

基于TensorFlow训练的人脸识别神经网络

基于TensorFlow训练的人脸识别神经网络

[</> 详细内容](#)

[评论 0](#)

[同类相比](#) [574](#)

[学习教程](#)

这段时间正在学习tensorflow的卷积神经网络部分，为了对卷积神经网络能够有一个更深的了解，自己动手实现一个例程是比较好的方式，所以就选了一个这样比较有点意思的项目。

项目的github地址:github 喜欢的话就给个Star吧。

想要她认得我，就需要给她一些我的照片，让她记住我的人脸特征，为了让她区分我和其他人，还需要给她一些其他人的照片做参照，所以需要两组数据集来让她学习，如果想让她多认识几个人，那多给她几组图片集学习就可以了。下面就开始让我们来搭建这个能认识我的“她”。

运行环境

下面为软件的运行搭建系统环境。

系统: window或linux

软件: python 3.x 、 tensorflow

python支持库:

tensorflow:

```
pip install tensorflow          #cpu版本
pip install tensorflow-gpu      #gpu版本，需要cuda与cudnn的支持，不清楚的可以选择cpu版
```

numpy:

```
pip install numpy
```

opencv:

```
pip install opencv-python
```

dlib:

```
pip install dlib
```

获取本人图片集

获取本人照片的方式当然是拍照了，我们需要通过程序来给自己拍照，如果你自己有照片，也可以用那些现成的照片，但前提是你的照片足够多。这次用到的照片数是10000张，程序运行后，得坐在电脑面前不停得给自己的脸摆各种姿势，这样可以提高训练后识别自己的成功率，在程序中加入了随机改变对比度与亮度的模块，也是为了提高照片样本的多样性。

程序中使用的是dlib来识别人脸部分，也可以使用opencv来识别人脸，在实际使用过程中，dlib的识别效果比opencv的好，但opencv识别的速度会快很多，获取10000张人脸照片的情况下，dlib大约花费了1小时，而opencv的花费时间大概只有20分钟。opencv可能会识别一些奇怪的部分，所以综合考虑之后我使用了dlib来识别人脸。

get_my_faces.py

```
import cv2
import dlib
import os
import sys
import random

output_dir = './my_faces'
size = 64

if not os.path.exists(output_dir):
    os.makedirs(output_dir)

# 改变图片的亮度与对比度
def relight(img, light=1, bias=0):
    w = img.shape[1]
    h = img.shape[0]
    #image = []
    for i in range(0,w):
        for j in range(0,h):
```

```

        for c in range(3):
            tmp = int(img[j,i,c]*light + bias)
            if tmp > 255:
                tmp = 255
            elif tmp < 0:
                tmp = 0
            img[j,i,c] = tmp

    return img

```

#使用dlib自带的frontal_face_detector作为我们的特征提取器

```
detector = dlib.get_frontal_face_detector()
```

打开摄像头 参数为输入流，可以为摄像头或视频文件

```
camera = cv2.VideoCapture(0)
```

```
index = 1
```

```
while True:
```

```
    if (index <= 10000):
```

```
        print('Being processed picture %s' % index)
```

从摄像头读取照片

```
        success, img = camera.read()
```

转为灰度图片

```
        gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

使用detector进行人脸检测

```
        dets = detector(gray_img, 1)
```

```
        for i, d in enumerate(dets):
```

```
            x1 = d.top() if d.top() > 0 else 0
```

```
            y1 = d.bottom() if d.bottom() > 0 else 0
```

```
            x2 = d.left() if d.left() > 0 else 0
```

```
            y2 = d.right() if d.right() > 0 else 0
```

```
            face = img[x1:y1,x2:y2]
```

调整图片的对比度与亮度， 对比度与亮度值都取随机数，这样能增加样本的多样性

```
            face = relight(face, random.uniform(0.5, 1.5), random.randint(-50, 50))
```

```
            face = cv2.resize(face, (size,size))
```

```
            cv2.imshow('image', face)
```

```
            cv2.imwrite(output_dir+'/'+str(index)+'.jpg', face)
```

```
            index += 1
```

```
        key = cv2.waitKey(30) & 0xff
```

```
        if key == 27:
```

```
        break
    else:
        print('Finished!')
        break
```

在这里我也给出一个opencv来识别人脸的代码示例：

```
import cv2
import os
import sys
import random

out_dir = './my_faces'
if not os.path.exists(out_dir):
    os.makedirs(out_dir)

# 改变亮度与对比度
def relight(img, alpha=1, bias=0):
    w = img.shape[1]
    h = img.shape[0]
    #image = []
    for i in range(0,w):
        for j in range(0,h):
            for c in range(3):
                tmp = int(img[j,i,c]*alpha + bias)
                if tmp > 255:
                    tmp = 255
                elif tmp < 0:
                    tmp = 0
                img[j,i,c] = tmp
    return img

# 获取分类器
haar = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# 打开摄像头 参数为输入流，可以为摄像头或视频文件
camera = cv2.VideoCapture(0)

n = 1
while 1:
    if (n <= 10000):
```

```

print('It`s processing %s image.' % n)
# 读帧
success, img = camera.read()

gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = haar.detectMultiScale(gray_img, 1.3, 5)
for f_x, f_y, f_w, f_h in faces:
    face = img[f_y:f_y+f_h, f_x:f_x+f_w]
    face = cv2.resize(face, (64,64))
    ...

    if n % 3 == 1:
        face = relight(face, 1, 50)
    elif n % 3 == 2:
        face = relight(face, 0.5, 0)
    ...

    face = relight(face, random.uniform(0.5, 1.5), random.randint(-50, 50))
    cv2.imshow('img', face)
    cv2.imwrite(out_dir+'/'+str(n)+'.jpg', face)
    n+=1

key = cv2.waitKey(30) & 0xff
if key == 27:
    break
else:
    break

```

获取其他人脸图片集

需要收集一个其他人脸的图片集，只要不是自己的人脸都可以，可以在网上找到，这里我给出一个我用到的图片集：

网站地址:<http://vis-www.cs.umass.edu/lfw/>

图片集下载:<http://vis-www.cs.umass.edu/lfw/lfw.tgz>

先将下载的图片集，解压到项目目录下的input_img目录下，也可以自己指定目录(修改代码中的input_dir变量)

接下来使用dlib来批量识别图片中的人脸部分，并保存到指定目录下

set_other_people.py

```

# -*- coding: utf-8 -*-
import sys

```

```

import os
import cv2
import dlib

input_dir = './input_img'
output_dir = './other_faces'
size = 64

if not os.path.exists(output_dir):
    os.makedirs(output_dir)

#使用dlib自带的frontal_face_detector作为我们的特征提取器
detector = dlib.get_frontal_face_detector()

index = 1
for (path, dirnames, filenames) in os.walk(input_dir):
    for filename in filenames:
        if filename.endswith('.jpg'):
            print('Being processed picture %s' % index)
            img_path = path+'/'+filename
            # 从文件读取图片
            img = cv2.imread(img_path)
            # 转为灰度图片
            gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            # 使用detector进行人脸检测 dets为返回的结果
            dets = detector(gray_img, 1)

            #使用enumerate 函数遍历序列中的元素以及它们的下标
            #下标i即为人脸序号
            #left : 人脸左边距离图片左边界的距离 ; right : 人脸右边距离图片左边界的距
            #top : 人脸上面距离图片上边界的距离 ; bottom : 人脸下边距离图片上边界的距
            for i, d in enumerate(dets):
                x1 = d.top() if d.top() > 0 else 0
                y1 = d.bottom() if d.bottom() > 0 else 0
                x2 = d.left() if d.left() > 0 else 0
                y2 = d.right() if d.right() > 0 else 0
                # img[y:y+h,x:x+w]
                face = img[x1:y1,x2:y2]
                # 调整图片的尺寸
                face = cv2.resize(face, (size,size))
                cv2.imshow('image',face)
                # 保存图片
                cv2.imwrite(output_dir+'/'+str(index)+'.jpg', face)
                index += 1

```

```
key = cv2.waitKey(30) & 0xff
if key == 27:
    sys.exit(0)
```

这个项目用到的图片数是10000张左右，如果是自己下载的图片集，控制一下图片的数量避免数量不足，或图片过多带来的内存不够与运行缓慢。

训练模型

有了训练数据之后，通过cnn来训练数据，就可以让她记住我的人脸特征，学习怎么认识我了。

train_faces.py

```
import tensorflow as tf
import cv2
import numpy as np
import os
import random
import sys
from sklearn.model_selection import train_test_split

my_faces_path = './my_faces'
other_faces_path = './other_faces'
size = 64

imgs = []
labs = []

def getPaddingSize(img):
    h, w, _ = img.shape
    top, bottom, left, right = (0,0,0,0)
    longest = max(h, w)

    if w < longest:
        tmp = longest - w
        # //表示整除符号
        left = tmp // 2
        right = tmp - left
    elif h < longest:
```

```

        tmp = longest - h
        top = tmp // 2
        bottom = tmp - top
    else:
        pass
    return top, bottom, left, right

def readData(path , h=size, w=size):
    for filename in os.listdir(path):
        if filename.endswith('.jpg'):
            filename = path + '/' + filename

            img = cv2.imread(filename)

            top,bottom,left,right = getPaddingSize(img)
            # 将图片放大， 扩充图片边缘部分
            img = cv2.copyMakeBorder(img, top, bottom, left, right, cv2.BORDER_CONSTANT)
            img = cv2.resize(img, (h, w))

            imgs.append(img)
            labs.append(path)

readData(my_faces_path)
readData(other_faces_path)
# 将图片数据与标签转换成数组
imgs = np.array(imgs)
labs = np.array([[0,1] if lab == my_faces_path else [1,0] for lab in labs])
# 随机划分测试集与训练集
train_x,test_x,train_y,test_y = train_test_split(imgs, labs, test_size=0.05, random_state=1)
# 参数：图片数据的总数，图片的高、宽、通道
train_x = train_x.reshape(train_x.shape[0], size, size, 3)
test_x = test_x.reshape(test_x.shape[0], size, size, 3)
# 将数据转换成小于1的数
train_x = train_x.astype('float32')/255.0
test_x = test_x.astype('float32')/255.0

print('train size:%s, test size:%s' % (len(train_x), len(test_x)))
# 图片块，每次取100张图片
batch_size = 100
num_batch = len(train_x) // batch_size

x = tf.placeholder(tf.float32, [None, size, size, 3])
y_ = tf.placeholder(tf.float32, [None, 2])

```



```

keep_prob_5 = tf.placeholder(tf.float32)
keep_prob_75 = tf.placeholder(tf.float32)

def weightVariable(shape):
    init = tf.random_normal(shape, stddev=0.01)
    return tf.Variable(init)

def biasVariable(shape):
    init = tf.random_normal(shape)
    return tf.Variable(init)

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

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

def dropout(x, keep):
    return tf.nn.dropout(x, keep)

def cnnLayer():
    # 第一层
    W1 = weightVariable([3,3,3,32]) # 卷积核大小(3,3)，输入通道(3)，输出通道(32)
    b1 = biasVariable([32])
    # 卷积
    conv1 = tf.nn.relu(conv2d(x, W1) + b1)
    # 池化
    pool1 = maxPool(conv1)
    # 减少过拟合，随机让某些权重不更新
    drop1 = dropout(pool1, keep_prob_5)

    # 第二层
    W2 = weightVariable([3,3,32,64])
    b2 = biasVariable([64])
    conv2 = tf.nn.relu(conv2d(drop1, W2) + b2)
    pool2 = maxPool(conv2)
    drop2 = dropout(pool2, keep_prob_5)

    # 第三层
    W3 = weightVariable([3,3,64,64])
    b3 = biasVariable([64])
    conv3 = tf.nn.relu(conv2d(drop2, W3) + b3)
    pool3 = maxPool(conv3)
    drop3 = dropout(pool3, keep_prob_5)

```

```

# 全连接层
Wf = weightVariable([8*16*32, 512])
bf = biasVariable([512])
drop3_flat = tf.reshape(drop3, [-1, 8*16*32])
dense = tf.nn.relu(tf.matmul(drop3_flat, Wf) + bf)
dropf = dropout(dense, keep_prob_75)

```

```

# 输出层
Wout = weightVariable([512,2])
bout = weightVariable([2])
#out = tf.matmul(dropf, Wout) + bout
out = tf.add(tf.matmul(dropf, Wout), bout)
return out

```

```

def cnnTrain():

```

```

    out = cnnLayer()

```

```

    cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(log

```

```

    train_step = tf.train.AdamOptimizer(0.01).minimize(cross_entropy)

```

```

    # 比较标签是否相等，再求的所有数的平均值，tf.cast(强制转换类型)

```

```

    accuracy = tf.reduce_mean(tf.cast(tf.equal(tf.argmax(out, 1), tf.argmax(y_

```

```

    # 将loss与accuracy保存以供tensorboard使用

```

```

    tf.summary.scalar('loss', cross_entropy)

```

```

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

```

```

    merged_summary_op = tf.summary.merge_all()

```

```

    # 数据保存器的初始化

```

```

    saver = tf.train.Saver()

```

```

    with tf.Session() as sess:

```

```

        sess.run(tf.global_variables_initializer())

```

```

        summary_writer = tf.summary.FileWriter('./tmp', graph=tf.get_default_

```

```

        for n in range(10):

```

```

            # 每次取128(batch_size)张图片

```

```

            for i in range(num_batch):

```

```

                batch_x = train_x[i*batch_size : (i+1)*batch_size]

```

```

                batch_y = train_y[i*batch_size : (i+1)*batch_size]

```

```

                # 开始训练数据，同时训练三个变量，返回三个数据

```

```

                _,loss,summary = sess.run([train_step, cross_entropy, merged
                    feed_dict={x:batch_x,y_:batch_

```

```

summary_writer.add_summary(summary, n*num_batch+i)
# 打印损失
print(n*num_batch+i, loss)

if (n*num_batch+i) % 100 == 0:
    # 获取测试数据的准确率
    acc = accuracy.eval({x:test_x, y:test_y, keep_prob_5:1})
    print(n*num_batch+i, acc)
    # 准确率大于0.98时保存并退出
    if acc > 0.98 and n > 2:
        saver.save(sess, './train_faces.model', global_step=n*num_batch+i)
        sys.exit(0)
    print('accuracy less 0.98, exited!')

cnnTrain()

```

训练之后的数据会保存在当前目录下。

使用模型进行识别

最后就是让她认识我了，很简单，只要运行程序，让摄像头拍到我的脸，她就可以轻松地识别出是不是我了。

is_my_face.py

```

output = cnnLayer()
predict = tf.argmax(output, 1)

saver = tf.train.Saver()
sess = tf.Session()
saver.restore(sess, tf.train.latest_checkpoint('.'))

def is_my_face(image):
    res = sess.run(predict, feed_dict={x: [image/255.0], keep_prob_5:1.0, keep_prob_1:1.0})
    if res[0] == 1:
        return True
    else:
        return False

#使用dlib自带的frontal_face_detector作为我们的特征提取器
detector = dlib.get_frontal_face_detector()

cam = cv2.VideoCapture(0)

```

```

while True:
    _, img = cam.read()
    gray_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    dets = detector(gray_image, 1)
    if not len(dets):
        #print('Can`t get face.')
        cv2.imshow('img', img)
        key = cv2.waitKey(30) & 0xff
        if key == 27:
            sys.exit(0)

    for i, d in enumerate(dets):
        x1 = d.top() if d.top() > 0 else 0
        y1 = d.bottom() if d.bottom() > 0 else 0
        x2 = d.left() if d.left() > 0 else 0
        y2 = d.right() if d.right() > 0 else 0
        face = img[x1:y1,x2:y2]
        # 调整图片的尺寸
        face = cv2.resize(face, (size,size))
        print('Is this my face? %s' % is_my_face(face))

        cv2.rectangle(img, (x2,x1),(y2,y1), (255,0,0),3)
        cv2.imshow('image',img)
        key = cv2.waitKey(30) & 0xff
        if key == 27:
            sys.exit(0)

sess.close()

```

本文标题:写个神经网络，让她认得我`(`••••`)(Tensorflow,opencv,dlib,cnn,人脸识别)

文章作者:Tumumu

发布时间:2017年05月02日 - 07时53分

最后更新:2017年05月03日 - 14时11分

原始链接:<http://tumumu.cn/2017/05/02/deep-learning-face/>



热门度与活跃度

0.0 ▶

0.7 ▶


Watchers: 1

Star: 16

Fork: 5

创建时间: 2017-05-03 13:56:36

最后Commits: 18天前



seathiefwang

@seathiefwang

基本信息

分类: 机器学习

收录时间: 2017-05-05 13:16:24

相关教程- 更多教程

1、卷积神经网络|TensorFlow官方文档中文版【TensorFlow 官方文档中文版】

2、总览|TensorFlow官方文档中文版【TensorFlow 官方文档中文版】

3、总览|TensorFlow官方文档中文版【TensorFlow 官方文档中文版】

4、TensorBoard:可视化学习|TensorFlow官方文档中文版【TensorFlow 官方文档中文版】

5、深入MNIST|TensorFlow官方文档中文版【TensorFlow 官方文档中文版】

6、MNIST数据下载|TensorFlow官方文档中文版【TensorFlow 官方文档中文版】

7、TensorFlow运作方式入门|TensorFlow官方文档中文版【TensorFlow 官方文档中文版】


8、字词的向量表示|TensorFlow官方文档中文版【TensorFlow 官方文档中文版】

相关主题- 发表话题

1、手把手教你如何用 OpenCV + Python 实现人脸识别


- 2、
- 3、图像技术在直播中的应用（下）——图像识别
- 4、
- 5、
- 6、
- 7、最全的优秀 TensorFlow 相关资源列表
- 8、如何用TensorFlow和TF-Slim实现图像分类与分割


相关的项目 - 更多比较

 机器学习


👁 928 ☆ 9.2k 📄 2.3k

不断更新的数据科学Python笔记，包括深度学习、大数据等



 10.0 ▶

 0.3 ▼

 ⌚ 29天前

 机器学习

👁 908 ☆ 8.2k 📄 2.3k

2016 深度学习，阿法狗复制品



2016 深度学习，阿法狗复制品

 10.0 ▶

 0.7 ▶

 ⌚ 10天前

 机器学习

👁 604 ☆ 7.7k 📄 1.1k

Awesome Tensorflow: 与Tensorflow相关的资源集合



Awesome Tensorflow: 与Tensorflow相关的资源集合

 10.0 ▶

 0.7 ▶

 ⌚ 10天前

POPULAR

 10.0 ▶

 6.2 ▲

  前天