Design of Handwritten Digit Recognition within Function of Record Storage in Cassandra

Author: Sichen Ma

Tutor: Fan Zhang

Date: 1/29/2019

Table of Contents

1. Introduction

Nowadays, people can use different kinds of applications, and a function of one of the e-dictionary like Youdao Dictionary is take picture or screenshot from a page with foreign language and translate it directly. The question of how the application could recognize those letters from a picture is the key of inspiration of this project. In this project, I suppose to start from basic letters, digits, and make a small application for recognizing handwritten digit then store all the records of recognition to NoSQL database, Cassandra.

To realize these functions, using TensorFlow for building a simple neural network model, training the network (feeding MNIST dataset) and then evaluating the result on the test set. The next step is using Flask to invoke the convolutional neural network model into a dynamic website meanwhile connecting to Cassandra through Docker Containers. Here, I also need to mention that the programming software I used is Python/PyCharm and the operation system is Linux/Ubuntu 18.04 in order to operate these applications smoothly.

2. Digit recognition model through TensorFlow

2.1 Preparing MNIST dataset

The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. This dataset is made up of images of handwritten digits, 28x28 pixels in size. Some sample images from MNIST test dataset is shown in figure 1.

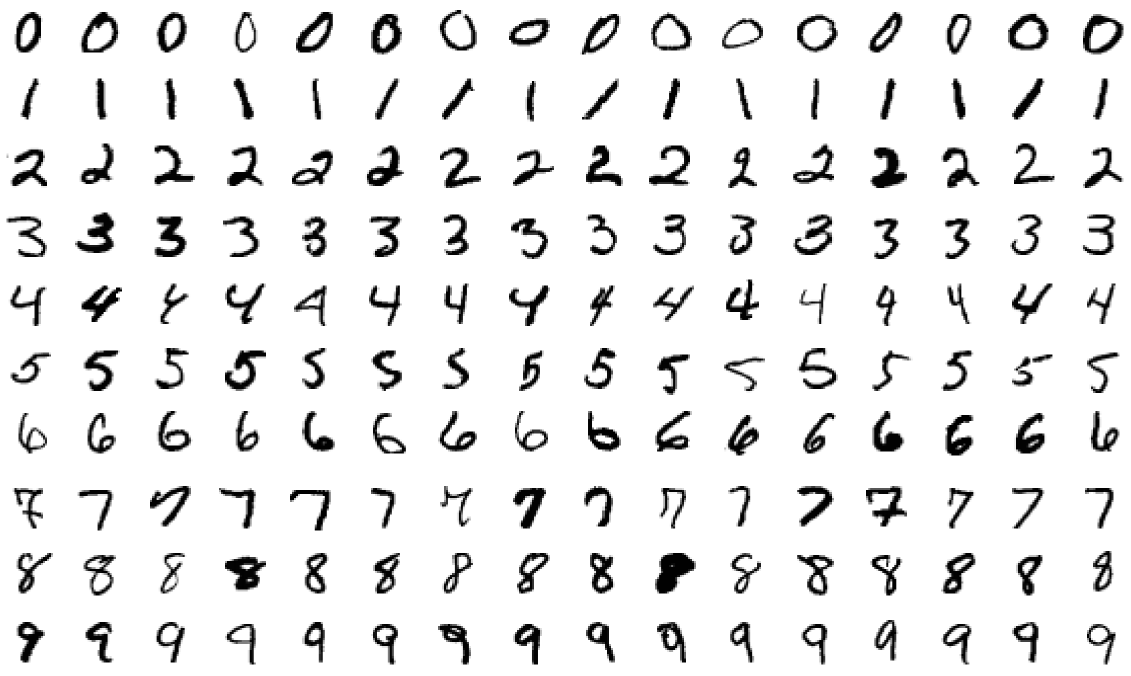
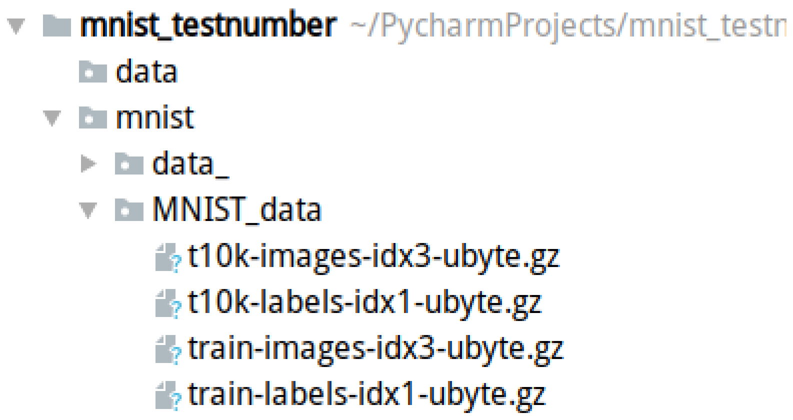


Figure 1. Sample images from MNIST

It is easy to import the MNIST dataset through TensorFlow from 'tensorflow.examples.tutorials.mnist'. After importing MNIST dataset, there would appear four folders in Python project, it has shown as following figure 2.



```
▼ mnist_testnumber ~/PycharmProjects/mnist_testn
    data
  ▼ mnist
    ▶ data_
    ▼ MNIST_data
        t10k-images-idx3-ubyte.gz
        t10k-labels-idx1-ubyte.gz
        train-images-idx3-ubyte.gz
        train-labels-idx1-ubyte.gz
```

Figure 2. Specific contents in MNIST dataset

It is obvious that there are handwritten digit images and corresponding labels under two parts: training and testing datasets.

2.2 Softmax regression model training and evaluation

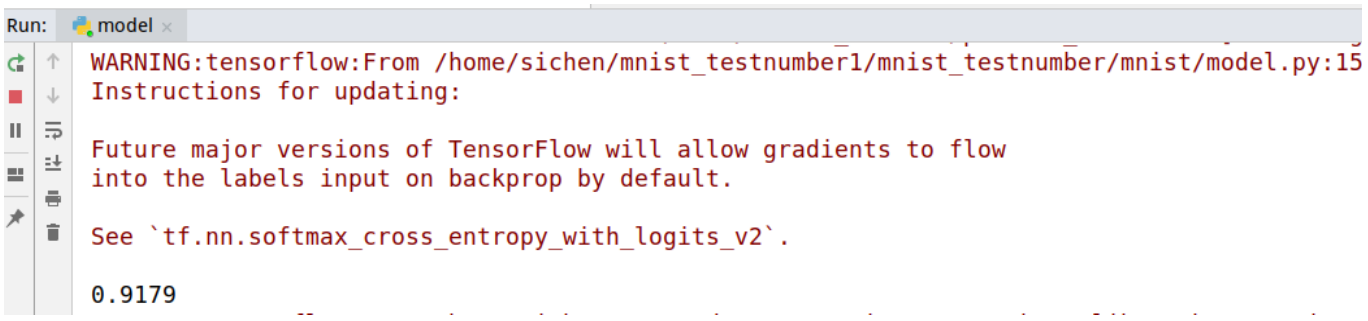
There are two kinds of architectures can be used for this project, I would choose the best one for the final recognizing process. One classic case is using a softmax regression model based on the weighted sum on the picture pixel values and add an extra bias for correcting. Therefore, the softmax model equation can be expressed by the following equation, $y = softmax(Wx + b)$. Initially, set the variables and parameters in this equation, then it can be achieved just by one line of code in python language. I list the codes for softmax model as following.

```
sess = tf.InteractiveSession()
x = tf.placeholder(tf.float32, shape=[None, 784])
y_ = tf.placeholder(tf.float32, shape=[None, 10])
W = tf.Variable(tf.zeros([784,10]))
b = tf.Variable(tf.zeros([10]))
sess.run(tf.global_variables_initializer())
y = tf.matmul(x,W) + b
```

The next step would be training the model and evaluate it by the accuracy of prediction. Here needs to take some things attention. For the training data, I use a placeholder that will be fed at run time with a training batch. The training batch I set is 100, then there are 1000 steps for predicting. And finally get the prediction accuracy to see how the model performance. The code of this part still listed as following.

```
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_ent
ropy)
for i in range(1000):
 batch = data.train.next_batch(100)
 train_step.run(feed_dict={x: batch[0], y_: batch[1]})
correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print(accuracy.eval(feed_dict={x: data.test.images, y_:
data.test.labels}))
```

Running these codes and see the performance of this model. The output is shown in the following figure 3. Therefore, the prediction accuracy is 0.9179.



```
WARNING:tensorflow:From /home/sichen/mnist_testnumber1/mnist_testnumber/mnist/model.py:15
Instructions for updating:

Future major versions of TensorFlow will allow gradients to flow
into the labels input on backprop by default.

See `tf.nn.softmax_cross_entropy_with_logits_v2`.

0.9179
```

Figure 3. Output of the softmax model

2.3 Convolutional Neural Network (CNN) architecture training and evaluation

The other architecture of this project is building a network with two convolutional layers, followed by one fully connected layer. The easy organization is shown in following table 1.

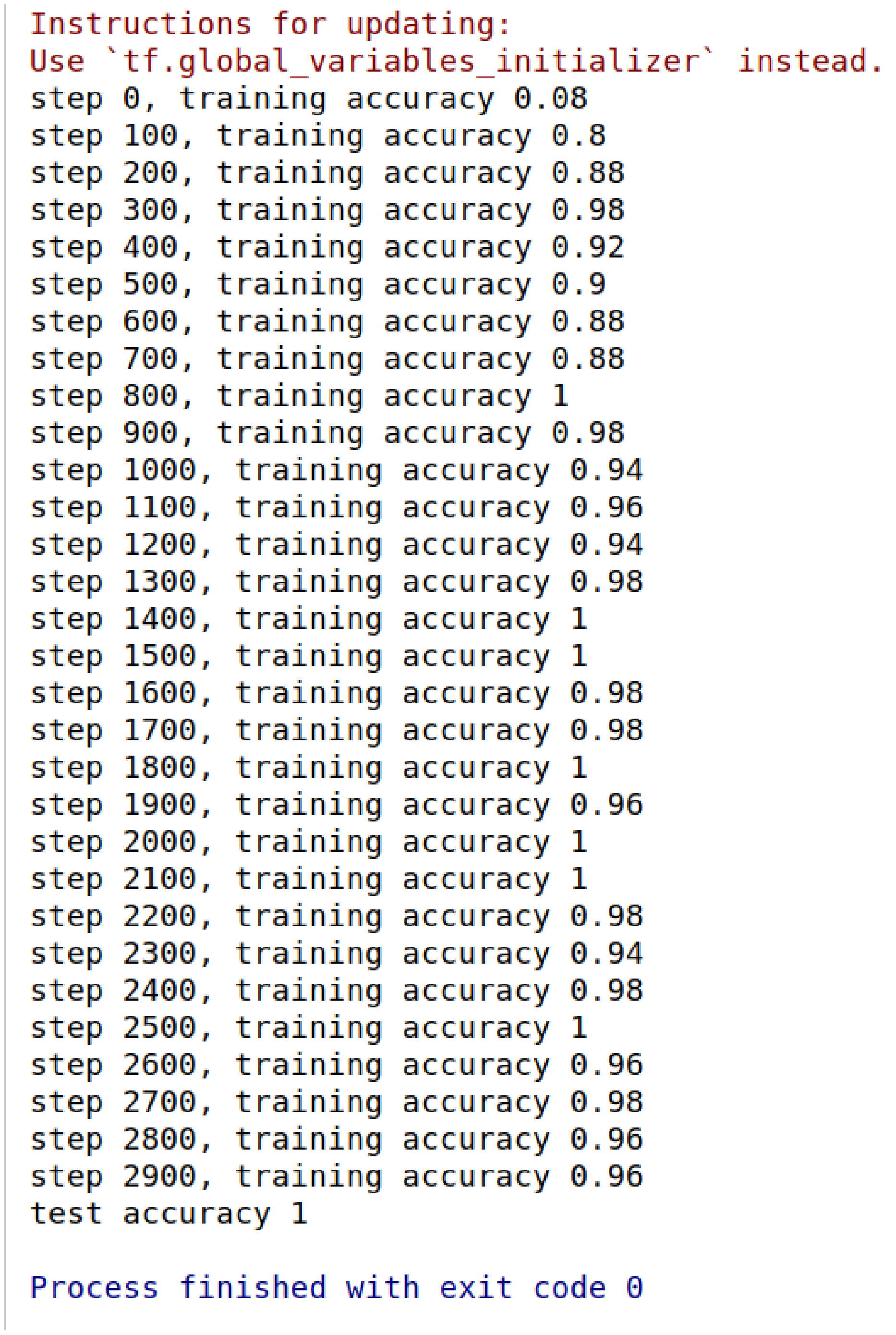
Table 1. Simple organization for CNN architecture

Convolutional Layer 1 + Max Pooling
Convolutional Layer 2 + Max Pooling
Fully Connected Layer 1 + Dropout
Fully Connected Layer 2 to Prediction

Besides the basic structure of this architecture, I need to start off with defining the training iterations and the batch size. Training iterations, also called training steps, indicate the number of times for training the network. The batch size means that how many parts of training images will be divided in a fixed batch size and at every batch it will take a fixed number of images and train them. In this model, I set train iterations as 20000 steps, and batch size as 50 for greater accuracy. I would list codes for training and evaluation parts as following.

```
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"))
for i in range(20000):
 batch = data.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})
```

Getting the output of these codes take time because of the large number of iterations and the final predicted accuracy is 1. Here, for showing the process of prediction more completely, I shorten the steps of prediction. The prediction accuracy is shown in the following figure 4. The accuracy is still 1.



```
Instructions for updating:
Use `tf.global_variables_initializer` instead.
step 0, training accuracy 0.08
step 100, training accuracy 0.8
step 200, training accuracy 0.88
step 300, training accuracy 0.98
step 400, training accuracy 0.92
step 500, training accuracy 0.9
step 600, training accuracy 0.88
step 700, training accuracy 0.88
step 800, training accuracy 1
step 900, training accuracy 0.98
step 1000, training accuracy 0.94
step 1100, training accuracy 0.96
step 1200, training accuracy 0.94
step 1300, training accuracy 0.98
step 1400, training accuracy 1
step 1500, training accuracy 1
step 1600, training accuracy 0.98
step 1700, training accuracy 0.98
step 1800, training accuracy 1
step 1900, training accuracy 0.96
step 2000, training accuracy 1
step 2100, training accuracy 1
step 2200, training accuracy 0.98
step 2300, training accuracy 0.94
step 2400, training accuracy 0.98
step 2500, training accuracy 1
step 2600, training accuracy 0.96
step 2700, training accuracy 0.98
step 2800, training accuracy 0.96
step 2900, training accuracy 0.96
test accuracy 1

Process finished with exit code 0
```

Figure 4. Output of the CNN architecture model

2.4 Summary

Comparing these two kinds of architectures, the softmax model is easier than CNN architecture with just a regression relationship. That is also why the predicted accuracy of softmax model worse than CNN model (based on comparison of figure 3 and 4). Therefore, I finally use CNN architecture for digit-recognition. The following picture shows the whole process of the training model.

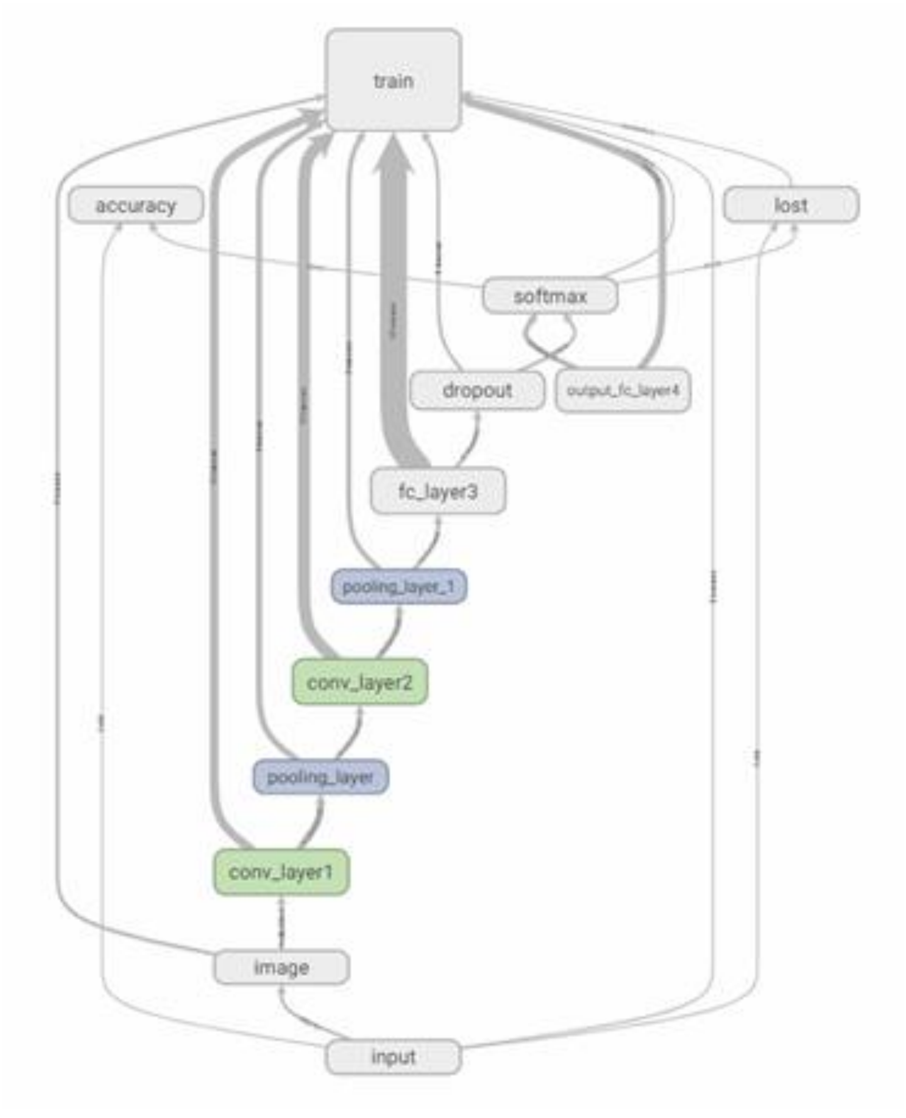


Figure 5. The model for the whole training model

3. Using flask to achieve recognition

In this project, the way to recognize picture is by uploading an image from local files. Flask is a good tool for creating a web server, then it is strong enough for defining the function of uploading files easily. Before the coding, we need to prepare a couple of imports, such as Flask, request, werkzeug.utils etc. Here, the last import I list is for werkzeug.secure_filename to store the filename from client. Now, it's time for using 'POST' method to achieve the function in the following codes.

```python
from flask import Flask, request
from werkzeug.utils import secure_filename
app = Flask(__name__)
result = -1
imgname = ""
idnum = 0
@app.route('/upload', methods=['POST'])
def upload():
   global result
   global imgname
   global idnum
   f = request.files['file']
   img = f.read()
   result = Number_recognition(img)
   imgname = str(secure_filename(f.filename))
   idnum = idnum + 1
   return result
```

Based on these codes, we can totally understand the process of uploading files and how to get the result through "Number_recognition" function, which is the usage of CNN architecture model I defined before. However, just depending on these codes is impossible to run in a website because ports and the web development is missed here. Remember that, using flask to achieve uploading in a specific website needs a definition of flask and expose the ports for the website you want to use. The codes are easy but most important.

```
@app.route('/')
def index():
   return '''
   <!doctype html>
   <html>
   <body>
   <form action='/upload' method='post' enctype='multipart/form-
data'>
       <input type='file' name='file'>
   <input type='submit' value='Upload'>
   </form>
   '''
if __name__ == '__main__':
   app.debug = True
   app.run(host='0.0.0.0', port=8000)
```

If we run these codes successfully, we can enter the website called http://0.0.0.8000, to upload files and get the prediction result. I would not like to go into detail here because when we use docker for composing this website would be useless at that moment. I would explain more in that part.

4. Using Cassandra for storing recognition record

Now we have finished the application of digit recognition. The next important step and the most motivating part is the storage of all recognition records. People are all use different kinds of database for storing data like SQL database. However, here we use CQL database, Cassandra, as a tool to let us view the recognition situations after client submitting.

4.1 Cassandra preparations and connection

The programmer for coding I used is Python, thus we need to install Cassandra Driver as a requirement. After the preparations of Cassandra, the first thing to do is connecting to cassandra to receive your data and records. Here are the codes for connecting to cassandra:

```
log = logging.getLogger()
log.setLevel('INFO')
handler = logging.StreamHandler()
handler.setFormatter(logging.Formatter("%(asctime)s
[%(levelname)s] %(name)s: %(message)s"))
log.addHandler(handler)
KEYSPACE = "mykeyspace"
session = 0
def createKeySpace():
   cluster = Cluster(contact_points=['cassandra'], port=9042)
   global session
   session = cluster.connect()
   log.info("Creating keyspace...")
   try:
       session.execute("""
           CREATE KEYSPACE %s
           WITH   replication   =   {   'class':   'SimpleStrategy',
'replication_factor': '2' }
           """ % KEYSPACE)
       log.info("setting keyspace...")
```

```
        session.set_keyspace(KEYSPACE)
        log.info("creating table...")
        session.execute("""
              CREATE TABLE record (
                   id int,
                   imgname text,
                   time text,
                   result text,
                   PRIMARY KEY (id)
               )
               """)
    except Exception as e:
        log.error("Unable to create keyspace")
        log.error(e)
createKeySpace();
```

To connect to cassandra, we need to have a better understanding of cluster. Cluster is a set of contact points, there are many different nodes in a cluster. Creating cluster and keyspace with your own name in case of using the keyspace for inquiring cassandra table. Next step is creating the cassandra table for needy, in this project, we want to return the information of id, image names, recognition results and current time. After these setting, Cassandra should be connected successfully when running this python file.

4.2 Store data to my cassandra database

Cassandra table is ready, the next thing would be how to let the wanted data store to cassandra.

```
session.execute("""INSERT INTO """ + KEYSPACE + """.""" + """record
(id,imgname,time,result)
   VALUES ("""  +  str(idnum)  +  """,'"""  +  imgname  +  """','"""  +
str(datetime.datetime.now()) + """','""" + str(
   result) + """');
```

```
    """)
```

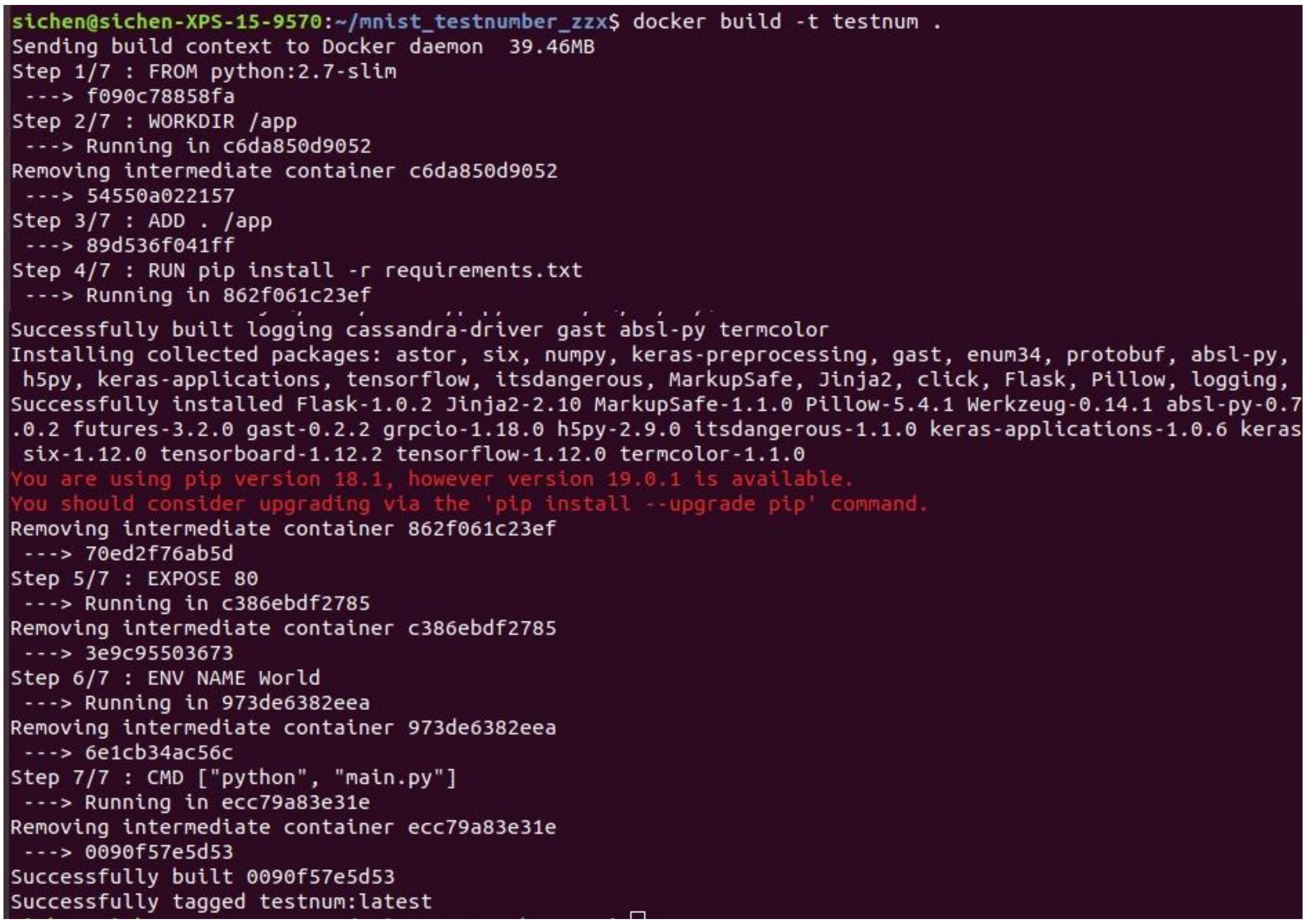
We need to get attention when we set these parameters. They are all strict to the form of them. For instance, image names are text forms and others are all strings which means real numbers. In addition, take care of the quotation marks in the lines, they should be on pairs and distinguished between single and double quotes. Finally, id in this table means people who enter that website for submitting, so id should be different every time and remember to let id add one every time, if not, the record would be always the latest submission.

5. Deploy the recognition application in Docker

Docker is one of the hottest open source projects that allows you to deploy your application inside containers, adding a layer of abstraction. Different package versions may cause different problems because of minor differences between environments. However, Docker would be a stable project to tight all the things in a container and develop more smoothly and convenient than without it. All in all, using docker is a great progress for programming and developing.

5.1 Define and build the image for digit recognition

Following the tutorial of docker, we need to use three basic files to create an image for our project. Firstly, dockerfile, which defines what goes on in the environment inside your container. Then, put 'requirements.txt' and our python file in the same folder with the dockerfile. Finally, use basic docker commands to create a Docker image for this project. I named this Docker image as 'testnum:latest'. I have shown process of building this image in following figure.

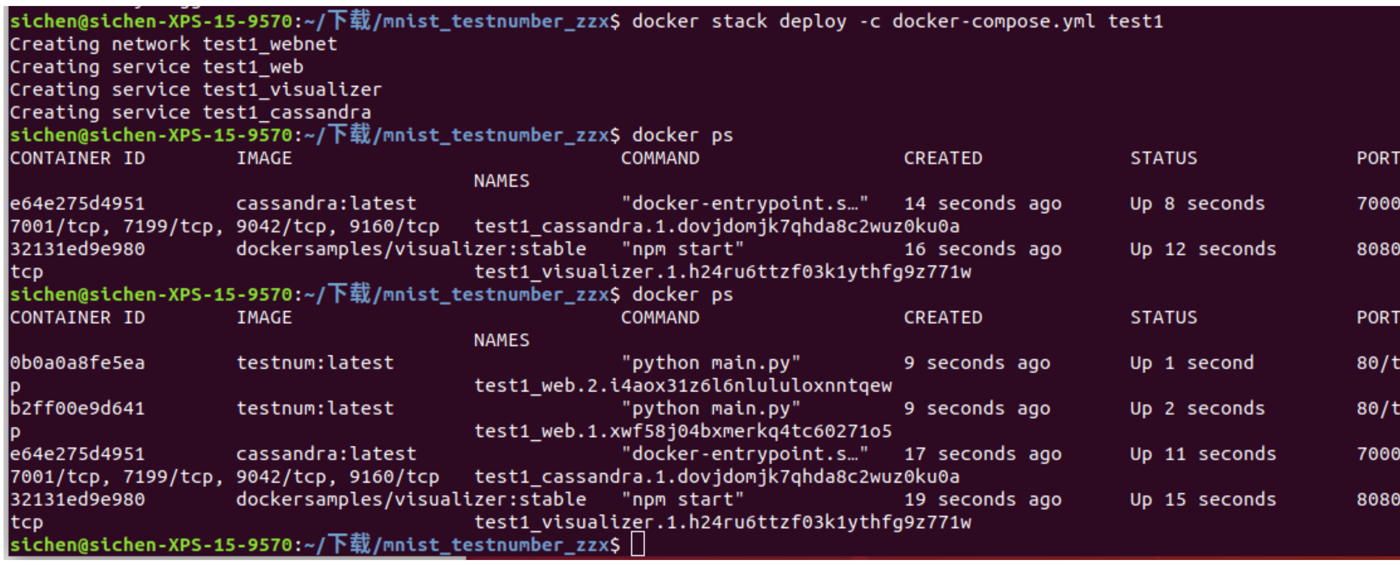


```
sichen@sichen-XPS-15-9570:~/mnist_testnumber_zzx$ docker build -t testnum .
Sending build context to Docker daemon  39.46MB
Step 1/7 : FROM python:2.7-slim
 ---> f090c78858fa
Step 2/7 : WORKDIR /app
 ---> Running in c6da850d9052
Removing intermediate container c6da850d9052
 ---> 54550a022157
Step 3/7 : ADD . /app
 ---> 89d536f041ff
Step 4/7 : RUN pip install -r requirements.txt
 ---> Running in 862f061c23ef
Successfully built logging cassandra-driver gast absl-py termcolor
Installing collected packages: astor, six, numpy, keras-preprocessing, gast, enum34, protobuf, absl-py,
 h5py, keras-applications, tensorflow, itsdangerous, MarkupSafe, Jinja2, click, Flask, Pillow, logging,
Successfully installed Flask-1.0.2 Jinja2-2.10 MarkupSafe-1.1.0 Pillow-5.4.1 Werkzeug-0.14.1 absl-py-0.7
.0.2 futures-3.2.0 gast-0.2.2 grpcio-1.18.0 h5py-2.9.0 itsdangerous-1.1.0 keras-applications-1.0.6 keras
 six-1.12.0 tensorboard-1.12.2 tensorflow-1.12.0 termcolor-1.1.0
You are using pip version 18.1, however version 19.0.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Removing intermediate container 862f061c23ef
 ---> 70ed2f76ab5d
Step 5/7 : EXPOSE 80
 ---> Running in c386ebdf2785
Removing intermediate container c386ebdf2785
 ---> 3e9c95503673
Step 6/7 : ENV NAME World
 ---> Running in 973de6382eea
Removing intermediate container 973de6382eea
 ---> 6e1cb34ac56c
Step 7/7 : CMD ["python", "main.py"]
 ---> Running in ecc79a83e31e
Removing intermediate container ecc79a83e31e
 ---> 0090f57e5d53
Successfully built 0090f57e5d53
Successfully tagged testnum:latest
```

Figure 6. Building testnum:latest image

5.2 Starting the service of Docker container

It's very easy to define, run, and scale services with the Docker platform -- just write a docker-compose.yml file. This file is a YAML file which defines how Docker containers should behave in this project. Following the tutorial of Docker and change as we needed, we set four services: two webs, one visualizer and one cassandra image. Similarly, use basic docker commands to deploy this for this project. I named this Docker service as 'test1'. I have shown process of starting the services in following figure 7.



```
sichen@sichen-XPS-15-9570:~/下载/mnist_testnumber_zzx$ docker stack deploy -c docker-compose.yml test1
Creating network test1_webnet
Creating service test1_web
Creating service test1_visualizer
Creating service test1_cassandra
sichen@sichen-XPS-15-9570:~/下载/mnist_testnumber_zzx$ docker ps
CONTAINER ID        IMAGE                            COMMAND                  CREATED             STATUS              PORT
                                        NAMES
e64e275d4951        cassandra:latest                 "docker-entrypoint.s…"   14 seconds ago      Up 8 seconds        7000
7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp   test1_cassandra.1.dovjdomjk7qhda8c2wuz0ku0a
32131ed9e980        dockersamples/visualizer:stable   "npm start"              16 seconds ago      Up 12 seconds       8080
tcp                                     test1_visualizer.1.h24ru6ttzf03k1ythfg9z771w
sichen@sichen-XPS-15-9570:~/下载/mnist_testnumber_zzx$ docker ps
CONTAINER ID        IMAGE                            COMMAND                  CREATED             STATUS              PORT
                                        NAMES
0b0a0a8fe5ea        testnum:latest                   "python main.py"         9 seconds ago       Up 1 second         80/t
p                                       test1_web.2.i4aox31z6l6nlululoxnntqew
b2ff00e9d641        testnum:latest                   "python main.py"         9 seconds ago       Up 2 seconds        80/t
p                                       test1_web.1.xwf58j04bxmerkq4tc60271o5
e64e275d4951        cassandra:latest                 "docker-entrypoint.s…"   17 seconds ago      Up 11 seconds       7000
7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp   test1_cassandra.1.dovjdomjk7qhda8c2wuz0ku0a
32131ed9e980        dockersamples/visualizer:stable   "npm start"              19 seconds ago      Up 15 seconds       8080
tcp                                     test1_visualizer.1.h24ru6ttzf03k1ythfg9z771w
sichen@sichen-XPS-15-9570:~/下载/mnist_testnumber_zzx$ 
```

Figure 7. The process of starting services in containers

Here, we have successfully opened the services in Docker, the next and last thing is about testing whether our codes are correct to recognize digit and whether cassandra database can store all the submitting record.

6. Result and Conclusion

It's time for testing outcome of the whole process. We have seen that in figure 7, test1_web has been begun. That indicates that we can enter website(http://localhost:80) to begin submitting images and get digit recognition result.

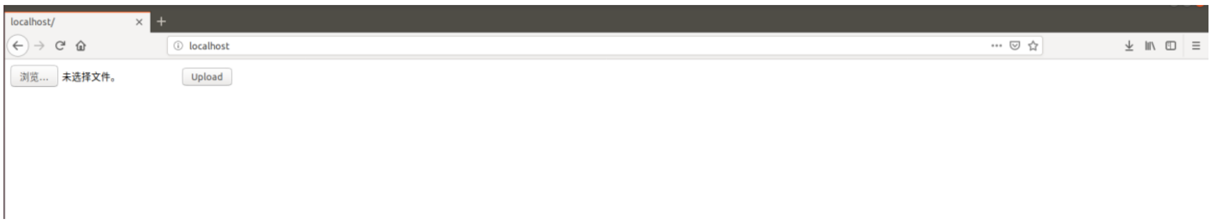
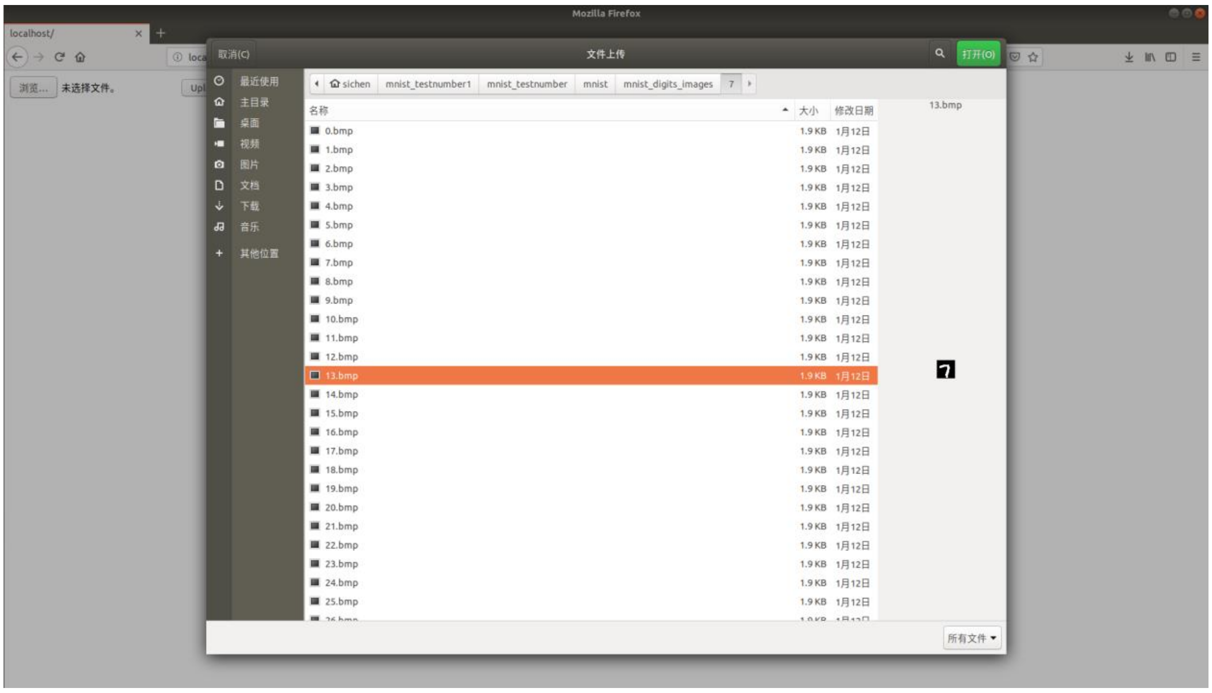
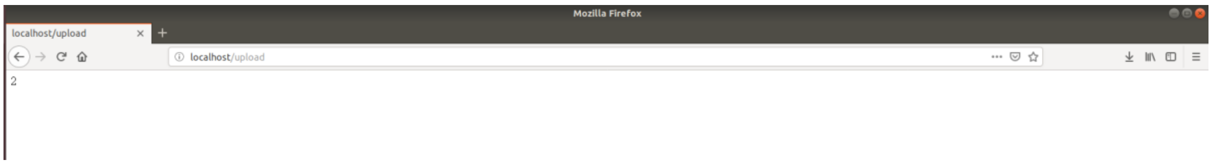


Figure 8-a: Web page for uploading images



名称	大小	修改日期
0.bmp	1.9 KB	1月12日
1.bmp	1.9 KB	1月12日
2.bmp	1.9 KB	1月12日
3.bmp	1.9 KB	1月12日
4.bmp	1.9 KB	1月12日
5.bmp	1.9 KB	1月12日
6.bmp	1.9 KB	1月12日
7.bmp	1.9 KB	1月12日
8.bmp	1.9 KB	1月12日
9.bmp	1.9 KB	1月12日
10.bmp	1.9 KB	1月12日
11.bmp	1.9 KB	1月12日
12.bmp	1.9 KB	1月12日
13.bmp	1.9 KB	1月12日
14.bmp	1.9 KB	1月12日
15.bmp	1.9 KB	1月12日
16.bmp	1.9 KB	1月12日
17.bmp	1.9 KB	1月12日
18.bmp	1.9 KB	1月12日
19.bmp	1.9 KB	1月12日
20.bmp	1.9 KB	1月12日
21.bmp	1.9 KB	1月12日
22.bmp	1.9 KB	1月12日
23.bmp	1.9 KB	1月12日
24.bmp	1.9 KB	1月12日
25.bmp	1.9 KB	1月12日

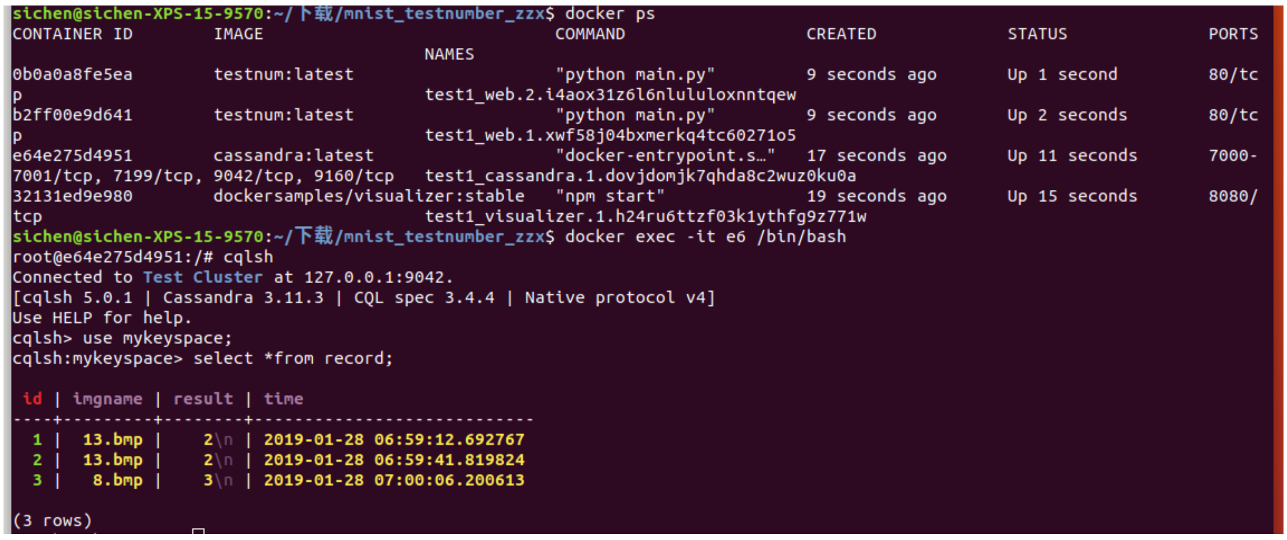
Figure 8-b: Selecting uploaded image



```
2
```

Figure 8-c: Result of digit recognition

From figure 8, the successful recognition and smooth process substantiate that the digit recognition part of this project is succeed. Then, we need to evaluate how does the cassandra database performance.



```
sichen@sichen-XPS-15-9570:~/下载/mnist_testnumber_zzx$ docker ps
CONTAINER ID        IMAGE                              COMMAND                  CREATED             STATUS              PORTS
                                       NAMES
0b0a0a8fe5ea        testnum:latest                     "python main.py"         9 seconds ago       Up 1 second         80/tc
p                                      test1_web.2.i4aox31z6l6nlululoxnntqew
b2ff00e9d641        testnum:latest                     "python main.py"         9 seconds ago       Up 2 seconds        80/tc
p                                      test1_web.1.xwf58j04bxmerkq4tc60271o5
e64e275d4951        cassandra:latest                   "docker-entrypoint.s…"   17 seconds ago      Up 11 seconds       7000-
7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp   test1_cassandra.1.dovjdomjk7qhda8c2wuz0ku0a
32131ed9e980        dockersamples/visualizer:stable    "npm start"              19 seconds ago      Up 15 seconds       8080/
tcp                                    test1_visualizer.1.h24ru6ttzf03k1ythfg9z771w
sichen@sichen-XPS-15-9570:~/下载/mnist_testnumber_zzx$ docker exec -it e6 /bin/bash
root@e64e275d4951:/# cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.3 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> use mykeyspace;
cqlsh:mykeyspace> select *from record;

 id | imgname | result | time
----+---------+--------+----------------------------
  1 |  13.bmp |    2\n | 2019-01-28 06:59:12.692767
  2 |  13.bmp |    2\n | 2019-01-28 06:59:41.819824
  3 |   8.bmp |    3\n | 2019-01-28 07:00:06.200613

(3 rows)
```

Figure 9. Performance of cassandra database for storing record

All in all, the whole process runs smoothly and successfully. In this project, I can not only realize the charm of python programmer, but also taste the sense of refreshing when using docker. Until now, this application is only for recognize single digit number in uploaded file. In the future, it still has many things to explore on other aspects like letters, real items and even languages. For me, I would more interested to continue exploring how to recognize a long array of numbers one time.

As a beginner of both programing and big data, this project taught me computer science is not just characteristic as interesting, functional and even profitable. My real feel about the whole project is patience, patience and patience again. Every step is hard for beginner, even though refer to sample codes from others, problems still be happened. Thanks to professor Zhang and colleague, give me more patience to let me finish the project greatly.

7. References

[1]MNIST database,from Wikipedia, accessed January 28,2019, URL:https://en.wikipedia.org/wiki/MNIST_database

[2][Python]基于 CNN 的 MNIST 手写数字识别, accessed January 28,2019, URL: https://www.cnblogs.com/Ran-Chen/p/9220739.html

[3]Convolutional Neural Networks with TensorFlow, accessed January 28,2019, URL: https://www.datacamp.com/community/tutorials/cnn-tensorflow-python

[4]Quickstart——Flask 1.0.2 documentation, accessed January 28,2019, URL:http://flask.pocoo.org/docs/1.0/quickstart/

[5]Uploading Files——Flask 1.0.2 documentation, accessed January 28,2019, URL: http://flask.pocoo.org/docs/1.0/patterns/fileuploads/

[6]Getting Started-Connecting to Cassandra, accessed January 28,2019, URL:https://datastax.github.io/python-driver/getting_started.html

[7]Get Started,Part 2:Containers|Docker Documentation, accessed January 28,2019, URL: https://docs.docker.com/get-started/part2/