# 模型结果可视化分析工具

## 1 绘制PR曲线

为了能比较不同模型的优劣，这里利用绘制PR曲线来进行对比。

1.1 检测结果列表保存

**实现代码：PR\_statistic.py**

**主要参数：**  
model\_def = **'deploy.prototxt'** *# 检测网络*model\_weights = **'COMPARE/NONE\_A75G20\_S\_D/NONE\_A75G20\_S\_D\_fix\_n\_iter\_200000.caffemodel'** *# 训练好的模型*ROOTDIR = **"\\\\192.168.1.186/PedestrianData/"** *# 待测试样本集所在根目录*imgList = **"../Data\_0825/val.txt"** *# 样本列表*

**deploy.prototxt中将confidence\_threshold设为0.001**

**输出：**与模型同名的测试结果列表.txt文件

**测试结果列表格式：**

<**img\_name**> <<**prior\_boxes\_num**> <**confidence**> <**prior\_box1\_coordinates**>**...**>

1.2 统计并保存不同conf阈值下的TP、FP、FN，绘制PR曲线

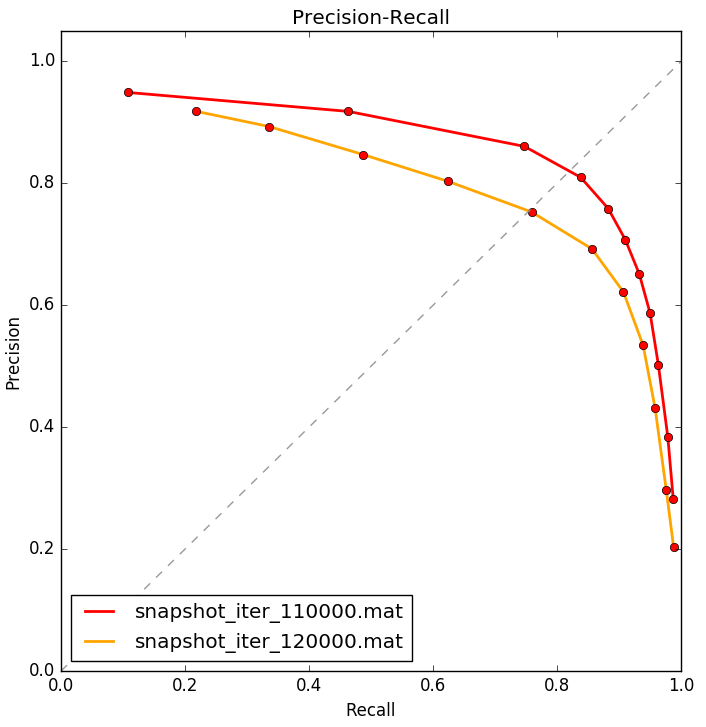
**实现代码：PRcurve.py**

save\_data()函数读取上一步的检测结果，统计不同conf阈值下的TP、FP、FN值并保存成.mat格式文件

conf\_thresholds = np.array([0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95], dtype=np.float64)

draw\_curve()函数读取mat文件绘制PR曲线。第一个参数是要绘制的曲线数量（1-3），后面跟着不同模型下的统计结果(mat文件名)。

由于可能要绘制多个模型结果的PR曲线，因此显示结果不会自动保存，可手动保存！



## 2 绘制训练过程中loss、mAP曲线

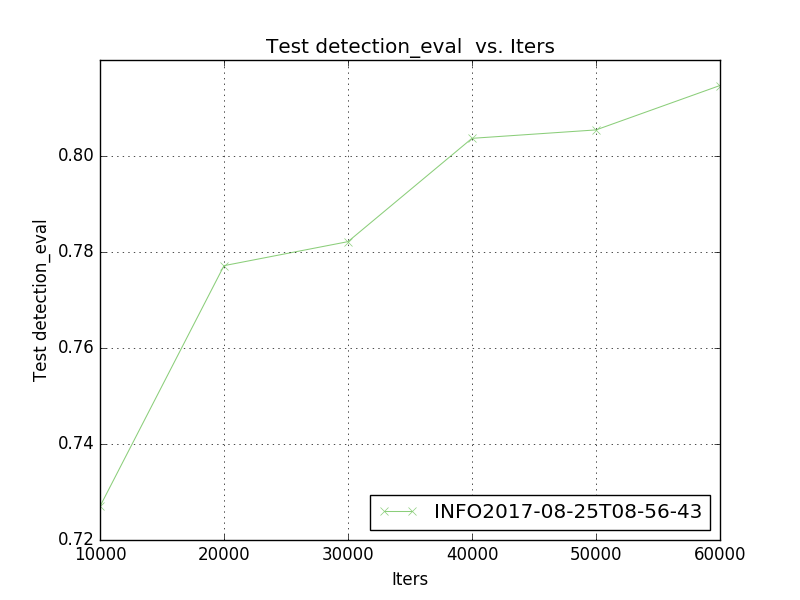
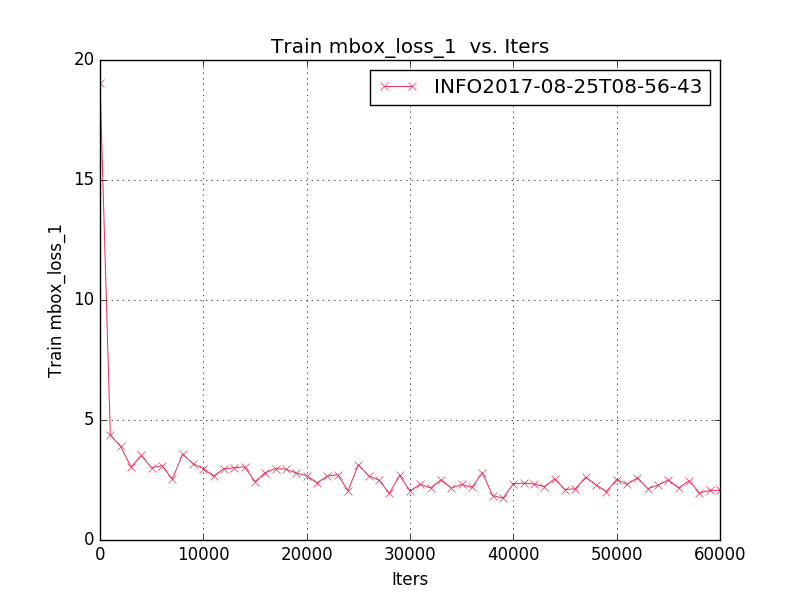
训练完成后，会在训练运行文件夹下的log文件夹下生成log文件，这个log文件存下的信息与训练时cmd窗口的输出信息一样。

**实现****代码：**plot\_training\_log.py、parse\_log.py、extract\_seconds.py

**主要参数：**

chart\_type = 0 *# 0：mAP, 4：learning rate, 6:loss*

path\_to\_logs = [**"INFO2017-08-26T14-00-20.log"**] *# log文件*



显示结果不会自动保存，可手动保存！

## 3 训练时输出test loss信息

1. 修改multibox\_loss\_layer.cpp中源文件在FindMatches()函数前加上

|  |
| --- |
| if (all\_match\_indices\_.size())  {  all\_match\_indices\_.clear();  all\_neg\_indices\_.clear();  } |

1. 修改solver.prototxt，添加

|  |
| --- |
| test\_compute\_loss: true |

1. 修改test.prototxt，在添加mbox\_priorbox\_1层之后添加

|  |
| --- |
| layer {  name: "mbox\_loss\_1"  type: "MultiBoxLoss"  bottom: "mbox\_loc\_1"  bottom: "mbox\_conf\_1"  bottom: "mbox\_priorbox\_1"  bottom: "label"  top: "mbox\_loss\_1"  include {  phase: TEST  }  propagate\_down: true  propagate\_down: true  propagate\_down: false  propagate\_down: false  loss\_param {  normalization: VALID  }  multibox\_loss\_param {  loc\_loss\_type: SMOOTH\_L1  conf\_loss\_type: SOFTMAX  loc\_weight: 1.0  num\_classes: 2  share\_location: true  match\_type: PER\_PREDICTION  overlap\_threshold: 0.5  use\_prior\_for\_matching: true  background\_label\_id: 0  #ignore\_label\_id: 13  use\_difficult\_gt: true  #do\_neg\_mining: true  neg\_pos\_ratio: 3.0  neg\_overlap: 0.5  code\_type: CENTER\_SIZE  mining\_type: MAX\_NEGATIVE  }  }  layer {  name: "silence"  type: "Silence"  bottom: "mbox\_loss\_1"  } |

## 4 测试分析单个模型在验证集上的性能

4.1 统计并保存不同目标大小区间的样本在不同conf阈值下的TP、FN，绘制RC曲线

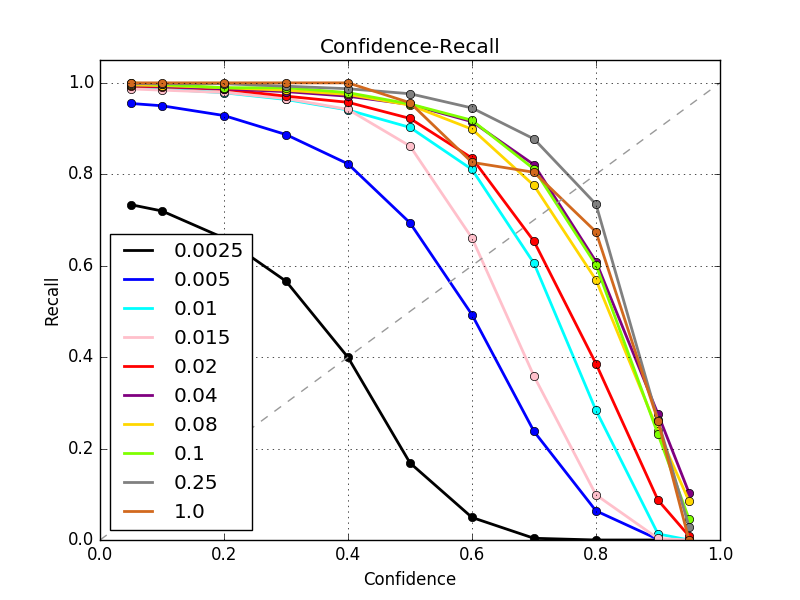
**实现代码：CRcurveObject.py**

save\_data()函数读取1.1节的检测结果，统计不同目标大小区间段样本在conf阈值下的TP、FN值并保存成.mat格式文件

area\_thresholds = np.array([0.0025, 0.005, 0.01, 0.015, 0.02, 0.04, 0.08, 0.1, 0.25, 1.0],dtype=np.float64) *# area*

conf\_thresholds = np.array([0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95], dtype=np.float64)

draw\_curve()函数读取mat文件绘制CR曲线。第一个参数是要绘制的曲线数量（1-3），后面跟着不同模型下的统计结果。图片默认保存在模型所在文件夹下CRcurveObject.png。



4.2 统计并保存各个共享层的分类情况，即不同conf阈值下的TP、FP、TN、FN

**实现代码：Conf\_FP\_layer.py**

layer\_priorbox\_num = np.array([9216, 2304, 576, 144, 36, 6],dtype=np.int32) *# layer层priorbox 数*

conf\_thresholds = np.array([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95], dtype=np.float64)

save\_data()函数利用训练得到的模型测试验证集，以不同的confirence为阈值，统计被划分为正例的default boxes的TP、FP有多少，被划分为负里的default boxes的TN、FN有多少。

draw\_curve()函数读取mat统计文件绘制不同共享层里default boxes的FPr-conf曲线。

## 5 分析感受野

5.1感受野计算公式

 感受野计算时有下面的几个情况需要说明：

　　（1）第一层卷积层的输出特征图像素的感受野的大小等于滤波器的大小

　　（2）深层卷积层的感受野大小和它之前所有层的滤波器大小和步长有关系

（3）计算感受野大小时，忽略了图像边缘的影响，即不考虑padding的大小

这里的每一个卷积层还有一个strides的概念，这个strides是之前所有层stride的乘积。

即strides（i） = stride(1) \* stride(2) \* ...\* stride(i-1)

关于感受野大小的计算采用top to down的方式， 即先计算最深层在前一层上的感受野，然后逐渐传递到第一层，使用的公式可以表示如下：　　　  
**RFdown = ((RFtop -1)\* stride) + fsize**

stride 表示卷积的步长； fsize表示卷积层滤波器的大小

5.2感受野计算代码

**实现代码：field.py**

layer output sizes given image = 256x256  
\*\*\*\*\*\*\*\*\*\*\*\*net structrue name **is** train\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Layer Name = conv1, Output size = 128, Stride = 2, RF size = 3  
Layer Name = pool1, Output size = 63, Stride = 4, RF size = 7  
Layer Name = fire2/squeeze1x1, Output size = 63, Stride = 4, RF size = 7  
Layer Name = fire2/expand3x3, Output size = 63, Stride = 4, RF size = 15  
Layer Name = fire3/squeeze1x1, Output size = 63, Stride = 4, RF size = 15  
Layer Name = fire3/expand3x3, Output size = 63, Stride = 4, RF size = 23  
Layer Name = pool3, Output size = 31, Stride = 8, RF size = 27  
Layer Name = fire4/squeeze1x1, Output size = 31, Stride = 8, RF size = 27  
Layer Name = fire4/expand3x3, Output size = 31, Stride = 8, RF size = 43  
Layer Name = fire5/squeeze1x1, Output size = 31, Stride = 8, RF size = 43  
Layer Name = fire5/expand3x3, Output size = 31, Stride = 8, RF size = 59  
Layer Name = pool5, Output size = 15, Stride = 16, RF size = 67  
Layer Name = fire6/squeeze1x1, Output size = 15, Stride = 16, RF size = 67  
Layer Name = fire6/expand3x3, Output size = 15, Stride = 16, RF size = 99  
Layer Name = fire7/squeeze1x1, Output size = 15, Stride = 16, RF size = 99  
Layer Name = fire7/expand3x3, Output size = 15, Stride = 16, RF size = 131  
Layer Name = fire8/squeeze1x1, Output size = 15, Stride = 16, RF size = 131  
Layer Name = fire8/expand3x3, Output size = 15, Stride = 16, RF size = 163  
Layer Name = fire9/squeeze1x1, Output size = 15, Stride = 16, RF size = 163  
Layer Name = fire9/expand3x3, Output size = 15, Stride = 16, RF size = 195  
Layer Name = conv6\_1, Output size = 15, Stride = 16, RF size = 195  
Layer Name = conv6\_2, Output size = 8, Stride = 32, RF size = 227  
Layer Name = conv7\_1, Output size = 8, Stride = 32, RF size = 227  
Layer Name = conv7\_2, Output size = 4, Stride = 64, RF size = 291  
Layer Name = conv8\_1, Output size = 4, Stride = 64, RF size = 291  
Layer Name = conv8\_2, Output size = 2, Stride = 128, RF size = 419  
Layer Name = pool6, Output size = 1, Stride = 256, RF size = 547