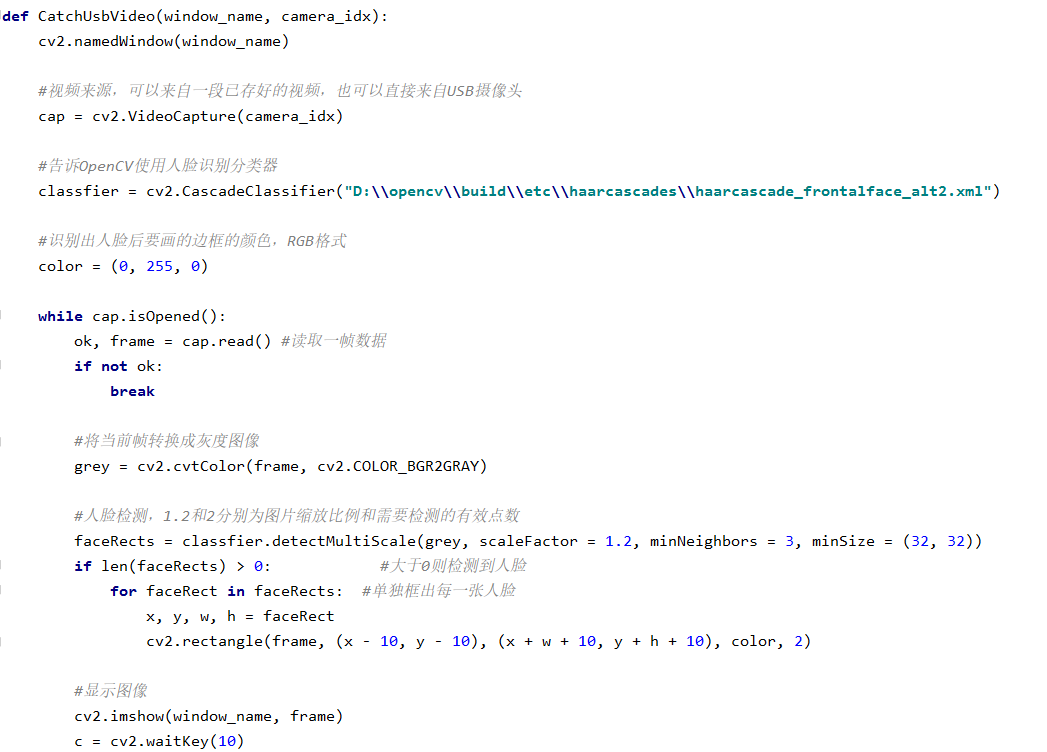
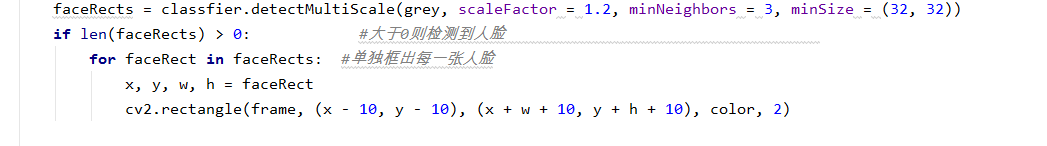


1,识别人脸

实现人脸识别简单程序没几行,但是我们要实现的是识别这个是谁的脸。首先我们让系统识别人脸,这是opencv的工作,我们只需要调用其中的API函数就可以了。下面是调用opencv实现对于人脸的识别。咱们在程序下面对程序进行一些解释：



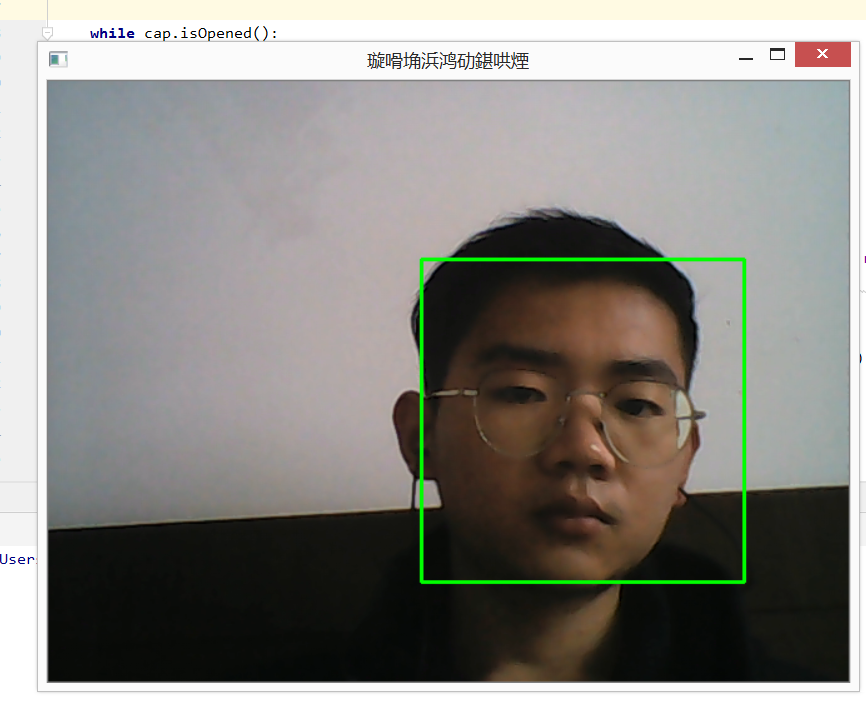
首先,第一行import cv2,实际上,”cv2”中的”2”并不表示OpenCV的版本号。我们知道,OpenCV是基于C/C++的,”cv”和”cv2”表示的是底层CAPI和C++API的区别,”cv2”表示使用的是C++API。这主要是一个历史遗留问题,是为了保持向后兼容性。PIL是一个模块,如果运行的过程中提示你缺少模块的时候你就要安装一个模块了,其余同理,就不再说了。另外在函数Catchusbvideo中,第二个参数指的是你电脑的摄像头的编号,例如是0,1,2等,如果0不行的话,试一下1。在下边的人脸识别分类器中是我自己下载的opencv,如果你是windows选择对应版本就好,还有就是“D: \\opencv\\build\\etc\\haarcascades\\haarcascade\_frontalface\_alt2.xml”这是我安装的一个路径,你也要找到这个路径并且复制到程序中,这个东西的作用主要是实现对人脸识别的功能。



这个函数完成对人脸的识别以及用一个框框给框起来,其中grey是要识别的图像数据,转化为灰度可以减少计算量。scaleFactor：图像缩放比例,可以理解为同一个物体与相机距离不同,其大小亦不同,必须将其缩放到一定大小才方便识别,该参数指定每次缩放的比例。minNeighbors：对特征检测点周边多少有效点同时检测,这样可避免因选取的特征检测点太小而导致遗漏。minSize：特征检测点的最小值。

对同一个画面有可能出现多张人脸,因此,我们需要用一个for循环将所有检测到的人脸都读取出来,然后逐个用矩形框框出来,这就是接下来的for语句的作用。Opencv会给出每张人脸在图像中的起始坐标（左上角,x、y）以及长、宽（h、w）,我们据此就可以截取出人脸。其中,cv2.rectangle()完成画框的工作,在这里外扩了10个像素以框出比人脸稍大一点的区域。cv2.rectangle()函数的最后两个参数一个用于指定矩形边框的颜色,一个用于指定矩形边框线条的粗细程度。

            运行结果：



好,看来可以顺利的识别出视频中的脸,搞定！但是我们想做的是识别这个人脸是谁的,这仅仅能识别这是谁的脸,完全不能满足我们的渴望,接下来我们进行下一步！

2.模型训练

模型训练的目的是让电脑知道,这个脸的特征是什么,从而可以在视频流中识别。在训练之前必须先准备足够的脸部照片作为机器学习的资料。

2.1准备机器学习的资料

所谓机器学习就是给程序投喂足够多的资料,资料越多,准确度和效率也会越高。要想识别出这张人脸属于谁,我们肯定需要大量的自己的脸和别人的脸,这样才能区别开。然后将这些数据输入到Tensorflow中建立我们自己脸的模型。

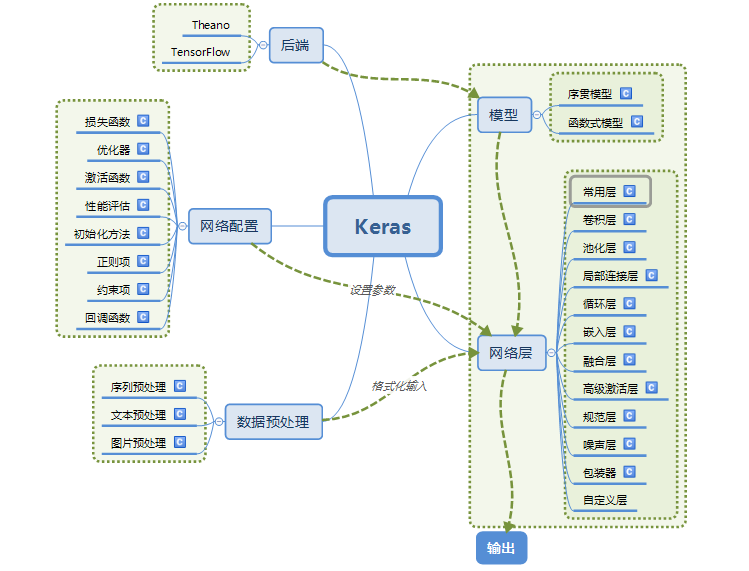
(1)Keras是由纯python编写的基于theano/tensorflow的深度学习框架。Keras是一个高层神经网络API,支持快速实验,能够把你的idea迅速转换为结果,如果有如下需求,可以优先选择Keras：

a）简易和快速的原型设计（keras具有高度模块化,极简,和可扩充特性）

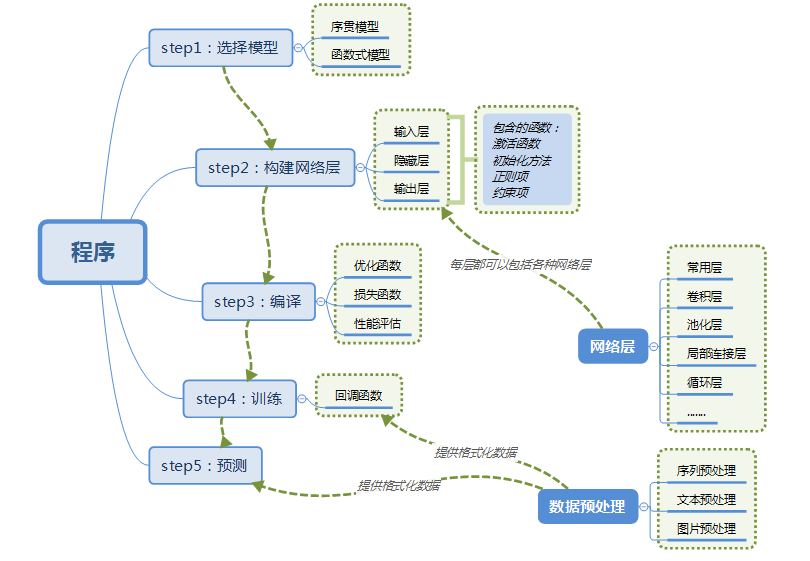
b）支持CNN和RNN,或二者的结合

c）无缝CPU和GPU切换

Keras的模块结构：



使用Keras搭建一个神经网络：



数据格式(data\_format)：

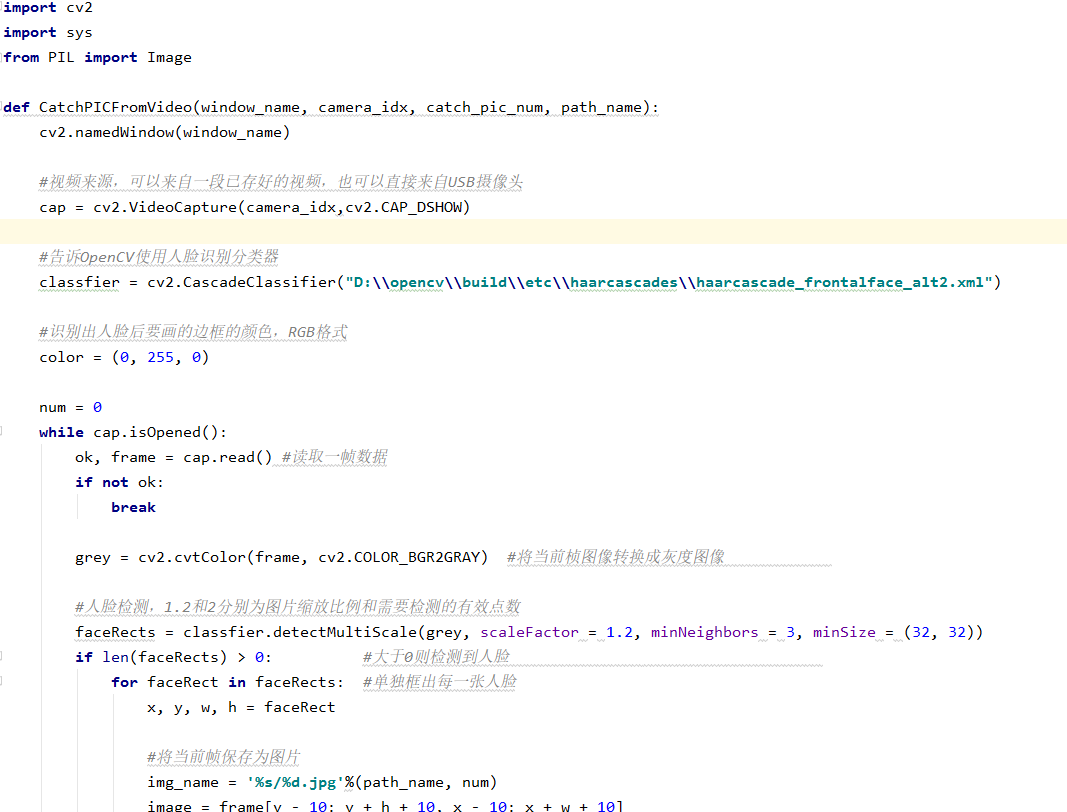
目前主要有两种方式来表示张量：

a) th模式或channels\_first模式,Theano和caffe使用此模式。

b）tf模式或channels\_last模式,TensorFlow使用此模式。

因为我装的是tensorflow因此我直接使用了keras的Tensorflow版,同时,为了验证其它深度学习库的效率和准确率,我还使用了Theano,利用CNN——卷积神经网络来训练我的人脸识别模型。本节专注把训练数据准备好。

代码如下：



这段代码我们只是在前面代码的基础上增加脸部图像存储功能,比较简单。

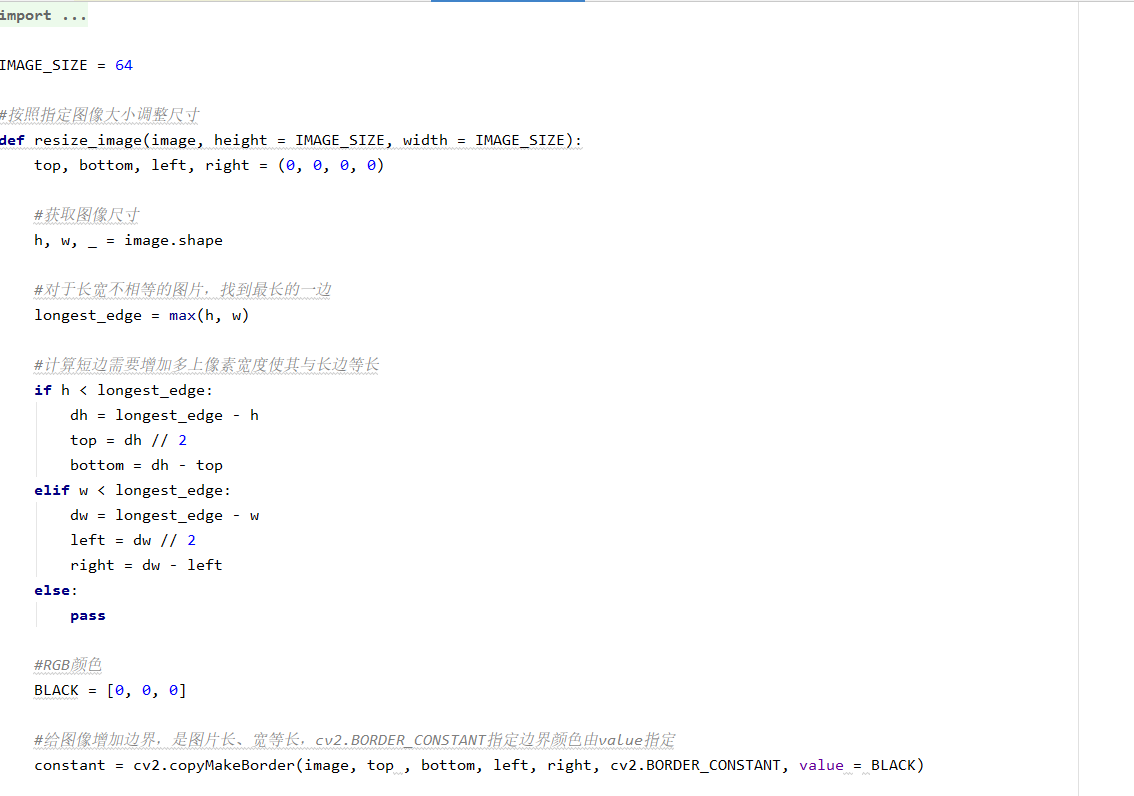


在函数定义中,几个参数,反别是窗口名字,摄像头系列号,捕捉照片数量,以及存储路径。根据自己需要进行修改,咱们这里为了精度高一点,选择捕捉1000张脸部照片。在你捕捉的时候由于精度的问题,会捕捉许多非脸部的照片,这时候需要你将不是脸部的照片清洗掉,使数据更加准确。另外,我们还需要捕捉另一个人的图片来提高模型的准确度。然后存储到另一个文件夹下,注意,一个人的照片存储到一个文件夹下,不可弄混。截图完成。

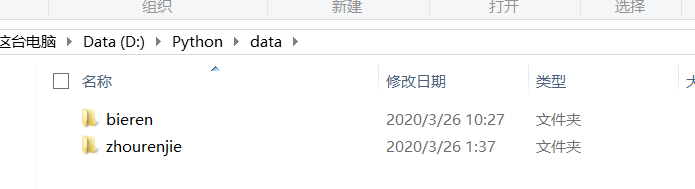
(2)模型训练

训练程序建立了一个包含4个卷积层的神经网络（CNN）,程序利用这个网络训练我的人脸识别模型,并将最终训练结果保存到硬盘上。

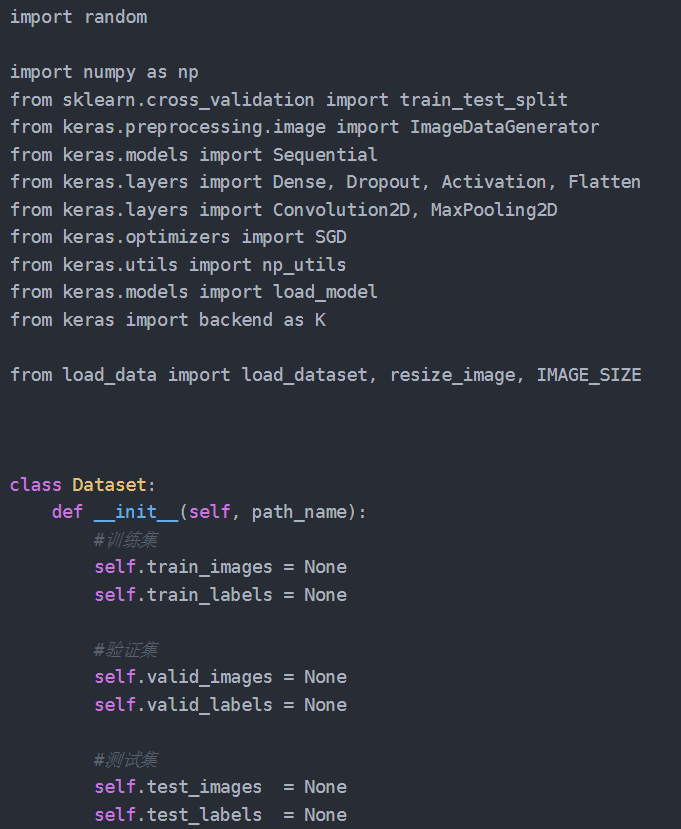
首先建立一个python文件,命名load\_data。代码如下：



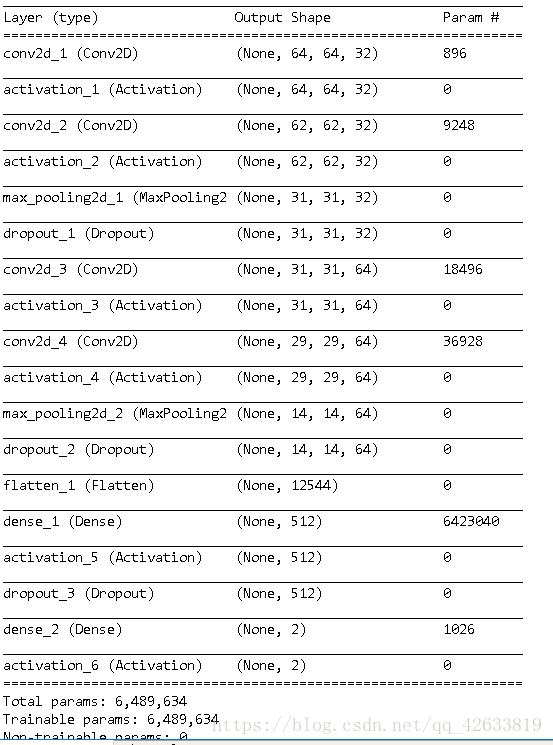
将你捕捉到的照片放不同的文件夹里，我在这里一块放在了data文件夹里。



然后再新建一个python文件，命名为：face\_train。添加如下代码。



程序运行结果：



3、识别人脸

新建python文件,命名：Face\_recognition。代码如下：



运行结果：

