**无人机跟随人脸移动实验报告**

**2018202072 王玺**

**Task 1：opencv级联分类器**

**工作内容概述：实现了在图片中找到人脸并输出人脸框图及中心点位置。方法采用opencv库自带的级联分类器及参数。**

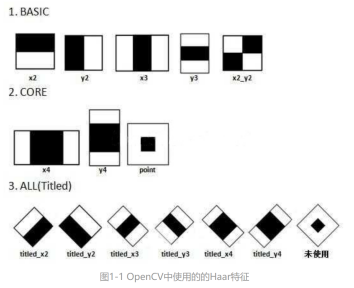
1. **原理分析**

*（1）特征的获取*

在分类器的实现中采用由Paul Viola等人提出的Haar特征。其获取步骤概述如下：

***Step1：***

采用不同的特征形式检测窗口在图片中放大，平移，得到不同大小，不同位置的Haar系列特征。不同检测窗口获取不同类型的图像信息，如横向信息，纵向信息。



***Step2：***

计算Haar特征值。其公式为：



即Haar特征值=整个Haar区域内像素和×权重 + 黑色区域内像素和×权重：不同特征检测窗口的对应的各区域权重不同。图1-2中的x3特征的weight\_black=-3，相当于白色区域像素-2\*黑色区域像素。

***Step3：***

标准化特征值。由于计算特征量时用到区域内像素和，导致仅仅12\*18大小的窗口计算出特征值变化范围为-2000-6000.不利于量化评定特征值。因此采用计算平均值，计算标准化因子等多种方式对同一位置检测窗口特征值进行标准化。

2）特征获取结果：

获取了一系列具有区分图像信息功能的一维特征值，在保存图像信息的同时实现降维。

特征值的个数与窗口大小和图片大小有关。

3）为什么Haar特征是有效的：

Haar特征的获取过程类似于CNN，特征窗口类似于CNN内核。只不过在CNN中，内核的值由训练确定，而Haar特征是手动确定的。从检测窗口的不同形式来看，这些特征用于检测边缘，线，矩形，斜线。

*（2）Adaboost级联分类器*

总体结构：树状结构，每个stage代表一级强分类器。某个检测窗口通过所有强分类器时被认为是目标，否则拒绝。

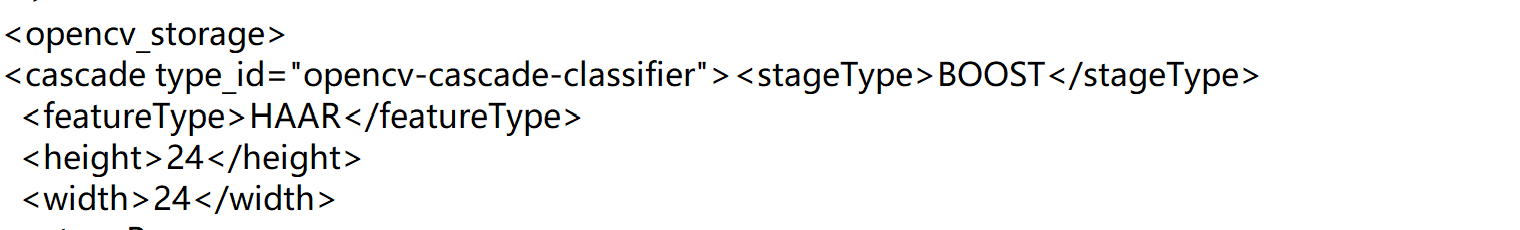
弱分类器结构及工作原理

弱分类器也是树状结构。以一个深度为2的弱分类器为例，通过将两个特征值Haar1和Haar2与阈值t1,t2对比，选择输出leftvalue, rightvalue中的哪一个。

训练弱分类器的目的即是确定特征值的最优阈值t，使得这个弱分类器对所有训练样本的分类误差最低。

*（3）Xml文件参数部分解释*

feature类型为HAAR, 检测窗口高度和宽度均为24



弱分类器包含了<internalNodes>和<leafValues>两个标签。其中<leafValues>标签中的2个浮点数由左向右依次是rightValue、leftValue；而<internalNodes>中有整数和浮点数，其中浮点数是弱分类器阈值，整数用于标示所属本弱分类器Haar特征存储在<features>标签中的位置，以及用于控制弱分类器树的形状

1. **结果分析及遇到的问题：**

用15s视频每秒10张做测试，保存返回结果的同时输出每张照片判别速率

照片结果见testvedio文件夹，结果统计如下：

|  |  |  |
| --- | --- | --- |
|  | 判别为人脸 | 未判别为人脸 |
| 有人脸 | 127 | 7 |
| 无人脸 | 16 | 0 |

|  |  |  |
| --- | --- | --- |
| precision | recall | F1\_score |
| 88.81% | 94.78% | 91.05% |

判别速率部分结果展示详见facedetect.ipynb，输出每张照片的判别速率。图片编号1-150中前40张图片用时较多，在0.3s左右，从后40张图片开始平均用时在0.1-0.2s之间，可以支持每秒5-10张的判别

**遇到的问题：**

从结果测试可以看出，此方法对人脸检测结果并不准确，因此后期尝试其他方法。

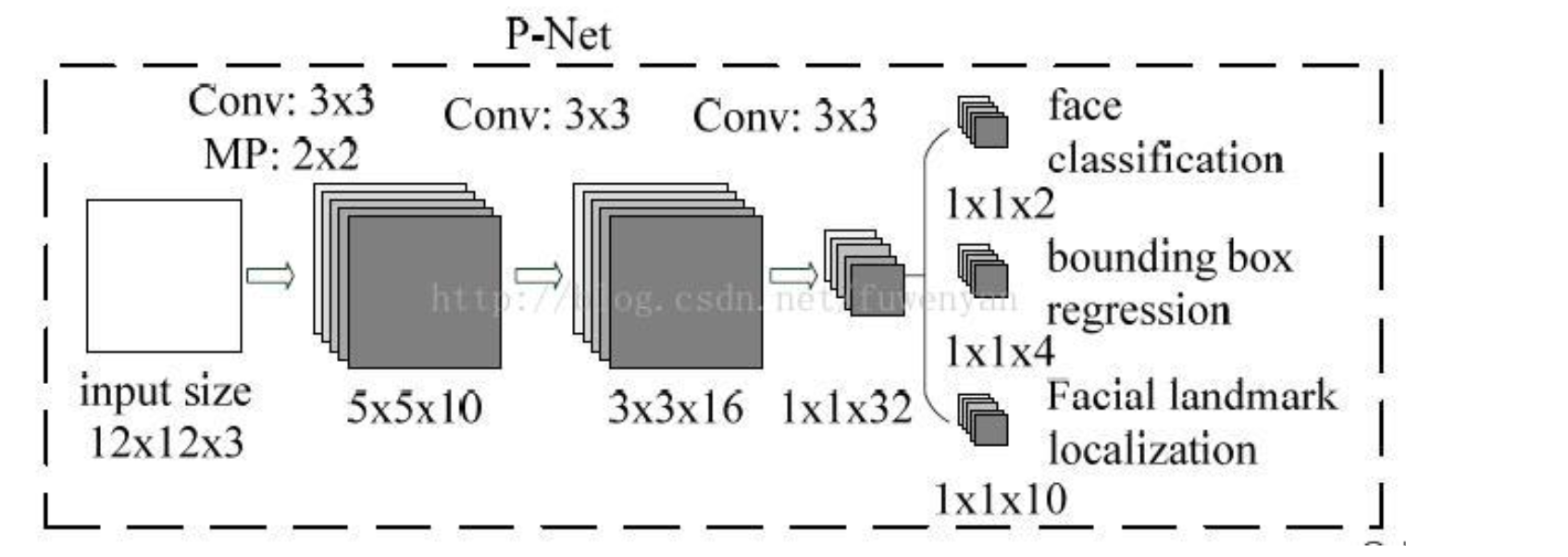
**Task 2: mtcnn**

**工作内容概述：同上，采用github上mtcnn\_pytorch开源项目，使用训练好的神经网络模型进行判别。**

**1原理分析**

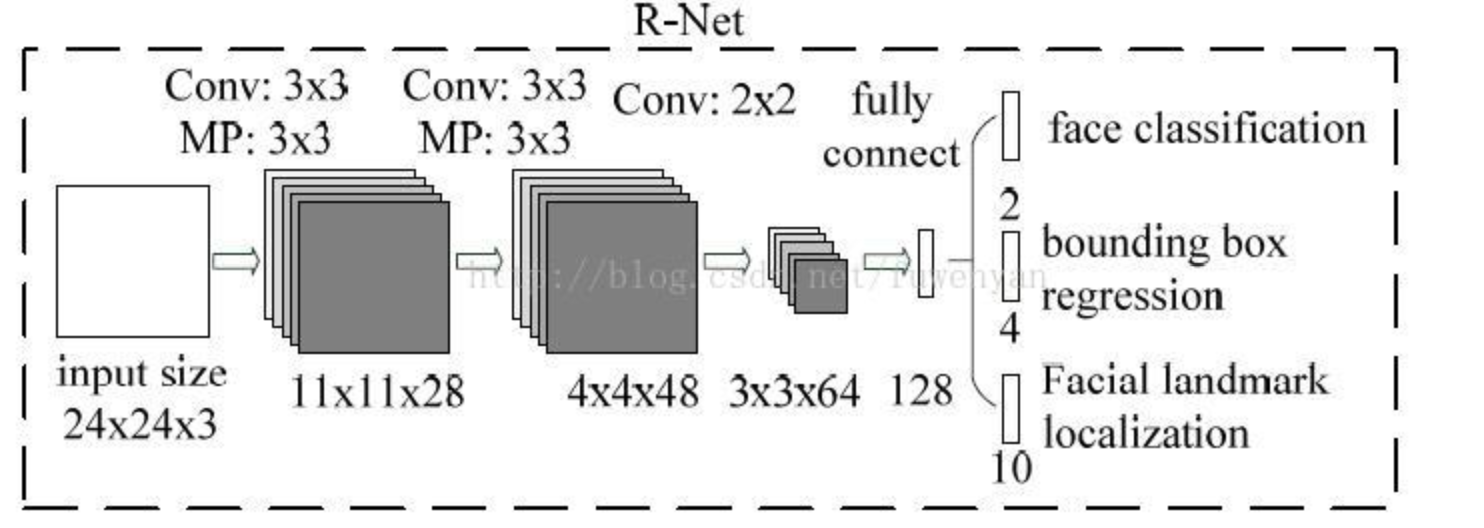
1. Pnet:

输入图片，输出一批人脸框



1. Rnet:

Rnet的作用是对Pnet得到的人脸框进一步打分筛选，结果为精选出来的人脸框



1. Onet

Onet的作用是对Rnet得到的人脸框进一步打分筛选，回归人脸框。同时在每个框上都计算特征点位置。

1. **结果分析及遇到的问题：**

用15s视频每秒10张做测试，保存返回结果的同时输出每张照片判别速率

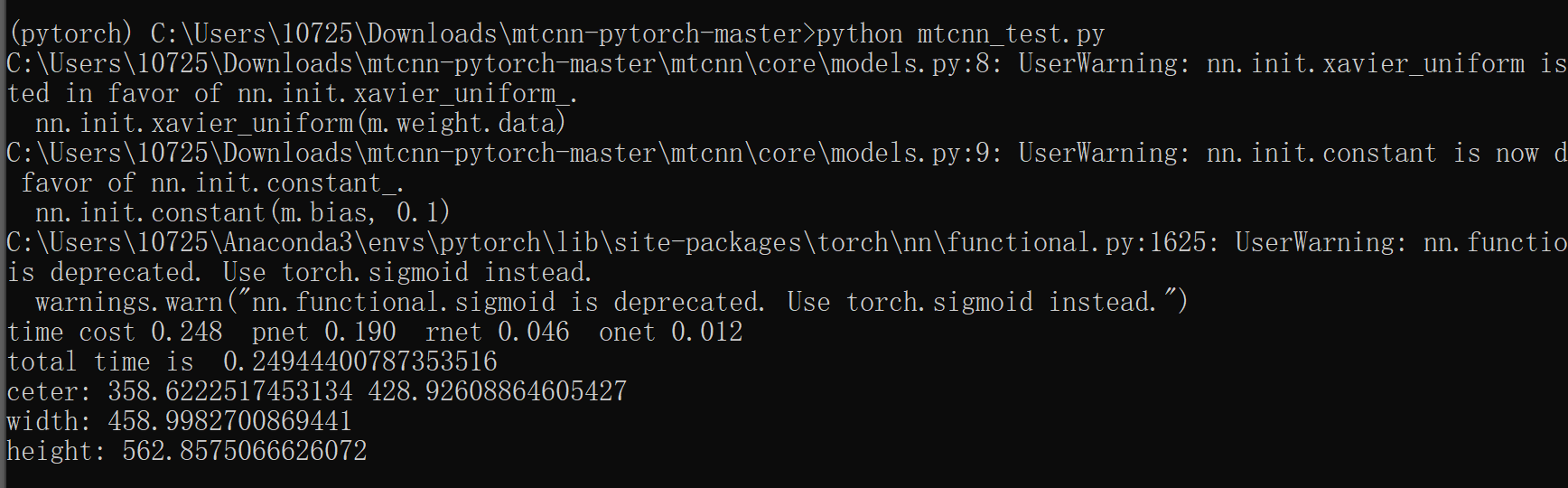
照片结果见vedioout文件夹

|  |  |  |
| --- | --- | --- |
|  | 判别为人脸 | 未判别为人脸 |
| 有人脸 | 121 | 13 |
| 无人脸 | 0 | 16 |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| precision | recall | F1\_score |
| 100% | 90.3% | 94.9% |

结果中可看出准确率很高，达到100%，召回率较低，但从应用角度来看，无人脸但判定为有人脸的情况更为严重，相比之下有人脸但判别为无人脸的情况不会导致严重错误。

判别速率相比opencv cascade稍高，但仍在0.2-0.3之间，可以支持1s内3-5张图像的判别



**收获与反思**

在应用了两个面部检测器后发现了opencv分类器和mtcnn分类器的差别。Opencv检测结果recall较高但precision较低。而mtcnn判别结果较为保守，precision较高但recall较低。

其原因是与二者识别架构有关，opencv的级联分类器虽然应用了多个弱分类器的联合，以像素点为单元经过了多次检验，但只有一次判断框的合并，因此其识别能力较强，即使是很远的人脸也可以识别出来，但判断能力较弱，无法保证识别出来的是人脸。相比之下mtcnn选择多个候选框，以候选框为单位进行多次检验，则在准确性上较好，保证识别出来的一定是人脸，但也会出现在模糊或者距离较远的情况下无法识别人脸。

这次经验让我理解了不同实现原理对功能的细节影响。在大目标都是人脸识别的条件下，如果重在人脸的识别与定位，那么opencv的级联分类器更好，更灵敏；而如果还要加入有无人脸的判定，则mtcnn更有优势。在我们的项目中无人机飞行在一些情况下很有可能拍不到人脸，因此采用mtcnn.

**Task3 :特定面孔识别（跟随特定人脸飞行）**

**工作内容概述：使用ORL人脸数据集进行人脸分类的试验。共包含40个不同人的400张图像，此数据集下包含40个目录，每个目录下有10张图像，每个目录表示一个不同的人。训练目标为对这些不同个人的照片进行分类。**

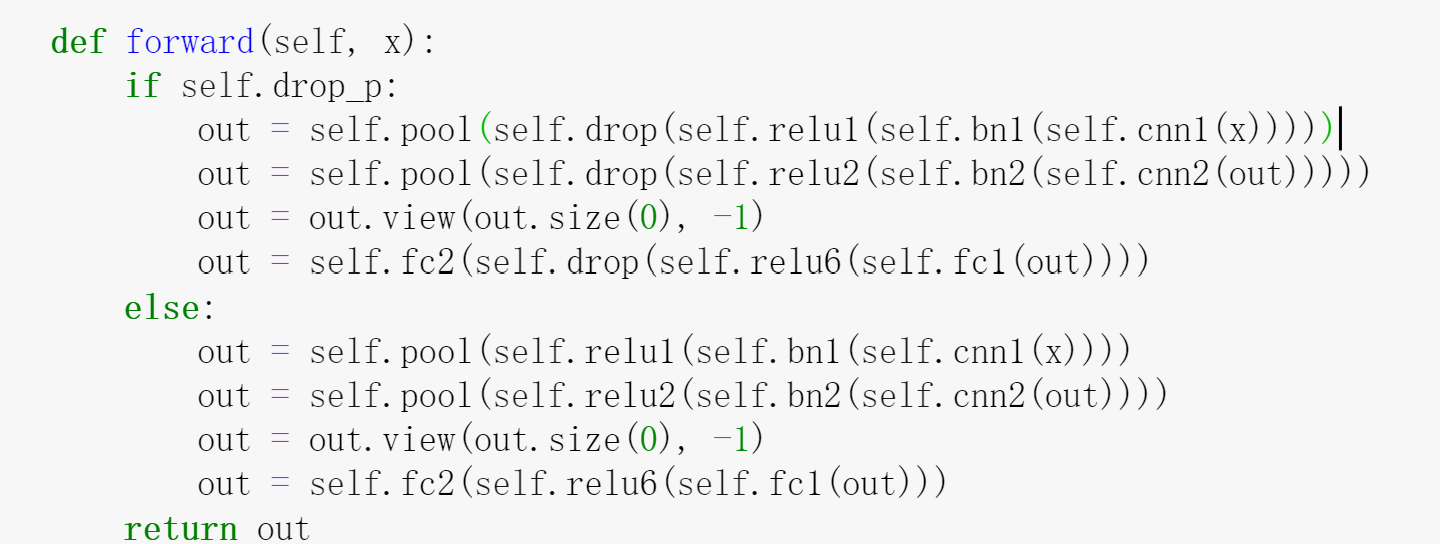
**1原理分析**

用pytorch搭建CNN。

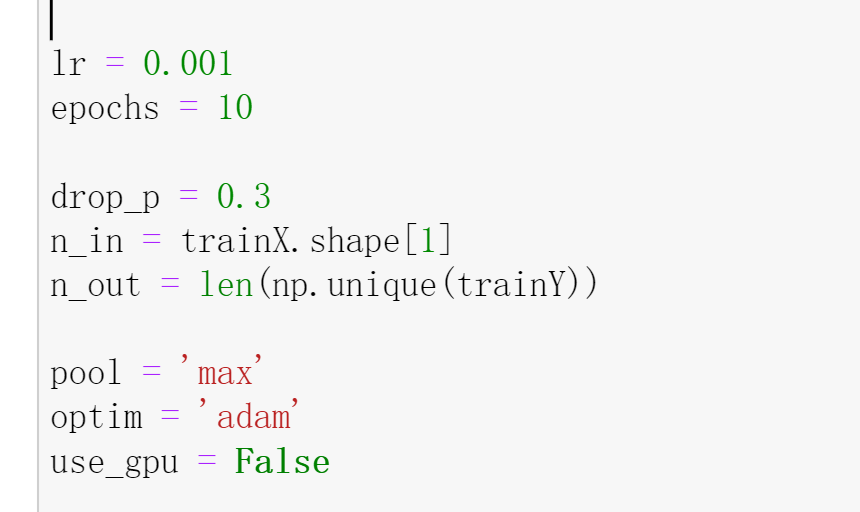
**2 关键部分代码解释**

1 forward函数：

CNN架构包含两层卷积层，一层全连接层。训练过程可使用dropout。

****

1. 超参的选择：可以在最后一个模块即对model类的调用运行处选择不同参数进行运行。尝试了epochs[10,15]; lr[0.001-0.005] 选择了结果最好的lr=0.001 epochs=10



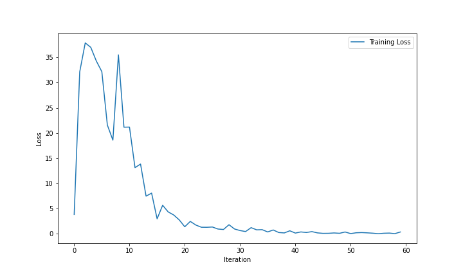
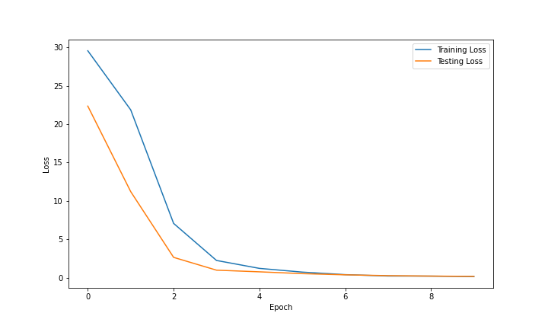
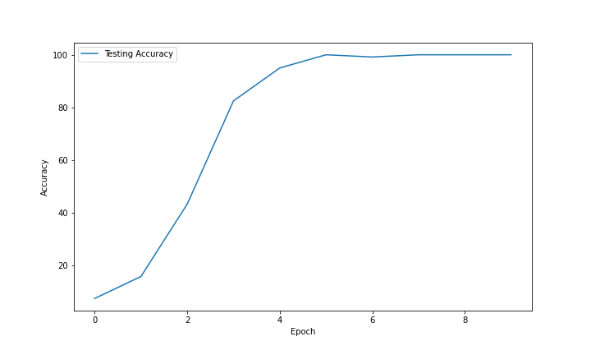
1. **遇到的问题：**
2. 图像数据类型转换问题：

在将图像转换为npz文件时使用的数据是int型，在使用torch.utils.data.TensorDatase函数时需要把类型转换为long型。

2) 设置CNN层维度时出错，例如最后一层fc2层输出维度与标记的输出结果维度不符合等

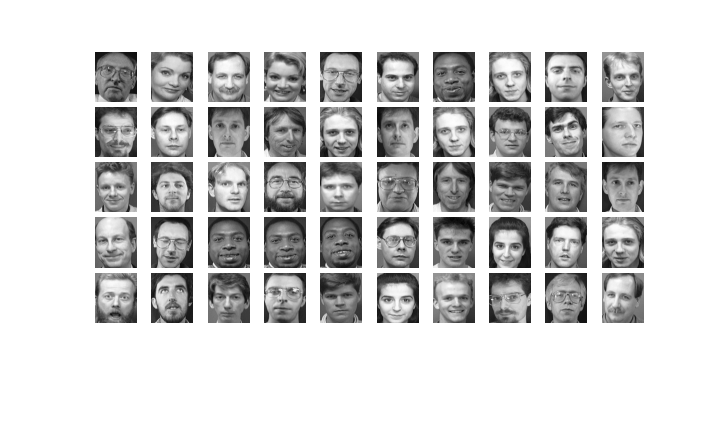
**4结果展示**

最优测试集准确率达到99.98%,训练集准确率90%。



**5模型在项目中的应用：**

实现特定人脸跟踪可以制作自己的类似ORL的人脸数据集：通过判断输出来决定无人机的移动。但由于时间限制，此部分并没有实际完成。



**一些其他的工作：**

1. 完成了人脸移动的决策函数。即检测出人脸中心点和期望中心点间的距离，检验框面积大小之间的比值来确定向左右前后移动的距离。

**本次项目的完成收获：**

1. 系统了解了图像识别，分类的原理，并检验了不同架构下识别效果的异同。

2 第一次尝试用数据集训练CNN神经网络，掌握了pytorch特别是forward函数的编程方法

1. 理解了检验结果需要的定量化的判别标准。例如在人脸检测中，比起个别图像的成功更重要的是在大量图像集上的表现。

**本次项目的缺点及反思**

1 此次在人脸检测部分的接口没有处理完善。现在所使用的接口是从文件夹读取图片然后进行检测，因此在设计接口时采用的是每秒截取图片保存至文件夹，再读入识别。这样的IO操作使得对每帧的决策速度较低，但时间大部分花在了传输上，如果没有IO操作直接传入图像矩阵，省去各部分转换，传输时间，根据测试，所用的检测器mtcnn可以支持每秒3-5帧的检测，从而提高实时性。

2 这次是我第一次完整地参与做一个项目，经过此次积累经验，下次做项目模块时会更加注重从项目整体的实现考虑而不是只从单一模块实现的角度考虑问题；同时，做出效果的好坏一定要定量检验。