目标检测-RCNN系列

原创 2017年01月11日 16:26:12

标签：RCNN (http://so.csdn.net/so/search/s.do?q=RCNN&t=blog) / YOLO (http://so.csdn.net/so/search/s.do?q=YOLO&t=blog) / SSD (http://so.csdn.net/so/search/s.do?q=SSD&t=blog) / 目标检测 (http://so.csdn.net/so/search/s.do?q=目标检测&t=blog)

4654

- **RCNN**

RCNN（Regions with CNN features）是将CNN方法应用到目标检测问题上的一个里程碑，由年轻有为的RBG大神提出，借助CNN良好的特征提取和分类性能，通过RegionProposal方法实现目标检测问题的转化。

算法可以分为四步：

1）候选区域选择

Region Proposal是一类传统的区域提取方法，可以看作不同宽高的滑动窗口，通过窗口滑动获得潜在的目标图像，关于Proposal大家可以看下SelectiveSearch，一般Candidate选项为2k个即可，这里不再详述；

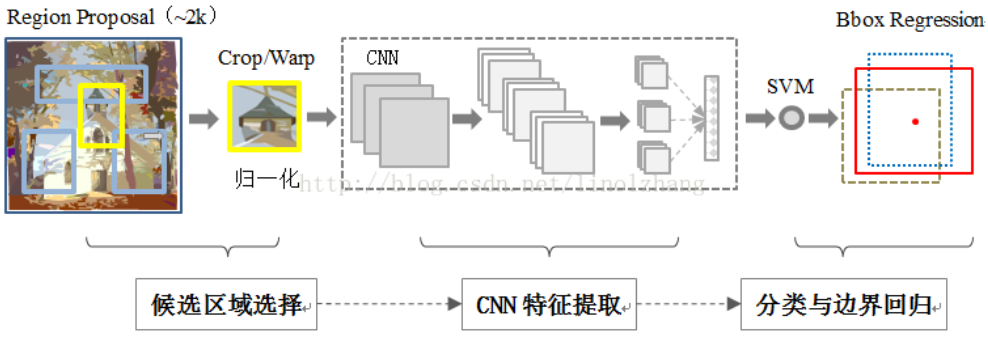
根据Proposal提取的目标图像进行归一化，作为CNN的标准输入。

2）CNN特征提取

标准CNN过程，根据输入进行卷积/池化等操作，得到固定维度的输出；

3）分类与边界回归

实际包含两个子步骤，一是对上一步的输出向量进行分类（需要根据特征训练分类器）；二是通过**边界回归**（bounding-box regression）得到精确的目标区域，由于实际目标会产生多个子区域，旨在对完成分类的前景目标进行精确的定位与合并，避免多个检出。



RCNN存在三个明显的问题：

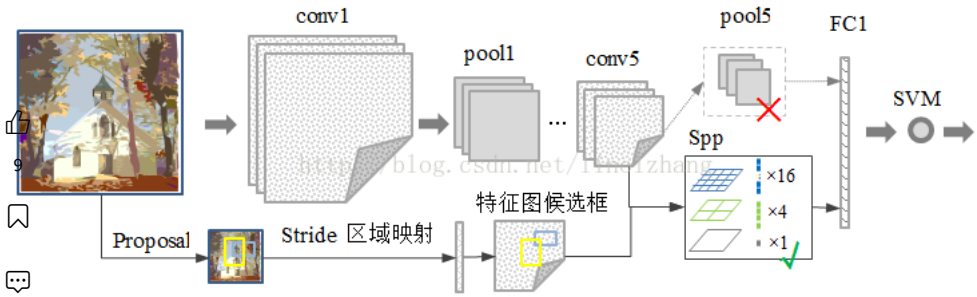
1）多个候选区域对应的图像需要预先提取，占用较大的磁盘空间；

2）针对传统CNN需要固定尺寸的输入图像，crop/warp（归一化）产生物体截断或拉伸，会导致输入CNN的信息丢失；

3）每一个ProposalRegion都需要进入CNN网络计算，上千个Region存在大量的范围重叠，重复的特征提取带来巨大的计算浪费。

- **SPP-Net**

智者善于提出疑问，既然CNN的特征提取过程如此耗时（大量的卷积计算），为什么要对每一个候选区域独立计算，而不是提取整体特征，仅在分类之前做一次Region截取呢？智者提出疑问后会立即付诸实践，于是SPP-Net诞生了。



SPP-Net在RCNN的基础上做了实质性的改进：

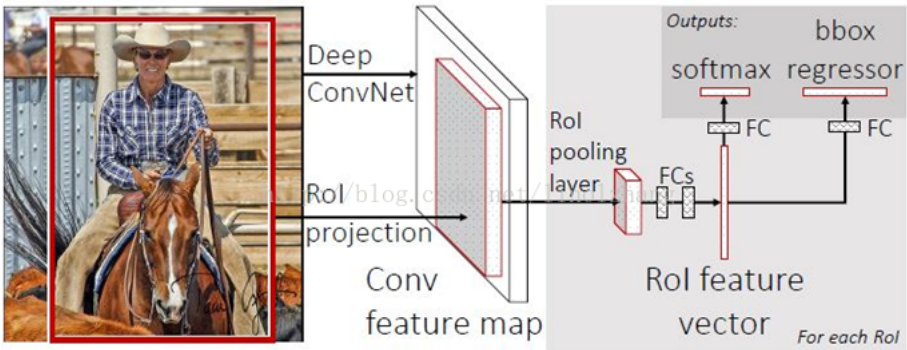
1）取消了crop/warp图像归一化过程，解决图像变形导致的信息丢失以及存储问题；

2）采用**空间金字塔池化**（SpatialPyramid Pooling ）替换了 全连接层之前的最后一个池化层（上图top），翠平说这是一个新词，我们先认识一下它。

为了适应不同分辨率的特征图，定义一种**可伸缩的池化层**，不管输入分辨率是多大，都可以划分成m*n个部分。这是SPP-net的第一个显著特征，它的输入是**conv5特征图** 以及**特征图候选框**（原图候选框
通过stride映射得到），输出是固定尺寸（m*n）特征；

还有金字塔呢？通过多尺度增加所提取特征的鲁棒性，这并不关键，在后面的Fast-RCNN改进中该特征已经被舍弃；

最关键的是SPP的位置，它放在所有的卷积层之后，有效解决了卷积层的重复计算问题（测试速度提高了24~102倍），这是论文的核心贡献。



尽管SPP-Net贡献很大，仍然存在很多问题：

1）和RCNN一样，训练过程仍然是隔离的，**提取候选框 | 计算CNN特征| SVM分类 | Bounding Box回归**独立训练，大量的中间结果需要转存，无法整体训练参数；

2）SPP-Net在无法同时Tuning在SPP-Layer两边的卷积层和全连接层，很大程度上限制了深度CNN的效果；

3）在整个过程中，Proposal Region仍然很耗时。

- **Fast-RCNN**

问题很多，解决思路同样也非常巧妙，ok，再次感谢 RBG 大神的贡献，直接引用论文原图（描述十分详尽）。

Fast-RCNN主要贡献在于对RCNN进行加速，快是我们一直追求的目标（来个山寨版的奥运口号- 更快、更准、更鲁棒），问题在以下方面得到改进：

1）卖点1 - **借鉴SPP思路，提出简化版的ROI池化层（注意，没用金字塔），同时加入了候选框映射功能，使得网络能够反向传播，解决了SPP的整体网络训练问题；**

2）卖点2 - **多任务Loss层**

A）SoftmaxLoss代替了SVM，证明了softmax比SVM更好的效果；

B）SmoothL1Loss取代Bouding box回归。

将分类和边框回归进行合并（又一个开创性的思路），通过多任务Loss层进一步整合深度网络，统一了训练过程，从而提高了算法准确度。

3）全连接层通过SVD加速

这个大家可以自己看，有一定的提升但不是革命性的。

4）结合上面的改进，模型训练时可对所有层进行更新，除了速度提升外（训练速度是SPP的3倍，测试速度10倍），得到了更好的检测效果（VOC07数据集mAP为70，注：mAP，mean Average Precision）。

接下来分别展开这里面的两大卖点：

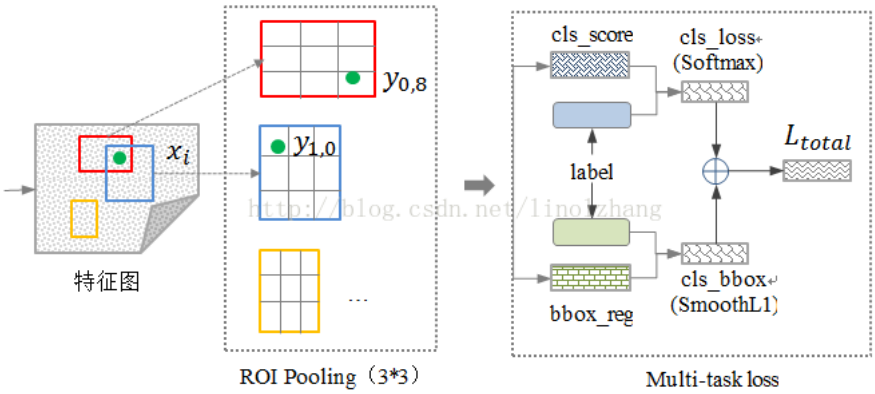
前面已经了解过可伸缩的池化层，那么在训练中参数如何通过ROI Pooling层传导的？根据链式求导法则，对于$y_j = max(x_i)$ 传统的max pooling的映射公式：

$$\frac{\partial L}{\partial x_i} = \begin{cases} 0 & \delta(i,j) = 0 \\ \frac{\partial L}{\partial y_j} & \delta(i,j) = 1 \end{cases}$$

其中 为判别函数，为1时表示选中为最大值，0表示被丢弃，误差不需要回传，即对应 权值不需要更新。如下图所示，对于输入 x_i 的扩展公式表示为：

$$\frac{\partial L}{\partial x_i} = \sum_{r,j} \delta(i,r,j) \frac{\partial L}{\partial y_{r,j}}$$

(i,r,j) 表示 x_i 在第 r 个框的第 j 个节点是否被选中为最大值（对应上图 $y_{0,8}$ 和 $y_{1,0}$），x_i 参数在前向传导时受后面梯度误差之和的影响。



多任务Loss层（全连接层）是第二个核心思路，如上图所示，其中cls_score用于判断分类，bbox_reg计算边框回归，label为训练样本标记。

其中L_{cls}为分类误差：

$$L_{cls} = -log\ p_l$$

p_x 为对应Softmax分类概率，p_l 即为label所对应概率（正确分类的概率），p_l = 1时，计算结果Loss为0， 越小，Loss值越大（0.01对应Loss为2）。

L_{reg}为边框回归误差：

$$L_{reg} = \sum_{i=0}^{3} g(t_i^u - v_i)$$

即在正确分类的情况下，回归框与Label框之间的误差（Smooth L1）， 对应描述边框的4个参数（上下左右or平移缩放），g对应单个参数的差异，|x|>1 时，变换为线性以降低离群噪声：

$$g(x) = \begin{cases} 0.5x^2 & |x| < 1 \\ |x| - 0.5 & otherwise \end{cases}$$

L_{total}为加权目标函数（背景不考虑回归Loss）：

$$L_{total} = \begin{cases} L_{cls} + \lambda * L_{reg} & 前景 \\ L_{cls} & 背景 \end{cases}$$

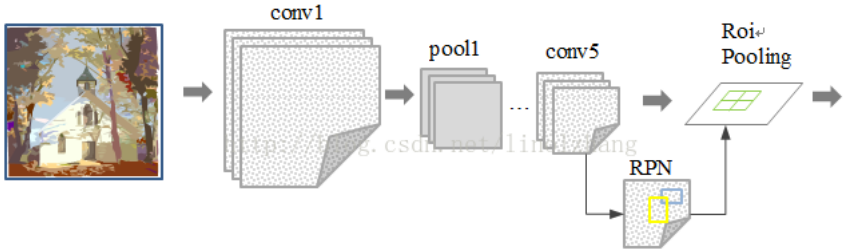
细心的小伙伴可能发现了，我们提到的SPP的第三个问题还没有解决，依然是耗时的候选框提取过程（忽略这个过程，Fast-RCNN几乎达到了实时），那么有没有简化的方法呢？

必须有，搞学术一定要有这种勇气。

- **Faster-RCNN**

对于提取候选框最常用的SelectiveSearch方法，提取一副图像大概需要2s的时间，改进的EdgeBoxes算法将效率提高到了0.2s，但是这还不够。

候选框提取不一定要在原图上做，特征图上同样可以，低分辨率特征图意味着更少的计算量，基于这个假设，MSRA的任少卿等人提出RPN（RegionProposal Network），完美解决了这个问题，我们先来看一下网络拓扑。

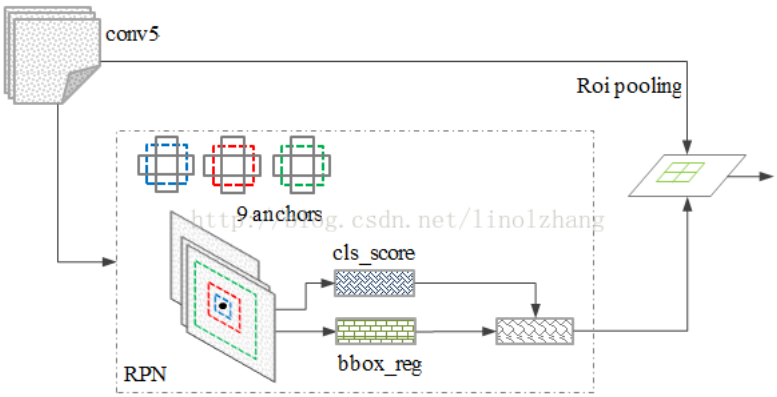


通过添加额外的RPN分支网络，将候选框提取合并到深度网络中，这正是Faster-RCNN里程碑式的贡献。

RPN网络的特点在于通过滑动窗口的方式实现候选框的提取，每个滑动窗口位置生成9个候选窗口（不同尺度、不同宽高），提取对应9个候选窗口（anchor）的特征，用于目标分类和边框回归，与FastRCNN类似。

目标分类只需要区分候选框内特征为前景或者背景。

边框回归确定更精确的目标位置，基本网络结构如下图所示：



训练过程中，涉及到的候选框选取，选取依据：

1）丢弃跨越边界的anchor；

2）与样本重叠区域大于0.7的anchor标记为前景，重叠区域小于0.3的标定为背景；

对于每一个位置，通过两个全连接层（目标分类+边框回归）对每个候选框（anchor）进行判断，并且结合概率值进行舍弃（仅保留约300个anchor），**没有显式地提取任何候选窗口**，完全使用网络自身完成判断和修正。

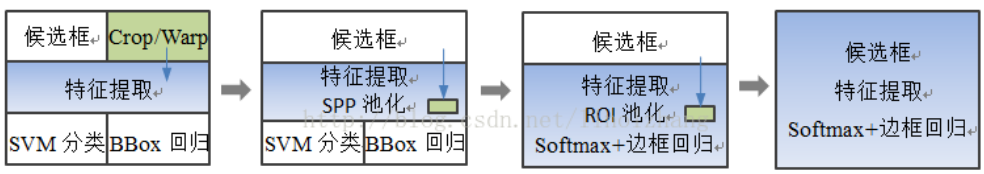
从模型训练的角度来看，通过使用共享特征交替训练的方式，达到接近实时的性能，交替训练方式描述为：

1）根据现有网络初始化权值w，训练RPN；

2）用RPN提取训练集上的候选区域，用候选区域训练FastRCNN，更新权值w；

3）重复1、2，直到收敛。

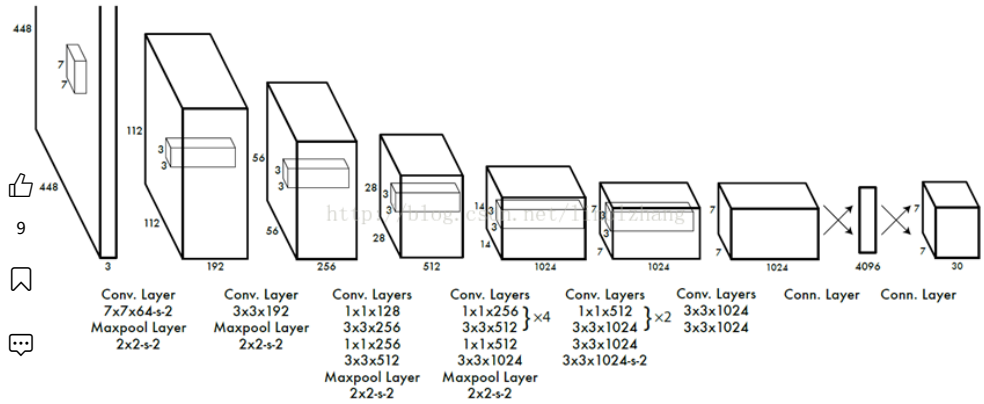
因为Faster-RCNN，这种基于CNN的real-time 的目标检测方法看到了希望，在这个方向上有了进一步的研究思路。至此，我们来看一下RCNN网络的演进，如下图所示：



Faster实现了端到端的检测，并且几乎达到了效果上的最优，速度方向的改进仍有余地，于是YOLO诞生了。

- **YOLO**

YOLO来自于“YouOnly Look Once”，你只需要看一次，不需要类似RPN的候选框提取，直接进行整图回归就可以了，简单吧？



算法描述为：

1）将图像划分为固定的网格（比如7*7），如果某个样本Object中心落在对应网格，该网格负责这个Object位置的回归；

2）每个网格预测包含Object位置与置信度信息，这些信息编码为一个向量；

3）网络输出层即为每个Grid的对应结果，由此实现端到端的训练。

YOLO算法的问题有以下几点：

1）7*7的网格回归特征丢失比较严重，缺乏多尺度回归依据；

2）Loss计算方式无法有效平衡（不管是加权或者均差），Loss收敛变差，导致模型不稳定。

Object（目标分类+回归）<=等价于=>背景（目标分类）

导致Loss对目标分类+回归的影响，与背景影响一致，部分残差无法有效回传；

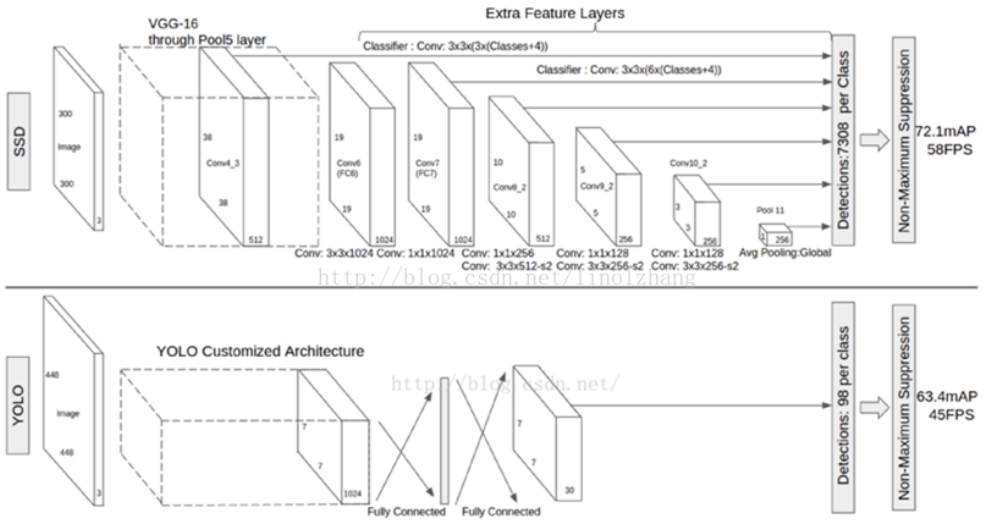
整体上YOLO方法定位不够精确，贡献在于提出给目标检测一个新的思路，让我们看到了目标检测在实际应用中真正的可能性。

这里备注一下，直接回归可以认为最后一层即是对应7*7个网格的特征结果，每一个网格的对应向量代表了要回归的参数（比如pred、cls、xmin、ymin、xmax、ymax），参数的含义在于Loss函数的设计。

- **SSD**

由于YOLO本身采用的SingleShot基于最后一个卷积层实现，对目标定位有一定偏差，也容易造成小目标的漏检。

借鉴Faster-RCNN的Anchor机制，SSD（Single Shot MultiBox Detector）(http://blog.csdn.net/u010167269/article/details/52563573)在一定程度上解决了这个问题，我们先来看下SSD的结构对比图。



基于多尺度特征的Proposal，SSD达到了效率与效果的平衡，从运算速度上来看，能达到接近实时的表现，从效果上看，要比YOLO更好。

对于目标检测网络的探索仍在一个快速的过程中，有些基于Faster-RCNN的变种准确度已经刷到了87%以上，而在速度的改进上，YOLO2也似乎会给我们带来一定的惊喜，“未来已来”，我们拭目以待！