

1. 简介

本文主要介绍了实时人脸检测与识别系统的详细方法。该系统基于python3.5/opencv2/tensorflow1.3-gpu环境，实现了从摄像头读取视频，检测人脸，识别人脸的功能。本代码地址：[代码](#)

代码中的0_unknown文件夹设定的是未知的人脸数据集。卷积训练至少需要2种数据集，需额外采集至少1种最多10种人脸数据。

项目用时：2017-8-1~2017-11-1 整个过程从零学习python、opencv、tensorflow。

项目主要内容：

- a. 获取1分钟人脸视频
- b. 读取视频并转存为图像
- c. 对图像数据集进行卷积神经网络训练
- d. 重载模型并通过摄像头实时检测人脸

2. 代码思路

- a. 获取1分钟人脸视频
 - i. 通过Opencv调用摄像头来获取1分钟人脸视频。视频保存为avi格式，大小为640*480，每秒20帧。
 - ii. 代码：

```
1 import cv2
2 cap = cv2.VideoCapture(0)
3 # Define the codec and create VideoWriter object
4 fourcc = cv2.VideoWriter_fourcc(*'XVID')
5 out = cv2.VideoWriter('./video/{}.avi'.format(input('请输入英文姓名：')), fourcc, 20.0, (640, 480))
6 count = 1
7 while(cap.isOpened()):
8
9     ret, frame = cap.read()
10    out.write(frame)
11    font = cv2.FONT_HERSHEY_SIMPLEX
12    cv2.putText(frame, 'time left :{} s'.format(int(1200-count)//20), (10, 30), font, 1, (255, 255, 255), 2, cv2.LINE_AA)
13    cv2.imshow('frame', frame)
14    count = count+1
15    if count > 1200:
16        break
17    if cv2.waitKey(1) & 0xFF == ord('q'):
18        break
19 # Release everything if job is finished
20 cap.release()
21 out.release()
22 cv2.destroyAllWindows()
```

- iii. 代码中1200代表1分钟共有1200帧，实际上有误差，结果并不影响。程序运行时如果计数达到1200或者键盘（英文）上按了‘Q’键，程序会保存视频退出。

b. 读取视频并转存为图像

- i. 需要调用dlib和cv2这2个库文件。
- ii. 函数1：读取文件夹中的视频目录

1. 代码

```
1 def read_video_names(path):
2     for filename in os.listdir(path):
3         if filename.endswith('.avi'):
4             filename = path + '/' + filename
5             print(filename)
6             file_names.append(filename)
```

2. 功能：如果发现视频格式为avi的视频，就记录下目录加文件名。例如：./video/zyz.avi。然后添加到file_names这个列表上。

iii. 函数2：读取文件夹中的图片目录

1. 代码

```
1 def read_pic_names(path, num):
2     for filename in os.listdir(path):
3         print(filename)
4         pic_names.append(filename.split('_')[-1])
5         num = num + 1
6     return num
```

2. 功能：函数返回num,num为图片目录下的文件夹数量，输入num默认为0.

iv. 函数3：改变图片的亮度与对比度

1. 代码

```
1 def relight(img, light=1, bias=0):
2     w = img.shape[1]
3     h = img.shape[0]
4     # image = []
5     for i in range(0,w):
6         for j in range(0,h):
7             for c in range(3):
8                 tmp = int(img[j,i,c]*light + bias)
9                 if tmp > 255:
10                     tmp = 255
```

```

11             elif tmp < 0:
12                 tmp = 0
13                 img[j,i,c] = tmp
14     return img

```

2. 功能：函数返回改变亮度和对比度的图片。

v. 函数4：旋转图片

1. 代码

```

1  def reangle(img, angle=0):
2      w = img.shape[1]
3      h = img.shape[0]
4      # 第一个参数旋转中心，第二个参数旋转角度，第三个参数：缩放比例
5      M = cv2.getRotationMatrix2D((w / 2, h / 2), angle, 1)
6      # 第三个参数：变换后的图像大小
7      img = cv2.warpAffine(img, M, (w, h))
8      return img

```

2. 功能：函数返回旋转一定角度后的图片。

vi. 函数5：读取视频并转化为图片

1. 代码

```

1  def video_read(path, num):
2      name_file = str(num) + '_' + path.split('/')[1].split('.')[0]
3      name_file = output_dir + '/' + name_file
4      # 如果不存在目录 就创造目录
5      if not os.path.exists(name_file):
6          os.makedirs(name_file)
7      # 打开摄像头 参数为输入流，可以为摄像头或视频文件
8      camera = cv2.VideoCapture(path)
9      index = 1
10     while True:
11         # 从摄像头读取照片
12         success, img = camera.read()
13         if not success:
14             camera.release()
15             print('finish')
16             break
17         else:
18             # 转为灰度图片
19             gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
20             # # 旋转图片

```

```

21         # gray_img = reangle(gray_img,
    random.randint(-65, 65))
22         # 使用detector进行人脸检测
23         dets = detector(gray_img, 1)
24         for i, d in enumerate(dets):
25             x1 = d.top() if d.top() > 0 else 0
26             y1 = d.bottom() if d.bottom() > 0 else 0
27             x2 = d.left() if d.left() > 0 else 0
28             y2 = d.right() if d.right() > 0 else 0
29
30             face = img[x1:y1, x2:y2]
31             # 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
32             face = relight(face, random.uniform(0.5,
    1.5), random.randint(-75, 75))
33             # 旋转图片
34             face = reangle(face, random.randint(-65, 65))
35             face = cv2.resize(face, (size, size))
36             cv2.imwrite(name_file+'/' + str(index) +
    '.jpg', face)
37             index += 1

```

2. 功能：函数调用为 video_read(file_name, pic_num)，file_name是未被转成图片的视频目录，pic_num是已被转成图片的文件夹个数。例如：
pic_num=2，file_name=./video/zyz.avi，则输出的文件夹名为3_zyz。
zyz.avi视频转成的图片将保存在3_zyz文件夹内，这样做的目的是避免重复处理已被转成图片的视频。

vii. 完整代码：

```

1  import cv2
2  import dlib
3  import os
4  import sys
5  import random
6
7  output_dir = './faces'
8  input_dir = './video'
9  file_names = []
10 pic_names = []
11 size = 64
12 pic_num = 0
13 # 如果不存在目录 就创造目录
14 if not os.path.exists(output_dir):
15     os.makedirs(output_dir)

```

```
16
17
18 # 读取文件夹中的视频目录
19 def read_video_names(path):
20     for filename in os.listdir(path):
21         if filename.endswith('.avi'):
22             filename = path + '/' + filename
23             print(filename)
24             file_names.append(filename)
25
26
27 # 读取文件夹中的图片目录
28 def read_pic_names(path, num):
29     for filename in os.listdir(path):
30         print(filename)
31         pic_names.append(filename.split('_')[-1])
32         num = num + 1
33     return num
34
35
36 # 改变图片的亮度与对比度
37 def relight(img, light=1, bias=0):
38     w = img.shape[1]
39     h = img.shape[0]
40     # image = []
41     for i in range(0,w):
42         for j in range(0,h):
43             for c in range(3):
44                 tmp = int(img[j,i,c]*light + bias)
45                 if tmp > 255:
46                     tmp = 255
47                 elif tmp < 0:
48                     tmp = 0
49                 img[j,i,c] = tmp
50     return img
51
52
53 # 旋转图片
54 def reangle(img, angle=0):
55     w = img.shape[1]
56     h = img.shape[0]
57     # 第一个参数旋转中心，第二个参数旋转角度，第三个参数：缩放比例
58     M = cv2.getRotationMatrix2D((w / 2, h / 2), angle, 1)
59     # 第三个参数：变换后的图像大小
60     img = cv2.warpAffine(img, M, (w, h))
61     return img
```

```

62
63
64 def video_read(path, num):
65     name_file = str(num) + '_' + path.split('/')[ -1].split('.')[0]
66     name_file = output_dir + '/' + name_file
67     # 如果不存在目录 就创造目录
68     if not os.path.exists(name_file):
69         os.makedirs(name_file)
70     # 打开摄像头 参数为输入流，可以为摄像头或视频文件
71     camera = cv2.VideoCapture(path)
72     index = 1
73     while True:
74         # 从摄像头读取照片
75         success, img = camera.read()
76         if not success:
77             camera.release()
78             print('finish')
79             break
80         else:
81             # 转为灰度图片
82             gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
83             # # 旋转图片
84             # gray_img = reangle(gray_img, random.randint(-65,
85 65))
86             # 使用detector进行人脸检测
87             dets = detector(gray_img, 1)
88             for i, d in enumerate(dets):
89                 x1 = d.top() if d.top() > 0 else 0
90                 y1 = d.bottom() if d.bottom() > 0 else 0
91                 x2 = d.left() if d.left() > 0 else 0
92                 y2 = d.right() if d.right() > 0 else 0
93
94                 face = img[x1:y1, x2:y2]
95                 # 调整图片的对比度与亮度， 对比度与亮度值都取随机数，
96                 # 这样能增加样本的多样性
97                 face = relight(face, random.uniform(0.5, 1.5),
98 random.randint(-75, 75))
99                 # 旋转图片
100                 face = reangle(face, random.randint(-65, 65))
101                 face = cv2.resize(face, (size, size))
102
103                 cv2.imwrite(name_file + '/' + str(index) + '.jpg',
104 face)
105
106                 index += 1
107

```

```

104
105 # 使用dlib自带的frontal_face_detector作为我们的特征提取器
106 detector = dlib.get_frontal_face_detector()
107 # 读取文件夹中的视频目录
108 read_video_names(input_dir)
109 pic_num = read_pic_names(output_dir, pic_num)
110 print(pic_num)
111 print(pic_names)
112 for file_name in file_names:
113     name = file_name.split('/')[-1].split('.')[0]
114     if name not in pic_names:
115         print(file_name+"正在处理")
116         video_read(file_name, pic_num)
117         pic_num = pic_num + 1
118     else:
119         pass
120 print('Done')
121

```

c. 对图像数据集进行卷积神经网络训练

i. 调用tensorflow库、cv2库、one_hot库、sklearn库

ii. one_hot是自己编写的库文件，只有一个函数：y_one_hot(num, max_num)，该函数的作用是输入一个小于10的数字 返回一个一维one_hot向量。例如：
num=4，max_num=5 时：函数返回[0, 0, 0, 0, 1]。

iii. 函数1：获取图像的边界

1. 代码

```

1 def get_padding_size(img):
2     h, w, _ = img.shape
3     top, bottom, left, right = (0, 0, 0, 0)
4     longest = max(h, w)
5
6     if w < longest:
7         tmp = longest - w
8         # //表示整除符号
9         left = tmp // 2
10        right = tmp - left
11    elif h < longest:
12        tmp = longest - h
13        top = tmp // 2
14        bottom = tmp - top
15    else:
16        pass
17    return top, bottom, left, right

```

2. 功能：函数返回 分别表示四个方向上边界的长度

iv. 函数2：读取图片数据

1. 代码

```
1 def read_data(path, h=size, w=size):
2     for filename in os.listdir(path):
3         for img_name in os.listdir(path+filename):
4             if img_name.endswith('.jpg'):
5                 path_name = path+filename + '/' + img_name
6                 img = cv2.imread(path_name)
7
8                 top, bottom, left, right =
9                     get_padding_size(img)
10                    # 将图片放大， 扩充图片边缘部分
11                    img = cv2.copyMakeBorder(img, top, bottom,
12                    left, right, cv2.BORDER_CONSTANT, value=[0, 0, 0])
13                    img = cv2.resize(img, (h, w))
14                    np_lab.append(filename)
15                    np_img.append(img)
```

2. 功能：读取文件夹中的所有图片，把图片全部扩充为64*64的图片

cv2.copyMakeBorder(src,top, bottom, left, right ,borderType,value)的作用是给源图像增加边界。src:源图像、top,bottom,left,right: 分别表示四个方向上边界的长度、borderType: 边界的类型

有以下几种方法：

```
1  BORDER_REPLICATE      # 直接用边界的颜色填充， aaaaaa |
   abcdefg | gggg
2  BORDER_REFLECT        # 倒映， abcdefg | gfedcbam | nmabcd
3  BORDER_REFLECT_101    # 倒映，和上面类似，但在倒映时，会把边界空
   开， abcdefg | egfedcbamne | nmabcd
4  BORDER_WRAP           # 额。类似于这种方式 abcdf | mmabcdf |
   mmabcd
5  BORDER_CONSTANT       # 常量，增加的变量通通为value色 [value]
   [value] | abcdef | [value][value][value]
```

3. value仅仅是常量型边界才有意义

v. 函数3：其实没有封装成一个函数

1. 代码

```
1 for lab in np_lab:
```






```

2     if lab not in lab_name:
3         lab_name.append(lab)
4 for lab in np_lab:
5     max_num = len(lab_name)
6     num_tot = max_num
7     for num in range(max_num):
8         if lab == lab_name[num]:
9             lab = one_hot.y_one_hot(num, max_num)
10            lab_full.append(lab)

```

2. 功能：第一个for循环添加标签的名字到列表lab_name中，例如文件夹中有如下目录：

 0_unknown	2017/9/26 15:38	文件夹
 1_lbq	2017/10/23 10:50	文件夹
 2_qilei	2017/10/23 11:00	文件夹
 3_yzy	2017/10/23 11:11	文件夹
 4_11	2017/10/23 21:25	文件夹

则列表lab_name=['0_unknown','1_lbq','2_qilei','3_yzy','4_11'].

第二个for循环是把这些标签转化成一维one_hot向量，便于分类。例如：

0_unknown转化为[1,0,0,0,0],1_lbq转化为[0,1,0,0,0],2_qilei转化为[0,0,1,0,0],3_yzy转化为[0,0,0,1,0],4_11转化为[0,0,0,0,1].

vi. 函数4：卷积池化

1. 代码

```

1  def cnnlayer():
2      # 第一层
3      W1 = weight_variable([5, 5, 3, 64]) # 卷积核大小(5,5),
        输入通道(3), 输出通道(64)
4      b1 = bias_variable([64])
5      # 卷积
6      conv1 = tf.nn.relu(conv_2d(x, W1) + b1)
7      # 池化
8      pool1 = max_pool(conv1)
9      # 减少过拟合，随机让某些权重不更新
10     drop1 = dropout(pool1, keep_prob_5)
11
12     # 第二层，此时输入图片大小为30*30
13     W2 = weight_variable([5, 5, 64, 128])
14     b2 = bias_variable([128])
15     conv2 = tf.nn.relu(conv_2d(drop1, W2) + b2)
16     pool2 = max_pool(conv2)
17     drop2 = dropout(pool2, keep_prob_5)
18

```

```

19     # 第三层，此时输入图片大小为14*14
20     W3 = weight_variable([5, 5, 128, 256])
21     b3 = bias_variable([256])
22     conv3 = tf.nn.relu(conv_2d(drop2, W3) + b3)
23     pool3 = max_pool(conv3)
24     drop3 = dropout(pool3, keep_prob_5)
25
26     # 全连接层，此时输入图片大小为5*5
27     Wf = weight_variable([5*5*256, 1024])
28     bf = bias_variable([1024])
29     drop_flat = tf.reshape(drop3, [-1, 5*5*256])
30     dense = tf.nn.relu(tf.matmul(drop_flat, Wf) + bf)
31     dropf = dropout(dense, keep_prob_75)
32
33     # 输出层
34     print('num_tot: ' + str(num_tot))
35     Wout = weight_variable([1024, num_tot])
36     bout = bias_variable([num_tot])
37     out = tf.add(tf.matmul(dropf, Wout), bout)
38     return out

```

2. 卷积操作中padding = 'VALID'，池化操作中padding = 'SAME'。

第一层：图片大小为64*64，共64个卷积核；

第二层：图片大小为30*30，共128个卷积核；

第三层：图片大小为14*14，共256个卷积核；

输出层：图片大小为5*5；

vii. 训练测试

1. 代码

```

1  def cnnTrain():
2      out = cnnlayer()
3
4      cross_entropy =
5          tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits
6              =out, labels=y_))
7
8      train_step = tf.train.AdamOptimizer(1e-
9          4).minimize(cross_entropy)
10
11     # 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
12     型)
13
14     accuracy = tf.reduce_mean(tf.cast(tf.equal(tf.argmax(out,
15         1), tf.argmax(y_, 1)), tf.float32))
16
17     # 将loss与accuracy保存以供tensorboard使用

```

```

10     tf.summary.scalar('loss', cross_entropy)
11     tf.summary.scalar('accuracy', accuracy)
12     merged_summary_op = tf.summary.merge_all()
13     # 数据保存器的初始化
14     saver = tf.train.Saver()
15
16     with tf.Session() as sess:
17
18         sess.run(tf.global_variables_initializer())
19
20         summary_writer = tf.summary.FileWriter('./tmp',
graph=tf.get_default_graph())
21
22         for n in range(500):
23             # 每次取128(batch_size)张图片
24             for i in range(num_batch):
25                 batch_x = train_x[i*batch_size:
(i+1)*batch_size]
26                 batch_y = train_y[i*batch_size:
(i+1)*batch_size]
27                 # 开始训练数据，同时训练三个变量，返回三个数据
28                 _, loss, summary = sess.run([train_step,
cross_entropy, merged_summary_op],
29                                             feed_dict={x:
batch_x, y_: batch_y, keep_prob_5: 0.55, keep_prob_75: 0.55})
30                 summary_writer.add_summary(summary,
n*num_batch+i)
31                 # 打印损失
32                 # print(n*num_batch+i, loss)
33                 if (n*num_batch+i) % 100 == 0:
34                     # 获取测试数据的准确率
35                     acc = accuracy.eval({x: test_x, y_:
test_y, keep_prob_5: 1.0, keep_prob_75: 1.0})
36                     print(n*num_batch+i, acc)
37                     # 准确率大于0.998时保存并退出
38                     if acc > 0.998 and n > 20:
39                         saver.save(sess,
'model/train_faces.model')
40                         sys.exit(0)
41                 saver.save(sess, 'model/train_faces.model')
42                 print('保存成功')

```

2. 功能：每次训练的批次为128，对整个训练集进行500次循环训练，当准确率大于0.998时，保存模型并退出，或者500次循环执行完毕时退出。

d. 重载模型进行实时人脸识别

- i. 调用和上一步相同的库文件
- ii. 函数1：读取文件夹中的图片目录
 - 1. 代码

```
1 # 读取文件夹中的图片目录
2 def read_pic_names(path):
3     for filename in os.listdir(path):
4         print(filename)
5         names.append(filename.split('_')[-1])
```

- 2. 功能：把目录名字按顺序返回到names列表中。

- iii. 函数2：获取图片的边界
- iv. 函数3：读取图片数据
- v. 函数4：判断人脸函数

- 1. 代码

```
1 def is_my_face(image):
2     res = sess.run(output, feed_dict={x: [image / 255.0],
3     keep_prob_5: 1.0, keep_prob_75: 1.0})
4     p = sess.run(tf.nn.softmax(res))
5     pre = sess.run(tf.argmax(p, 1))
6     p = p[0]
7     if pre[0] == 0:
8         return names[0], p[0]
9     elif pre[0] == 1:
10        return names[1], p[1]
11    elif pre[0] == 2:
12        return names[2], p[2]
13    elif pre[0] == 3:
14        return names[3], p[3]
15    elif pre[0] == 4:
16        return names[4], p[4]
17    elif pre[0] == 5:
18        return names[5], p[5]
19    elif pre[0] == 6:
20        return names[6], p[6]
21    elif pre[0] == 7:
22        return names[7], p[7]
23    elif pre[0] == 8:
24        return names[8], p[8]
25    elif pre[0] == 9:
26        return names[9], p[9]
```

2. 功能：输入一个人脸图片，经过模型预测，返回人脸对应的名字，和其相似概率。pre代表模型预测后返回的标签，names是一个列表，内有与标签对应的名字。

vi. 主循环函数：

1. 代码

```
1 while True:
2     ret, img = cam.read()
3     gray_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
4     dets = detector(gray_image, 1)
5
6     for i, d in enumerate(dets):
7         x1 = d.top() if d.top() > 0 else 0
8         y1 = d.bottom() if d.bottom() > 0 else 0
9         x2 = d.left() if d.left() > 0 else 0
10        y2 = d.right() if d.right() > 0 else 0
11        x_mid = (x1 + y1)//2
12        y_mid = (x2 + y2) // 2
13        face = img[x1:y1, x2:y2]
14        # 调整图片的尺寸
15        face = cv2.resize(face, (size, size))
16        Id, Probability = is_my_face(face)
17
18        cv2.rectangle(img, (x2, x1), (y2, y1), (255, 255, 0),
19                        3)
20        font = cv2.FONT_HERSHEY_SIMPLEX
21        cv2.putText(img, str(Id) + " : " + str(Probability),
22                    (x_mid, y_mid), font, 1, (255, 255, 255), 2, cv2.LINE_AA)
23        cv2.imshow('image', img)
24        key = cv2.waitKey(30) & 0xff
25        if key == 27:
26            sys.exit(0)
```

2. 功能：打开摄像头，不停的读取人脸图片，并调用人脸判断函数。
按'Esc'键退出循环。

3. 总结

- a. 训练准确率一般能达到98%以上，但具体在现实环境中进行识别时，受环境光线，摄像头角度，离摄像头的距离，被识别人脸部分遮挡或面貌特征发生较大改变等等因素，实际现场识别的准确度还达不到90%。
- b. 看到一个统计，如下图：



c. 以后改进：

- 数据集。增加多样化数据集，提高系统的鲁棒性。
- CNN模型。提高模型的复杂度+适当的模型调参能够更好的改善CNN模型。