说明文档：

整个人脸识别流程分为三个部分（1）人脸检测；（2）人脸特征提取；（3）基于欧几里得距离的比较。

为了便于测试我将对于数据库数据的处理单独交给一个工程（align）处理，之后可能会考虑将功能合并。在DB文件夹下，为每一名用户单独创建文件夹保存照片，照片名随意，文件夹名为用户名。在align\_dataset\_mtcnn.py文件的main函数中调用Dect\_for\_batch方法，填入DB文件夹路径，处理后的图片会保存在DB\_dect文件夹中。（注意：工程内包含两个align\_dataset\_mtcnn.py文件，分别用作数据库处理和服务过程，这里使用image\_dect文件夹下的align\_dataset\_mtcnn.py。）

API\_VideoCapture.py文件作为启动程序，包括加载网络模型、获取数据库的全体特征向量并且调用其他模块完成识别流程。

完整的服务流程:

1. 加载神经网络检测模型MTCNN和特征提取模型Res-Net，返回网络的输入和输出节点。
2. 启动会话。
3. 启动获取数据库的全体特征向量（在数据库中由.txt格式保存），检测目标在（1:15000）的规模里进行检测。返回特征值格式包括两部分：(1)包含全体用户特征值的列表，列表元素为np的narray对象；（2）与全体特征值下标对应的列表，列表元素为用户姓名。
4. 启动摄像头：
   1. 从视频流里截取一帧图片，转为numpy三维数组格式。（这里获取图片选择了VideoCapture，获取的图片为传统的rgb格式与网络模型训练集一致，但是在进行现实的时候我选择了opencv的方法，opencv的图片格式是bgr,所以显示窗口颜色有问题，但不影响但不影响图片的处理）。
   2. 将图片作为输入，由MTCNN模型计算，计算结果返回boundingbox回归框标定检测结果的四个点。裁剪图片，通过回归点计算目标面积，将裁减后的统一化像素为256x256。
   3. 将检测后的图片作为Res-Net的输入，返回（16x128=2048）的特征值。（Res-Net的返回值是（n\*128）的形式，根据输入像素的不同返回不同的n。）
   4. 将2048长度的特征值与数据库加载的用户特征计算欧氏距离，取最小的欧式距离。

（为了充分利用计算资源这里首先将图片特征值转换为narray格式，然后扩展为一个与数据库特征值数量对应的矩阵，如[15000,2048]。然后将该矩阵和数据库特征值整体求欧几里得距离。注：在这里我为了放大差距并没有做开方运算，所以并不是一个标准的欧氏距离。）

* 1. 标定一个阈值（或者转换为相似度），当最小的欧氏距离小于阈值时认为确定识别。（经过反复测试，准确率和识别成功率难以兼得。当阈值小于16时基本不会出错，但同时也会难识别；如果以人脸签到的标准让人脸很端正地靠近摄像头，测得的欧式距离基本在14以内；为了保证识别速度，暂时将阈值调高，并通过多次确认的方式提高准确率）
  2. 将识别结果和目标面积加入一个全局字典。在字典中查找，当发现同一个目标连续被识别出三次（暂定三次）以上时，通过计算目标面积的变化可以判断目标是否有靠近的倾向。
  3. 当判断目标有靠近倾向时可以认为是找到了我们需要的人，为银行发送提示信息。提示信息包括了用户的基本信息和保存在数据库中的用户照片（可以考虑加上一张现场采集的照片作为比对）。
  4. 为了避免对于同一个用户反复生成提示，可以设定时间，比如在2分钟之内不再重复生成提示。

历史遗留问题，我对文件的处理没写好，文件夹需要用绝对路径。（inference.py的路径改为自己的DB\_dect文件路径）

只需要先在image\_dect下启动align\_dataset\_mtcnn.py文件处理数据库（可以为自己创建文件夹做测试），然后再在Grape下启动API\_VideoCaputure.py就可以使用服务。

测试的时候角度比较重要，人脸要和摄像头保持水平，然后要有向前进的动作才会提示消息。

文件有一些注释其实是草稿，还有一些废弃的函数没舍得删，看起来比较乱。

最后文件显示有点乱。

dect\_face.py不是我写的，我完全看不懂。