实现回归模型时，TensorFlow 定义了两个个占位符（placeholder），和四个变量：（前提是先 import tensorflow as tf）

X = tf.placeholder (tf.float32, shape = [None, input_size]) （这里的 input_size = 784）

Y = tf.placeholder (tf.float32, shape = [None, num_classes])

W1 = tf.Variable (tf.random_normal ([input_size, hidden_units_size], stddev = 0.1))#hidden_units_size = 30#正态分布随机数

B1 = tf.Variable (tf.constant (0.1),[hidden_units_size])#常数为 1，形状为（1,1）

W2 = tf.Variable (tf.random_normal ([hidden_units_size, num_classes], stddev = 0.1))#正态分布随机数

B2 = tf.Variable (tf.constant (0.1), [num_classes])

我们在 TensorFlow 运行计算时输入这个值。我们希望能够输入任意数量的 MNIST 图像，每一张图展平成 784 维的向量。我们用 2 维的浮点数张量来表示这些图，这个张量的形状是[None，input_size]。（None 表示此张量的第一个维度可以是任何长度的。）我们的模型也需要权重值和偏置量，当然我们可以把它们当做是另外的输入（使用占位符），但 TensorFlow 有一个更好的方法来表示它们：Variable 。 一个 Variable 代表一个可修改的张量，存在在 TensorFlow 的用于描述交互性操作的图中。它们可以用于计算输入值，也可以在计算中被修改。对于各种机器学习应用，一般都会有模型参数，可以用 Variable 表示。我们赋予 tf.Variable 不同的初值来创建不同的 Variable：在这里，我们都用全为零的张量来初始化 W 和 b。因为我们要学习 W 和 b 的值，它们的初值可以随意设置。注意，W 的维度是[784，10]，因为我们想要用 784 维的图片向量乘以它以得到一个 10 维的证据值向量，每一位对应不同数字类。b 的形状是[10]，所以我们可以直接把它加到输出上面。现在，我们只需要一行代码就可实现我们的模型：

tf.nn.softmax_cross_entropy_with_logits (labels = Y, logits = final_opt)

3. 训练模型

为了训练我们的模型，我们首先需要定义一个指标来评估这个模型是好的。其实，在机器学习，我们通常定义指标来表示一个模型是坏的，这个指标称为成本（cost）或损失（loss），然后尽量最小化这个指标。但是，这两种方式是相同的。

一个非常常见的，非常漂亮的成本函数是“交叉熵”（cross-entropy）。交叉熵产生于信息论里面的信息压缩编码技术，但是它后来演变成为从博弈论到机器学习等其他领域里的重要技术手段。它的定义如下：

$$H_{y'}(y) = -\sum_i y'_i \log(y_i)$$

y 是我们预测的概率分布, y' 是实际的分布（我们输入的 one-hot vector)。比较粗糙的理解是，交叉熵是用来衡量我们的预测用于描述真相的低效性。更详细的关于交叉熵的解释超出本教程的范畴，但是你很有必要好好反向传播算法(backpropagation algorithm)来有效地确定你的变量是如何影响你想要最小化的那个成本值的。然后，TensorFlow 会用你选择的优化算法来不断地修改变量以降低成本。

在训练时，我们会首先定义一个初始化的变量，并且在 session 里面启动它，然后让模型训练 50000 次，每 10000 次都会有一个准确度，而这个准确度都是越来越接近 1 的。

4. 实现识别并将其可视化

在上一步的训练模型中，我们得到了一个 final_opt 的训练矩阵，实际上，在训练的过程中识别到的数字可以提取出来，并且将其可视化，可以观察结果是否正确。在训练集中的图片，先将其转换成 28✕28 的灰度图像，然后将其投入训练模型中，得到最终的识别结果，返回至一个数组中，将其遍历出来。同时通过 imshow 和 pyshow 函数将其图像输出出来，进行观察。

5. 存储识别结果

通过模型识别，我们得到了 final_opt_a[i]，这样一个数组，里面存放了训练集中的图片的数字识别结果，从第一张开始一直到我们设定的图片数量上限。我们准备将其识别结果，时间戳存放在数据库中，在此我们使用了非关系型数据库 Cassandra。

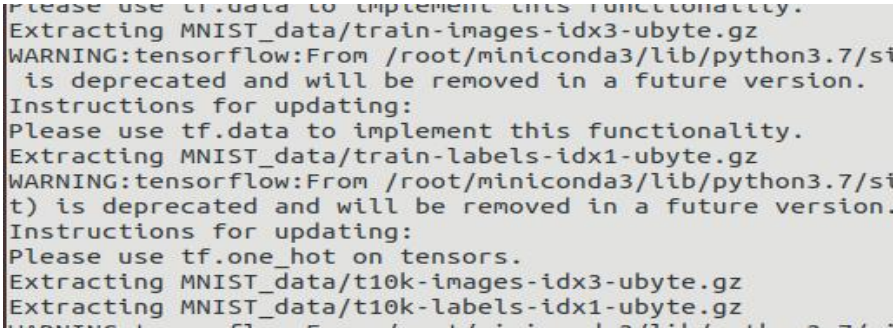
首先，我们在 Python 代码中先于 Cassandra 数据库相连接，在此之前我们需要安装后 Cassandra 和配置和 Cassandra-driver，同时在 Python 代码中 import 进来 Cassandra 包，并且为其设置端口 cluster = Cluster(contact_points=['127.0.0.1'],port=9042)，方便连接。然后通过先创建命名空间、表，来定义好我们的存储位置，最后直接使用 session.execute()函数，在括号中输入 Cassandra 的插入语句，即可完成对数据的添加。在进行数据查看时，也同样运用 session.execute()函数，通过 cqlsh SELECT 语句即可实现。

List：在加入数据库之前，我们先在 Python 中定义了一个列表 LIST，用于存储处理好的数据，当做一个中介。列表也能够很方便的像 Cassandra 的 table 一样存放各个属性的数据。

三、项目结果

1. 程序运行截图

（1）从 Mnist 数据集中导入数据



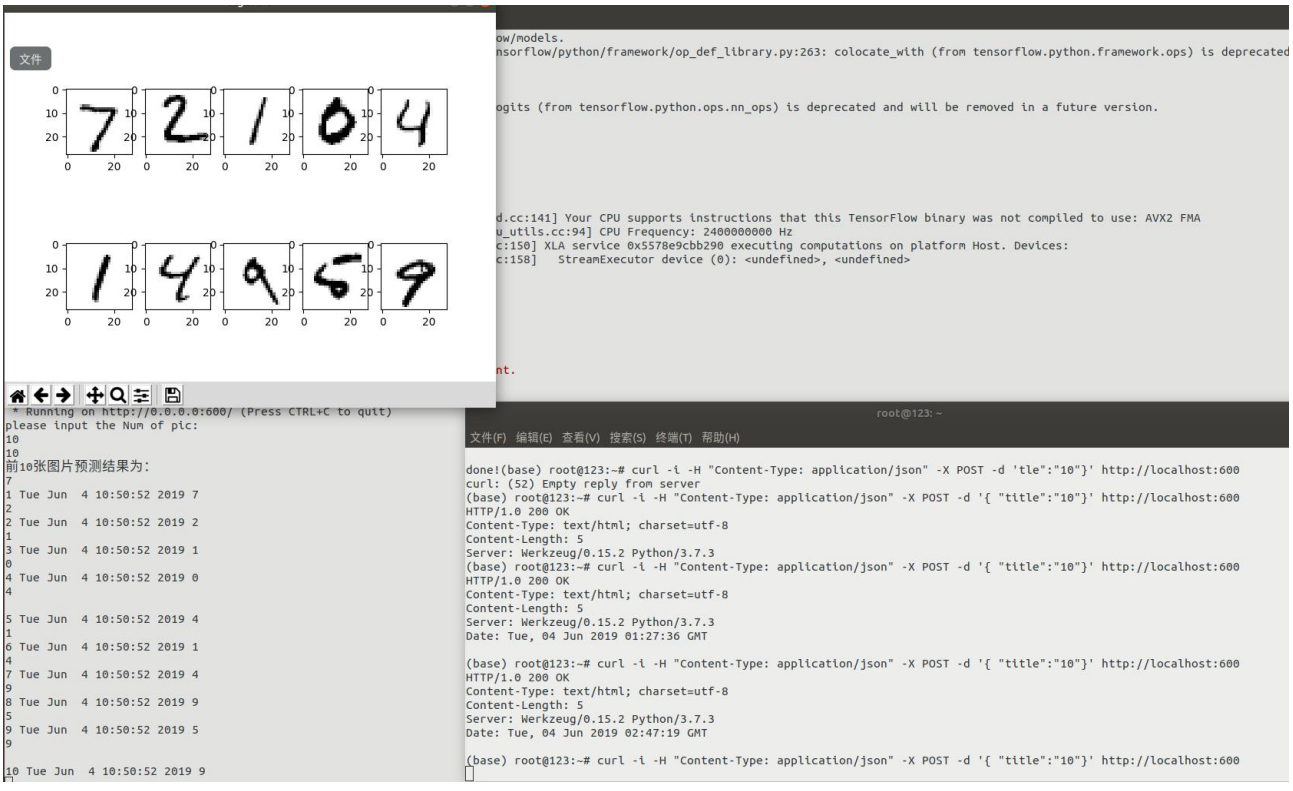
（2）运行程序，开始训练模型

```
step : 10000, training accuracy = 0.98
step : 20000, training accuracy = 0.98
step : 30000, training accuracy = 0.99
step : 40000, training accuracy = 1
step : 50000, training accuracy = 1
```

（3）训练完毕，通过 curl -X POST 命令指定识别的图片数量

```
(base) root@123:~# curl -i -H "Content-Type: application/json" -X POST -d '{ "title":"10"}' http://localhost:600
```

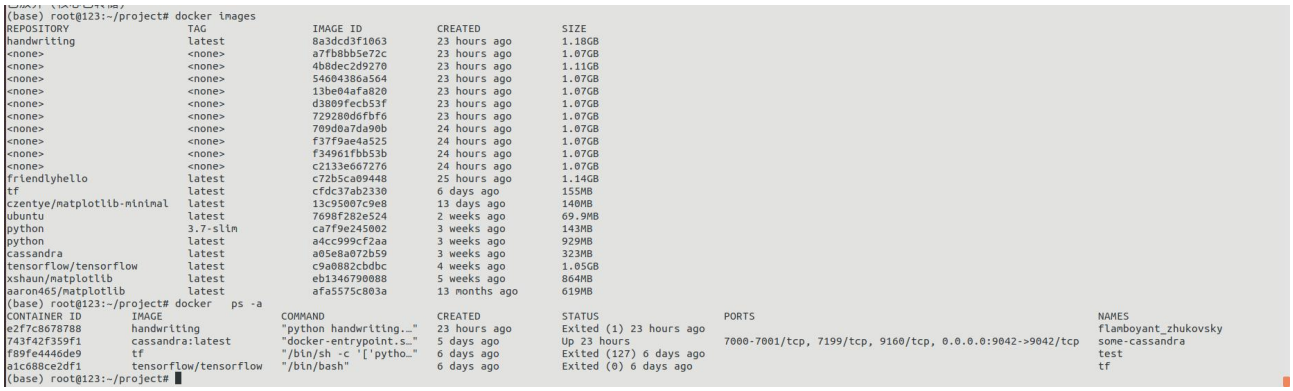
（4）输出识别结果及所指定的图片



（5）存入 Cassandra 数据库，并且遍历出来

```
OrderedDict([('number', <cassandra.metadata.ColumnMetadata
 object at 0x7f630c3e5940>), ('result', <cassandra.metadat
a.ColumnMetadata object at 0x7f630c3e52e8>), ('time', <cas
sandra.metadata.ColumnMetadata object at 0x7f630c3e5978>)]
)
Row(number=5, result=4, time='Tue Jun  4 10:50:52 2019')
Row(number=10, result=9, time='Tue Jun  4 10:50:52 2019')
Row(number=1, result=7, time='Tue Jun  4 10:50:52 2019')
Row(number=8, result=9, time='Tue Jun  4 10:50:52 2019')
Row(number=2, result=2, time='Tue Jun  4 10:50:52 2019')
Row(number=4, result=0, time='Tue Jun  4 10:50:52 2019')
Row(number=7, result=4, time='Tue Jun  4 10:50:52 2019')
Row(number=6, result=1, time='Tue Jun  4 10:50:52 2019')
Row(number=9, result=5, time='Tue Jun  4 10:50:52 2019')
Row(number=3, result=1, time='Tue Jun  4 10:50:52 2019')
[7, 2, 1, 0, 4, 1, 4, 9, 5, 9]
测试集前10张图片为:
the result is:
[7, 2, 1, 0, 4, 1, 4, 9, 5, 9]
127.0.0.1 - - [04/Jun/2019 10:51:38] "POST / HTTP/1.1" 200
 -
```

2. 项目中所涉及的 images 和 container

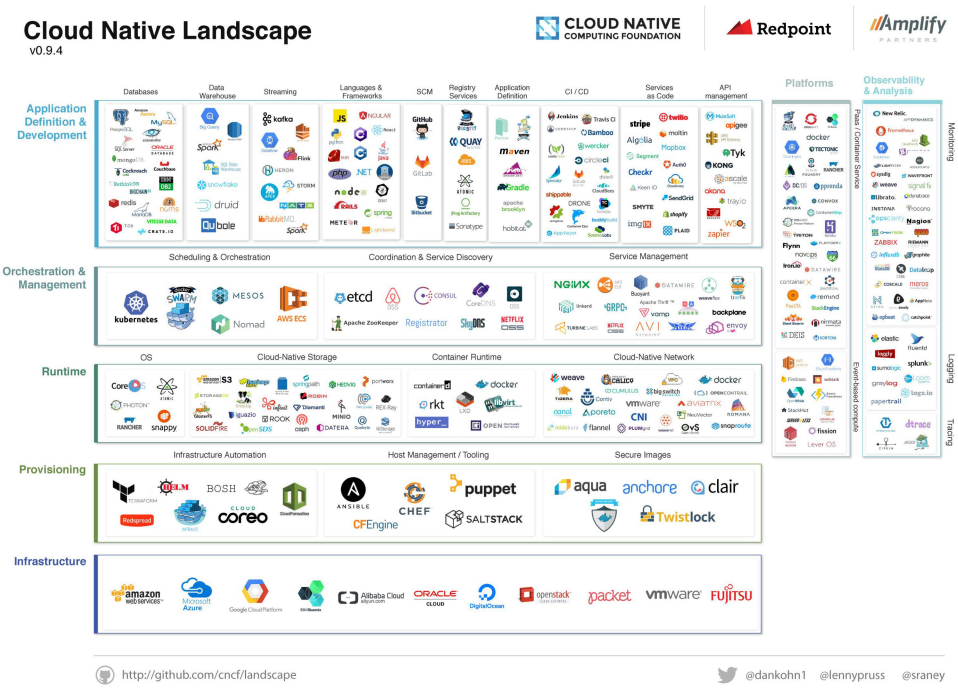


```
(base) root@123:~/project# docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
handwriting                 latest              8a3dcd3f1063        23 hours ago        1.18GB
<none>                      <none>              a7fb8bb5e72c        23 hours ago        1.07GB
<none>                      <none>              4b8dec2d9270        23 hours ago        1.11GB
<none>                      <none>              54604386a564        23 hours ago        1.07GB
<none>                      <none>              13be04afa820        23 hours ago        1.07GB
<none>                      <none>              d3809fecb53f        23 hours ago        1.07GB
<none>                      <none>              729280d6fbf6        23 hours ago        1.07GB
<none>                      <none>              709d0a7da90b        24 hours ago        1.07GB
<none>                      <none>              f37f9ae4a525        24 hours ago        1.07GB
<none>                      <none>              f34961fbb53b        24 hours ago        1.07GB
<none>                      <none>              c2133e667276        24 hours ago        1.07GB
friendlyhello               latest              c72b5ca09448        25 hours ago        1.14GB
tf                          latest              cfdc37ab2330        6 days ago          155MB
czentye/matplotlib-minimal  latest              13c95007c9e8        13 days ago         140MB
ubuntu                      latest              7698f282e524        2 weeks ago         69.9MB
python                      3.7-slim            ca7f9e245002        3 weeks ago         143MB
python                      latest              a4cc999cf2aa        3 weeks ago         929MB
cassandra                   latest              a05e8a072b59        3 weeks ago         323MB
tensorflow/tensorflow       latest              c9a0882cbdbc        4 weeks ago         1.05GB
xshaun/matplotlib           latest              eb1346790088        5 weeks ago         864MB
aaron465/matplotlib         latest              afa5575c803a        13 months ago       619MB
(base) root@123:~/project# docker   ps -a
CONTAINER ID        IMAGE                  COMMAND                  CREATED             STATUS                      PORTS                                              NAMES
e2f7c8678788        handwriting            "python handwriting.…"   23 hours ago        Exited (1) 23 hours ago                                                        flamboyant_zhukovsky
743f42f359f1        cassandra:latest       "docker-entrypoint.s…"   5 days ago          Up 23 hours                 7000-7001/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:9042->9042/tcp   some-cassandra
f89fe4446de9        tf                     "/bin/sh -c '['pytho…"   6 days ago          Exited (127) 6 days ago                                                        test
a1c688ce2df1        tensorflow/tensorflow  "/bin/bash"              6 days ago          Exited (0) 6 days ago                                                          tf
(base) root@123:~/project#
```

四、心得体会

在本次项目中，张老师给我们上了四次课，涉及到的知识层面非常广，也有很多是我从来没有接触过的，比如说 Linux 系统，Linux 系统这门课我们这学期才开始上，平时都是用虚拟机来使用，老师教我们安装乌班图双系统，在 Linux 环境下完成我们的项目，而对于 Linux 系统的操作使用，我是很陌生的，大部分的代码都是通过命令行来实现，许多的软件都通过 pip 来安装，这在 windows 下都是很少接触的。刚接触 Linux 的时候也是非常的不熟悉，也有很多低级的问题就想着向老师提问了。而关于 Python 语言，我也是第一次运用于实战，以前也听说过 Python 代码非常简洁明了，所以我也就很快上手了。最让我头疼的是 Docker 这个小鲸鱼，整个项目中大部分的问题都出在它身上。要么就是镜像 pull 不进去，要么就是端口没设置好，还有服务器、命令格式，尤其是-t -a -p 这些东西，基本每次使用都要调出 command help 来看一看是不是对的，老师也说了他现在的工作中，Docker 是必不可少的一部分，我觉得在我完成这个项目之后，也要好好的研究一下 Docker，对于我今后的不管是学习还是工作肯定都有很大的帮助。

这也是我第一次接触大数据方面的知识，以前也了解过人工智能、机器学习这方面的知识，国内这一部分也应该是刚刚起步，所以也需要花很多的时间学习这方面的知识。而对于老师最后一节课讲的 cloud native 云原生，我觉得这个东西包含的知识量非常的多，里面也有很多我们接触过的东西，我相信在之后的学习工作生活中，我也会不断地学习这个架构里面的各个工具，也算是走在时代的前沿吧。



最后，通过这次远程科研项目，我的知识面扩充了非常多，linux,python,github,cassandra,docker,flask,matplotlib,tensorflow,spark,pip,anaconda,http api,zoom 等等等等，总的来说，这是一次可以影响我一生的科研项目，我也希望在以后的生活中，遇到困难时能够想起这一次的项目，不断地利用身边的资源来解决问题，每一次的困难都是一个新的项目，通过这一次的学习，我相信我的问题解决能力已经很好地提升了，也要塑造一种不畏难的品格。

五、参考文献

1. http://www.w3school.com.cn/ w3cshool
2. https://blog.csdn.net/Finley1991/article/details/81151840 神经网络初识——MINIST 手写数字识别
3. https://docs.docker.com/get-started/ Docker tutorial
4. https://blog.miguelgrinberg.com/post/designing-a-restful-api-with-python-and-flask flask
5. https://blog.csdn.net/huaweizte123/article/details/79672479 深度学习 TENSORFLOW 入门
6. http://www.tensorfly.cn/ tensorflow 中文社区

....