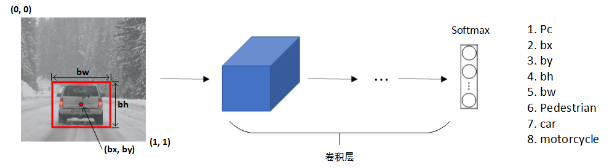
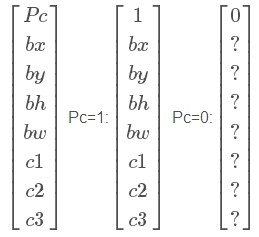
目标检测与目标定位

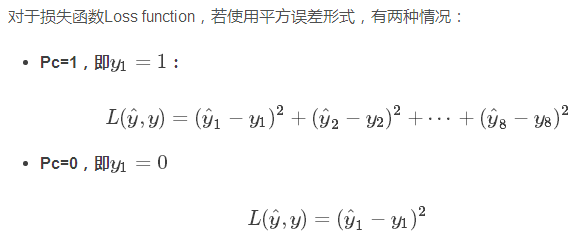
http://blog.csdn.net/red\_stone1/article/details/79028058

目标检测和定位的输出形式





目标检测和定位的代价函数，使用平方误差已经能够取得较好效果



简单滑动窗口算法

首先使用目标占比较大或完全没有目标的图片集训练CNN，达到较高识别率。然后在待检测图片中划定大小适合的窗口，以适当的步长，从左到右从上到下滑动，依次检测各个窗口中是否有目标。

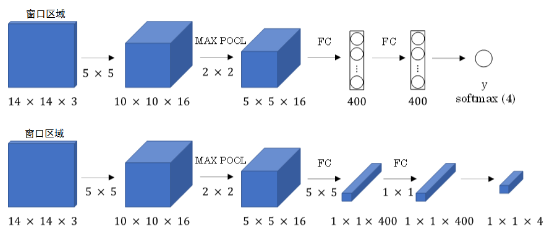
此方法原理简单，但是效果受窗口和步长大小影响明显，且一张图片需要计算若干次，效率太低。



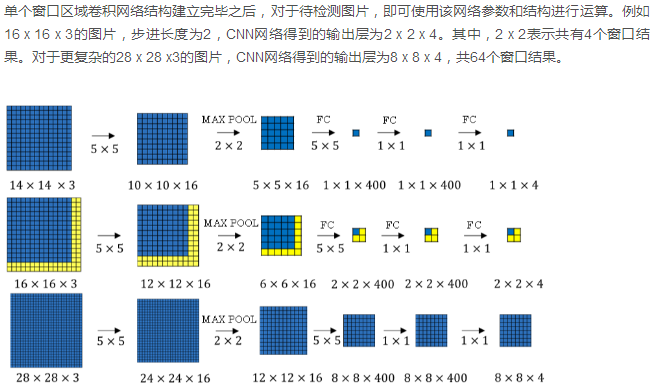
卷积滑动窗口算法 卷积通常是减小通道数，而池化通常会减小高度和宽度，这句加到前面去

需要将普通卷积网络中的全连接层，转换为1\*1卷积网络。

必须经此转换是因为前一层输出变量增加后，全连接层神经元个数必须相应增加才能完成运算。而1\*1卷积网络却可以直接应用于增加部分。



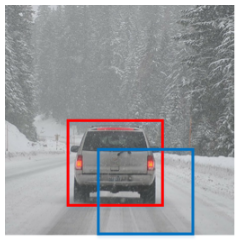
滑动窗口算法卷积化后，使用训练好的CNN对待检测图片整体进行一次运算，即包含了所有滑动过程的运算。每个窗口运算结果在整体运算结果中的位置，称为一个网格，对应于每个窗口在图片中的位置



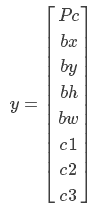
窗口滑动步长，由池化层表征。2\*2 pooling则步长为2，4\*4 pooling则步长为4。

YOLO算法，you only look once

滑动窗口算法有时会出现窗口不能完全覆盖目标的情况（红色），使用YOLO算法，可以生成更加准确的bounding box（蓝色）

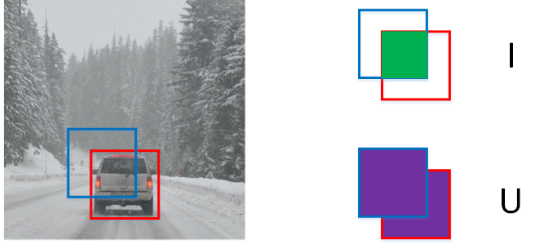


YOLO算法对卷积滑动窗口算法的输出做了变动。YOLO算法只有目标中心在某个网格中时，才会Pc=0。bx和by为目标中心位置，数值必须在[0,1]内，因为是相对于窗口的比例位置。bh和bw为目标高度和宽度，可以大于1。c1，c2和c3表明图片中的目标是车辆，行人还是自行车。



交并比，intersection over union

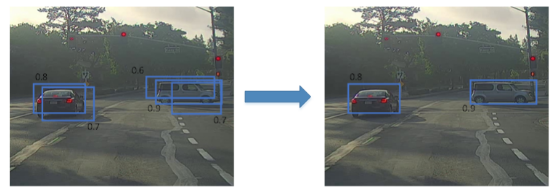
用于评估目标检测准确度。



非最大值抑制算法，non-max suppression

解决YOLO算法中相邻几个网格都检测出同一目标中心在各自内部时，判断哪个最准确的问题

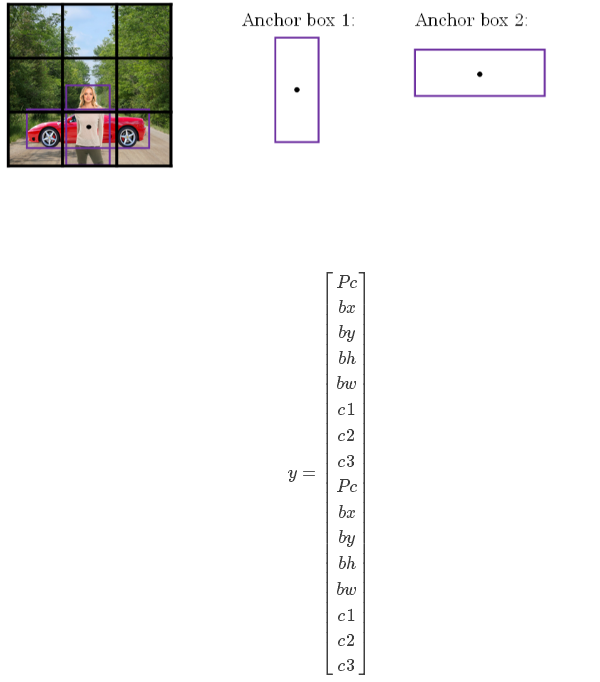
首先选出Pc值最接近1的网格，然后删除与其交并比大于设定阈值（通常0.5）的其他网格，即非最大值抑制。从而获得目标所在最准确的网格。之后在选出其他Pc值最大的网格重复非最大值抑制，以获得其他目标所在的最准确网格



Anchor boxes

解决多目标重叠问题

根据目标形状，设置额外的anchor box，并在CNN中增加相应输出。如下两个archor box分别用于标定行人和车辆。CNN输出变为原来的2倍，两个Pc分别表示网格中出现对应archor box类型目标的概率



Region proposals ？？？？