基于TensorFlow的CAPTCHA注册码识别实践

武汉大学 谷歌 联合实验室

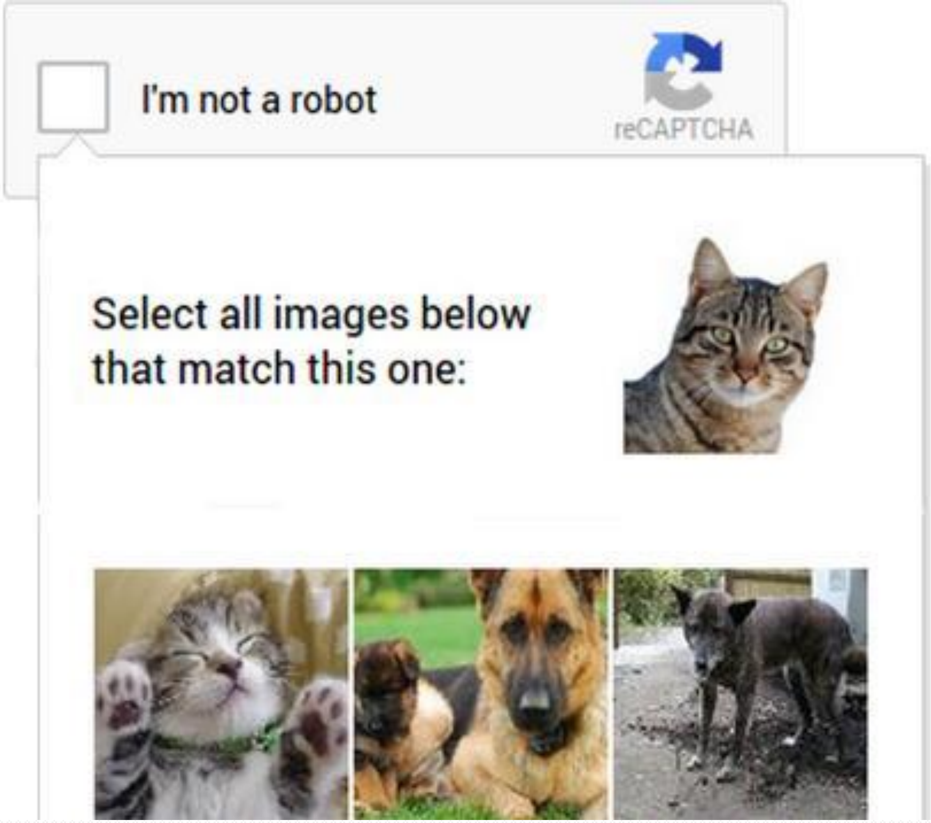
目录

- CAPTCHA 库及数据集
- CAPTCHA注册码识别实践
  - 生成注册码
  - 定义卷积神经网络
  - 训练网络模型参数
  - 测试网络效果
- 调试技巧：模型参数存储与加载
- 小结

CAPTCHA 库

简介：Captcha（全自动区分计算机和人类的图灵测试，俗称验证码）是目前用于区分人和机器主要办法，其工作原理是通过提供模糊或是有歧义的图片，并要求用户进行回答，以此来区分人和机器

下图为Google目前采用的验证码形式



□ I'm not a robot

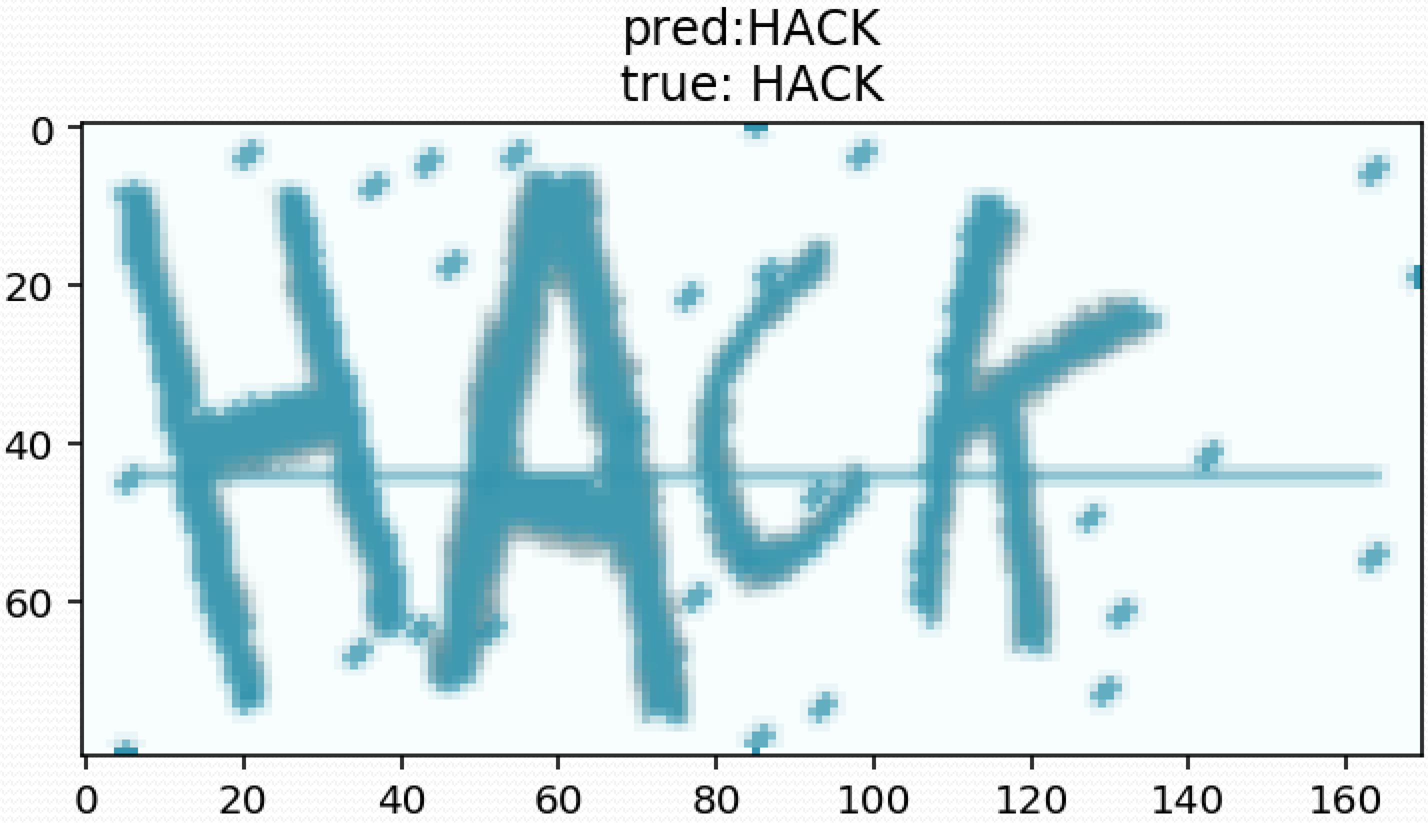
Select all images below that match this one:

CAPTCHA 库

简介：同时，captcha 是用 python 写的生成验证码的库，它支持图片验证码和语音验证码

图片像素、字符个数均可指定

外观：



CAPTCHA 库

用例:

```
from captcha.image import ImageCaptcha


#生成一张图片
image = ImageCaptcha()

#生成一个字符
 captcha_text = random_captcha_text()
```

目录

- CAPTCHA 库及数据集
- CAPTCHA注册码识别实践
  - 生成注册码
  - 定义卷积神经网络
  - 训练网络模型参数
  - 测试网络效果
- 调试技巧：模型参数存储与加载
- 小结

生成注册码

```
from captcha.image import ImageCaptcha  # pip install captcha
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
 #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) #连接字符串

  captcha = image.generate(captcha_text)

  captcha_image = Image.open(captcha)
  captcha_image = np.array(captcha_image)
  return captcha_text, captcha_image
```

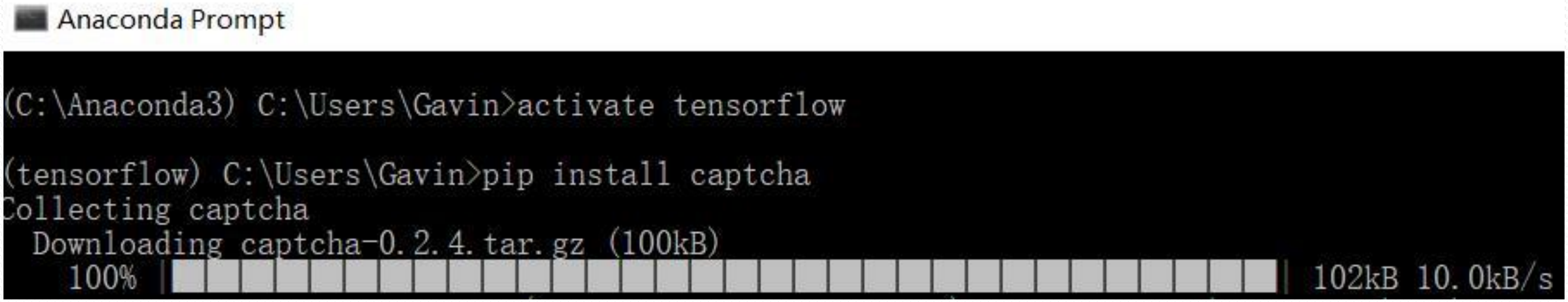
课程代码：Machine_Learning\04_CAPTCHA_demo\ cnn_data.py

生成注册码

cnn_data.py文件可单独运行：

通过cmd命令行界面 或 Anaconda Prompt切换到工作目录， 输入 python cnn_data.py

如果报错并提示找不到captcha， 则在Prompt下安装该库: pip install captcha

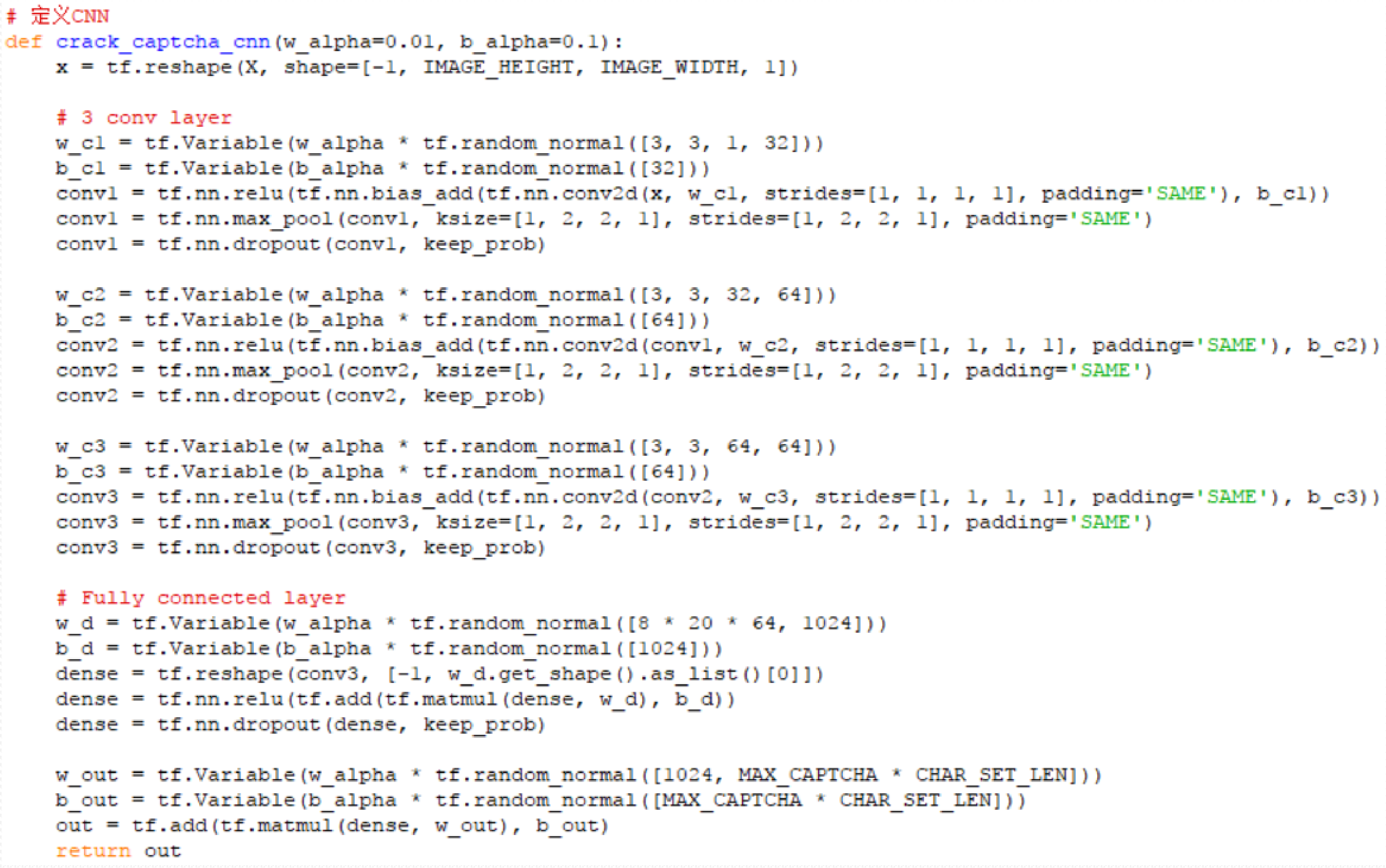


```
(C:\Anaconda3) C:\Users\Gavin>activate tensorflow

(tensorflow) C:\Users\Gavin>pip install captcha
Collecting captcha
  Downloading captcha-0.2.4.tar.gz (100kB)
    100% |████████████████████████████████| 102kB 10.0kB/s
```

目录

- CAPTCHA 库及数据集
- CAPTCHA注册码识别实践
  - 生成注册码
  - 定义卷积神经网络
  - 训练网络模型参数
  - 测试网络效果
- 调试技巧：模型参数存储与加载
- 小结

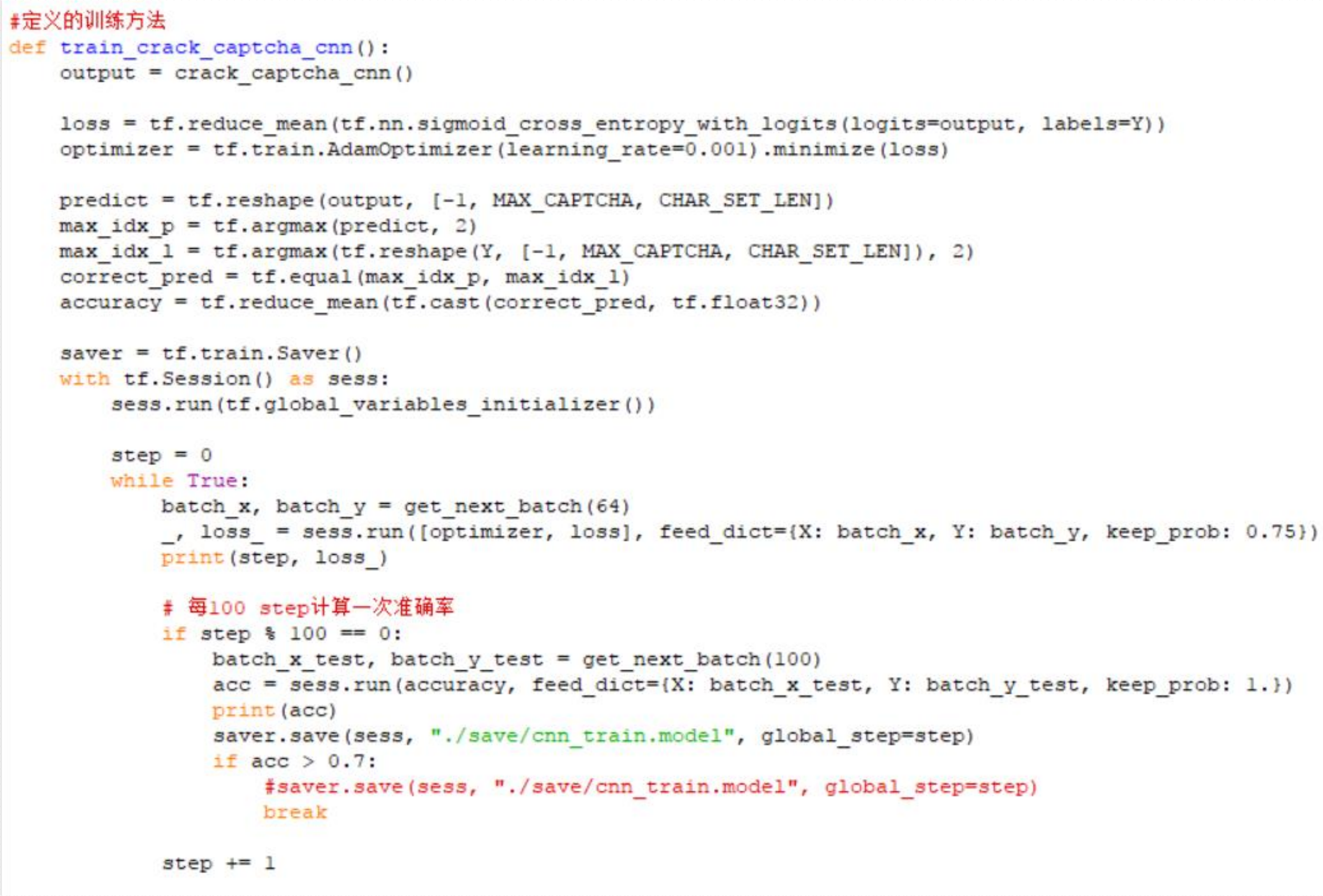
定义卷积神经网络



```
# 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])

    # 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)

    w_out = tf.Variable(w_alpha * tf.random_normal([1024, MAX_CAPTCHA * CHAR_SET_LEN]))
    b_out = tf.Variable(b_alpha * tf.random_normal([MAX_CAPTCHA * CHAR_SET_LEN]))
    out = tf.add(tf.matmul(dense, w_out), b_out)
    return out
```

课程代码：Machine_Learning\04_CAPTCHA_demo\ cnn_train.py

训练网络模型参数



```
#定义的训练方法
def train_crack_captcha_cnn():
    output = crack_captcha_cnn()

    loss = tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(logits=output, labels=Y))
    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)
    accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

    saver = tf.train.Saver()
    with tf.Session() as sess:
        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_)

            # 每100 step计算一次准确率
            if step % 100 == 0:
                batch_x_test, batch_y_test = get_next_batch(100)
                acc = sess.run(accuracy, feed_dict={X: batch_x_test, Y: batch_y_test, keep_prob: 1.})
                print(acc)
                saver.save(sess, "./save/cnn_train.model", global_step=step)
                if acc > 0.7:
                    #saver.save(sess, "./save/cnn_train.model", global_step=step)
                    break

            step += 1
```

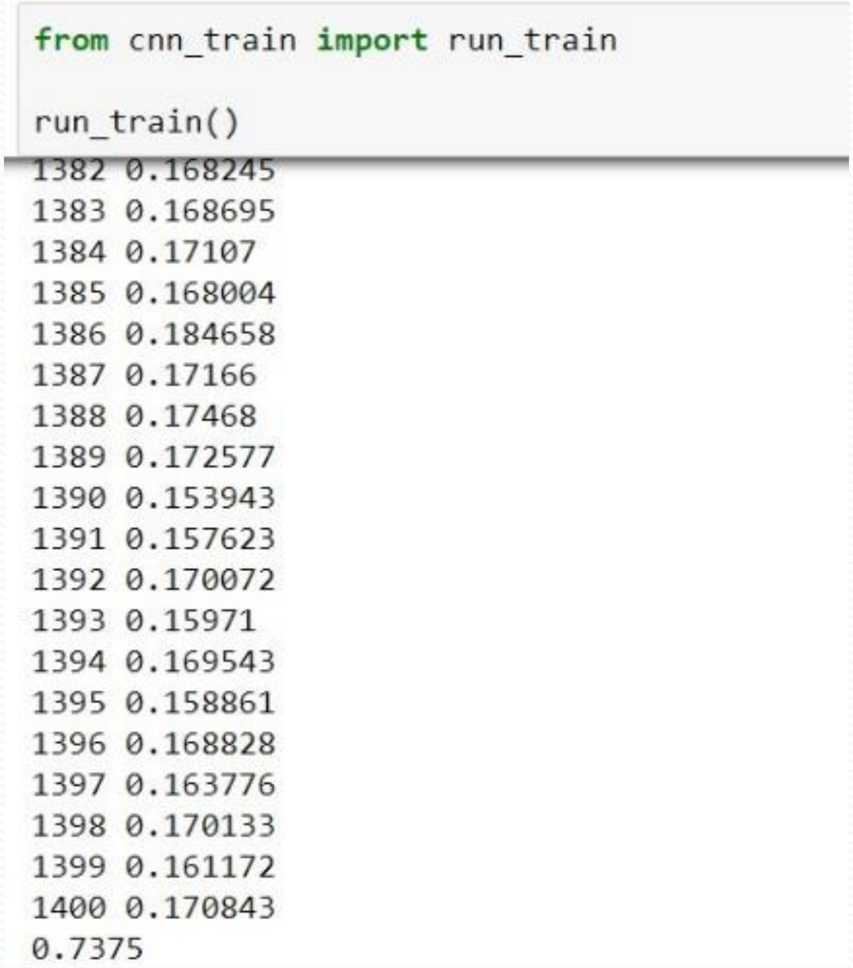
课程代码：Machine_Learning\04_CAPTCHA_demo\ cnn_train.py

目录

- CAPTCHA 库及数据集
- CAPTCHA注册码识别实践
    - 生成注册码
    - 定义卷积神经网络
    - 训练网络模型参数
    - 测试网络效果
- 调试技巧：模型参数存储与加载
- 小结

训练网络模型参数

预计训练用时30分钟

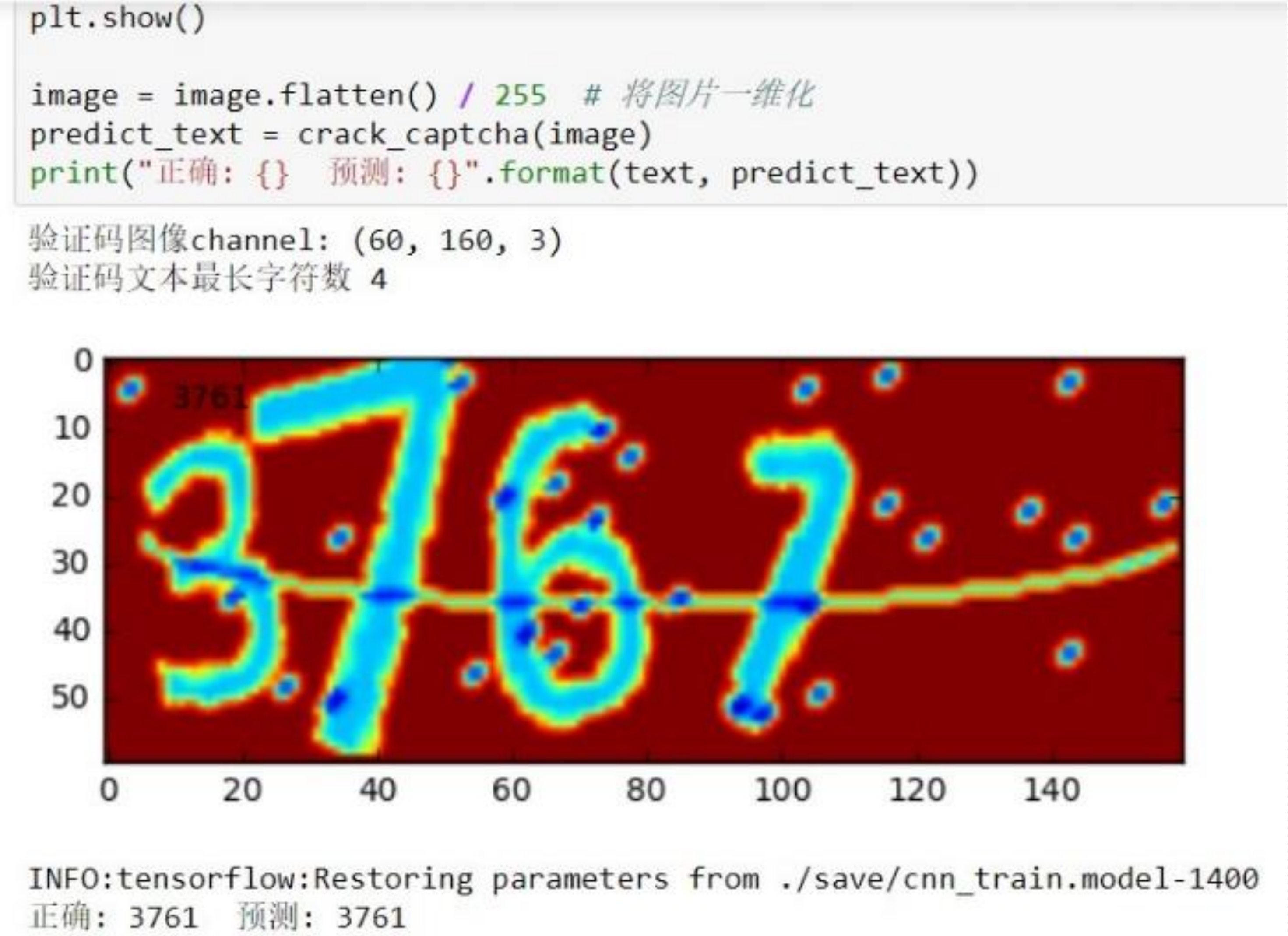


```
from cnn_train import run_train

run_train()
```

```
1382 0.168245
1383 0.168695
1384 0.17107
1385 0.168004
1386 0.184658
1387 0.17166
1388 0.17468
1389 0.172577
1390 0.153943
1391 0.157623
1392 0.170072
1393 0.15971
1394 0.169543
1395 0.158861
1396 0.168828
1397 0.163776
1398 0.170133
1399 0.161172
1400 0.170843
0.7375
```

课程代码：Machine_Learning\04_CAPTCHA_demo\ run_cnn_train.ipynb

测试网络效果



```
plt.show()

image = image.flatten() / 255  # 将图片一维化
predict_text = crack_captcha(image)
print("正确: {}  预测: {}".format(text, predict_text))
```

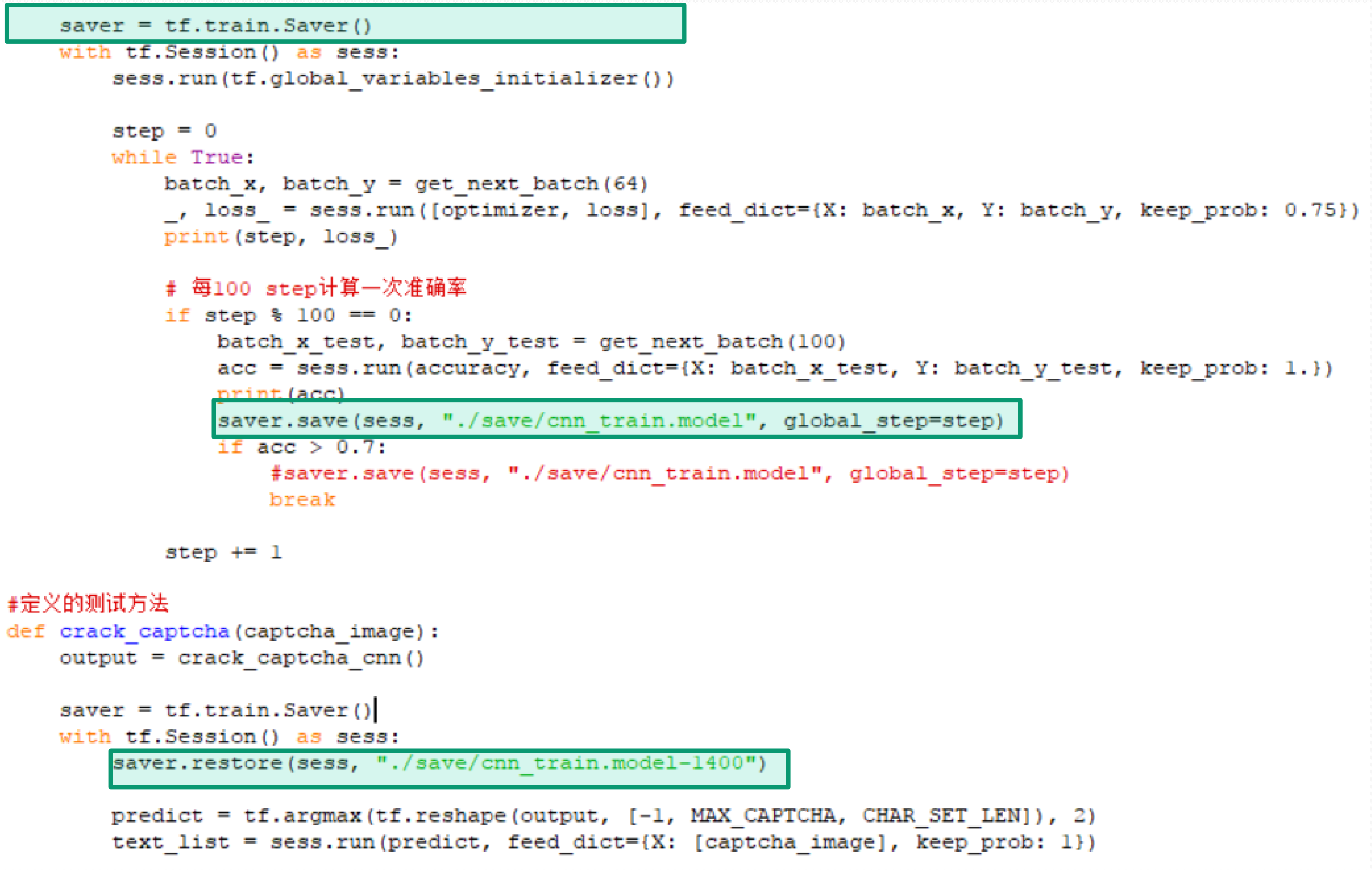
```
验证码图像channel: (60, 160, 3)
验证码文本最长字符数 4
```

```
INFO:tensorflow:Restoring parameters from ./save/cnn_train.model-1400
正确: 3761  预测: 3761
```

目录

- CAPTCHA 库及数据集
- CAPTCHA注册码识别实践
  - 生成注册码
  - 定义卷积神经网络
  - 训练网络模型参数
  - 测试网络效果
- 调试技巧：模型参数存储与加载
- 小结

模型加载与存储



```
    saver = tf.train.Saver()
    with tf.Session() as sess:
        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_)

            # 每100 step计算一次准确率
            if step % 100 == 0:
                batch_x_test, batch_y_test = get_next_batch(100)
                acc = sess.run(accuracy, feed_dict={X: batch_x_test, Y: batch_y_test, keep_prob: 1.})
                print(acc)
                saver.save(sess, "./save/cnn_train.model", global_step=step)
                if acc > 0.7:
                    #saver.save(sess, "./save/cnn_train.model", global_step=step)
                    break

            step += 1

#定义的测试方法
def crack_captcha(captcha_image):
    output = crack_captcha_cnn()

    saver = tf.train.Saver()
    with tf.Session() as sess:
        saver.restore(sess, "./save/cnn_train.model-1400")

        predict = tf.argmax(tf.reshape(output, [-1, MAX_CAPTCHA, CHAR_SET_LEN]), 2)
        text_list = sess.run(predict, feed_dict={X: [captcha_image], keep_prob: 1})
```

小结

- 一个完整的神经网络构建过程：

  生成训练数据集

  定义网络模型

  训练模型参数并保存

  读入参数并测试新数据

Thanks