BIG DATA REASERCH

PROJECT REPORT

Handwritten Recognition System Based on Convolutional Neural Network

Name ZIXUAN ZHAO

Mentor FAN ZHANG

09.20.2018 - 11.27.2018

https://github.com/azuredream/NumberRec

Content

1. Introduction

The primary goal of this research project was to build a website that is capable of recognizing handwritten number and saving the result in a NoSQL database. The project can be divided into three parts.

First, using TensorFlow build a convolution neural network which included two hidden layers, one full connection layer and one dropout layer. The model was trained with MNIST dataset.

Second, constructing a dynamic website with Flask+Cassandra and deploy the server and database separately in two Docker Containers. Clients can upload pictures with handwritten number, and then the website will recognize the number and record the picture and result in Cassandra.

Third, a writing board was set on the homepage of the website. Users can write numbers on this board and submit it to get recognition result.

2. CNN architecture and training

2.1 CNN architecture

Figure 1.CNN architecture illustrates the architecture of CNN in this project.

Including 2 hidden layers, one full connection layer and one dropout layer, the model uses ReLU as activation function and ADAM as optimizer.

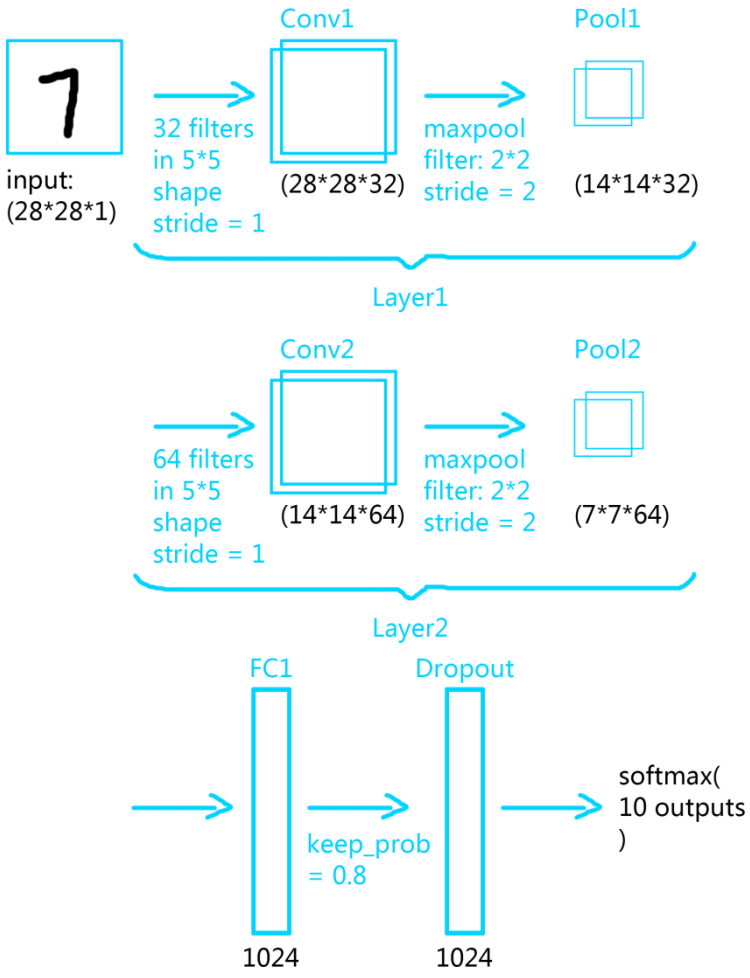

Figure 1.CNN architecture

2.2 Training

Ideally, MNIST is a great dataset to train this model, because it contains large amount of data and well-rounded. And after the training , the model should be able to recognize anyone’s handwritten number picture as long as the picture is in the right shape and clear enough.

However, at the first time, the recognition accuracy of author’s pictures is less than 20% while over 96% of MNIST test dataset was recognized correctly. Though a series of analysis and comparison of the MNIST test data (Figure 2.MNIST test pic) and the author’s data (Figure 3.Untreated test pic), I adjusted the brush attributes such as line width and hardness to make the test pictures closer to “MNIST style”. Also, sharping the picture was important because it highlighted the outline of the number, made convolution and pooling more effective (Figure 4.Adjusted test pic).

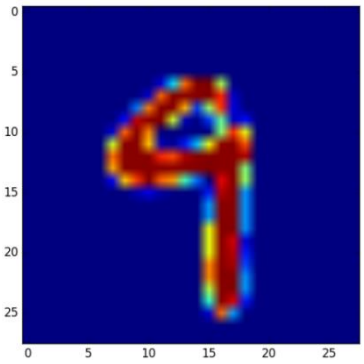

Figure 2.MNIST test pic

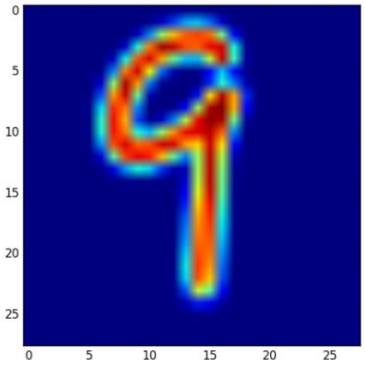

Figure 3.Untreated test pic

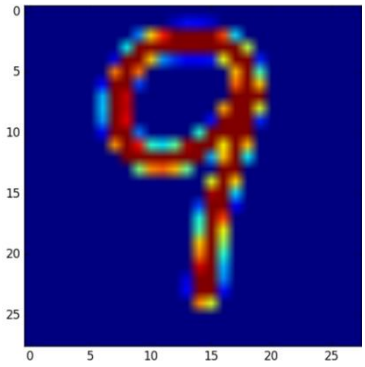

Figure 4.Adjusted test pic

Admittedly, using a small batch of the author’s data to fine tune the model instead of trying to write in “MNIST style” is the standard step, but it’s hard to realize in this project, because of the limited human source (one person).

2.3 evaluation

As shown in the figure, accuracy of recognizing MNIST test dataset is 96% and author’s test dataset is 80%.

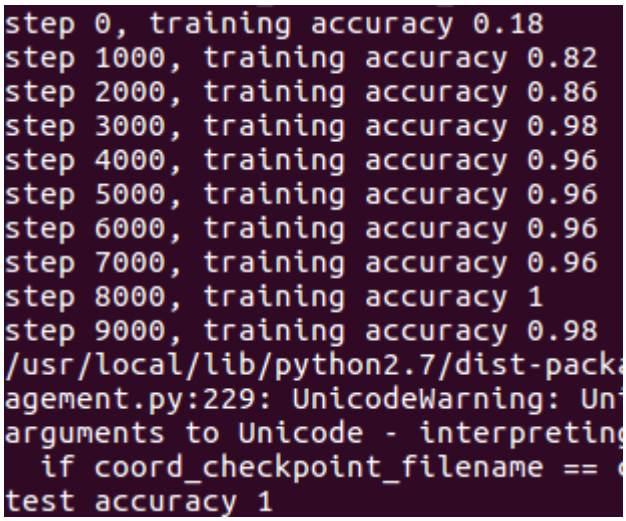

Figure 5.Training and test

3. Deploy the recognition application in Docker

3.1 Constructing a Docker application

For users' convenience, the application was shown as a dynamic website. Users write on a writing board, submit their produce, the web server will run the recognizing program and give result.

Figure 6.Write a number

Figure 7.Get the result

3.2 File tree

<Dockerfile> lists commands that will be execute in Container such as configuring work directory and installing dependent libraries.<requirements.txt> lists all the dependent libraries for the app.<app.py> contains core codes of the app including a flask website and the CNN model.

Build these files as an image and run the image in a container will start number recognizing website service. Cassandra image can be downloaded from Cassandra official website.

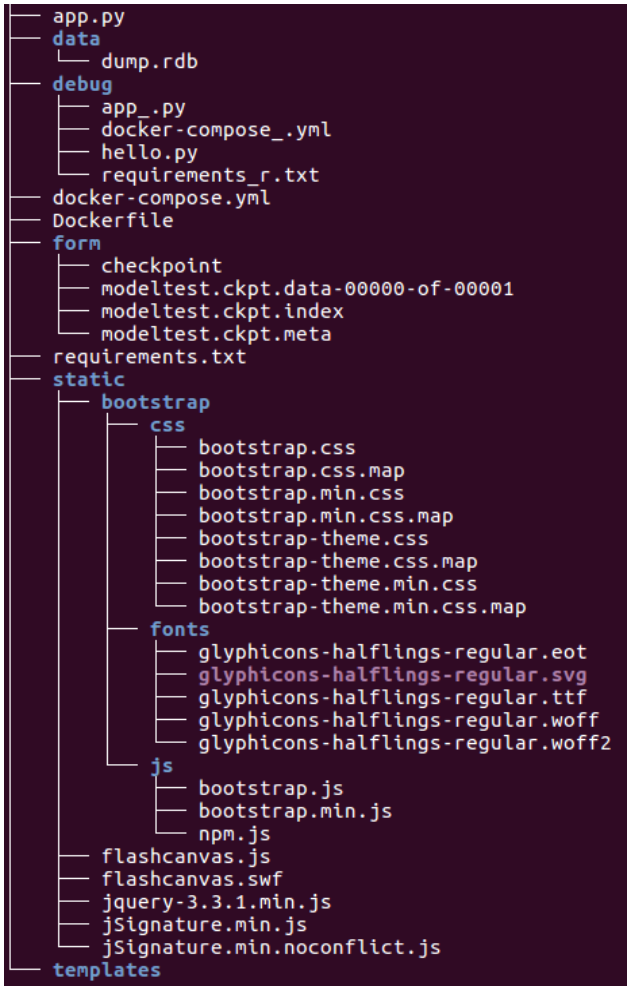

Figure 8.File tree

3.3 Compose Docker services

3.3.1 Load Balance

By configuring services of a app in <docker-compose.yml> ,one can set up the port mapping, performance limitation and network of a batch of containers. In this project, we need three services: web server, Cassandra database and docker visualization program. Taking load balance into consideration, I started two web server containers to handle requests. Also, I constructed docker cluster and set my computer as the manager node. If this app gets more computers to use, it will be able to divide the work on different computers reasonably to make the website more productive.

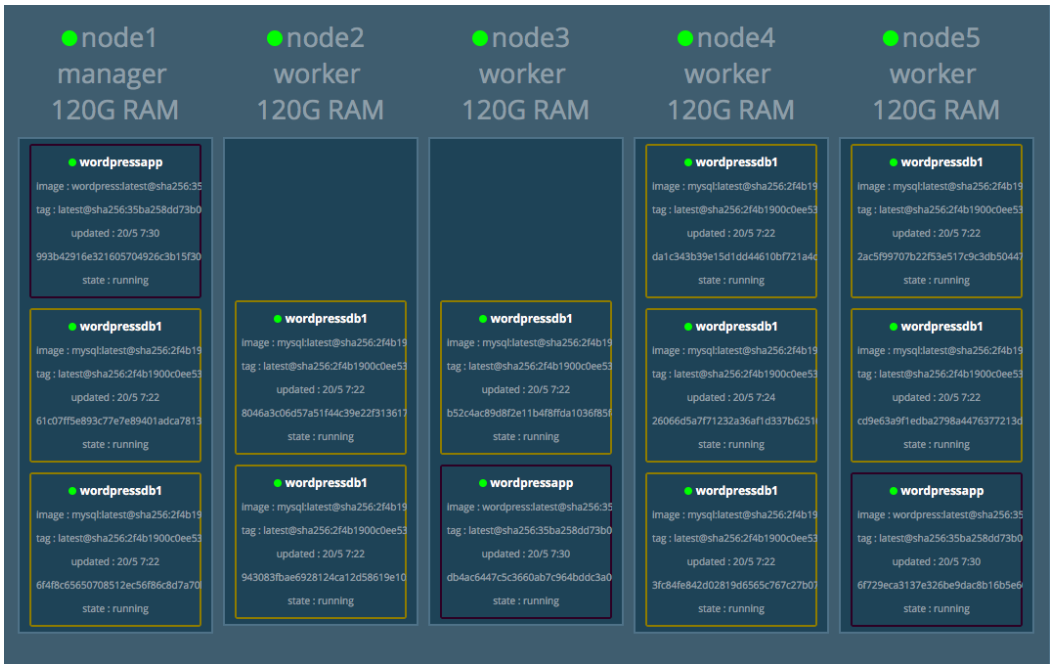

Figure 9.Example of Docker load balance

3.3.2 Docker network

I encountered a problem that the web service and the Database which settled in different Containers cannot communicate to each other. According to my knowledge, once the Docker Service is configured, these two programs in one host, as two processes, should be able to send data to each other through specific ports. However, I was confused when Container A with Flask wasn't able to connect to Container B with Cassandra. I checked the Docker network structure from official documents and realized that the host and every containers was connected by the virtual bridge docker0, and sharing data based on DNAT and SNAT, which was in line with the structure of Virtual Machine and the Host. I employed command "Docker Network Inspect bridge" to acquire the network segment of the host. According to the rule that containers should be assigned within the network segment of the host, I located IP addresses of Container A and Container B, and successfully linked them up.

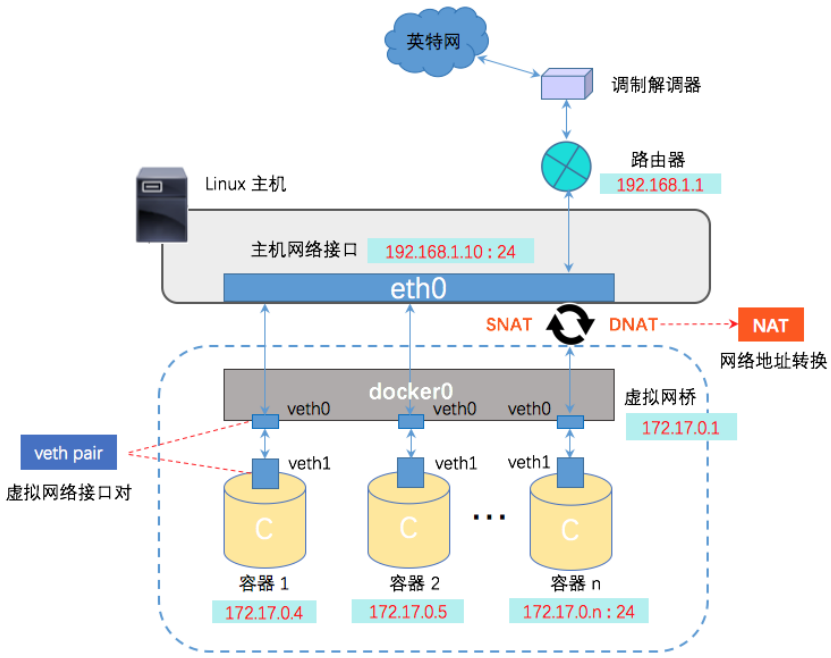

Figure 10.Docker network

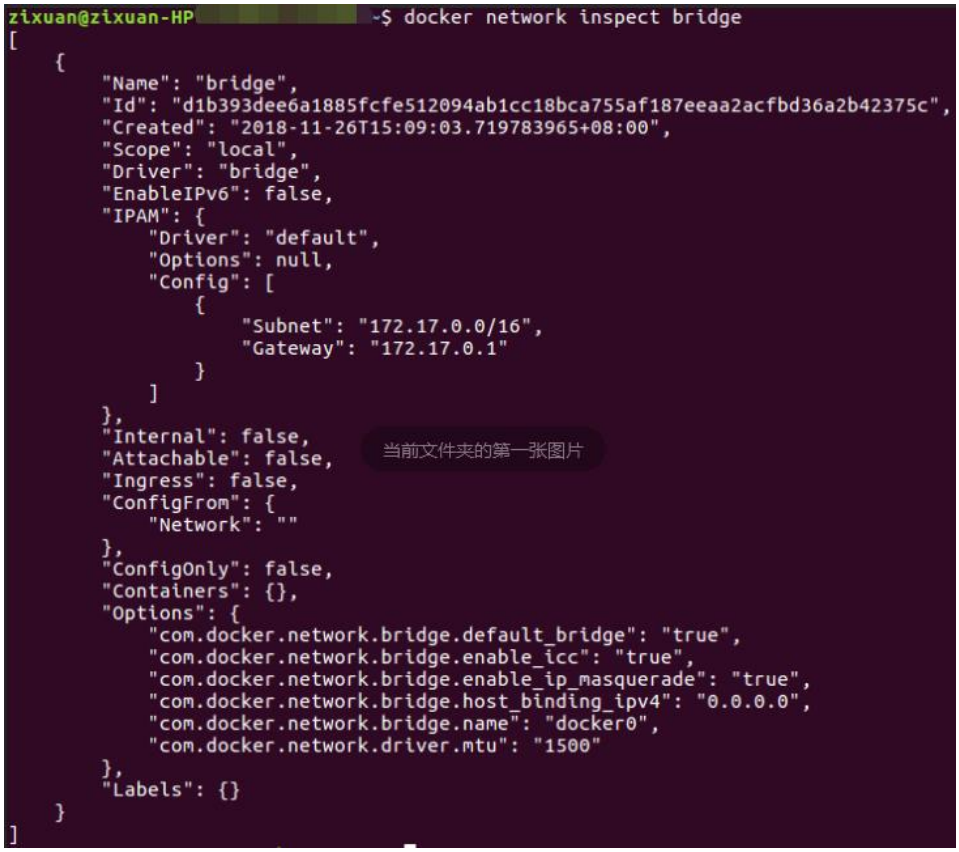

Figure 11.Docker network

3.4 Record result in Cassandra

3.3.1 Why Cassandra

As a NoSQL Database, Cassandra specializes in storing various kinds of data (in blob form) and performs excellent in high concurrent application scenarios. These advantages make Cassandra very suitable for my website because I need to save pictures, text, numbers and timestamps and I hope a large amount of people can use my website at the same time smoothly. Besides, eventual consistency is acceptable to this app.

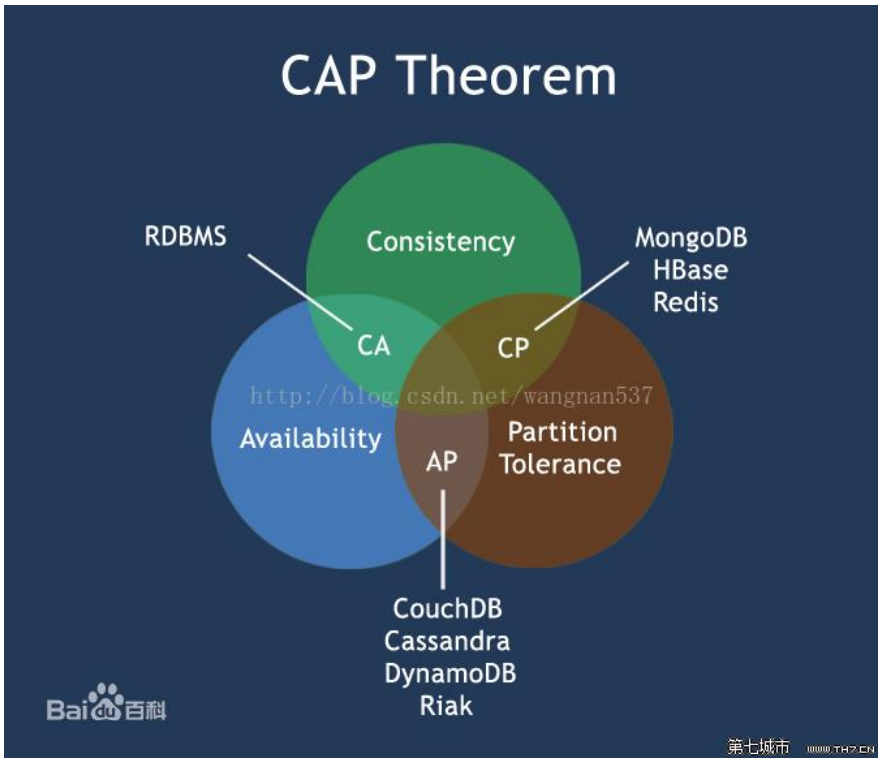

Figure 12.CAP Theorem

3.3.2 Operating Cassandra

When the server is started, it will connect to Cassandra and create a data table.

```
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
2018-11-27 02:39:57,086 [INFO] werkzeug:  * Running on http://0.0.0.0:80/ (Press CTRL+C to qu
it)
2018-11-27 02:39:57,087 [INFO] werkzeug:  * Restarting with stat
2018-11-27 02:40:01,355 [WARNING] cassandra.cluster: Cluster.__init__ called with contact_poi
nts specified, but no load_balancing_policy. In the next major version, this will raise an er
ror; please specify a load-balancing policy. (contact_points = ['cassandra'], lbp = None)
2018-11-27 02:40:01,380 [INFO] cassandra.policies: Using datacenter 'datacenter1' for DCAware
RoundRobinPolicy (via host '10.0.2.3'); if incorrect, please specify a local_dc to the constr
uctor, or limit contact points to local cluster nodes
2018-11-27 02:40:01,463 [INFO] root: Creating keyspace...
2018-11-27 02:40:01,465 [ERROR] root: Unable to create keyspace
2018-11-27 02:40:01,465 [ERROR] root: Keyspace 'keyspace1' already exists
2018-11-27 02:40:01,476 [WARNING] werkzeug:  * Debugger is active!
2018-11-27 02:40:01,477 [INFO] werkzeug:  * Debugger PIN: 331-863-973
2018-11-27 02:42:00,271 [INFO] werkzeug: 10.255.0.2 - - [27/Nov/2018 02:42:00] "GET /favicon.
ico HTTP/1.1" 404 -
```

Figure 13.Connect to Cassandra

After image recognition, the image, file name, recognition result and current time will be recorded in Cassandra database.

```
>>> rows = ss.execute("""SELECT * FROM data;""")
>>> for row in rows:
...     print(row)
...
Row(id=1, img=u'
+UAAAAj0lEQVRIie3UywmFMBSEYUtJCXZkSrGTm2IswBKswoCb380ILnTha/RCZpvAd3IGUlUlJX8dIA
AN8AMy28k6b4BwBTub8SqYgLg3vbYQgU73p9OvPDjgIDA6sChseBwTOL3xutLdLaC1u+DubllncoHJtk
6BS3/BgbXCsgOrVx977QB7Ye3j2ArsLVjJJzMDy+k3eWRstF8AAAAASUVORK5CYII=', imgname=u't
estpic1', result=u'[7]', time=u'2018-11-27 02:42:10.348680')
```

Figure 14.Insert success

4. The front end of the website

The front end of the website was realized by Bootstrap and three.js.

5. Conclusion and future work

5.1 Conclusion

The technologies used in this project, such as TensorFlow , Docker, and Cassandra, are crucial technologies in industrial production. In this project, they were organically combined to build a usable image recognition application and reflected their respective strengths and characteristics. For a student who is a beginner in AI and big data field, this is a valuable, systematic learning experience. After this project, I am confident that I will be able to make more powerful and attractive big data applications in my next project.

Mr. Zhang’s curriculum is comprehensive and systematic, from Cassandra, Docker to Spark and data visualization technologies, which has enabled me not only to get started in this field, but also to clarify the direction of learning in the coming years.

5.2 future work

There are two directions for this project to continue to improve.

5.2.1 Text recognition

By optimizing the model and providing more training data, the model will be able to recognize letters and even Chinese characters. Such improvements will enable it to be applied in areas of practical significance such as handwriting input.

5.2.2 Front page

First of all, there is no doubt that this page has the potential to become more charming.

In addition, the brush of the tablet can add hardness parameters, to make the handwriting closer to the training data. When the picture exported from the tablet, it can be processed, such as sharpen, in the front end so as to share the computing pressure of the server.

Appendix

https://github.com/azuredream/NumberRec

app.py

```
# Author: ZIXUAN ZHAO
# -*- coding: utf-8 -*-
from flask import Flask ,redirect, url_for
from redis import Redis, RedisError
from flask import request
from flask import render_template
import re
from cStringIO import StringIO
from PIL import Image, ImageFilter
import os,base64
import socket
import numpy
import tensorflow as tf
import logging
from cassandra.cluster import Cluster
from cassandra.query import SimpleStatement
import datetime

# Connect to Redis
redis = Redis(host="redis", db=0, socket_connect_timeout=2,
socket_timeout=2)

app = Flask(__name__)

# Prepare 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 = "keyspace1"
session = 0

def createKeySpace():
    global session
    cluster = Cluster(contact_points=['cassandra'],port=9042)
    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 data (
       id int,
       imgname text,
       img text,
       time text,
       result text,
               PRIMARY KEY (id)
            )
            """)
    except Exception as e:
        log.error("Unable to create keyspace")
        log.error(e)
idcount = 1
createKeySpace();
# Prepare Cassandra end

@app.route('/')
def hello_world():
    global idcount
    global session
    isrc = request.args.get('pic')

    print(isrc)
    if(isrc != None):
   #图片转为矩阵
   isrctrans = isrc.replace(' ','+')  #trans" " back to "+"
   img = base64_to_image(isrctrans)
   #plt.imshow(img)
   #plt.show()
   tv = list(img.getdata())
   tva = [ round((x/255.000)*1.000,8) for x in tv]
   #识别图片
   result = recnub(tva)
   #存入 Cassandra
   imgname  = "testpic"+str(idcount)
   img
   session.execute("""INSERT INTO """+KEYSPACE+"""."""+"""data
(id,imgname, img, time, result) VALUES ("""+str(idcount)+""",'"""+
imgname+"""','"""+isrctrans+"""','"""+str(datetime.datetime.now
())+"""','"""+ str(result)+"""');""")
```

```
        #每次存好显示数据表
    session.set_keyspace(KEYSPACE)
    ctable = session.execute("""SELECT * FROM data;""")
    print(ctable)
    idcount = idcount+1;
        return render_template('jSig.html', message=result,picsrc
= isrctrans)
    name = "test"
    return render_template('jSig.html', message="none")

def base64_to_image(base64_str, image_path=None):
    base64_data = re.sub('^', '', base64_str)
    binary_data = base64.b64decode(base64_data)
    img_data = StringIO(binary_data)
    img = Image.open(img_data).convert('L')
    if image_path:
        img.save(image_path)
    return img

def recnub(data):#(784 维数组)
#model
    x = tf.placeholder(tf.float32, [None, 784])
    W = tf.Variable(tf.zeros([784, 10]))
    b = tf.Variable(tf.zeros([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(tf.float32)
    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)
    init_op = tf.initialize_all_variables()
    #model
    ndresult = numpy.array(data)
    prediction=tf.argmax(y_conv,1)
    #saver
    saver = tf.train.Saver()
    with tf.Session() as sess:
```

```
        sess.run(init_op)
        saver.restore(sess, "./form/modeltest.ckpt")#这里使用了之前
保存的模型参数
    #saver
        print(ndresult)
        a = prediction.eval(feed_dict={x: [ndresult], keep_prob:
1.0})
        print('result:')
        print(a)
        return a

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)

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

def max_pool_2x2(x):
  return tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2,
1], padding='SAME')

@app.route("/file",methods=['POST','GET'])
def filein():
    pic = request.args.get('imgdata')
    return redirect(url_for('hello_world',pic = pic,result = 15))

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80)
```

Dockerfile

```
// Author:ZIXUAN ZHAO
# 将官方 Python 运行时用作父镜像
FROM python:2.7-slim

# 将工作目录设置为 /app
WORKDIR /app

# 将当前目录内容复制到位于 /app 中的容器中
ADD . /app

# 安装 requirements.txt 中指定的任何所需软件包
RUN pip install -i https://pypi.tuna.tsinghua.edu.cn/simple -r
requirements.txt
```

```
# 使端口 80 可供此容器外的环境使用
EXPOSE 80

# 定义环境变量
ENV NAME World

# 在容器启动时运行 app.py
CMD ["python", "app.py"]
```

requirements.txt

```
// Author:ZIXUAN ZHAO
Flask
Redis
cassandra-driver
Pillow
numpy
tensorflow
logging
```

docker-compose.yml

```
// Author:ZIXUAN ZHAO
 version: "3"
 services:
   web:
     image: azuredream/numberrec:latest
     deploy:
       replicas: 2
       restart_policy:
         condition: on-failure
       resources:
         limits:
           cpus: "0.8"
           memory: 500M
     ports:
       - "80:80"
     networks:
       - webnet
   visualizer:
     image: dockersamples/visualizer:stable
     ports:
       - "8080:8080"
     volumes:
       - "/var/run/docker.sock:/var/run/docker.sock"
     deploy:
       placement:
         constraints: [node.role == manager]
     networks:
       - webnet
```

```
  redis:
    image: redis
    ports:
      - "6379:6379"
    volumes:
      - ./data:/data
    deploy:
      placement:
        constraints: [node.role == manager]
    networks:
      - webnet
  cassandra:
    image: cassandra
    ports:
      - "9042:9042"
    volumes:
      - ./data:/data
    deploy:
      placement:
        constraints: [node.role == manager]
    networks:
      - webnet
networks:
  webnet:
```

jSig.html

```
// Author:ZIXUAN ZHAO
<!DOCTYPE html>
<html lang="zh-CN">
<head>
<title>手写数字识别</title>
<!-- Bootstrap -->
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/boo
tstrap.min.css" rel="stylesheet">
</head>

<body style="background:#d1d1d1;">

<div class="container">

    <div class="container"> </div>
    <div id="signature" style=""></div>
    <h6>请在上方书写数字</h6>
    <input class="btn btn-primary btn-xs" type="button" value="
识别数字" id="yes"/>
    <div class="container"> </div>
    <!<div class="container" id="result"
style="background:#d1d1d1;border:2px;">
```

```
    <img src= {{picsrc}}></img>
    <h6>识别结果: {{message}}</h6>
    <!</div>
    <!<div class="container">
    <!</div>

    <!<input type="button" value="下载" id="download"/>
    <!<input type="button" value="重写" id="reset"/>
    <div class="container" id="someelement"></div>
</div>

    <script src="{{ url_for('static',
filename='jquery-3.3.1.min.js') }}"></script>
    <!-- 加载 Bootstrap 的所有 JavaScript 插件。你也可以根据需要只加
载单个插件。 -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/js/boots
trap.min.js"></script>
    <!--[if lt IE 9]>
        <script src="jSignature/flashcanvas.js"></script>
    <![endif]-->
    <script src="{{ url_for('static',
filename='jSignature.min.js') }}"></script>
    <script>
    document.body.style.zoom = 4;
    function uploadComplete(){
        }
    function uploadFailed(){
        }
    function progressFunction(){
        }

        $(function() {
            var $sigdiv = $("#signature");
            $sigdiv.jSignature({height:28,width:28,'color' :
'#ffffff','lineWidth' : '2','background-color': '#000'}); // 初
始化 jSignature 插件.
            $("#yes").click(function(){
                //将画布内容转换为图片
                var datapair = $sigdiv.jSignature("getData",
"image");
                var i = new Image();
                i.src = "data:" + datapair[0] + "," + datapair[1];
                $(i).appendTo($("#someelement")); // append the
image (SVG) to DOM.

       //利用 canvas 生成 png
       // canvas
```

```
        var canvas = document.createElement('canvas');
        var context = canvas.getContext('2d');
        // canvas 对图片进行缩放
        targetWidth = 28;
        targetHeight = 28;
                canvas.width = targetWidth;
                canvas.height = targetHeight;
                // 清除画布
                context.clearRect(0, 0, targetWidth,
targetHeight);
                // 图片压缩
                context.drawImage(i, 0, 0, targetWidth,
targetHeight);
        //准备上传图片数据
        //datapair = $sigdiv.jSignature("getData","native");

        //datapair[0]=
        //datapair[1]=svgbase64 的字符串

        window.location.href = "/file?imgdata="+i.src;
            });

            //$sigdiv.jSignature("setData", "data:" +
datapair.join(","));
            $("#download").click(function(){
                downloadFile("a.png",
convertBase64UrlToBlob($("img").attr("src")));
            });
            $("#reset").click(function(){
                $sigdiv.jSignature("reset"); //重置画布，可以进行重
新作画.
                $("#someelement").html("");
            });
        });

        function downloadFile(fileName, blob){
            var aLink = document.createElement('a');
            var evt = document.createEvent("HTMLEvents");
            evt.initEvent("click", false, false);//initEvent 不加
后两个参数在 FF 下会报错，感谢 Barret Lee 的反馈
            aLink.download = fileName;
            aLink.href = URL.createObjectURL(blob);
            aLink.dispatchEvent(evt);
        }
        /**
         * 将以 base64 的图片 url 数据转换为 Blob
         * @param urlData
         *            用 url 方式表示的 base64 图片数据
```

```
         */
        function convertBase64UrlToBlob(urlData){

            var bytes=window.atob(urlData.split(',')[1]);
//去掉 url 的头，并转换为 byte

            //处理异常,将 ascii 码小于 0 的转换为大于 0
            var ab = new ArrayBuffer(bytes.length);
            var ia = new Uint8Array(ab);
            for (var i = 0; i < bytes.length; i++) {
                ia[i] = bytes.charCodeAt(i);
            }

            return new Blob( [ab] , {type : 'image/png'});
        }
    </script>
</body>
</html>
```