Captcha_CNN

Machine Learning EL-GY 6132

Mingdi Han mh4928
Mo Wu mw3774

In [2]:

```
pip install captcha
```

```
Collecting captcha
  Downloading https://files.pythonhosted.org/packages/90/fe/d4ddf1e657
6073b5eaea76e9b2afa022c626212a30c871968480be3ccb7b/captcha-0.3-py3-non
e-any.whl (https://files.pythonhosted.org/packages/90/fe/d4ddf1e657607
3b5eaea76e9b2afa022c626212a30c871968480be3ccb7b/captcha-0.3-py3-none-a
ny.whl) (101kB)
    |████████████████████████████████| 102kB 8.3
MB/s
Requirement already satisfied: Pillow in /usr/local/lib/python3.6/dist
-packages (from captcha) (4.3.0)
Requirement already satisfied: olefile in /usr/local/lib/python3.6/dis
t-packages (from Pillow->captcha) (0.46)
Installing collected packages: captcha
Successfully installed captcha-0.3
```

In [1]:

```
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

```
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 12471404877416461321
, name: "/device:XLA_CPU:0"
device_type: "XLA_CPU"
memory_limit: 17179869184
locality {
}
incarnation: 13296973567740588015
physical_device_desc: "device: XLA_CPU device"
, name: "/device:XLA_GPU:0"
device_type: "XLA_GPU"
memory_limit: 17179869184
locality {
}
incarnation: 4937162778655325005
physical_device_desc: "device: XLA_GPU device"
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 14800692839
locality {
  bus_id: 1
  links {
  }
}
incarnation: 10809061159631309841
physical_device_desc: "device: 0, name: Tesla T4, pci bus id: 0000:00:
04.0, compute capability: 7.5"
]
```

In [0]:

```
import random
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from captcha.image import ImageCaptcha
import os
import tensorflow as tf
```

In [0]:

```
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',

ALPHABET = ['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R',

CAPTCHA_LIST = Number + alphabet + ALPHABET

CAPTCHA_LEN = 4

CAPTCHA_HEIGHT = 60

CAPTCHA_WIDTH = 160
```

In [0]:

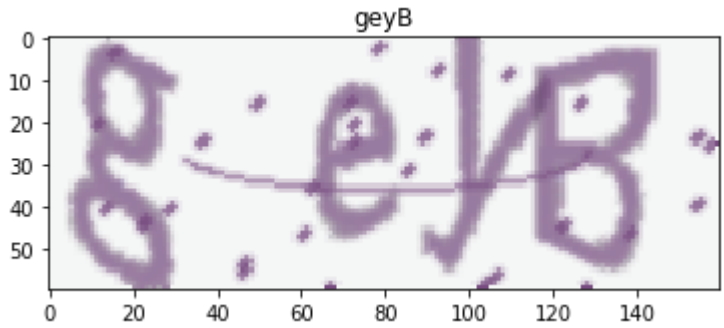
```
def random_captcha_text(char_set= CAPTCHA_LIST, captcha_size=CAPTCHA_LEN):
    captcha_text = []
    for i in range(captcha_size):
        c = random.choice(char_set)
        captcha_text.append(c)
    return captcha_text

def gen_captcha_text_and_image(width=CAPTCHA_WIDTH, height=CAPTCHA_HEIGHT,save=None)
    image = ImageCaptcha(width=width, height=height)
    captcha_text = random_captcha_text()
    captcha_text = ''.join(captcha_text)
    captcha = image.generate(captcha_text)
    if save: image.write(captcha_text, captcha_text + '.jpg')
    captcha_image = Image.open(captcha)
    captcha_image = np.array(captcha_image)

    return captcha_image, captcha_text
```

In [5]:

```
img,text = gen_captcha_text_and_image(CAPTCHA_WIDTH, CAPTCHA_HEIGHT, save=False)
plt.imshow(img)
plt.title(text)
```

Out[5]:

```
Text(0.5, 1.0, 'geyB')
```



In [0]:

```
def text2vec(text, captcha_len=CAPTCHA_LEN, captcha_list=CAPTCHA_LIST):
    text_len = len(text)
    if text_len > captcha_len:
        raise ValueError('Error in Length')
    vector = np.zeros(captcha_len * len(captcha_list))
    for i in range(text_len):
        vector[captcha_list.index(text[i])+i*len(captcha_list)] = 1
    return vector


def vec2text(vec, captcha_list=CAPTCHA_LIST, size=CAPTCHA_LEN):
    vec_idx = vec
    text_list = [captcha_list[v] for v in vec_idx]
    return ''.join(text_list)



def get_next_batch(batch_size, width=CAPTCHA_WIDTH, height=CAPTCHA_HEIGHT):
    batch_x = np.zeros([batch_size, width * height])
    batch_y = np.zeros([batch_size, CAPTCHA_LEN * len(CAPTCHA_LIST)])
    for i in range(batch_size):
        image,text = gen_captcha_text_and_image()
        if len(image.shape) > 2:
            image = np.mean(image, -1)
        batch_x[i, :] = image.flatten()/255
        batch_y[i, :] = text2vec(text)
    return batch_x, batch_y
```

In [0]:

```
X, y = get_next_batch(batch_size = 1)
```

In [0]:

```
def cnn_graph(x, keep_prob, size, captcha_list=CAPTCHA_LIST, captcha_len=CAPTCHA_LEN
    # reshape image
    image_height, image_width = size
    x_image = tf.reshape(x, shape=[-1, image_height, image_width, 1])

    # conv1
    w_conv1 = tf.Variable(w_alpha * tf.random_normal([3, 3, 1, 32]), name='w_conv1')
    b_conv1 = tf.Variable(b_alpha * tf.random_normal([32]),name='b_conv1')
    #  activition function :ReLU
    h_conv1 = tf.nn.relu(tf.nn.bias_add(tf.nn.conv2d(x_image, w_conv1, strides=[1, 1
    # pooling
    h_pool1 = tf.nn.max_pool(h_conv1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padd
    # dropout
    h_drop1 = tf.nn.dropout(h_pool1, keep_prob)

    # convc2
    #w_conv2 = weight_variable([3, 3, 32, 64])
    #b_conv2 = bias_variable([64])
    w_conv2 = tf.Variable(w_alpha * tf.random_normal([3, 3, 32, 64]),name='w_conv2')
    b_conv2 = tf.Variable(b_alpha * tf.random_normal([64]),name='b_conv2')
    h_conv2 = tf.nn.relu(tf.nn.bias_add(tf.nn.conv2d(h_drop1, w_conv2, strides=[1, 1
    h_pool2 = tf.nn.max_pool(h_conv2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padd
    h_drop2 = tf.nn.dropout(h_pool2, keep_prob)

    # conv3
    #w_conv3 = weight_variable([3, 3, 64, 64])
    #b_conv3 = bias_variable([64])
    w_conv3 = tf.Variable(w_alpha * tf.random_normal([3, 3, 64, 64]),name='w_conv3')
    b_conv3 = tf.Variable(b_alpha * tf.random_normal([64]),name='b_conv3')
    h_conv3 = tf.nn.relu(tf.nn.bias_add(tf.nn.conv2d(h_drop2, w_conv3, strides=[1, 1
    h_pool3 = tf.nn.max_pool(h_conv3, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padd
    h_drop3 = tf.nn.dropout(h_pool3, keep_prob)

    # fully connection
    image_height = int(h_drop3.shape[1])
    image_width = int(h_drop3.shape[2])
    w_fc = tf.Variable(w_alpha * tf.random_normal([image_height*image_width*64, 1024
    b_fc = tf.Variable(w_alpha * tf.random_normal([1024]),name = 'b_fc')
    h_drop3_re = tf.reshape(h_drop3, [-1, image_height*image_width*64])
    h_fc = tf.nn.relu(tf.add(tf.matmul(h_drop3_re, w_fc), b_fc))
    h_drop_fc = tf.nn.dropout(h_fc, keep_prob)

    # output
    #w_out = weight_variable([1024, len(captcha_list)*captcha_len])
    #b_out = bias_variable([len(captcha_list)*captcha_len])
    w_out = tf.Variable(w_alpha * tf.random_normal([1024, len(captcha_list)*captcha_
    b_out = tf.Variable(w_alpha * tf.random_normal([len(captcha_list)*captcha_len]),
    y_conv = tf.add(tf.matmul(h_drop_fc, w_out), b_out)

    variables_dict = {'w_conv1': w_conv1, 'b_conv1': b_conv1,'w_conv2': w_conv2, 'b_
                      'w_conv3': w_conv3, 'b_conv3': b_conv3 ,'w_fc': w_fc, 'b_fc':

    return y_conv, variables_dict



# 最小化loss
```

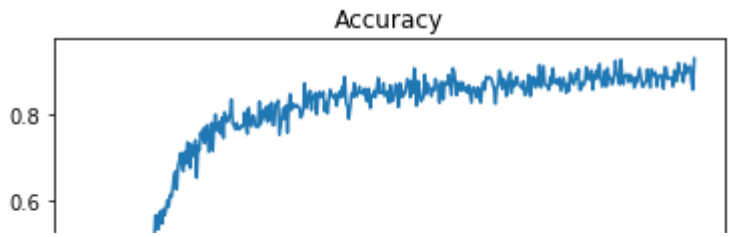
```
def optimize_graph(y, y_conv):

    loss = tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(logits=y_conv, lab
    #
    optimizer = tf.train.AdamOptimizer(learning_rate=0.001).minimize(loss)

    return optimizer



# 偏差计算

def accuracy_graph(y, y_conv, width=len(CAPTCHA_LIST), height=CAPTCHA_LEN):

    predict = tf.reshape(y_conv, [-1, height, width])
    max_predict_idx = tf.argmax(predict, 2)

    # labels
    label = tf.reshape(y, [-1, height, width])
    max_label_idx = tf.argmax(label, 2)
    correct_p = tf.equal(max_predict_idx, max_label_idx)
    accuracy = tf.reduce_mean(tf.cast(correct_p, tf.float32))

    return accuracy

def train(height=CAPTCHA_HEIGHT, width=CAPTCHA_WIDTH, y_size=len(CAPTCHA_LIST)*CAPTC
    acc_rate = 0.92
    #pre-occupied
    x = tf.placeholder(tf.float32, [None, height * width])
    y = tf.placeholder(tf.float32, [None, y_size])
    keep_prob = tf.placeholder(tf.float32) # prevent from overfit ,only use in trai

    # CNN model
    y_conv ,var_dict= cnn_graph(x, keep_prob, (height, width))
    #
    optimizer = optimize_graph(y, y_conv)
    #
    accuracy = accuracy_graph(y, y_conv)

    #begin
    saver = tf.train.Saver(var_dict)
    sess = tf.Session()
    sess.run(tf.global_variables_initializer())
    step = 0
    Acc=[]
    xplt=[]

    while 1:
        batch_x, batch_y = get_next_batch(64)
        sess.run(optimizer, feed_dict={x: batch_x, y: batch_y, keep_prob: 0.75})
        #
        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, ke
            Acc.append(acc)
            xplt.append(step)


            print(' step:', step, ' accuracy:', acc)

            # save model when satisfied
```

```
            if acc >= acc_rate:
                model_path = os.getcwd() + os.sep + str(acc_rate) + "captcha.model"
                saver.save(sess, model_path, global_step=step)
                acc_rate += 0.01
                if acc_rate > 0.93:
                    break

        step += 1
    plt.plot(Acc)
    plt.title('Accuracy')
    plt.show()

    sess.close()
```

In [9]:

```
train()
```

```
 step: 44600  accuracy: 0.9
 step: 44700  accuracy: 0.8975
 step: 44800  accuracy: 0.8875
 step: 44900  accuracy: 0.875
 step: 45000  accuracy: 0.9175
 step: 45100  accuracy: 0.895
 step: 45200  accuracy: 0.9075
 step: 45300  accuracy: 0.91
 step: 45400  accuracy: 0.9125
 step: 45500  accuracy: 0.8775
 step: 45600  accuracy: 0.8575
 step: 45700  accuracy: 0.93
```



In [0]:

```
def captcha_decode(image_list, height=CAPTCHA_HEIGHT, width=CAPTCHA_WIDTH):
  x = tf.placeholder(tf.float32, [None, height * width])
  keep_prob = tf.placeholder(tf.float32)
  y_conv, var_dict = cnn_graph(x, keep_prob, (height, width))
  saver = tf.train.Saver(var_dict)

  #saver = tf.train.import_meta_graph(os.getcwd()+'/0.02captcha.model-500.meta')
  with tf.Session() as sess:
    saver.restore(sess, tf.train.latest_checkpoint('.'))
    predict = tf.argmax(tf.reshape(y_conv, [-1, CAPTCHA_LEN, len(CAPTCHA_LIST)]), 2)
    vector_list = sess.run(predict, feed_dict={x: image_list, keep_prob: 1})
    vector_list = vector_list.tolist()
    text_list = [vec2text(vector) for vector in vector_list]

  return text_list
```

In [20]:

```
from tensorflow.python import pywrap_tensorflow

reader = pywrap_tensorflow.NewCheckpointReader('/content/0.93captcha.model-45700')
var_to_shape_map = reader.get_variable_to_shape_map()
for key in var_to_shape_map:
  print("tensor_name: ", key)
```

```
tensor_name:  w_fc
tensor_name:  w_conv2
tensor_name:  b_conv2
tensor_name:  b_fc
tensor_name:  w_out
tensor_name:  w_conv3
tensor_name:  b_conv1
tensor_name:  b_conv3
tensor_name:  b_out
tensor_name:  w_conv1
```

In [24]:

```
image, text = gen_captcha_text_and_image()
plt.imshow(image)
plt.title(text)
if len(image.shape) > 2:
  image = np.mean(image, -1)
  image = image.flatten()/255
pre_text = captcha_decode([image])
print('Label:', text, ' Predict:', pre_text)
```

```
INFO:tensorflow:Restoring parameters from /content/0.93captcha.model-4
5700
Label: 7u13  Predict: ['7u13']
```