目录

[MTCNN解读 2](#_Toc503274807)

[数据与处理： 2](#_Toc503274808)

[第一阶段， 2](#_Toc503274809)

[第二阶段， 3](#_Toc503274810)

[第三阶段， 3](#_Toc503274811)

[基于caffe的mtcnn训练实现 5](#_Toc503274812)

[一、训练 5](#_Toc503274813)

[（1）样本问题： 6](#_Toc503274814)

[（2）网络问题 7](#_Toc503274815)

[二、训练步骤 8](#_Toc503274816)

[三、使用阶段 8](#_Toc503274817)

[视频车牌定位—mtcnn 9](#_Toc503274818)

[颜色定位和形态学定位改进后的mtcnn车牌定位算法 10](#_Toc503274819)

MTCNN解读

解读论文为《Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks》基于多任务级联卷积神经网络的人脸检测和对齐

论文地址：<https://kpzhang93.github.io/MTCNN_face_detection_alignment/>

本文作者：非文艺小燕儿，博客地址：<http://blog.csdn.net/fuwenyan/article/details/73201680>

这篇文章在人脸检测和特征点定位任务上，精度较之前state-of-art的算法有明显的提升，而且具有实时处理的性能。

文中有2个点：

(1)通过三阶的级联卷积神经网络对任务进行从粗到细的处理；

(2)还提出一种新的在线困难样本生成策略可以进一步提升性能。

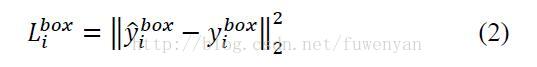
最主要的点，应该算是三阶的级联卷积神经网络。

每个阶段的网络都是一个多任务网络。处理任务有三个：人脸/非人脸判定、人脸框回归和特征点定位。

人脸/非人脸判定采用cross-entropy损失函数：

http://img.blog.csdn.net/20170614104608998?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvZnV3ZW55YW4=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

人脸框回归采用欧式距离损失函数：



特征点定位也采用欧式距离损失函数：

http://img.blog.csdn.net/20170614110820100?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvZnV3ZW55YW4=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

其中a表示三个任务在当前阶段的网络中损失所占比重。B是采样类型指示，取值为{0,1}，当人脸/非人脸判定为非人脸时，box和landmark的B取值0，而det取值1；判定为人脸时，全部取值为1.

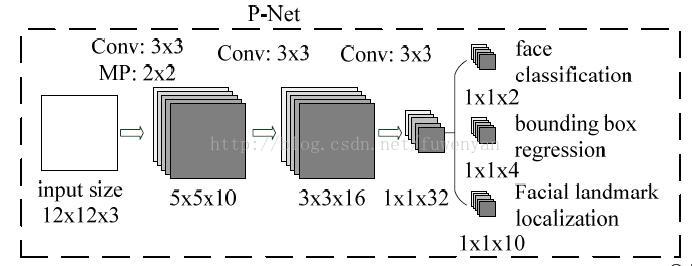
接下来详细讲述各阶段：

数据与处理：为应对目标多尺度问题，将原始图像resize到不同尺寸，构建图像金字塔，作为三阶级联架构的输入。

第一阶段，通过一个浅层的CNN快速生成候选窗口。

该阶段是一个全部由卷积层组成的CNN，取名P-Net，获取候选人脸窗口以及人脸框回归向量。基于人脸框回归向量对候选窗口进行校正。之后采用NMS合并高重叠率的候选窗口。

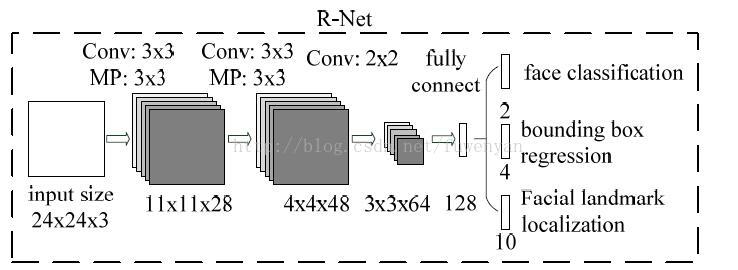
该阶段在三个任务det、box、landmark任务上，a的对应取值为{1.0,0.5,0.5}.



第二阶段，通过一个更复杂的CNN否决大量非人脸窗口从而精化人脸窗口。

第一阶段输出的候选窗口作为R-Net的输入，R-Net能够进一步筛除大量错误的候选窗口，再利用人脸框回归向量对候选窗口做校正，并执行NMS。

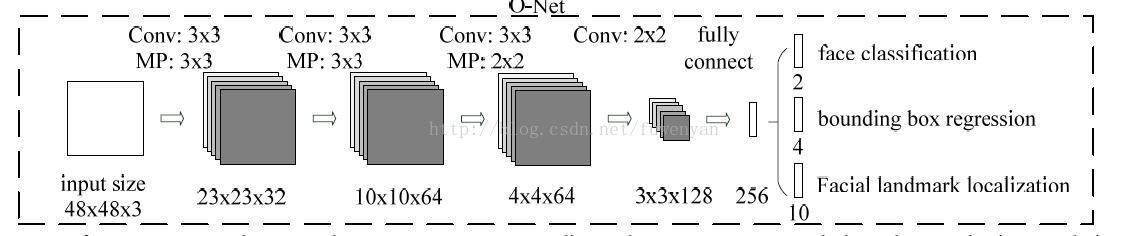
该阶段在三个任务det、box、landmark任务上，a的对应取值为{1.0,0.5,1.0}.



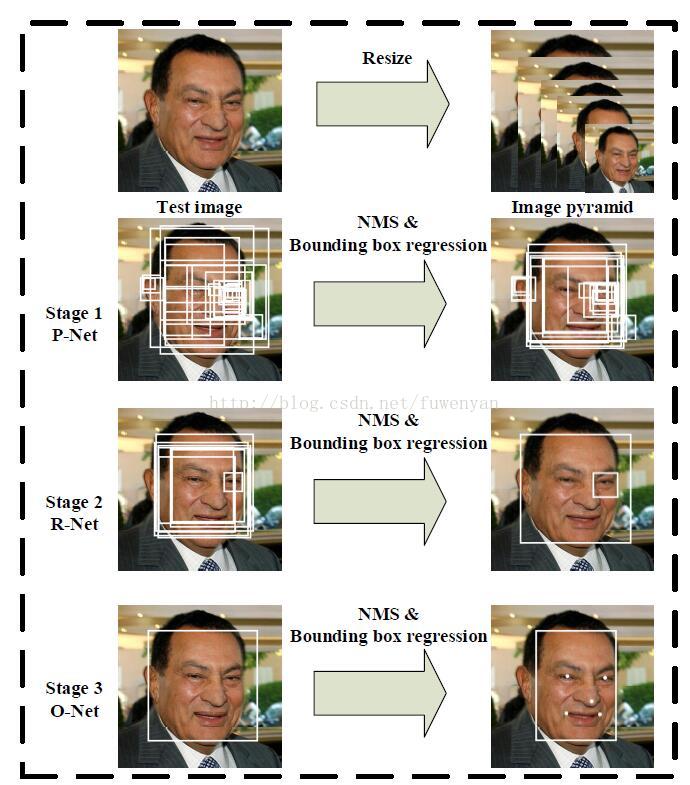
第三阶段，使用更复杂的CNN进一步精化结果并输出5个人脸特征点。

与stage2相似，但这一阶段用更多的监督来识别人脸区域，而且网络能够输出五个人脸特征点位置坐标。

该阶段在三个任务det、box、landmark任务上，a的对应取值为{1.0,0.5,1.0}.



以上四个步骤的直观图示如下：



关于另外一个在线困难样本生成的策略，是指不同于以往离线生成采样的方式，本文的处理方式是在每个mini-batch中选出70%大loss的样本作为困难样本，并且在BP阶段只利用这部分计算梯度。也就是说抛弃了一些对强化分类没有多大作用的简单样本。文中表示，通过实验表明这样做能够进一步提升精度。

基于caffe的mtcnn训练实现

可以训练一个自己的有效的目标检测算法,非常容易非常简单，并且有配套的纯c++版本的mtcnn-light。该算法证实可以被用来做其他目标检测，效果非常好，效率也非常高

一、caffemodel\_2\_mtcnnmodel:

caffemodel文件转换为mtcnn-light支持的.h头文件类型的模型，可以直接丢进去替换编译就行了

二、mtcnn-light:

改进版本的mtcnn-light，配合训练程序，和转换程序，实现一整套的流程到最后部署

三、train:

训练过程所使用的程序和脚本

四、mtcnn-caffe:

一个caffe实现的简单，稳定有效的mtcnn版本

## 一、训练

论文中作者主要使用了Wider\_face 和CelebA数据库，Wider\_face包含人脸边框标注数据，大概人脸在20万,CelebA包含边框标注数据和5个点的关键点信息．对于三个网络，提取过程类似，但是图像尺寸不同．其中Wider\_face主要用于检测任务的训练，CelebA主要用于关键点的训练．训练集分为四种:负样本，正样本 ，部分样本，关键点样本. 三个样本的比例为3: 1: 1: 2

P-net的任务是人脸检测和人脸框回归，所以该阶段仅需要使用WIDER FACE数据集。为什么没带特征点标定任务？P-net输入12\*12的图像，图像太小，不适合做特征点标定。

（1）取候选窗，生成训练图片

下载的原始数据集并不能直接用于训练，而是在原始图像上截取候选框图像。随机截取候选框，根据IoU计算候选框所属类别，并将其resize到12\*12大小，将resize后的候选框图像保存到对应的类别文件夹下。每个类别生成一个txt文档，存放图片路径以及对应的候选框位置信息。

根据IoU计算选择的候选框是属于negative（IoU<0.3）、positive（IoU>0.65）、part（0.4<IoU<0.65），三者的比例为3:1:1。

其中positive和negative用于人脸检测，positive和part用于人脸框回归。

（2）生成imdb，训练数据

使用positive和negative类别中的图像生成人脸检测所需的数据cls.imdb

使用positive和part类别中的图像生成人脸框回归所需的数据roi.imdb

(3)构建网络和solver，开始训练

训练主要包括三个任务

人脸分类任务：利用正样本和负样本进行训练

人脸边框回归任务：利用正样本和部分样本进行训练

关键点检测任务：利用关键点样本进行训练

### （1）样本问题：

#### 1、训练数据整理:

mtcnn训练时，会把训练的原图样本，通过目标所在区域进行裁剪，得到三类训练样本，即：正样本、负样本、部分(part)样本

#### 2、正负样本，部分样本提取：

１．从Wider\_face随机选出边框，然后和标注数据计算IOU，如果大于0.65，则为正样本，大于0.4小于0.65为部分样本，小于0.4为负样本．

２．计算边框偏移．对于边框，(x1,y1)为左上角坐标，(x2,y2)为右下角坐标，新剪裁的边框坐标为(xn1,yn1),(xn2,yn2),width,height．则offset\_x1 = (x1 - xn1)/width,同上，计算另三个点的坐标偏移．

３．对于正样本，部分样本均有边框信息，而对于负样本不需要边框信息

#### 3、关键点样本提取

１．从celeba中提取，可以根据标注的边框，在满足正样本的要求下，随机裁剪出图片，然后调整关键点的坐标．

其中：裁剪方式：对目标区域，做平移、缩放等变换得到裁剪区域。IoU：目标区域和裁剪区域的重合度

#### 4、Loss修改

由于训练过程中需要同时计算３个loss,但是对于不同的任务，每个任务需要的loss不同．

所有在整理数据中，对于每个图片进行了15个label的标注信息

1.第1列：为正负样本标志，１正样本,0负样本,2部分样本,3关键点信息

2.第2-5列：为边框偏移，为float类型，对于无边框信息的数据，全部置为-1

3.第6-15列：为关键点偏移，为floagt类型，对于无边框信息的数据，全部置为-1

修改softmax\_loss\_layer.cpp　增加判断，只对于1,0计算loss值

修改euclidean\_loss\_layer.cpp　增加判断，对于置为-1的不进行loss计算

#### 5、困难样本选择

论文中作者对与人脸分类任务，采用了在线困难样本选择，实现过程如下：修改softmax\_loss\_layer.cpp，根据计算出的loss值，进行排序，只对于70%的值较低的数据，进行反向传播．

#### 6、此时三类样本如下定义：

正样本：IoU >= 0.65，标签为1

负样本：IoU < 0.3，标签为0

部分(part)样本：0.65 > IoU >= 0.4，标签为-1

对于训练数据，分为四个组成部分

1. 负样本 3

2. 正样本 1

3. 部分样本 1

4. 关键点样本 2

比例为 3:1:1:2

其中对于 1,2,3中数据，来自wider\_face数据库

对于4中数据，来自CelebA数据库

包括 label box landmark

label

negative 0

positive 1

part 2

landmark 3

（2）网络问题，mtcnn分为三个小网络，分别是PNet、RNet、ONet，新版多了一个关键点回归的Net（这个不谈）。

PNet：12 x 12，负责粗选得到候选框，功能有：分类、回归

RNet：24 x 24，负责筛选PNet的粗筛结果，并微调box使得更加准确和过滤虚警，功能有：分类、回归

ONet：48 x 48，负责最后的筛选判定，并微调box，回归得到keypoint的位置，功能有：分类、回归、关键点

分为三层网路PNet RNet ONet

#### 1.输入:积分图像

输出:prob1 置信度

conv4-2 边框偏移

2.输入：根据第一步提取的边框，提取图片，作为batch进行输入

输出：prob1： batch\_size \* 2 置信度

conv5-2: batch\_size \* 4 边框偏移

3.输入：根据第二步提取的边框，提取图片，作为batch进行输入

输出：prob1: batch\_size \* 2 置信度

conv6-2: batch\_size \* 4 边框偏移

conv6-3: batch\_size \* 10 人脸关键点

4.网络大小的问题，训练时输入图像大小为网络指定的大小，例如12 x 12，而因为PNet没有全连接层，是全卷积的网络，所以预测识别的时候是没有尺寸要求的，那么PNet可以对任意输入尺寸进行预测得到k个boundingbox和置信度，通过阈值过滤即可完成候选框提取过程，而该网络因为结构小，所以效率非常高。

## 二、训练步骤

参考：<https://github.com/dlunion/mtcnn/tree/master/train>

一般训练几万次后，loss到0.0x的时候就可以接受了

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 三、使用阶段

将训练的caffemodel，复制到caffemodel\_2\_mtcnnmodel里面，编译执行他（代码写的必须3个网络同时存在，所以自己看情况改下），这时候产生的mtcnn\_models.h，就是我们要的网络头文件，添加到mtcnn-light覆盖下就可以执行看效果了

引用：

<https://github.com/kpzhang93/MTCNN_face_detection_alignment> <https://github.com/Seanlinx/mtcnn>

<https://github.com/CongWeilin/mtcnn-caffe>

<https://github.com/AlphaQi/MTCNN-light>

<https://github.com/dlunion/CCDL>

注意：

该mtcnn项目的计算方法，和mtcnn-light、和官方的mtcnn计算方法有区别，所以不能被通用，当然如果你修改下也是很容易就跟原版一样了，只是我修改的版本更加容易理解吧

# 视频车牌定位—mtcnn

这段时间一直在弄关于对视频的车牌定位，学了一点Python。真是人生苦短，我用Python啊。

对视频进行车牌定位是一类对时间效率要求很高的车牌定位。将视频的每一帧进行处理之后得到图中的类似车牌的位置。当Fps达到30之后，一连串的图片在人眼中就成为了视频，所以对于每一帧的处理时间要在30ms以下。

都说神经网络很火，所以我最先采用的就是利用神经网络的方法来进行车牌定位。

采用的是mtcnn（Multi-task convolutional neural networks）这个算法。mtcnn之前是用来做人脸识别的，关于这一方面有很多文章就不多描述了。

MTCNN由3个网络结构组成（P-Net,R-Net,O-Net），如下图所示

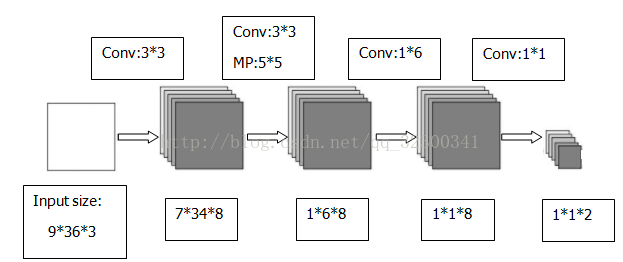


ProposalNetwork (P-Net)：利用滑动窗口在图片中选出候选车牌并用NMS算法去除掉重叠的窗口

RefineNetwork (R-Net)：该网络结构还是通过边界框回归和NMS来去掉那些false-positive区域。该网络结构和P-Net网络结构有差异，多了一个全连接层，所以会取得更好的抑制false-positive的作用。

OutputNetwork (O-Net)：该层比R-Net层又多了一层卷积层，所以处理的结果会更加精细。作用和R-Net层作用一样。

第一层详细的网络结构图如下



运行结果:

对视频（视频每帧图片大小1080\*1920\*3）中的车牌基本能够识别出来，然而误检率较高，时间效率较差。

第一层耗时2.2s左右

第二层耗时0.2左右

第三层耗时0.1s左右

与用在人脸识别时达到的15Fps有较大的差距。

代码已经上传到地址如下：<http://download.csdn.net/download/qq_32300341/9953142>

# 颜色定位和形态学定位改进后的mtcnn车牌定位算法

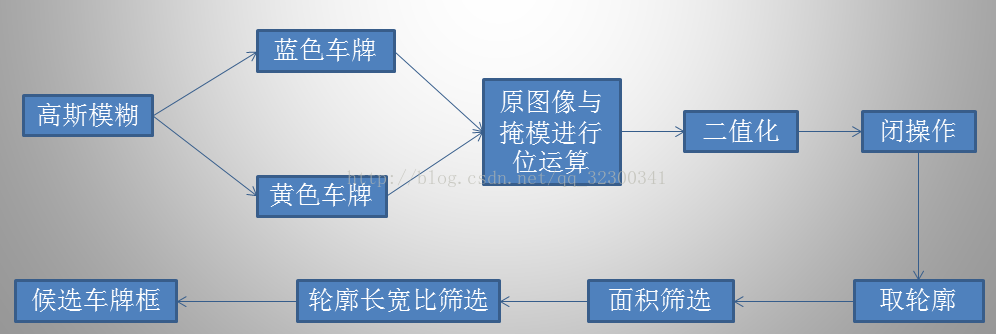
在之前已经介绍过原始的mtcnn车牌定位算法了，网址如下：

<http://blog.csdn.net/qq_32300341/article/details/77628648>

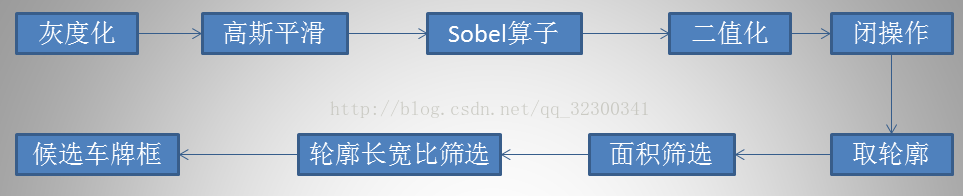
但是原始的定位算法效率不高，利用滑动窗口产生候选框的方法效率太低。在1080\*1920图片上产生的候选车牌框有50万个。

这就导致了第一层大致过滤候选框的工作耗时很大。单独采用颜色定位和候选车牌定位的方法会产生很多误检的车牌。所以综合传统车牌定位和神经网络车牌定位。采用颜色定位和形态学定位产生的候选车牌框来作为mtcnn第二层的输入。

颜色车牌定位的流程图如下：



形态学车牌定位流程图如下：



然后将这两个算法产生的候选车牌框作为mtcnn算法中RNet也就是第二层的输入来进一步排除误检车牌。

改进过后算法的时间效率为：

第一层耗时0.06s左右

第二层耗时0.005s左右

第三层耗时0.003s左右

达到了Fps15左右。识别率较高，但是存在漏检。

源代码地址如下：<http://download.csdn.net/download/qq_32300341/9954391>