简介

任务时间：时间未知

传统的验证码识别算法一般需要把验证码分割为单个字符，然后逐个识别。本教程将验证码识别问题转化为分类的问题，实现对验证码进行整体识别。

步骤简介

本教程一共分为四个部分

* generate\_captcha.py - 利用 Captcha 库生成验证码；
* captcha\_model.py - CNN 模型；
* train\_captcha.py - 训练 CNN 模型；
* predict\_captcha.py - 识别验证码。

数据学习

任务时间：时间未知

安装 captcha 库

pip install captcha

获取训练数据

本教程使用的验证码由数字、大写字母、小写字母组成，每个验证码包含 4 个字符，总共有 62^4 种组合，所以一共有 62^4 种不同的验证码。

**示例代码：**

现在您可以在 /home/ubuntu 目录下创建源文件 generate\_captcha.py，内容可参考：

**示例代码：/home/ubuntu/generate\_captcha.py**

#-\*- coding:utf-8 -\*-

from captcha.image import ImageCaptcha

from PIL import Image

import numpy as np

import random

import string

class generateCaptcha():

def \_\_init\_\_(self,

width = 160,#验证码图片的宽

height = 60,#验证码图片的高

char\_num = 4,#验证码字符个数

characters = string.digits + string.ascii\_uppercase + string.ascii\_lowercase):#验证码组成，数字+大写字母+小写字母

self.width = width

self.height = height

self.char\_num = char\_num

self.characters = characters

self.classes = len(characters)

def gen\_captcha(self,batch\_size = 50):

X = np.zeros([batch\_size,self.height,self.width,1])

img = np.zeros((self.height,self.width),dtype=np.uint8)

Y = np.zeros([batch\_size,self.char\_num,self.classes])

image = ImageCaptcha(width = self.width,height = self.height)

while True:

for i in range(batch\_size):

captcha\_str = ''.join(random.sample(self.characters,self.char\_num))

img = image.generate\_image(captcha\_str).convert('L')

img = np.array(img.getdata())

X[i] = np.reshape(img,[self.height,self.width,1])/255.0

for j,ch in enumerate(captcha\_str):

Y[i,j,self.characters.find(ch)] = 1

Y = np.reshape(Y,(batch\_size,self.char\_num\*self.classes))

yield X,Y

def decode\_captcha(self,y):

y = np.reshape(y,(len(y),self.char\_num,self.classes))

return ''.join(self.characters[x] for x in np.argmax(y,axis = 2)[0,:])

def get\_parameter(self):

return self.width,self.height,self.char\_num,self.characters,self.classes

def gen\_test\_captcha(self):

image = ImageCaptcha(width = self.width,height = self.height)

captcha\_str = ''.join(random.sample(self.characters,self.char\_num))

img = image.generate\_image(captcha\_str)

img.save(captcha\_str + '.jpg')

X = np.zeros([1,self.height,self.width,1])

Y = np.zeros([1,self.char\_num,self.classes])

img = img.convert('L')

img = np.array(img.getdata())

X[0] = np.reshape(img,[self.height,self.width,1])/255.0

for j,ch in enumerate(captcha\_str):

Y[0,j,self.characters.find(ch)] = 1

Y = np.reshape(Y,(1,self.char\_num\*self.classes))

return X,Y

理解训练数据

* X：一个 mini-batch 的训练数据，其 shape 为 [ batch\_size, height, width, 1 ]，batch\_size 表示每批次多少个训练数据，height 表示验证码图片的高，width 表示验证码图片的宽，1 表示图片的通道。
* Y：X 中每个训练数据属于哪一类验证码，其形状为 [ batch\_size, class ] ，对验证码中每个字符进行 One-Hot 编码，所以 class 大小为 4\*62。

**执行:**

* 获取验证码和对应的分类

cd /home/ubuntu;

python

from generate\_captcha import generateCaptcha

g = generateCaptcha()

X,Y = g.gen\_test\_captcha()

* 查看训练数据

X.shape

Y.shape

可以在 /home/ubuntu 目录下查看生成的验证码，jpg 格式的图片可以点击查看。

模型学习

任务时间：时间未知

CNN 模型

总共 5 层网络，前 3 层为卷积层，第 4、5 层为全连接层。对 4 层隐藏层都进行 dropout。网络结构如下所示： input——>conv——>pool——>dropout——>conv——>pool——>dropout——>conv——>pool——>dropout——>fully connected layer——>dropout——>fully connected layer——>output

**示例代码：**

现在您可以在 /home/ubuntu 目录下创建源文件 captcha\_model.py，内容可参考：

**示例代码：/home/ubuntu/captcha\_model.py**

# -\*- coding: utf-8 -\*

import tensorflow as tf

import math

class captchaModel():

def \_\_init\_\_(self,

width = 160,

height = 60,

char\_num = 4,

classes = 62):

self.width = width

self.height = height

self.char\_num = char\_num

self.classes = classes

def conv2d(self,x, W):

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

def max\_pool\_2x2(self,x):

return tf.nn.max\_pool(x, ksize=[1, 2, 2, 1],

strides=[1, 2, 2, 1], padding='SAME')

def weight\_variable(self,shape):

initial = tf.truncated\_normal(shape, stddev=0.1)

return tf.Variable(initial)

def bias\_variable(self,shape):

initial = tf.constant(0.1, shape=shape)

return tf.Variable(initial)

def create\_model(self,x\_images,keep\_prob):

#first layer

w\_conv1 = self.weight\_variable([5, 5, 1, 32])

b\_conv1 = self.bias\_variable([32])

h\_conv1 = tf.nn.relu(tf.nn.bias\_add(self.conv2d(x\_images, w\_conv1), b\_conv1))

h\_pool1 = self.max\_pool\_2x2(h\_conv1)

h\_dropout1 = tf.nn.dropout(h\_pool1,keep\_prob)

conv\_width = math.ceil(self.width/2)

conv\_height = math.ceil(self.height/2)

#second layer

w\_conv2 = self.weight\_variable([5, 5, 32, 64])

b\_conv2 = self.bias\_variable([64])

h\_conv2 = tf.nn.relu(tf.nn.bias\_add(self.conv2d(h\_dropout1, w\_conv2), b\_conv2))

h\_pool2 = self.max\_pool\_2x2(h\_conv2)

h\_dropout2 = tf.nn.dropout(h\_pool2,keep\_prob)

conv\_width = math.ceil(conv\_width/2)

conv\_height = math.ceil(conv\_height/2)

#third layer

w\_conv3 = self.weight\_variable([5, 5, 64, 64])

b\_conv3 = self.bias\_variable([64])

h\_conv3 = tf.nn.relu(tf.nn.bias\_add(self.conv2d(h\_dropout2, w\_conv3), b\_conv3))

h\_pool3 = self.max\_pool\_2x2(h\_conv3)

h\_dropout3 = tf.nn.dropout(h\_pool3,keep\_prob)

conv\_width = math.ceil(conv\_width/2)

conv\_height = math.ceil(conv\_height/2)

#first fully layer

conv\_width = int(conv\_width)

conv\_height = int(conv\_height)

w\_fc1 = self.weight\_variable([64\*conv\_width\*conv\_height,1024])

b\_fc1 = self.bias\_variable([1024])

h\_dropout3\_flat = tf.reshape(h\_dropout3,[-1,64\*conv\_width\*conv\_height])

h\_fc1 = tf.nn.relu(tf.nn.bias\_add(tf.matmul(h\_dropout3\_flat, w\_fc1), b\_fc1))

h\_fc1\_drop = tf.nn.dropout(h\_fc1, keep\_prob)

#second fully layer

w\_fc2 = self.weight\_variable([1024,self.char\_num\*self.classes])

b\_fc2 = self.bias\_variable([self.char\_num\*self.classes])

y\_conv = tf.add(tf.matmul(h\_fc1\_drop, w\_fc2), b\_fc2)

return y\_conv

训练 CNN 模型

每批次采用 64 个训练样本，每 100 次循环采用 100 个测试样本检查识别准确度，当准确度大于 99% 时，训练结束，采用 GPU 需要 4-5 个小时左右，CPU 大概需要 20 个小时左右。

**示例代码：**

现在您可以在 /home/ubuntu 目录下创建源文件 train\_captcha.py，内容可参考：

**示例代码：/home/ubuntu/train\_captcha.py**

#-\*- coding:utf-8 -\*-

import tensorflow as tf

import numpy as np

import string

import generate\_captcha

import captcha\_model

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

captcha = generate\_captcha.generateCaptcha()

width,height,char\_num,characters,classes = captcha.get\_parameter()

x = tf.placeholder(tf.float32, [None, height,width,1])

y\_ = tf.placeholder(tf.float32, [None, char\_num\*classes])

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

model = captcha\_model.captchaModel(width,height,char\_num,classes)

y\_conv = model.create\_model(x,keep\_prob)

cross\_entropy = tf.reduce\_mean(tf.nn.sigmoid\_cross\_entropy\_with\_logits(labels=y\_,logits=y\_conv))

train\_step = tf.train.AdamOptimizer(1e-4).minimize(cross\_entropy)

predict = tf.reshape(y\_conv, [-1,char\_num, classes])

real = tf.reshape(y\_,[-1,char\_num, classes])

correct\_prediction = tf.equal(tf.argmax(predict,2), tf.argmax(real,2))

correct\_prediction = tf.cast(correct\_prediction, tf.float32)

accuracy = tf.reduce\_mean(correct\_prediction)

saver = tf.train.Saver()

with tf.Session() as sess:

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

step = 1

while True:

batch\_x,batch\_y = next(captcha.gen\_captcha(64))

\_,loss = sess.run([train\_step,cross\_entropy],feed\_dict={x: batch\_x, y\_: batch\_y, keep\_prob: 0.75})

print ('step:%d,loss:%f' % (step,loss))

if step % 100 == 0:

batch\_x\_test,batch\_y\_test = next(captcha.gen\_captcha(100))

acc = sess.run(accuracy, feed\_dict={x: batch\_x\_test, y\_: batch\_y\_test, keep\_prob: 1.})

print ('###############################################step:%d,accuracy:%f' % (step,acc))

if acc > 0.99:

saver.save(sess,"./capcha\_model.ckpt")

break

step += 1

**然后执行:**

cd /home/ubuntu;

python train\_captcha.py

**执行结果：**

step:75193,loss:0.010931

step:75194,loss:0.012859

step:75195,loss:0.008747

step:75196,loss:0.009147

step:75197,loss:0.009351

step:75198,loss:0.009746

step:75199,loss:0.010014

step:75200,loss:0.009024

###############################################step:75200,accuracy:0.992500

**使用训练好的模型：**

作为实验，你可以通过调整 train\_captcha.py 文件中 if acc > 0.99: 代码行的准确度节省训练时间(比如将 0.99 为 0.01)，体验训练过程；我们已经通过长时间的训练得到了一个训练好的模型，可以通过如下命令将训练集下载到本地。

wget http://tensorflow-1253902462.cosgz.myqcloud.com/captcha/capcha\_model.zip

unzip -o capcha\_model.zip

识别验证码

**测试数据集：**

我们在腾讯云的 COS 上准备了 100 个验证码作为测试集，使用 wget 命令获取：

wget http://tensorflow-1253902462.cosgz.myqcloud.com/captcha/captcha.zip

unzip -q captcha.zip

**示例代码：**

现在您可以在 /home/ubuntu 目录下创建源文件 predict\_captcha.py，内容可参考：

**示例代码：/home/ubuntu/predict\_captcha.py**

#-\*- coding:utf-8 -\*-

from PIL import Image, ImageFilter

import tensorflow as tf

import numpy as np

import string

import sys

import generate\_captcha

import captcha\_model

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

captcha = generate\_captcha.generateCaptcha()

width,height,char\_num,characters,classes = captcha.get\_parameter()

gray\_image = Image.open(sys.argv[1]).convert('L')

img = np.array(gray\_image.getdata())

test\_x = np.reshape(img,[height,width,1])/255.0

x = tf.placeholder(tf.float32, [None, height,width,1])

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

model = captcha\_model.captchaModel(width,height,char\_num,classes)

y\_conv = model.create\_model(x,keep\_prob)

predict = tf.argmax(tf.reshape(y\_conv, [-1,char\_num, classes]),2)

init\_op = tf.global\_variables\_initializer()

saver = tf.train.Saver()

gpu\_options = tf.GPUOptions(per\_process\_gpu\_memory\_fraction=0.95)

with tf.Session(config=tf.ConfigProto(log\_device\_placement=False,gpu\_options=gpu\_options)) as sess:

sess.run(init\_op)

saver.restore(sess, "capcha\_model.ckpt")

pre\_list = sess.run(predict,feed\_dict={x: [test\_x], keep\_prob: 1})

for i in pre\_list:

s = ''

for j in i:

s += characters[j]

print(s)

**然后执行:**

cd /home/ubuntu;

python predict\_captcha.py captcha/0hWn.jpg

**执行结果：**

0hWn

完成实验

任务时间：时间未知

实验内容已完成