Faster-RCNN算法精读 - hunterlew的专栏 - CSDN博客

论文：《Faster R-CNN: Towards Real-Time ObjectDetection with Region Proposal Networks》

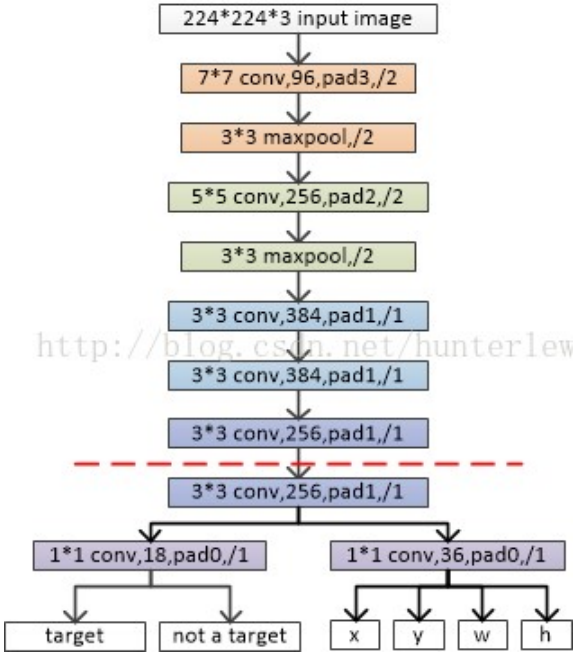
摘要：算法主要解决两个问题：

1、提出区域建议网络RPN，快速生成候选区域；

2、通过交替训练，使RPN和Fast-RCNN网络共享参数。

一、 RPN网络结构

RPN网络的作用是输入一张图像，输出一批矩形候选区域，类似于以往目标检测中的Selective Search一步。网络结构是基于卷积神经网络，但输出包含二类softmax和bbox回归的多任务模型。网络结果如下（以ZF网络为参考模型）：



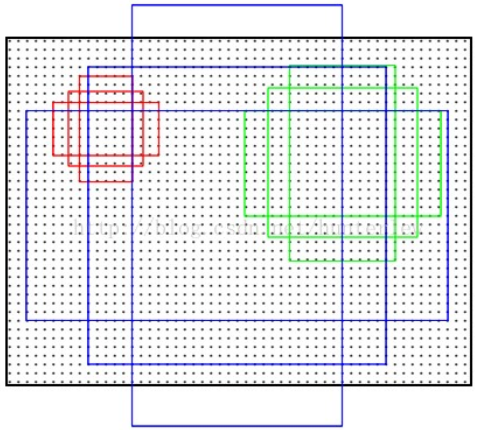
其中，虚线以上是ZF网络最后一层卷积层前的结构，虚线以下是RPN网络特有的结构。首先是3*3的卷积，然后通过1*1卷积输出分为两路，其中一路输出是目标和非目标的概率，另一路输出box相关的四个参数，包括box的中心坐标x和y，box宽w和长h。

（至于之前为什么要用3*3的卷积核，我觉得是和感受野大小相对应的。在原来的ZF模型中，3*3卷积核对应map比例是3/13，相当于在型如1000*600的图片中采用180左右的感受野。对于1000*600的图片中大部分目标而言，这个大小的感受野是比较合适的吧。）

从卷积运算本身而言，卷积相当于滑窗。假如输入图像是1000*600，则经过了几次stride后，map大小缩小了16倍，最后一层卷积层输出大约为60*40大小，那么相当于用3*3的窗口滑窗（注意有padding），对于左边一支路而言，输出18个通道，每个通道map大小仍为60*40，代表每个滑窗中心对应感受野内存在目标与否的概率。右支路同理。

二、 anchor机制

anchor是rpn网络的核心。刚刚说到，需要确定每个滑窗中心对应感受野内存在目标与否。由于目标大小和长宽比例不一，需要多个尺度的窗。Anchor即给出一个基准窗大小，按照倍数和长宽比例得到不同大小的窗。例如论文中基准窗大小为16，给了（8、16、32）三种倍数和（0.5、1、2）三种比例，这样能够得到一共9种尺度的anchor，如图（摘自http://blog.csdn.net/shenxiaolu1984/article/details/51152614）。



因此，在对60*40的map进行滑窗时，以中心像素为基点构造9种anchor映射到原来的1000*600图像中，映射比例为16倍。那么总共可以得到60*40*9大约2万个anchor。

三、 训练

RPN网络训练，那么就涉及ground truth和loss function的问题。对于左支路，ground truth为anchor是否为目标，用0/1表示。那么怎么判定一个anchor内是否有目标呢？论文中采用了这样的规则：1）假如某anchor与任一目标区域的IoU最大，则该anchor判定为有目标；2）假如某anchor与任一目标区域的IoU>0.7，则判定为有目标；3）假如某anchor与任一目标区域的IoU<0.3，则判定为背景。所谓IoU，就是预测box和真实box的覆盖率，其值等于两个box的交集除以两个box的并集。其它的anchor不参与训练。

于是，代价函数定义为：

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*).$$

代价函数分为两部分，对应着RPN两条支路，即目标与否的分类误差和bbox的回归误差，其中Leg(ti,ti*) = R(ti-ti*)采用在Fast-RCNN中提出的平滑L1函数，作者认为其比L2形式的误差更容易调节学习率。注意到回归误差中Leg与pi相乘，因此bbox回归只对包含目标的anchor计算误差。也就是说，如果anchor不包含目标，box输出位置无所谓。所以对于bbox的groundtruth，只考虑判定为有目标的anchor，并将其标注的坐标作为ground truth。此外，计算bbox误差时，不是比较四个角的坐标，而是tx，ty，tw，th，具体计算如下：

$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a, \quad t_w = \log(w/w_a), \quad t_h = \log(h/h_a),$$
$$t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a, \quad t_w^* = \log(w^*/w_a), \quad t_h^* = \log(h^*/h_a),$$

四、 联合训练

作者采用四步训练法：

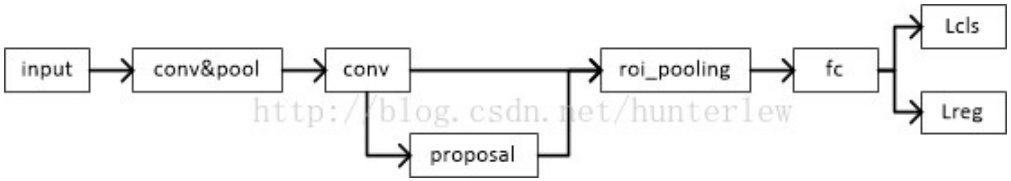
1） 单独训练RPN网络，网络参数由预训练模型载入；

2） 单独训练Fast-RCNN网络，将第一步RPN的输出候选区域作为检测网络的输入。具体而言，RPN输出一个候选框，通过候选框截取原图像，并将截取后的图像通过几次conv-pool，然后再通过roi-pooling和fc再输出两条支路，一条是目标分类softmax，另一条是bbox回归。截止到现在，两个网络并没有共享参数，只是分开训练了；

3） 再次训练RPN，此时固定网络公共部分的参数，只更新RPN独有部分的参数；

4） 那RPN的结果再次微调Fast-RCNN网络，固定网络公共部分的参数，只更新Fast-RCNN独有部分的参数。

至此，网络训练结束，网络集检测-识别于一体，测试阶段流程图如下：



有一些实现细节，比如RPN网络得到的大约2万个anchor不是都直接给Fast-RCNN，因为有很多重叠的框。文章通过非极大值抑制的方法，设定IoU为0.7的阈值，即仅保留覆盖率不超过0.7的局部最大分数的box（粗筛）。最后留下大约2000个anchor，然后再取前N个box（比如300个）给Fast-RCNN。Fast-RCNN将输出300个判定类别及其box，对类别分数采用阈值为0.3的非极大值抑制（精筛），并仅取分数大于某个分数的目标结果（比如，只取分数60分以上的结果）。