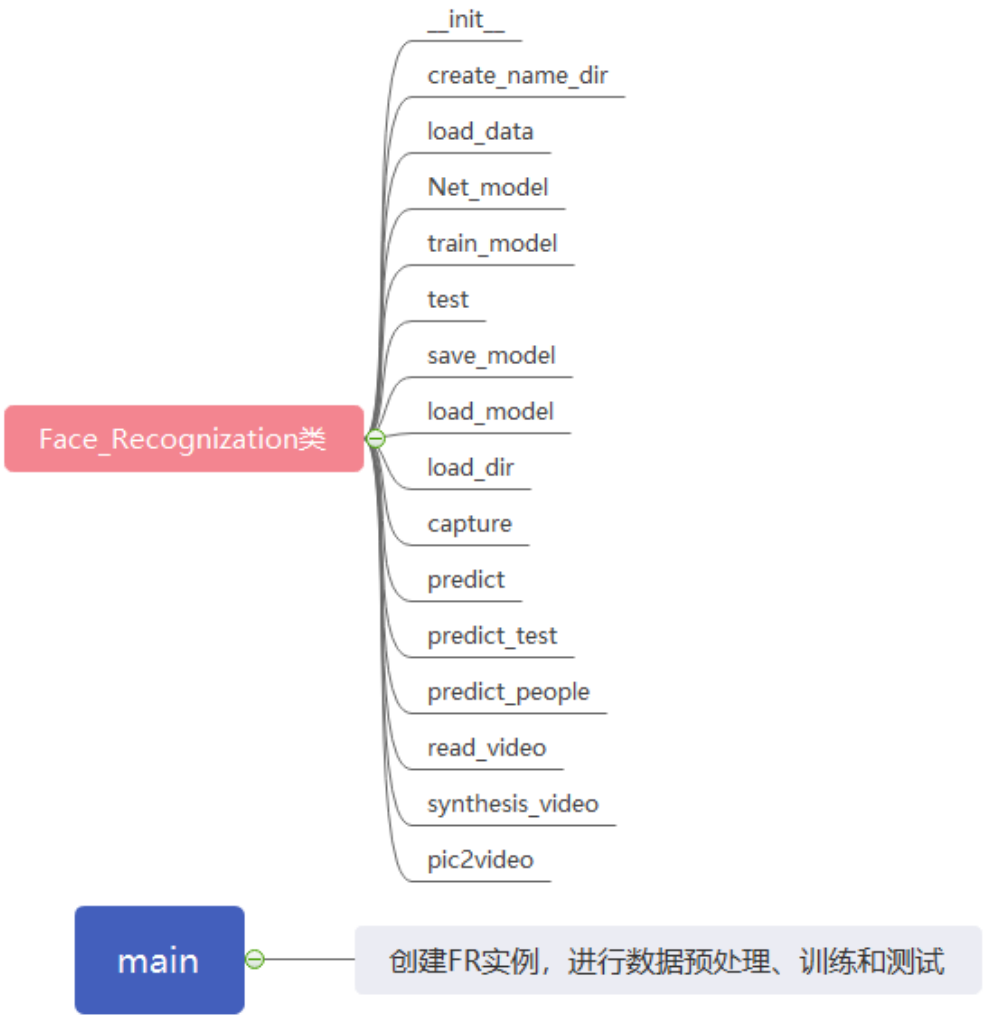
一. 编译环境

在 ubuntu16.04,python3.7 下进行编程,以 keras 为网络框架,调用了 numpy、keras、cv2、os、collection、face_recognition、pickle、collections 模块

二. 代码思路简述

一共一个 Face_Recognization 类，类内含有 16 个函数，关系如下：



1. Main 函数：

分为 4 个部分

1）对模型进行 train 和 test

实例化 Race_Recognization 类为 FR;

调用 create_name_dir 创建学号和名字对应的字典;

调用 load_data 加载和预处理数据;

调用 net_model 创建神经网络;

调用 train_model 训练模型;

调用 save_model 保存训练模型;

调用 test 检测模型

2）加载训练好的模型并用准备好的图片进行测试
实例化 Race_Recognization 类为 FR；
调用 load_dir 来加载字典模型；
调用 net_model 创建神经网络；
调用 load_model 来加载模型；
调用 predit_test 来对单张图片进行测试

3）加载训练好的模型并在摄像头下进行预测
实例化 Race_Recognization 类为 FR；
调用 load_dir 来加载字典模型；
调用 net_model 创建神经网络；
调用 load_model 来加载模型；
调用 capture 来

4）加载训练好的模型并处理视频流
实例化 Race_Recognization 类为 FR；
调用 load_dir 来加载字典模型；
调用 net_model 创建神经网络；
调用 load_model 来加载模型；
调用 read_video 来读入视频
调用 synthesis_video 来逐帧检测并生成新的视频

2. Face_Recognization 类：
   1) __init__函数：初始化空的训练和测试集，初始化标签和学号、学号和姓名的转换字典
   2) create_name_dir 函数：根据选课名单，创建学号和名字相对应的字典
   3) load_data 函数：在运行程序前，先结合该函数人工筛选了一些不符合规范的数据，删掉了部分数据集。此函数预处理数据，将每位同学的前 700 张图片读入了训练集，并对每张图片进行了 2 次亮度增强的变化，2 次亮度减弱的变换，即每个同学的训练数据为 700*5=3500，后 300 张图片读入了测试集，并建立了概率标签。创建了初始化标签和学号的转换字典。
   4) Net_model 函数：定义神经网络：输入层→卷积→池化→dropout→卷积→池化→dropout→扁平化处理→第一个全连接层→第二个全连接层→第三个全连接层→第四个全连接层(输出层)。优化器为 adagrad，损失函数为交叉熵函数。
   5) Train_model 函数：训练模型
   6) Test 函数：评估模型
   7) Save_model 函数：保存模型
   8) Load_model 函数：加载模型
   9) Load_dir 函数：打开摄像头，调用 face_location 来捕捉人脸，对捕捉的人脸分别进行预测，并加结果展示在图片上
   10) predict 函数：预测传入的图片，返回为学号和姓名，此函数共其他函数调用
   11) predict_test 函数：输入图片的路径，输出为姓名和学号(此时的输

入必须为 128x128x3 的图片）

12）predct_people 函数：输入任意大小的图片，可检测多人，进行预测，绘图，对人脸进行框选并在框附近输出学号和人名

13）read_vido 函数：读取视频流，并逐帧进行预测

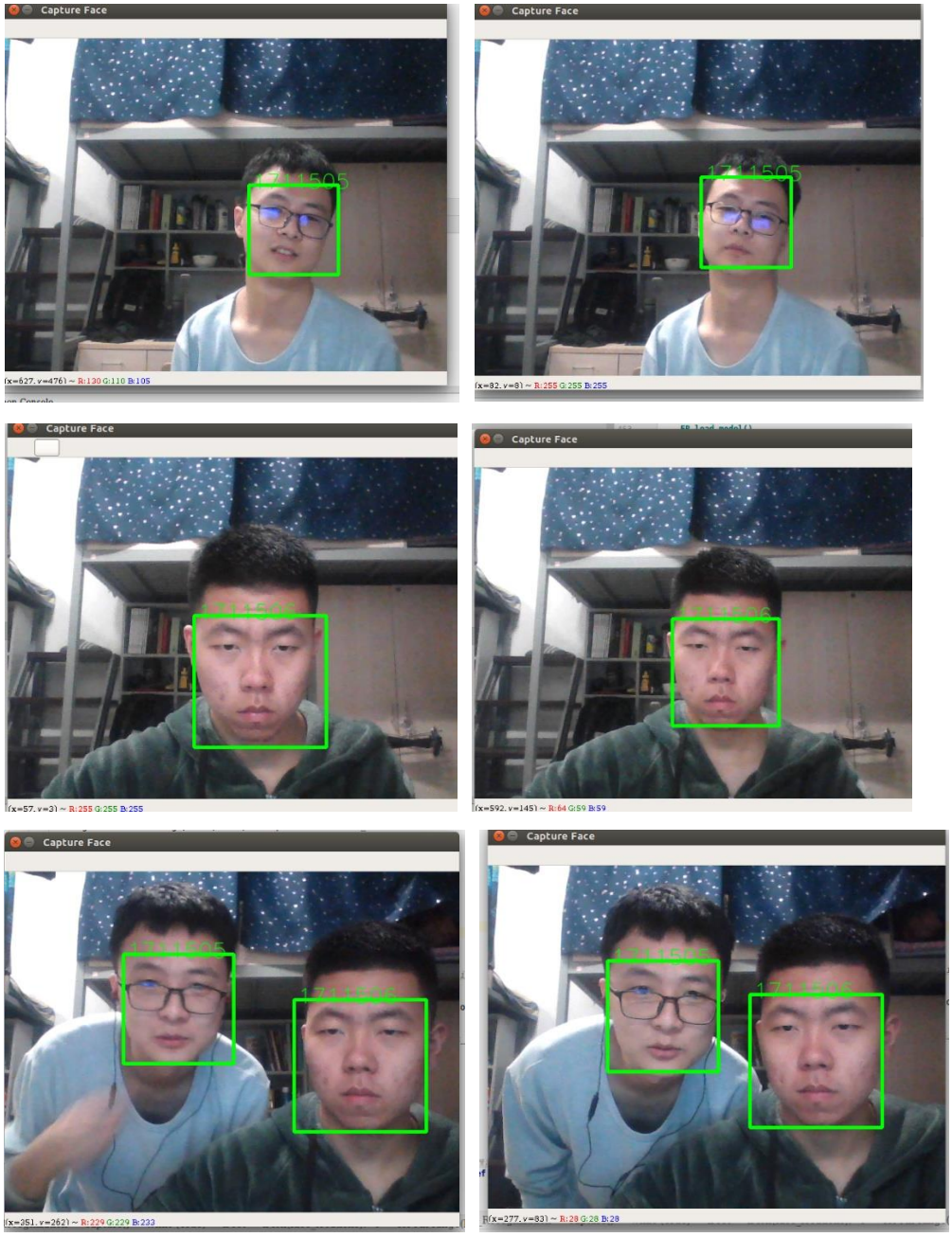
14）synthesis_video 函数：将逐帧预测完的图片合成为 MP4 类型的视频并进行保存

15）pic2video 函数：将图片加载成视频，供后期使用

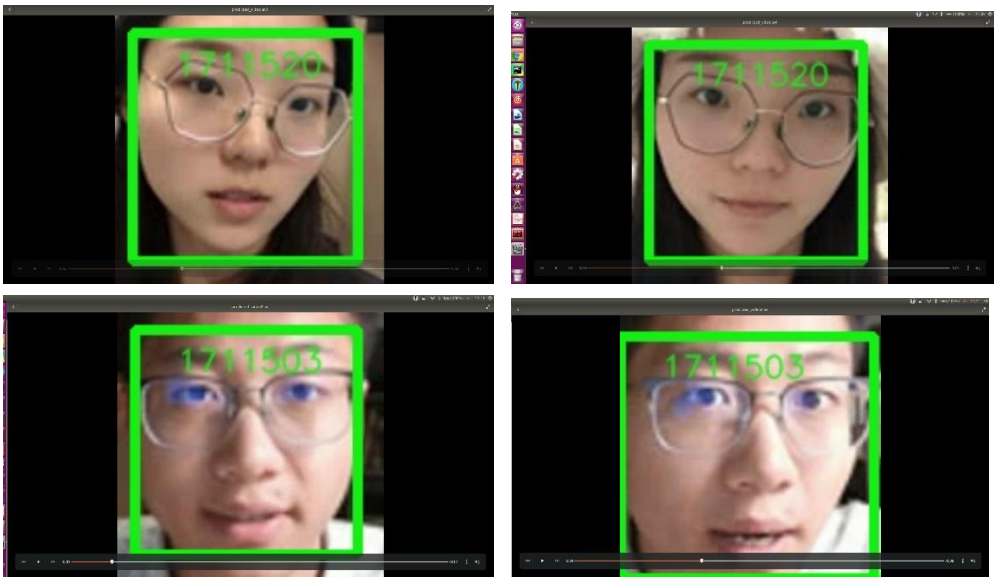
三. 效果展示

训练的准确率在 99.9%以上，测试集的准确率约为 84%

1. 摄像头预测效果



2. 处理视频流（部分截图）



四. 总结

1. 预处理：
   在数据预处理的时候有一部分图片有问题，而且有很多图片，人脸占的比例很小在这里用 face_recognition 的库对所有的图片进行了一次处理；训练集的亮度对预测产生了很大影响，同一个人在不同的光照强度下，预测出来的效果差别很大，在这里对所有的图片均进行了 4 次亮度变换，两次变亮，两次变暗，将训练集成了原来的 5 倍。
2. 电脑的性能问题：
   网络搭太大的话跑不动，所以这里只用了 10 层网络，预测的效果不是很好，对于 test 集，准确率大概在 80%多，但是对于其他生活中的照片（角度、光照、穿的衣服等均与训练集、测试集差别较大），预测效果不好；针对 memory error 的问题，自行分批次对网络进行了训练（读入部分数据、训练后保存模型、释放数据；读取新的数据并进行训练，如此反复）
3. 关于神经网络：
   通过这次学习，对 tensorflow、keras 和里面的激活函数、loss 值计算方法等都有了一定的了解，对 cv 和 face_recognition 等进行了浅显的了解和应用
4. 未来的改进方式：
   增加网络层数；
   调整参数；
   增大数据集、增高数据集的质量