Final project report
for
Big Data

MNIST handwritten digit recognition with Docker and Cassandra

Member:

Hao.Chen
Xi'an Jiaotong-Liverpool University

Project Instructor:
Fan Zhang

25th March 2019

Contents

1 Introduction

Big Data has become a popular term in recent years. Unlike its meaning in earlier years, which refers to massive data sets, Big Data is in fact a capacity for searching, aggregating and cross-referencing large data sets [1]. With the fast development of networking, data storage, and the data collection capacity, Big Data are now rapidly expanding in all science and engineering domains, including physical, biological and biomedical sciences [2]. The era of Big Data is underway.

Since Big Data has both merits and shortcomings, it should be considered critically. On one hand, Big Data seems a powerful tool to analyze, predict and address various societal issues. For example, McKinsey & Company launched an in-depth research on the U.S. healthcare, the EU public sector administration, the U.S. retail, the global manufacturing and the global personal location data, which represent the global economy [3]. As a result, the McKinsey report pointed out that Big Data has the ability to fulfill the economic function, improve the productivity and competitiveness of enterprises and create huge profits for consumers. On the other, Big Data will lead to loss of privacy and security. Since many personal information and other significant messages in other fields are collected and stored in data sets, if they are leaked, a deleterious and serious problems will follow.

This project is required to deploy MNIST applications into Docker container to fulfill handwritten digit recognition, then save results into Cassandra database.

2 MNIST

2.1 Background Summary

MNIST, or Modified National of Standards and Technology, is a database of handwritten digits, which is commonly used for training various image processing systems [4]. Handwritten digit recognition by MNIST is a classic application in the field of deep learning. The MNIST database downloaded from the official website contains 60,000 training images and 10,000 testing images. Each size of images is 28 * 28 pixels. It is necessary to divide the training set and the test set because in the design of machine learning model, there must be a separate test set not for training but for evaluating the performance of the model, so that it is easier to generalize the designed model to other data sets, which is called generalization.

2.2 Specific Procedure

(1) **Data loading**
According to the MNIST database, it mainly includes two parts: 60,000 training set (mnist.train) and 10,000 test set (mnist.test). As shown in figure 1, inside each row is a 28 * 28 array, the essence of which is to convert 28 * 28 pixels images to corresponding pixel lattice.

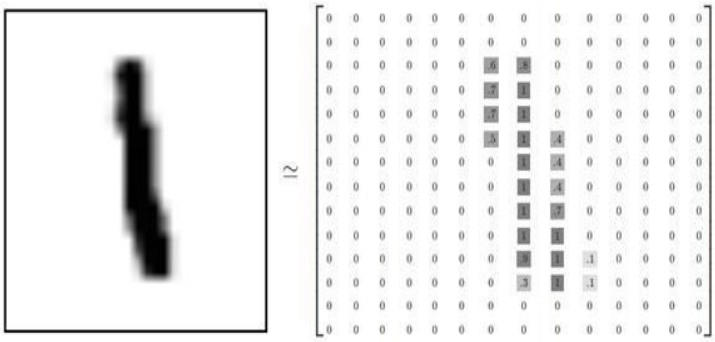

Figure 1: Converting handwritten number 1 into a matrix

(2) **Model building**
Softmax regression is recommended, which can be used to assign probabilities to different objects. In order to get evidence of a given picture belonging to digital class, 784 features of the images (each pixel value of lattices) are used to get the weighted sum. If a feature (pixels) has very strong evidence that this image does not belong to the digital class, then the corresponding weighted sum should be negative. On the contrary, if a feature (pixels) have favorable evidence to support this image belongs to this class, then the weighted sum is a positive number. The process can be converted into a mathematical formula:

$$evidence_i = \sum_j W_{i,j} x_j + b_i$$

i is the number needs predicting. In the case of this, b_i represents the bias of i and $W_{i,j}$ is the weighted sum. x_j is the value of 784 features.

For better understanding, the whole process can be concluded as figure 2:

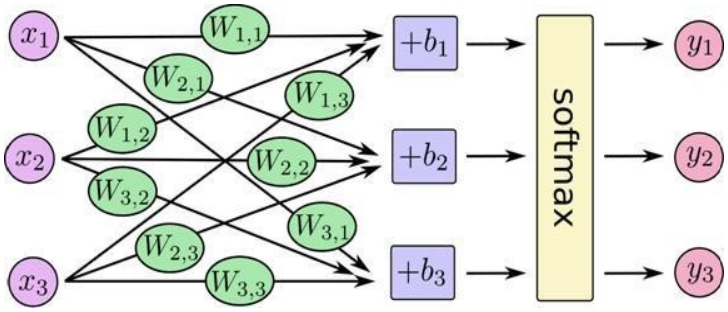

Figure 2: Model building

It can also be shown in matrix multiplication as figure 3:

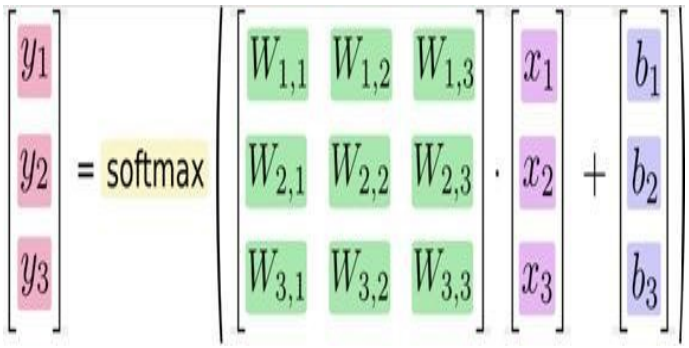

Figure 3: Matrix multiplication

Finally, Softmax will convert these evidence into probability:

$$y = softmax(W_x + b)$$

(3) **Constructing loss functions and optimizing settings**
An indicator is defined to measure the model, which is called cost or loss. Then the indicator will be minimized by cross-entropy to improve the accuracy.

2.3 Result

The model program for training is here:

```
from tensorflow.examples.tutorials.mnist import input_data
import tensorflow as tf
#Load data
mnist = input_data.read_data_sets('MNIST_data/', one_hot=True)

#Set the placeholder size to be the size of the sample input and
    output
x = tf.placeholder(tf.float32, [None, 784])

y_ = tf.placeholder(tf.float32, [None, 10])

def weight_variable(shape):
    initial = tf.truncated_normal(shape,stddev = 0.1)
    return tf.Variable(initial)

def bias_variable(shape):
    initial = tf.constant(0.1,shape = shape)
    return tf.Variable(initial)

#Custom convolution function
def conv2d(x,W):
    return tf.nn.conv2d(x, W, strides = [1,1,1,1], padding = '
        SAME')

#Custom pooling function
```

```
24 def max_pool_2x2(x):
25     return tf.nn.max_pool(x, ksize=[1,2,2,1], strides=[1,2,2,1],
           padding='SAME')
26
27 #Set the first convolution layer and pooling layer
28 W_conv1 = weight_variable([5, 5, 1, 32])
29 b_conv1 = bias_variable([32])
30
31 x_image = tf.reshape(x,[-1,28,28,1])
32
33 h_conv1 = tf.nn.relu(conv2d(x_image,W_conv1) + b_conv1)
34 h_pool1 = max_pool_2x2(h_conv1)
35
36 #Set the second convolution layer and pooling layer
37 W_conv2 = weight_variable([5, 5, 32, 64])
38 b_conv2 = bias_variable([64])
39
40 h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
41 h_pool2 = max_pool_2x2(h_conv2)
42
43 #Set the first full connection layer
44 W_fc1 = weight_variable([7 * 7 * 64, 1024])
45 b_fc1 = bias_variable([1024])
46
47 h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
48 h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
49
50 #dropout
51 keep_prob = tf.placeholder("float")
52 h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)
53
54 #Set the second full connection layer
55 W_fc2 = weight_variable([1024, 10])
56 b_fc2 = bias_variable([10])
57
58 y_conv=tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
59
60 #Build loss function as cross entropy
61 cross_entropy = -tf.reduce_sum(y_*tf.log(y_conv))
62
63 #Configure the Adam optimizer with a learning rate of 1e-4
64 train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy
       )
65
66 #The expression of correct rate is established
67 correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(y_
       ,1))
68 accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
```

```
max_acc=0
saver = tf.train.Saver(max_to_keep=1)

#Begin training
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))
            if train_accuracy>=max_acc:
                max_acc=train_accuracy
                saver.save(sess,'SAVE/model.ckpt')
        train_step.run(feed_dict={x: batch[0], y_: batch[1],
           keep_prob: 0.5})
        # saver.save(sess, 'SAVE/model.ckpt')

#After the training, the test set is used for testing, and the
   final result is output
    print('test accuracy %g' % accuracy.eval(feed_dict={
        x: mnist.test.images, y_: mnist.test.labels, keep_prob:
           1.0}))


}
```

Finally, run the model program and the result is as follows:

```
step 19800, training accuracy 1
step 19900, training accuracy 1
2019-02-13 17:44:31.272565: W tensorflow/core/framework/allocator.cc:108] Allocation of 1003520000 exceeds
2019-02-13 17:44:35.355082: W tensorflow/core/framework/allocator.cc:108] Allocation of 250880000 exceeds
2019-02-13 17:44:35.851316: W tensorflow/core/framework/allocator.cc:108] Allocation of 501760000 exceeds
2019-02-13 17:51:26.418997: W tensorflow/core/framework/allocator.cc:108] Allocation of 125440000 exceeds
2019-02-13 17:51:27.099304: W tensorflow/core/framework/allocator.cc:108] Allocation of 40960000 exceeds 1
test accuracy 0.9921

Process finished with exit code 0
```

Figure 4: Accuracy

It is clear that accuracy is 0.9921.

3 Docker

3.1 Background Summary

Docker is a platform for developers and system administrators to develop, deploy, and run applications with containers, which can also be depicted as a container virtualization technology [5] . The use of Linux containers to deploy applications is called containerization. Generally, Docker extends Linux containers with kernel-level and application-level API, which can work together to run processes like CPU, memory, I/O, network and so on independently [6].

Docker containers are based on images [6]. A Docker image is an executable package, which contains everything needed to run a program, such as the code, a runtime, libraries, environment variables, and configuration files. In addition, constructing a container requires a Dockerfile in advance, which is used to run some commands and ensure some statements automatically.

Nowadays, Docker becomes quite popular mainly because of flexibility and lightweight. Unlike traditional virtual machines, Docker can containerize most of applications and leverage the host kernel. Virtual machines require installing all the software and application code manually, which is rather time-consuming. However, in the Docker world, it just takes seconds to build a container and fulfill the same function [5]. As a result, Docker is convenient for peoples life and work.

3.2 Specific Procedure

From the Docker official documentation, building a container needs three parts: a Dockerfile, the app itself and a requirement text. Specification below will combine with this project.

The Dockerfile is used to define the environment in the container. As the figure 5 & 6 shows below, the version of Python used in the project is 3.7. The working path set in the container is named **/app**. Also, everything in the folder will be copied to the working path in the container. Then the container will install some necessary packages following the requirement text. Finally, port 80 will be exposed to connect with outside and the app itself will be run in the container.

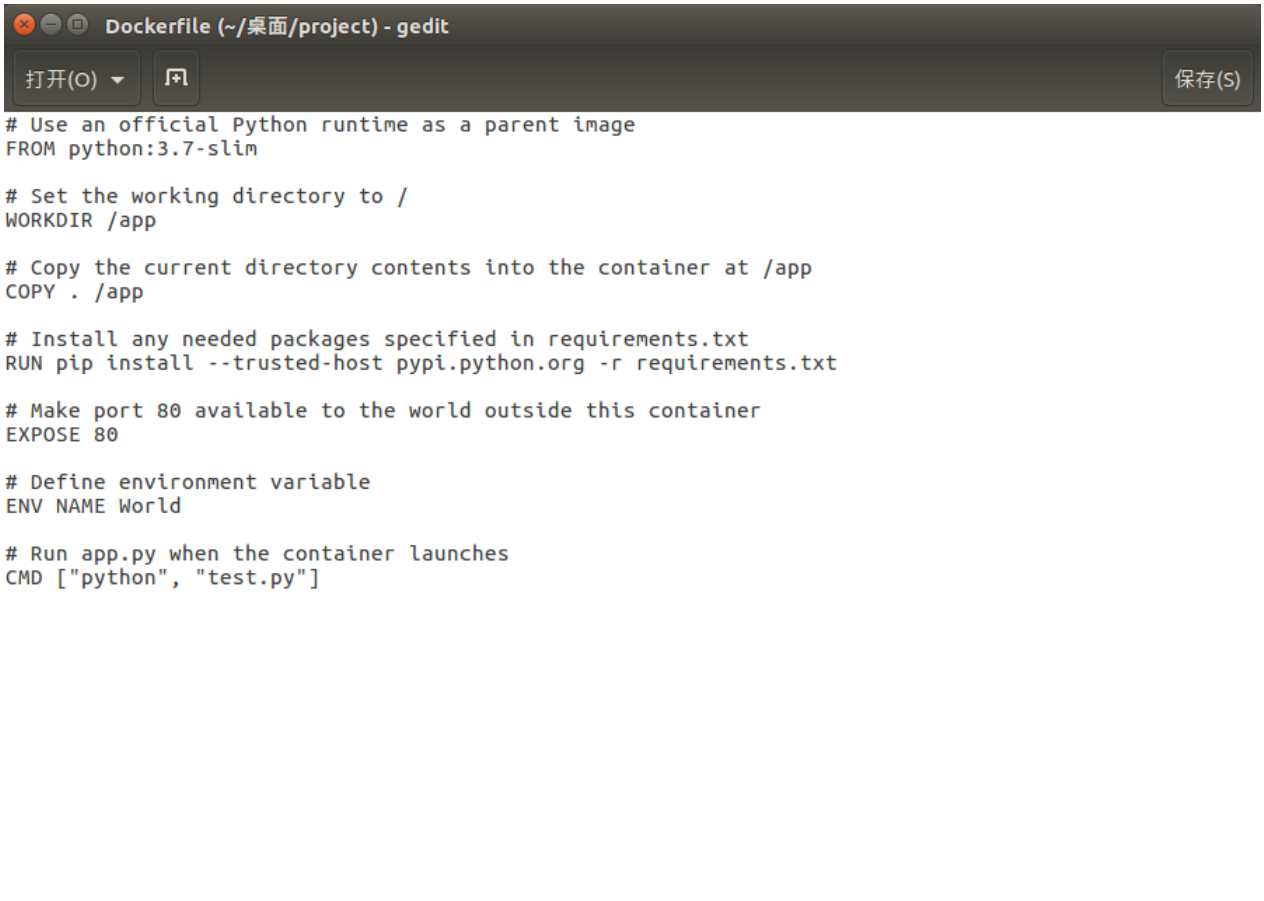

Figure 5: Docker file

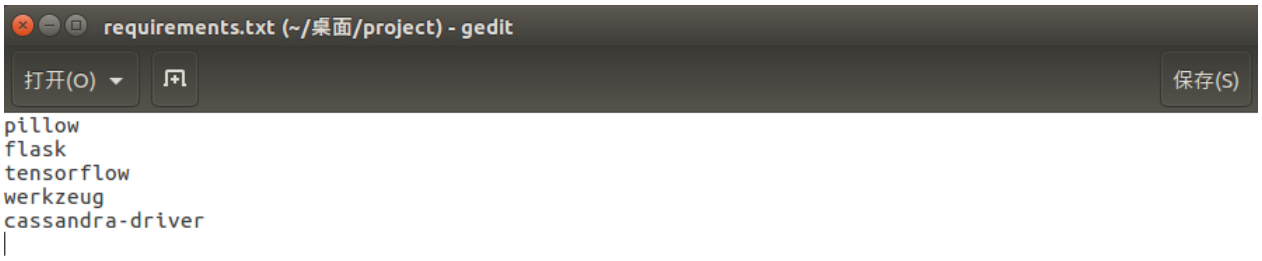

Figure 6: requirements.txt

3.3 Result

After creating the Docker image and mapping the port, all running containers can be inspected by commands as figure 7:

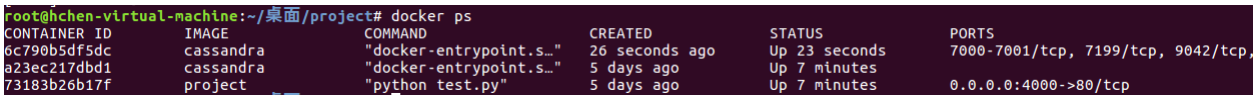

Figure 7: Existing containers

4 Cassandra

4.1 Background Summary

Apache Cassandra is an open source distributed database management system, aimed to handle very large amounts of data spread out across many commodity severs while providing a highly available service with no single point of failure. Its core is NoSQL solution, initially developed by Facebook for reliability and scalability [7].

Cassandra has many excellent features. Firstly, it is flexible. With Cassandra, like document storage, you don't have to solve fields in a record ahead of time. You can add or remove fields as you wish while the system is running. This is an amazing efficiency boost, especially on large deployments. In addition, it has high expandability. Cassandra is a purely horizontal extension. To add more capacity to the cluster, you can point to another computer. You don't have to restart any processes, change application queries, or manually migrate any data. Furthermore, it has a multi-data center. In this case, you can adjust your node layout to avoid a single data center catching fire, and a backup data center will have at least a full copy of each record. In conclusion, these features make Cassandra more competitive.

4.2 Specific Procedure

It is convenient to download Cassandra from Docker. According to the official documentation, executing several commands can satisfying our requirement. In the terminal, we can input: **docker pull cassandra** to download Cassandra; **docker run –name some-cassandra –network some-network -d cassandra:tag** to start an instance; **docker run -it –network some-network –rm cassandra cqlsh some-cassandra** to connect to Cassandra from cqlsh.

It's worth mentioning that to connect handwritten digit recognition container to the Cassandra container, we use the method of network. Two specific commands are as follows:
docker run -d –name mnist -p 4000:80 test
docker run -d –name hchen-cassandra –network container:mnist cassandra

4.3 Result

The result is the same as the result in section 3.3 and the figure 6.

5 Completed Codes

```
1  from PIL import Image
2  import tensorflow as tf
3  from cassandra.cluster import Cluster
4  import time
5  from flask import Flask,request,redirect, flash
6  from werkzeug.utils import secure_filename
7  import os
8
9  import logging
10 log = logging.getLogger()
11 log.setLevel('INFO')
12 handler = logging.StreamHandler()
13 handler.setFormatter(logging.Formatter("%(asctime)s [%(levelname
       )s] %(name)s: %(message)s"))
14 log.addHandler(handler)
```

```
15
16
17 UPLOAD_FOLDER ='/app'    #'/home/hchen/        /project'
18 ALLOWED_EXTENSIONS = set(['png', 'jpg', 'JPG', 'PNG', 'bmp'])
19 localtime = time.strftime("%Y-%m-%d %H:%M:%S")
20 KEYSPACE = "imagesapce"
21 app = Flask(__name__)
22 app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
23
24 def allowed_file(filename):
25     return '.' in filename and filename.rsplit('.', 1)[1] in
       ALLOWED_EXTENSIONS
26
27
28 @app.route('/upload', methods=['POST', 'GET'])
29 def upload():
30     if request.method == 'POST':
31         f = request.files['file']
32         if 'file' not in request.files:
33             flash('No file part')
34             return redirect(request.url)
35         file = request.files['file']
36         # if user does not select file, browser also
37         # submit an empty part without filename
38         if file.filename == '':
39             flash('No selected file')
40             return redirect(request.url)
41         if file and allowed_file(file.filename):
42             filename = secure_filename(file.filename)
43             file.save(os.path.join(app.config['UPLOAD_FOLDER'],
                   filename))
44         result = useModel(f.filename)
45         insertValues(localtime,f.filename,result)
46         return "The number in the picture is {}".format(result)
47
48
49     return '''
50     <!DOCTYPE html>
51     <html lang="en">
52     <head>
53         <meta charset="UTF-8">
54         <title>Title</title>
55     </head>
56     <body>
57         <h1>Please upload your picture.</h1>
58         <form action="" enctype='multipart/form-data' method='
               POST'>
59             <input type="file" name="file">
```

```
60              <input type="submit" value="Upload">
61          </form>
62      </body>
63      </html>
64      '''
65
66  def imageprepare(file_name):
67      """
68      This function returns the pixel values.
69      The input is a png file location.
70      """
71
72      im = Image.open(file_name)
73      im = im.convert('L')
74      tv = list(im.getdata()) #get pixel values
75
76      #normalize pixels to 0 and 1. 0 is pure white, 1 is pure
            black.
77      tva = [ (255-x)*1.0/255.0 for x in tv]
78      print(tva)
79      return tva
80
81      """
82      This function returns the predicted integer.
83      The input is the pixel values from the imageprepare()
            function.
84      """
85
86
87
88
89
90
91  def weight_variable(shape):
92      initial = tf.truncated_normal(shape,stddev = 0.1)
93      return tf.Variable(initial)
94
95  def bias_variable(shape):
96      initial = tf.constant(0.1,shape = shape)
97      return tf.Variable(initial)
98
99  def conv2d(x,W):
100     return tf.nn.conv2d(x, W, strides = [1,1,1,1], padding =
            'SAME')
101
102 def max_pool_2x2(x):
103     return tf.nn.max_pool(x, ksize=[1,2,2,1], strides=[1,2,2,1],
            padding='SAME')
```

```
104
105 def GetData():
106     x = tf.placeholder(tf.float32, [None, 784])
107
108     y_ = tf.placeholder(tf.float32, [None, 10])
109
110     W_conv1 = weight_variable([5, 5, 1, 32])
111     b_conv1 = bias_variable([32])
112
113     x_image = tf.reshape(x,[-1,28,28,1])
114
115
116     h_conv1 = tf.nn.relu(conv2d(x_image,W_conv1) + b_conv1)
117     h_pool1 = max_pool_2x2(h_conv1)
118
119     W_conv2 = weight_variable([5, 5, 32, 64])
120     b_conv2 = bias_variable([64])
121
122     h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
123     h_pool2 = max_pool_2x2(h_conv2)
124
125     W_fc1 = weight_variable([7 * 7 * 64, 1024])
126     b_fc1 = bias_variable([1024])
127
128     h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
129     h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
130
131     keep_prob = tf.placeholder("float")
132     h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)
133
134     W_fc2 = weight_variable([1024, 10])
135     b_fc2 = bias_variable([10])
136
137     y_conv=tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
138     return y_conv, keep_prob, h_conv2, x, y_
139
140 def useModel(file_name):
141     result = imageprepare(file_name)
142     y_conv, keep_prob,h_conv2, x, y_ = GetData()
143     cross_entropy = -tf.reduce_sum(y_*tf.log(y_conv))
144     train_step = tf.train.AdamOptimizer(1e-4).minimize(
            cross_entropy)
145     correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax
            (y_,1))
146     accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float
            "))
147
148     saver = tf.train.Saver()
```

```
149     with tf.Session() as sess:
150         sess.run(tf.global_variables_initializer())
151         saver.restore(sess, "/app/model.ckpt")  #"/home/haochen/
                /BigData/Docker/model.ckpt")#

152         #print ("Model restored.")
153 
154         prediction=tf.argmax(y_conv,1)
155         predint=prediction.eval(feed_dict={x: [result],keep_prob
              : 1.0}, session=sess)
156         print(h_conv2)
157         print('result:')
158         print(predint[0])
159         return predint[0]
160 
161 def insertValues(localtime, filename, result):
162     cluster = Cluster(contact_points=['0.0.0.0'],port=9042)
163     session = cluster.connect()
164     try:
165 
166         session.execute("""
167 
168             CREATE KEYSPACE %s
169 
170             WITH replication = { 'class': 'SimpleStrategy', '
                  replication_factor': '2' }
171 
172             """ % KEYSPACE)
173 
174 
175         log.info("setting keyspace...")
176 
177         session.set_keyspace(KEYSPACE)
178 
179 
180         log.info("creating table...")
181 
182         session.execute("""
183 
184             CREATE TABLE mytable (
185 
186                 time text,
187 
188                 filename text,
189 
190                 result int,
191 
192                 PRIMARY KEY (time, filename, result)
```

```
193
194                 )
195
196                 """)
197             session.execute("""
198
199                     INSERT INTO mytable (time,filename,result)
200
201                     VALUES (%s, %s, %s)
202
203                         """ ,(localtime,filename,result))
204
205         except Exception as e:
206
207             log.error("Unable to create keyspace")
208
209             log.error(e)
210
211
212
213 if __name__ == '__main__':
214     app.run(host='0.0.0.0', port=80)
215
216
217 }
```

6 Conclusion

6.1 Summary

The goal of this project, deploying MNIST applications into Docker container to fulfill handwritten digit recognition and saving results into Cassandra database, has been achieved as follows. Figure 8 is the interface for uploading our pictures and figure 9 is results in Cassandra.

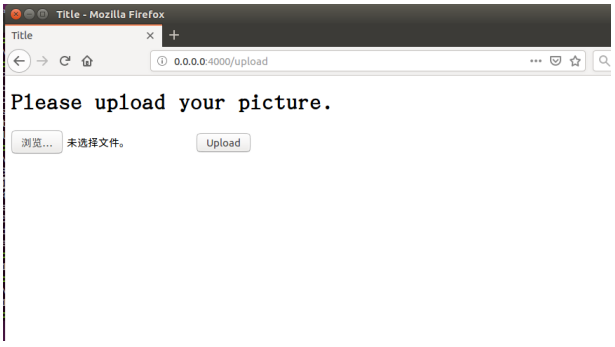

Figure 8: Interface

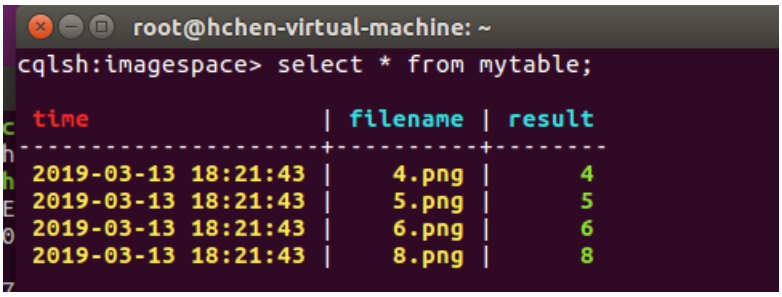

Figure 9: Results

6.2 Limitations

Firstly, one month's study is largely insufficient. Some part of learning is in a rush time. In addition, for the code provided in the previous section, there are many issues about low efficiency and unreasonable structure in spite of that it still can work.

6.3 Future study

Since Big Data is a gigantic concept in the field of computer science, what we have learned is just a small part of it. Handwritten digit recognition is only an entry level. At the same time, many new techniques are making great progress every day. What we need to do is keeping learning and carrying on.

References

[1] D. Boyd and K. Crawford,"Critical Questions for Big Data," *Information, Commun. Soc.,* vol. 15, no. 5, pp. 662-679, 2012.

[2] J. E. Cardama, J. C. Gatti, and C. F. Gatti, "Almohadillas Articulares (Knuckle Pads)," *Rev. Argentina Dermatologia,* vol. 63, no. 4, pp. 331-333, 1982.

[3] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mob. Networks Appl.,* vol. 19, no. 2, pp. 171-209, 2014.

[4] S. Gangaputra, "Handwritten digit databasele," 2013. [Online]. Available: http://cis.jhu.edu/ sachin/digit/digit.html. [Accessed: 18-Mar-2019].

[5] C. Anderson, "Docker," *Softw. Eng.,* pp. 102-104, 2015.

[6] D. Bernstein, "Containers and cloud: From LXC to docker to kubernetes," *IEEE Cloud Comput.,* vol. 1, no. 3, pp. 81-84, 2014.

[7] J. James, "Cassandra," *Notes Queries,* vol. s2-X, no. 241, p. 111, 1860.