利用python和OpenCV实现静态图片和视频人脸识别和性别检测

github：

https://github.com/Freebreeze/face_recognition

静态图片人脸识别的实现

导入所需要的CV2模块，OpenCV是一个基于BSD许可发行的跨平台计算机视觉库，可以运行在Linux、Windows和Mac OS操作系统上，轻量而且高效，用C / C++编写，同时提供了Python、Ruby、MATLAB等接口，实现了图像处理和计算机视觉方面的很多通用算法。

```
import cv2
```

采用默认的人脸分类器haarcascade_frontalface_default.xml，仅可以检查睁开的眼睛haarcascade_eye.xml 检查带眼镜的眼睛haarcascade_eye_tree_eyeglasses.xml

```
filename =  'F:/python/test/example/1.jpg
    casvade_face_name='F:/python/test/cascades/haarcascade_frontalface_default.xml
    casvade_eye = 'F:/python/test/cascades/haarcascade_eye.xml
    eye_glasses='F:/python/test/cascades/haarcascade_eye_tree_eyeglasses.xm
```

定义了detect函数

```
def detect(filename) :
face_cascade=cv2.CascadeClassifier(casvade_face_name)#人脸检测
eye_cascade=cv2.CascadeClassifier(casvade_eye)#人眼检测
eye_glass_cascade=cv2.CascadeClassifier(eye_glasses)#带眼睛的人眼检测
#通过CV2imread加载图片，并把它转换为灰度图片
img=cv2.imread(filename)
gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
# 用face_cascade.detectMultiScale进行人脸检测
faces=face_cascade.detectMultiScale(gray,1.2,3)
for (x,y,w,h) in faces:
```

cv2.rectangle用来允许通过坐标绘制矩形,x,y表示左上角的坐标，w和h表示人脸矩形的宽度和高度

```
img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
roi_gray=gray[y:y+h,x:x+w]
#print(roi_gray)
roi_color = img[y:y + h, x:x + w]
glass=eye_glass_cascade.detectMultiScale(roi_gray,1.03,5,0,(40,40))
for (gx, gy, gw, gh) in glass:
cv2.rectangle(roi_color, (gx, gy), (gx + gw, gy + gh), (0, 255, 0), 2)
eyes = eye_cascade.detectMultiScale(roi_gray,1.03,5,0,(26,26))
for (ex, ey, ew, eh) in eyes:
cv2.rectangle(roi_color, (ex, ey), (ex + ew, ey + eh), (0, 255, 0), 2)
```

在图片上显示性别

```
font = cv2.FONT_HERSHEY_SIMPLEX  # 使用默认字体
img = cv2.putText(faces, 'man', (roi_gray[0][0], roi_gray[0][1]), font, 1.2, (255, 255,
255), 2)  # 添加文字，1.2表示字体大小，（0,40）是初始的位置
```

显示窗口

```
cv2.namedWindow('find')
cv2.imshow('face',img)
cv2.imwrite('F:/python/test/人脸.jpg ',img)
cv2.waitKey(0)
detect(filename)`
```

静态图片性别检测的实现

采用bp神经网络实现人脸识别，首先先框出人脸，并保存人脸区域，再将人脸图片灰度化，改为28*28大小，对图片处理结束，下进行神经网络的设计 神经网络设计 网络层设计：一层隐藏层，一层输出层 输入层：一张图片的灰度值矩阵reshape后的784（2828）个数，也就是x_train中的某一列 输出层：输出为pre的值，预测结果pre大于0.5，为男；预测结果小于或等于0.5为女 激励函数：sigmoid函数（公式) 更新法则：后向传播算法（参考） 测试：统计预测正确的个数  反向传播算法 我们把各个模块写成函数，在后面测试时直接调用前面的函数，下面介绍每个函数：

所需要的库

```
from PIL import Image
    import dlib
    import cv2
    import os.path
    import PIL.Image
    from pylab import *
    import os
    import numpy as np
    import sys
```

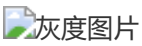
获取人脸照片

```
def get_face_from_photo(i,path,spath):

    detector = dlib.get_frontal_face_detector() #获取人脸分类
    # 读取path路径下的图片，获得所有的图片名字
    filenames = os.listdir(path)

    for f1 in filenames:
        f = os.path.join(path,f1)
        # iimag = PIL.Image.open(f)
        # opencv 读取图片，并显示，将OpenCV转换成PIL.image格式
        img = cv2.imread(f, cv2.IMREAD_COLOR)
        iimag = Image.fromarray(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

        b, g, r = cv2.split(img)    # 分离三个颜色通道
        img2 = cv2.merge([r, g, b])   # 生成新图片

        counts = detector(img, 1) #人脸检测

        for index, face in enumerate(counts):

            # 在图片中标注人脸，并显示
            left = face.left()
            top = face.top()
            right = face.right()
            bottom = face.bottom()

            #保存人脸区域
            j =str(i)
            j = j+'.jpg'
            save_path = os.path.join(spath,j)
            region = (left,top,right,bottom)
            #裁切图片
            cropImg = iimag.crop(region)

            #保存裁切后的图片
            cropImg.save(save_path)
            i +=1

            cv2.rectangle(img, (left, top), (right, bottom), (0, 255, 0), 3)
            cv2.namedWindow(f, cv2.WINDOW_AUTOSIZE)
```

```
            cv2.imshow(f, img)

    # 等待按键，退出，销毁窗口
    k = cv2.waitKey(0)
    cv2.destroyAllWindows()
    return i
```

将人脸图片转化为28*28的灰度图片， path:人脸图片存储路径，spath:灰度图片存储路径

```
def change_photo_size28(path,spath):`


    filenames = os.listdir(path)

    for filename in filenames:
        f = os.path.join(path,filename)
        iimag = PIL.Image.open(f).convert('L').resize((28,28))
        savepath = os.path.join(spath,filename)
        #savepath = spath + '/' + filename
        iimag.save(savepath)
```



读取训练图片，对照片从0开始编号

```
def read_photo_for_train(k,photo_path):`
    for i in range(k):
        j = i
        j = str(j)
        st = '.jpg'
        j = j+st
        j = os.path.join(photo_path,j)
        im1 = array(Image.open(j).convert('L'))
        # (28, 28) -->(28*28,1)
        im1 = im1.reshape((784,1))
        #把所有的图片灰度值放到一个矩阵中
        #一列代表一张图片的信息
        if i == 0:
            im = im1
        else:
            im = np.hstack((im,im1))
    return im
```

修正函数layerout

```
def layerout(w,b,x):

        '''
        sigmoid函数实现
        '''

    y = np.dot(w,x) + b
    t = -1.0*y
    # n = len(y)
    # for i in range(n):
        # y[i]=1.0/(1+exp(-y[i]))
    y = 1.0/(1+exp(t))
    return y
```

训练函数

设置一个隐藏层，784-->隐藏层神经元个数-->1

```
def mytrain(x_train,y_train):`
    step=int(input('mytrain迭代步数: '))
    a=double(input('学习因子: '))
    # step=1000
    # a=0.4
    inn = 784  #输入神经元个数
    # inn=int(input('输入神经元的个数'))
    hid = int(input('隐藏层神经元个数: '))#隐藏层神经元个数
    # hid=28
    out = 1  #输出层神经元个数

    w = np.random.randn(out,hid)
    w = np.mat(w)
    b = np.mat(np.random.randn(out,1))
    w_h = np.random.randn(hid,inn)
    w_h = np.mat(w_h)
    b_h = np.mat(np.random.randn(hid,1))

    for i in range(step):
        #打乱训练样本
        r=np.random.permutation(206)
        x_train = x_train[:,r]
        y_train = y_train[:,r]
        #mini_batch
        for j in range(206):
            x = np.mat(x_train[:,j])
            x = x.reshape((784,1))
            y = np.mat(y_train[:,j])
            y = y.reshape((1,1))
            hid_put = layerout(w_h,b_h,x)
            out_put = layerout(w,b,hid_put)

            #更新公式的实现
            o_update = np.multiply(np.multiply((y-out_put),out_put),(1-out_put))
```

```
            h_update = np.multiply(np.multiply(np.dot((w.T),np.mat(o_update)),hid_put),(1-
hid_put))

            outw_update = a*np.dot(o_update,(hid_put.T))
            outb_update = a*o_update
            hidw_update = a*np.dot(h_update,(x.T))
            hidb_update = a*h_update

            w = w + outw_update
            b = b+ outb_update
            w_h = w_h +hidw_update
            b_h =b_h +hidb_update

    return w,b,w_h,b_h
```

测试函数

预测结果pre大于0.5，为男；预测结果小于或等于0.5为女

```
  def mytest(x_test,w,b,w_h,b_h):`

      hid = layerout(w_h,b_h,x_test);
      pre = layerout(w,b,hid);
      print(pre)
      if pre > 0.5:
          print("hello,boy!")
      else:[]
          print("hello,girl!")
```

开始训练

```
      #框出人脸，并保存到faces中,i为保存的名字
      i = 0
      #女孩
      path = 'F:/python/test/beauty'
      spath = 'F:/python/test/faces'
      i = get_face_from_photo(i,path,spath)
      # 男孩
      path = 'F:/python/test/nan'
      i = get_face_from_photo(i,path,spath)

      #将人脸图片转化为28*28的灰度图片
      path = 'F:/python/test/faces'
      spath = 'F:/python/test/grayfaces'
      change_photo_size28(path,spath)


      #获取图片信息
      im = read_photo_for_train(206,spath)

      #归一化
      immin = im.min()
```

```
  immax = im.max()
  im = (im-immin)/(immax-immin)

  x_train = im

  #制作标签，前97张是女生，为0,后109张为男生，为1
  y1 = np.zeros((1,97))
  y2 = np.ones((1,109))
  y_train = np.hstack((y1,y2))

  #开始训练
  print("----------------------开始训练-----------------------------------------")
  w,b,w_h,b_h = mytrain(x_train,y_train)
  print("----------------------训练结束-----------------------------------------")
```

测试女生

```
`print("--------------------测试女生-----------------------------------------")
#框出人脸，并保存到girltests中,i为保存的名字
i = 0
#女孩测试集
path = 'F:/python/test/girltest'
spath = 'F:/python/test/girltest-face'
i = get_face_from_photo(i,path,spath)

#将人脸图片转化为28*28的灰度图片
path = 'F:/python/test/girltest-face'
spath = 'F:/python/test/girltest-grayface'
change_photo_size28(path,spath)

#获取图片信息
im = read_photo_for_train(19,spath)

#归一化
immin = im.min()
immax = im.max()
im = (im-immin)/(immax-immin)

x_test = im
#print(x_test.shape)
for i in range(19):
    xx = x_test[:,i]
    xx = xx.reshape((784,1))
    mytest(xx,w,b,w_h,b_h)
```

测试男生

```
print("--------------------测试男生-----------------------------")
#框出人脸，并保存到boytests中,i为保存的名字
i = 0
#男孩测试集
```

```
    path = 'F:/python/test/boytest'
    spath = 'F:/python/test/boytest_face'
    i = get_face_from_photo(i,path,spath)

    #将人脸图片转化为28*28的灰度图片
    path = 'F:/python/test/boytest_face'
    spath = 'F:/python/test/boytest_grayface'
    change_photo_size28(path,spath)


    #获取图片信息
    im = read_photo_for_train(19,spath)

    #归一化
    immin = im.min()
    immax = im.max()
    im = (im-immin)/(immax-immin)

    x_test = im
    for i in range(19):
    xx = x_test[:,i]
    xx = xx.reshape((784,1))
    mytest(xx,w,b,w_h,b_h)
```

视频人脸识别

视频中人脸识别和图片中识别人脸的算法库相同，不同的是图片来源不同，用camear的read函数获得摄像头的每一帧，并把每一帧进行灰度处理，把灰度图片传给haar进行灰度处理，返回值是人脸左上角坐标，宽度和高度.

```
import cv2

def detect():
casvade_face='F:/python/test/cascades/haarcascade_frontalface_default.xml'
eye_glasses='F:/python/test/cascades/haarcascade_eye_tree_eyeglasses.xml'

    face_cascade=cv2.CascadeClassifier(casvade_face)
    eye_glass_cascade=cv2.CascadeClassifier(eye_glasses)
    camera= cv2.VideoCapture(0)

    while(True):
        # 文件来自摄像头读取内容
        ret, frame=camera.read()#获摄像头得每一帧
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)#把每一帧进行灰度处理
        faces = face_cascade.detectMultiScale(gray, 1.3, 5,0)#把灰度图片传给haar进行灰度处理，
返回值是人脸左上角坐标，宽度和高度


        for (x,y,w,h) in faces:
            img = cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)

            roi_gray=gray[y:y+h,x:x+w]
```

```
            roi_color = img[y:y + h, x:x + w]
            glass = eye_glass_cascade.detectMultiScale(roi_gray, 1.03, 5, 0, (20, 20))
            for (gx, gy, gw, gh) in glass:
                cv2.rectangle(roi_color, (gx, gy), (gx + gw, gy + gh), (0, 255, 0), 2)
            cv2.imshow("camera",frame)
        if cv2.waitKey(1) & 0xff == ord("q"):
            break
    camera.release()
    cv2.destroyAllWindows()
if __name__ =="__main__":
    detect()
```

视频中性别识别

```
# 测试
print("--------------------视频测试----------------------------------------")
camera= cv2.VideoCapture(0)
while(True):
    ret, f = camera.read()
    detector = dlib.get_frontal_face_detector()  # 获取人脸分类
    iimag = Image.fromarray(cv2.cvtColor(f, cv2.COLOR_BGR2RGB))
    counts = detector(f, 1)  # 人脸检测
    for index, face in enumerate(counts):
        # 在图片中标注人脸，并显示
        left = face.left()
        top = face.top()
        right = face.right()
        bottom = face.bottom()
        region = (left, top, right, bottom)
        cropImg = iimag.crop(region)
        finaImg=cropImg.convert('L').resize((28,28))
        im1 = array(finaImg)
        #  (28, 28) -->(28*28,1)
        im1 = im1.reshape((784, 1))
        im = im1
        # print(im)
        #归一化
        immin = im.min()
        immax = im.max()
        im = (im - immin) / (immax - immin)
        i=0
        x_test = im
        xx = x_test[:, i]
        xx = xx.reshape((784, 1))
        mytest(xx, w, b, w_h, b_h)
        pre=mytest(xx, w, b, w_h, b_h)
        font = cv2.FONT_HERSHEY_SIMPLEX  # 使用默认字体
        if pre > 0.5:
            img = cv2.putText(f, 'man', (left, top - 30), font, 1.2, (255, 255, 255),2)
 # 添加文字，1.2表示字体大小，（0,40）是初始的位置，
            # 保存
```

```
        else:
            img = cv2.putText(f, 'girl', (left, top - 30), font, 1.2, (255, 255, 255),
2)  # 添加文字，1.2
        cv2.rectangle(f, (left, top), (right, bottom), (0, 255, 0), 2)
    cv2.imshow("camera", f)
    if cv2.waitKey(1) & 0xff == ord("q"):
        break
camera.release()
cv2.destroyAllWindows()
```