校徽

**学校名字**

**图像处理-课程报告**

**项目题目:**

**学 院：**

**专 业：**

**年级班别：**

**学 号：**

**学生姓名：xw\_cqx**

**任课老师:**

**日 期：2018.6.30**

# TensorFlow 验证码识别

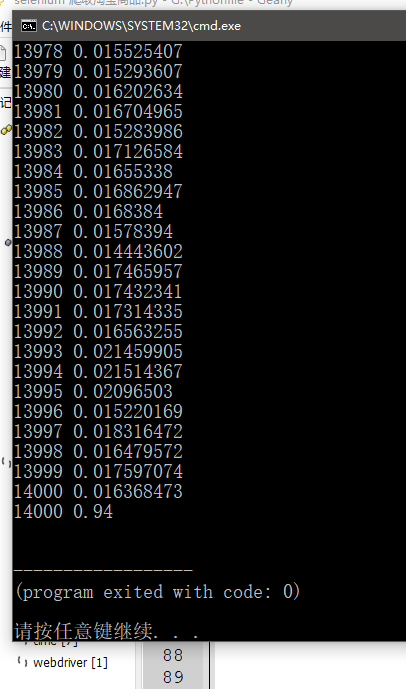
# 一 实验目的：大数据时代，数据为王，爬虫就是获取数据的主要手段之一，而在爬取数据的过程中时常会遇到验证码来反爬虫，该project提供一套TensorFlow构建和训练自己的CNN来做**四位数、数字字母混合**验证码识别的算法及训练成功的model。

**二 流程**：

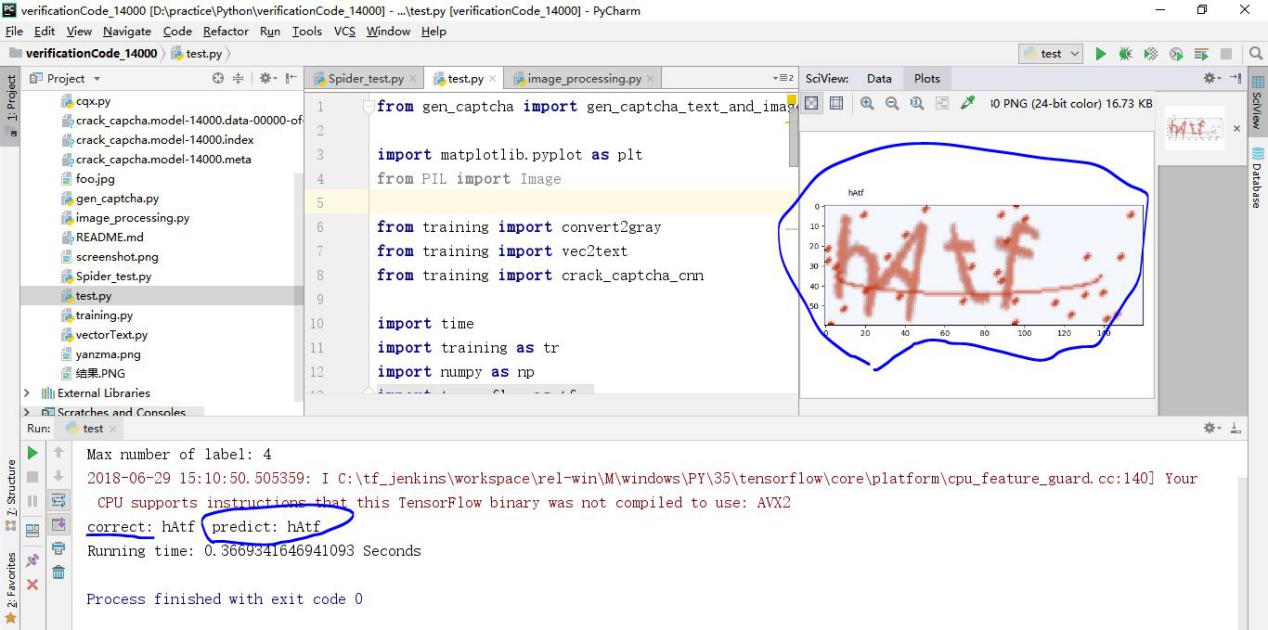
**流程：**依次随机选择0~9和a~Z共62字符，用 Python 的 captcha 库生成验证码，然后对 生成的验证码进行预处理，然后放进自己构建的深度学习cnn模型中进行训练每训练 100个batch就再随机生成100张验证码进行验证检验识别率，训练完成之后，用 selenium模拟登录慕课网(和微信公众号网页版)获取其验证码，然后用训练的模型去 识别，得出结果上传然后登录。

**三 实验结果分析：**

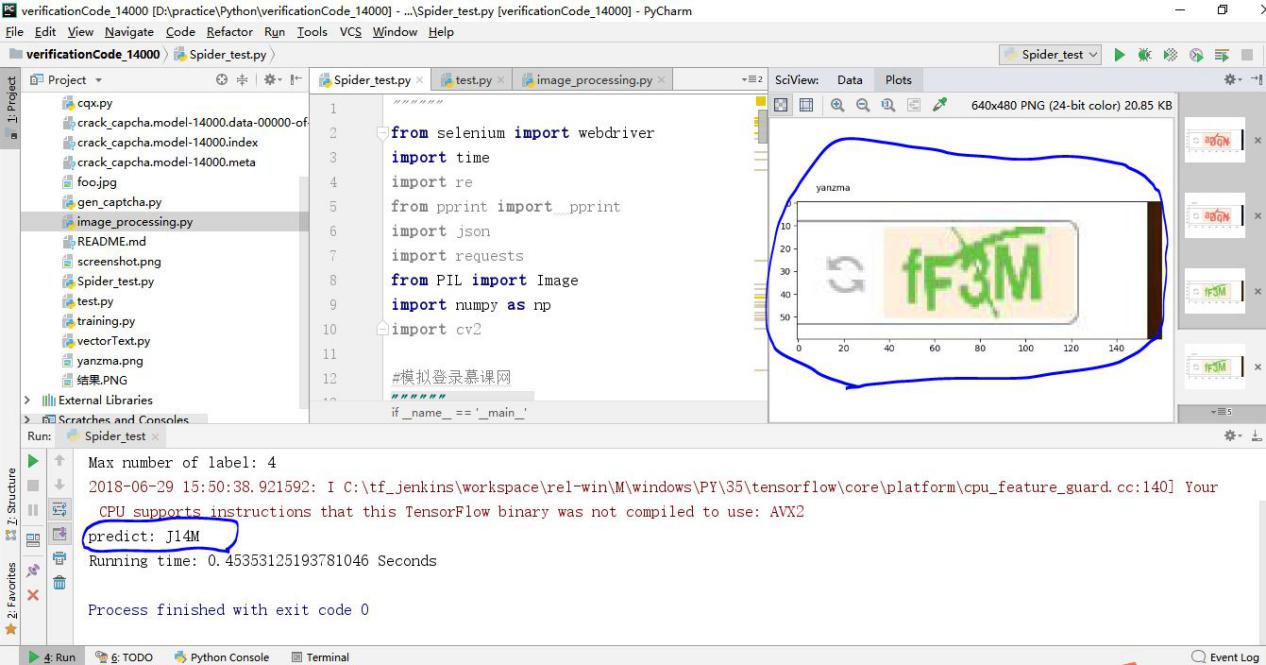
**1运行training.py 文件**，经过6个多小时的训练，命中率达到94%(每次训练128张图片，训练了14000次)(见下图 )



**2运行test**。可以随机生成一张验证码(在右边展示，要保存下来的话可以在gen\_captcha.py中把注释掉的保存验证码的代码的注释符删除)，并识别，然后打印出正确的打印和预测的答案。



**3.运行Spider\_test.py。** 会自动打开一个网页，然后跳转到慕课网(也可以跳转到微信公众，只需把代码中注释的对象对换)然后登陆，然后会跳出需要填入验证码，然后会把其获取下来在屏幕的右边显示，然后底下会 打印出识别的结果，见下图

**游侠**

可以看见预测得结果和实际获取的验证码的是不一样的，识别自己生成的验证码识别率高达94%，识别网上验证码识别率近乎0%，原因主要是训练集和要识别的验证码差别太大（分辨率，像素特点等不同,而且自己的图片是截图得到的，不是直接下载网页上的图片，因为上面的图片链接是一样的，但是每一次点击返回的验证码是不一样的）

**解决办法**：知道要目标网站的验证码的来源，然后用相同的验证码的生成模式去生成验证码，然后重新训练，cnn算法和爬虫模块，验证模块等都不需改变，只需更换训练集即可。由于时间问题来不及去找目标网站的验证码的生成方式，故本项目还留下一个小瑕疵。

**四 代码：train.py是训练的代码 、gen\_captcha.py是生成验证码的代码 、vectorText.py是处理生成的验证码的代码，test.py是检测训练模型代码，Spider\_test.py是爬取慕课网（或者微信公众号）然后调用生成的模型去识别获取到网络验证码的代码，image\_processing.py是处理网络获取的验证码的代码。**

train.py

from gen\_captcha import gen\_captcha\_text\_and\_image

from gen\_captcha import number

from gen\_captcha import alphabet

from gen\_captcha import ALPHABET

import time

import numpy as np

import tensorflow as tf

text, image = gen\_captcha\_text\_and\_image()

print("verification code iamge channel:", image.shape) # (60, 160, 3)

# 图像大小

IMAGE\_HEIGHT = 60

IMAGE\_WIDTH = 160

MAX\_CAPTCHA = len(text)

print("Max number of label:", MAX\_CAPTCHA) # 验证码最长4字符; 我全部固定为4,可以不固定. 如果验证码长度小于4，用'\_'补齐

# 把彩色图像转为灰度图像（色彩对识别验证码没有什么用）

def convert2gray(img):

if len(img.shape) > 2:

gray = np.mean(img, -1)

# 上面的转法较快，正规转法如下

# r, g, b = img[:,:,0], img[:,:,1], img[:,:,2]

# gray = 0.2989 \* r + 0.5870 \* g + 0.1140 \* b

return gray

else:

return img

"""

cnn在图像大小是2的倍数时性能最高, 如果你用的图像大小不是2的倍数，可以在图像边缘补无用像素。

np.pad(image,((2,3),(2,2)), 'constant', constant\_values=(255,)) # 在图像上补2行，下补3行，左补2行，右补2行

"""

# 文本转向量

char\_set = number + alphabet + ALPHABET + ['\_'] # 如果验证码长度小于4, '\_'用来补齐

CHAR\_SET\_LEN = len(char\_set)

def text2vec(text):

text\_len = len(text)

if text\_len > MAX\_CAPTCHA:

raise ValueError('验证码最长4个字符')

vector = np.zeros(MAX\_CAPTCHA \* CHAR\_SET\_LEN)

def char2pos(c):

if c == '\_':

k = 62

return k

k = ord(c) - 48

if k > 9:

k = ord(c) - 55

if k > 35:

k = ord(c) - 61

if k > 61:

raise ValueError('No Map')

return k

for i, c in enumerate(text):

idx = i \* CHAR\_SET\_LEN + char2pos(c)

vector[idx] = 1

return vector

# 向量转回文本

def vec2text(vec):

char\_pos = vec.nonzero()[0]

text = []

for i, c in enumerate(char\_pos):

char\_at\_pos = i # c/63

char\_idx = c % CHAR\_SET\_LEN

if char\_idx < 10:

char\_code = char\_idx + ord('0')

elif char\_idx < 36:

char\_code = char\_idx - 10 + ord('A')

elif char\_idx < 62:

char\_code = char\_idx - 36 + ord('a')

elif char\_idx == 62:

char\_code = ord('\_')

else:

raise ValueError('error')

text.append(chr(char\_code))

return "".join(text)

"""

#向量（大小MAX\_CAPTCHA\*CHAR\_SET\_LEN）用0,1编码 每63个编码一个字符，这样顺利有，字符也有

vec = text2vec("F5Sd")

text = vec2text(vec)

print(text) # F5Sd

vec = text2vec("SFd5")

text = vec2text(vec)

print(text) # SFd5

"""

# 生成一个训练batch

def get\_next\_batch(batch\_size=128):

batch\_x = np.zeros([batch\_size, IMAGE\_HEIGHT \* IMAGE\_WIDTH])

batch\_y = np.zeros([batch\_size, MAX\_CAPTCHA \* CHAR\_SET\_LEN])

# 有时生成图像大小不是(60, 160, 3)

def wrap\_gen\_captcha\_text\_and\_image():

while True:

text, image = gen\_captcha\_text\_and\_image()

if image.shape == (60, 160, 3):

return text, image

for i in range(batch\_size):

text, image = wrap\_gen\_captcha\_text\_and\_image()

image = convert2gray(image)

batch\_x[i, :] = image.flatten() / 255 # (image.flatten()-128)/128 mean为0

batch\_y[i, :] = text2vec(text)

return batch\_x, batch\_y

####################################################################

with tf.name\_scope('input'):

X = tf.placeholder(tf.float32, [None, IMAGE\_HEIGHT \* IMAGE\_WIDTH])

Y = tf.placeholder(tf.float32, [None, MAX\_CAPTCHA \* CHAR\_SET\_LEN])

keep\_prob = tf.placeholder(tf.float32) # dropout

# 定义CNN

def crack\_captcha\_cnn(w\_alpha=0.01, b\_alpha=0.1):

x = tf.reshape(X, shape=[-1, IMAGE\_HEIGHT, IMAGE\_WIDTH, 1])

# w\_c1\_alpha = np.sqrt(2.0/(IMAGE\_HEIGHT\*IMAGE\_WIDTH)) #

# w\_c2\_alpha = np.sqrt(2.0/(3\*3\*32))

# w\_c3\_alpha = np.sqrt(2.0/(3\*3\*64))

# w\_d1\_alpha = np.sqrt(2.0/(8\*32\*64))

# out\_alpha = np.sqrt(2.0/1024)

# 3 conv layer

w\_c1 = tf.Variable(w\_alpha \* tf.random\_normal([3, 3, 1, 32]))

b\_c1 = tf.Variable(b\_alpha \* tf.random\_normal([32]))

conv1 = tf.nn.relu(tf.nn.bias\_add(tf.nn.conv2d(x, w\_c1, strides=[1, 1, 1, 1], padding='SAME'), b\_c1))

conv1 = tf.nn.max\_pool(conv1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

conv1 = tf.nn.dropout(conv1, keep\_prob)

w\_c2 = tf.Variable(w\_alpha \* tf.random\_normal([3, 3, 32, 64]))

b\_c2 = tf.Variable(b\_alpha \* tf.random\_normal([64]))

conv2 = tf.nn.relu(tf.nn.bias\_add(tf.nn.conv2d(conv1, w\_c2, strides=[1, 1, 1, 1], padding='SAME'), b\_c2))

conv2 = tf.nn.max\_pool(conv2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

conv2 = tf.nn.dropout(conv2, keep\_prob)

w\_c3 = tf.Variable(w\_alpha \* tf.random\_normal([3, 3, 64, 64]))

b\_c3 = tf.Variable(b\_alpha \* tf.random\_normal([64]))

conv3 = tf.nn.relu(tf.nn.bias\_add(tf.nn.conv2d(conv2, w\_c3, strides=[1, 1, 1, 1], padding='SAME'), b\_c3))

conv3 = tf.nn.max\_pool(conv3, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

conv3 = tf.nn.dropout(conv3, keep\_prob)

# Fully connected layer

w\_d = tf.Variable(w\_alpha \* tf.random\_normal([8 \* 20 \* 64, 1024]))

b\_d = tf.Variable(b\_alpha \* tf.random\_normal([1024]))

dense = tf.reshape(conv3, [-1, w\_d.get\_shape().as\_list()[0]])

dense = tf.nn.relu(tf.add(tf.matmul(dense, w\_d), b\_d))

dense = tf.nn.dropout(dense, keep\_prob)

with tf.name\_scope('w\_out'):

w\_out = tf.Variable(w\_alpha \* tf.random\_normal([1024, MAX\_CAPTCHA \* CHAR\_SET\_LEN]))

with tf.name\_scope('b\_out'):

b\_out = tf.Variable(b\_alpha \* tf.random\_normal([MAX\_CAPTCHA \* CHAR\_SET\_LEN]))

out = tf.add(tf.matmul(dense, w\_out), b\_out)

# out = tf.nn.softmax(out)

return out

# 训练

def train\_crack\_captcha\_cnn():

# with tf.device('/cpu:0'):

output = crack\_captcha\_cnn()

# loss

# loss = tf.reduce\_mean(tf.nn.softmax\_cross\_entropy\_with\_logits(output, Y))

with tf.name\_scope('loss'):

loss = tf.reduce\_mean(tf.nn.sigmoid\_cross\_entropy\_with\_logits(logits=output, labels=Y))

tf.summary.scalar('loss', loss) # 可视化loss常量

# optimizer 为了加快训练 learning\_rate应该开始大，然后慢慢衰

optimizer = tf.train.AdamOptimizer(learning\_rate=0.001).minimize(loss)

predict = tf.reshape(output, [-1, MAX\_CAPTCHA, CHAR\_SET\_LEN])

max\_idx\_p = tf.argmax(predict, 2)

max\_idx\_l = tf.argmax(tf.reshape(Y, [-1, MAX\_CAPTCHA, CHAR\_SET\_LEN]), 2)

correct\_pred = tf.equal(max\_idx\_p, max\_idx\_l)

with tf.name\_scope('accuracy'):

accuracy = tf.reduce\_mean(tf.cast(correct\_pred, tf.float32))

tf.summary.scalar('accuracy', accuracy)

saver = tf.train.Saver()

with tf.Session(config=tf.ConfigProto(log\_device\_placement=True)) as sess:

merged = tf.summary.merge\_all()

writer = tf.summary.FileWriter("log/", sess.graph)

sess.run(tf.global\_variables\_initializer())

step = 0

while True:

batch\_x, batch\_y = get\_next\_batch(64)

\_, loss\_ = sess.run([optimizer, loss], feed\_dict={X: batch\_x, Y: batch\_y, keep\_prob: 0.75})

print(step, loss\_)

# writer.add\_summary(summary,step)

# 每100 step计算一次准确率

if step % 100 == 0:

batch\_x\_test, batch\_y\_test = get\_next\_batch(100)

summary, acc = sess.run([merged, accuracy], feed\_dict={X: batch\_x\_test, Y: batch\_y\_test, keep\_prob: 1.})

print(step, acc)

writer.add\_summary(summary, step)

# 训练7000步，正常就有超80%的准确率

if step == 7000:

saver.save(sess, ".\crack\_capcha.model", global\_step=step)

break

step += 1

# performance test

# if step == 20:

# break

if \_\_name\_\_ == '\_\_main\_\_':

start = time.clock()

train\_crack\_captcha\_cnn()

end = time.clock()

print('Running time: %s Seconds' % (end - start))

**gen\_captcha.py**

from captcha.image import ImageCaptcha # pip install captcha

import numpy as np

import matplotlib.pyplot as plt

from PIL import Image

import random

# 验证码中的字符, 就不用汉字了

number = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u',

'v', 'w', 'x', 'y', 'z']

ALPHABET = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U',

'V', 'W', 'X', 'Y', 'Z']

# 验证码一般都无视大小写；验证码长度4个字符

def random\_captcha\_text(char\_set=number + alphabet + ALPHABET, captcha\_size=4):

captcha\_text = []

for i in range(captcha\_size):

c = random.choice(char\_set)

captcha\_text.append(c)

return captcha\_text

# 生成字符对应的验证码

def gen\_captcha\_text\_and\_image():

image = ImageCaptcha()

captcha\_text = random\_captcha\_text()

captcha\_text = ''.join(captcha\_text)

captcha = image.generate(captcha\_text)

# image.write(captcha\_text, captcha\_text + '.jpg') # 写到文件

captcha\_image = Image.open(captcha)

captcha\_image = np.array(captcha\_image)

return captcha\_text, captcha\_image

**vectorText.py**

from gen\_captcha import gen\_captcha\_text\_and\_image

from gen\_captcha import number

from gen\_captcha import alphabet

from gen\_captcha import ALPHABET

import time

import numpy as np

text, image = gen\_captcha\_text\_and\_image()

print("verification code iamge channel:", image.shape) # (60, 160, 3)

# 图像大小

IMAGE\_HEIGHT = 60

IMAGE\_WIDTH = 160

MAX\_CAPTCHA = len(text)

print("Max number of label:", MAX\_CAPTCHA) # 验证码最长4字符; 我全部固定为4,可以不固定. 如果验证码长度小于4，用'\_'补齐

# 把彩色图像转为灰度图像（色彩对识别验证码没有什么用）

def convert2gray(img):

if len(img.shape) > 2:

gray = np.mean(img, -1)

# 上面的转法较快，正规转法如下

# r, g, b = img[:,:,0], img[:,:,1], img[:,:,2]

# gray = 0.2989 \* r + 0.5870 \* g + 0.1140 \* b

return gray

else:

return img

"""

cnn在图像大小是2的倍数时性能最高, 如果你用的图像大小不是2的倍数，可以在图像边缘补无用像素。

np.pad(image,((2,3),(2,2)), 'constant', constant\_values=(255,)) # 在图像上补2行，下补3行，左补2行，右补2行

"""

# 文本转向量

char\_set = number + alphabet + ALPHABET + ['\_'] # 如果验证码长度小于4, '\_'用来补齐

CHAR\_SET\_LEN = len(char\_set)

def text2vec(text):

text\_len = len(text)

if text\_len > MAX\_CAPTCHA:

raise ValueError('验证码最长4个字符')

vector = np.zeros(MAX\_CAPTCHA \* CHAR\_SET\_LEN)

def char2pos(c):

if c == '\_':

k = 62

return k

k = ord(c) - 48

if k > 9:

k = ord(c) - 55

if k > 35:

k = ord(c) - 61

if k > 61:

raise ValueError('No Map')

return k

for i, c in enumerate(text):

idx = i \* CHAR\_SET\_LEN + char2pos(c)

vector[idx] = 1

return vector

# 向量转回文本

def vec2text(vec):

char\_pos = vec.nonzero()[0]

text = []

for i, c in enumerate(char\_pos):

char\_at\_pos = i # c/63

char\_idx = c % CHAR\_SET\_LEN

if char\_idx < 10:

char\_code = char\_idx + ord('0')

elif char\_idx < 36:

char\_code = char\_idx - 10 + ord('A')

elif char\_idx < 62:

char\_code = char\_idx - 36 + ord('a')

elif char\_idx == 62:

char\_code = ord('\_')

else:

raise ValueError('error')

text.append(chr(char\_code))

return "".join(text)

# 向量（大小MAX\_CAPTCHA\*CHAR\_SET\_LEN）用0,1编码 每63个编码一个字符，这样顺利有，字符也有

vec = text2vec("F5Sd")

print(vec)

text = vec2text(vec)

print(text) # F5Sd

vec = text2vec("SFd5")

print(vec)

text = vec2text(vec)

print(text) # SFd5

**test.py**

from gen\_captcha import gen\_captcha\_text\_and\_image

import matplotlib.pyplot as plt

from PIL import Image

from training import convert2gray

from training import vec2text

from training import crack\_captcha\_cnn

import time

import training as tr

import numpy as np

import tensorflow as tf

def crack\_captcha(captcha\_image):

output = crack\_captcha\_cnn()

saver = tf.train.Saver()

with tf.Session() as sess:

saver.restore(sess, tf.train.latest\_checkpoint("./"))

predict = tf.argmax(tf.reshape(output, [-1, tr.MAX\_CAPTCHA, tr.CHAR\_SET\_LEN]), 2)

text\_list = sess.run(predict, feed\_dict={tr.X: [captcha\_image], tr.keep\_prob: 1})

text = text\_list[0].tolist()

vector = np.zeros(tr.MAX\_CAPTCHA \* tr.CHAR\_SET\_LEN)

i = 0

for n in text:

vector[i \* tr.CHAR\_SET\_LEN + n] = 1

i += 1

return vec2text(vector)

if \_\_name\_\_ == '\_\_main\_\_':

start = time.clock()

text, image = gen\_captcha\_text\_and\_image()

f = plt.figure()

ax = f.add\_subplot(111)

ax.text(0.1, 1.1, text, ha='center', va='center', transform=ax.transAxes)

plt.imshow(image)

image = convert2gray(image)

image = image.flatten() / 255

predict\_text = crack\_captcha(image)

print("correct: {} predict: {}".format(text, predict\_text))

end = time.clock()

print('Running time: %s Seconds' % (end - start))

plt.show()

**Spider\_test.py**

""""""

from selenium import webdriver

import time

import re

from pprint import pprint

import json

import requests

from PIL import Image

import numpy as np

import cv2

#模拟登录慕课网

""""""

browser = webdriver.Chrome()

browser.set\_window\_size(1200, 800)

browser.get('https://www.imooc.com/#')#慕课登陆页面

button1 = browser.find\_element\_by\_id('js-signin-btn')#点击出现输入密码及账户

button1.click()

time.sleep(3)

input1 = browser.find\_element\_by\_xpath('//\*[@id="signup-form"]/div[1]/input')##如果不是唯一class用xpath

input1.send\_keys('1912490575@qq.com')

time.sleep(3)

input2 = browser.find\_element\_by\_xpath('//\*[@id="signup-form"]/div[2]/input')

input2.send\_keys('dasdqrqwea')

button2 = browser.find\_element\_by\_xpath('//\*[@id="signup-form"]/div[5]/input')

button2.click()

time.sleep(3)

# 获取截图

browser.get\_screenshot\_as\_file('screenshot.png')

# 获取指定元素位置

element = browser.find\_element\_by\_xpath('//\*[@id="signup-form"]/div[3]/a[1]/img')

left = int(element.location['x'])-38

top = int(element.location['y'])-10

right = int(element.location['x'] + element.size['width'])+38

bottom = int(element.location['y'] + element.size['height'])+10

# 通过Image处理图像

im = Image.open('screenshot.png')

im = im.crop((left, top, right, bottom))

im.save('yanzma.png')

time.sleep(1)

img = Image.open('yanzma.png')

img = np.array(img)

print(img.shape) # img.shape打印高宽 (60, 160, 3) muke(40, 84, 3)

browser.close()

#模拟登录微信公众号

"""

browser = webdriver.Chrome()

browser.get('https://mp.weixin.qq.com')

time.sleep(2)

browser.find\_element\_by\_name('account').clear() # 清空里面有的东西

input1 = browser.find\_element\_by\_name('account') # 这里没有ID是直接那么是

input1.send\_keys('1922664316@qq.com')

browser.find\_element\_by\_name('password').clear()

input2 = browser.find\_element\_by\_name('password')

input2.send\_keys('12345789bnm')

button1 = browser.find\_element\_by\_class\_name('icon\_checkbox')

button1.click()

button = browser.find\_element\_by\_class\_name('btn\_login')

button.click()

time.sleep(1)

button.click()

time.sleep(1)

button.click()

time.sleep(1)

button.click()

time.sleep(1)

# 获取截图

browser.get\_screenshot\_as\_file('screenshot.png')

# 获取指定元素位置

element = browser.find\_element\_by\_xpath('//\*[@id="header"]/div[2]/div/div/form/div[2]/img')

left = int(element.location['x'])-19

top = int(element.location['y'])-5

right = int(element.location['x'] + element.size['width'])+18

bottom = int(element.location['y'] + element.size['height'])+5

# 通过Image处理图像

im = Image.open('screenshot.png')

im = im.crop((left, top, right, bottom))

im.save('yanzma.png')

# im1 = Image.open('yanzma.png')

# im1.save('yanzma.jpg')

time.sleep(1)

img = Image.open('yanzma.png')

img = np.array(img)

# print(img)

# print(img.shape) # img.shape打印高宽 (60, 160, 3) gzh(53, 130, 3)

browser.close()

"""

#测试

""""""

from image\_processing import gen\_captcha\_text\_and\_image

import matplotlib.pyplot as plt

from PIL import Image

from training import convert2gray

from training import vec2text

from training import crack\_captcha\_cnn

import time

import training as tr

import numpy as np

import tensorflow as tf

def crack\_captcha(captcha\_image):

output = crack\_captcha\_cnn()

saver = tf.train.Saver()

with tf.Session() as sess:

saver.restore(sess, tf.train.latest\_checkpoint("./"))

predict = tf.argmax(tf.reshape(output, [-1, tr.MAX\_CAPTCHA, tr.CHAR\_SET\_LEN]), 2)

text\_list = sess.run(predict, feed\_dict={tr.X: [captcha\_image], tr.keep\_prob: 1})

text = text\_list[0].tolist()

vector = np.zeros(tr.MAX\_CAPTCHA \* tr.CHAR\_SET\_LEN)

i = 0

for n in text:

vector[i \* tr.CHAR\_SET\_LEN + n] = 1

i += 1

return vec2text(vector)

if \_\_name\_\_ == '\_\_main\_\_':

start = time.clock()

text, image = gen\_captcha\_text\_and\_image()

f = plt.figure()

ax = f.add\_subplot(111)

ax.text(0.1, 1.1, text, ha='center', va='center', transform=ax.transAxes)

plt.imshow(image)

image = convert2gray(image)

image = image.flatten() / 255

predict\_text = crack\_captcha(image)

print("correct: {} predict: {}".format(text, predict\_text))

end = time.clock()

print('Running time: %s Seconds' % (end - start))

plt.show()

**image\_processing.py**

#模拟浏览器下载验证码，失败因为会变化

"""

import urllib.request

import urllib.parse

url = 'https://www.imooc.com/passport/user/verifycode?t=1530100117665'

# 携带cookie进行访问

headers = {

'GET https': '//weibo.cn/5273088553/info HTTP/1.1',

'Host': ' www.imooc.com',

'Accept-Encoding': 'gzip, deflate, br',

'Connection':'keep-alive',

'User-Agent': ' Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.62 Safari/537.36',

'Accept': ' image/webp,image/apng,image/\*,\*/\*;q=0.8',

'Referer':'https://www.imooc.com/',

'Accept-Language': 'zh-CN,zh;q=0.9',

'Cookie': 'PHPSESSID=fj0un6n568jdf5j1bt2rud06b7; imooc\_uuid=e38efac0-eacc-4a7d-a1ad-bbe7e0bc42a2; imooc\_isnew=1; imooc\_isnew\_ct=1530099807; IMCDNS=0; Hm\_lvt\_fb538fdd5bd62072b6a984ddbc658a16=1530099812; Hm\_lvt\_f0cfcccd7b1393990c78efdeebff3968=1530099815; Hm\_lpvt\_fb538fdd5bd62072b6a984ddbc658a16=1530100095; Hm\_lpvt\_f0cfcccd7b1393990c78efdeebff3968=1530100098; cvde=5b33785fc62f0-10',

}

request = urllib.request.Request(url=url, headers=headers)

response = urllib.request.urlopen(request)

# 输出所有

# print(response.read().decode('gbk'))

# 将内容写入文件中

with open('weibo.jpg', 'wb') as fp:

fp.write(response.read())

"""

#用来查看照片的宽高

import cv2

import numpy as np

from PIL import Image

import matplotlib.pyplot as plt

def gen\_captcha\_text\_and\_image():

png = Image.open('yanzma.png')

png.load() # required for png.split()

background = Image.new("RGB", png.size, (255, 255, 255))

background.paste(png, mask=png.split()[3]) # 3 is the alpha channel

background.save('foo.jpg', 'JPEG', quality=80)

text = 'yanzma'

img1 = Image.open('foo.jpg')

img1 = np.array(img1)

plt.imshow(img1)

return text,img1

# print(img,img.shape) #img.shape打印高宽 (60, 160, 3) gzh(53, 130, 3)

#测试

if \_\_name\_\_ == '\_\_main\_\_':

text, image = gen\_captcha\_text\_and\_image()

print(image.shape)