链滴

RCNN 系列模型记录

作者：lenks

原文链接：https://ld246.com/article/1646722880868

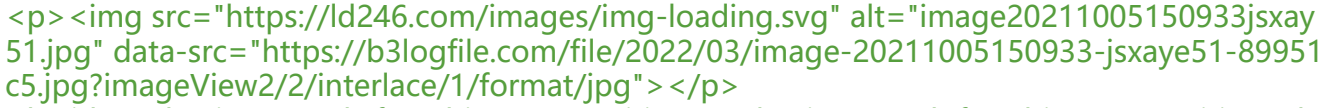
来源网站：链滴

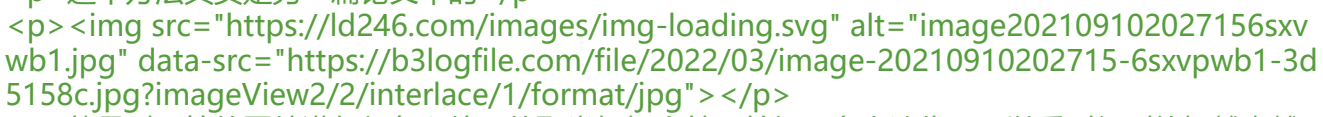
<h2 id="前言">前言</h2>
<h3 id="回顾与展望">回顾与展望</h3>
<hr>
<blockquote>
相关资料：
作者的论文的海报 [rcnn-poster.pdf]
作者的 CVPR 上的 ppt[rcnn-cvpr14-sliders]
目标检测的历史 [HistoryObjectRecognition.png]
同济子豪兄的批注[R+CNN 论文-同济子豪兄批注.pdf]
霹雳吧啦的 ppt[b 站的讲解视频不错]
</blockquote>
下面是近些年比较典型的模型
RCNN(CVPR2014) -> SPP-Net(ECCV2014)->Fast R-CNN(ICCV2015) ->Faster R-NN(NIPS2015) ->YOLO(CVPR2016) ->SSD(ECCV2016)->R-FCN(NIPS2016) <code>以总结每个网络的典型的突出贡献，以及操作</code>
这张图是较为全面的发展过程，其中红色标注的是较为经典的模型。
可以看到，从 14 年开始，每年的成果也越来越多。所以，现在的研究主要是一方面在细节性上突破，比如下面引用；另一方面是学术研究到实际应用的落地。

更好的建议区域算法
更好的结果处理方式（更好的 nms，或把 nms 进化到检测主网络中）
更精准的定位精度
弱监督目标检测（例如只有 image-level 标注的目标检测）
更好的多尺度处理（现有检测网络大都需要输入图像 resize，其实不合理）
少样本目标检测

<h3 id="作者介绍">作者介绍</h3>
Ross.B.Girshick (RBG) 看名字就像天生研究图像的。RGB。。
作者个人主页 主页链接
通过作者的谷歌学术主页看到作者发表的文章和学术报告等内容，按照引用量排名，发现作者在个门派都混的风生水起。
<h2 id="RCNN">RCNN</h2>
<h3 id="RCNN流程-">RCNN 流程：</h3>
<blockquote>
图片来自于作者的 ppt。
</blockquote>

输入图像
通过 selective search 方法提取差不多 2k 个区域 然后 resize 到 227 * 227 的大小（因为含有连接层，所以必须 resize 到相同的大小）
逐一的喂到 CNN 模型中，获得一个 4096 维的特征。
再用提取到的特征分别进行分类和回归 （并行）。比如 Pascal voc 有 20 个类别，就用 20 个线支持向量机对这个 4096 维的向量进行分类。也直接用来 bbox 的回归。

<blockquote>
将 2000x4096 维特征与 20 个 SVM 组成的权值矩阵 4096x20 相乘，获得 2000x20 维矩阵表每个建议框是某个目标类别的得分。分别对上述 2000x20 维矩阵中的每一列(即每一类)进非极大值抑制 ,剔除重叠建议框，得到该列即该类得分最高的一些建议框。
</blockquote>
<hr>
整个网络的步骤比较多，依赖上下游多个模块协作完成的（提取候选框，resize，卷积处理，bbo reg 和 SVMs ）每个步骤都需要单的优化，如果一个环节没有处理好，整体的性能都会受到影响。且，后面的每一类都需要训练一个相应的 SVMs，然后逐一比对。如下图


<h3 id="Selective-Search-for-object-recognition">Selective Search for object recognition</h3>
这个方法其实是另一篇论文中的


就是对原始的图片进行颜色和纹理的聚类加权合并，并经历多次迭代。可以看到得到的框越来越，越来越准确，所以 recall 比较高。
通过类似聚类的方法，在图中找到一些初始的分割区域，比如找到一些颜色，纹理，大小相似的域，加权合并，并经历多次迭代
<blockquote>
算法流程 ：
</blockquote>

生成区域集：R
计算区域集 R 中每个相邻区域的相似度 S
找出最相似的两个区域，将其合并成新区域，并添加到 R 中
从 S 中移除所有与 step2 中相关的区域
计算新集与所有子集的相似度
跳转到 step2 指导 S 集为空为止

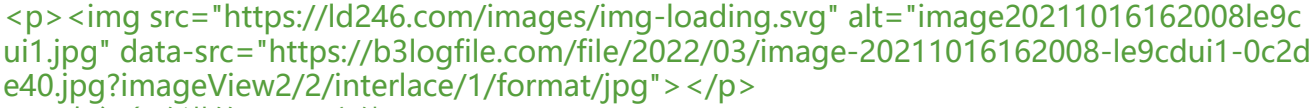
<hr>
<blockquote>
相似度计算：
</blockquote>

颜色相似度（colour similarity）：每个区域分别计算 25bins 的直方图，这样每个区域就有 75 的特征向量，再通过公式计算
纹理相似度（texture similarity）：使用 σ=1 的高斯函数计算每个通道(共 3 通道)每个方向(共 8 个方向)上的高斯微分, 这样共可以得到 24 个微分图, 每个图上计算 10bins 的直方图(也归一化了), 这每个区域可以得到一个维度为 240 的纹理直方图特征向量
尺度相似度（size similarity）：
交叠相似度（shape compatibility measure）

<blockquote>
最终相似度：上面所有相似度的加权求和
</blockquote>
<h3 id="非极大值抑制-NMS--">非极大值抑制（NMS）：</h3>
<blockquote>
非极大值抑制（Non-Maximum Suppression，NMS），顾名思义就是抑制不是极大值的元素可以理解为局部最大搜索。用于目标检测中提取分数最高的窗口，因为滑动窗口会导致很多窗口与其窗口存在包含或大部分交叉的情况。
</blockquote>

</blockquote>

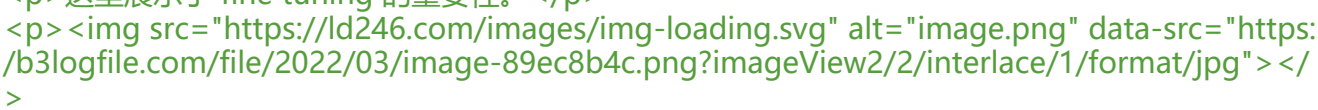
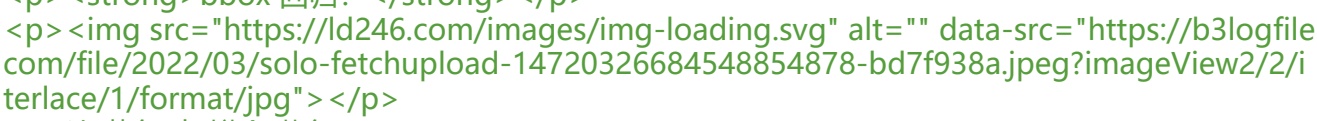
寻找该类得分最高的目标
计算其他目标与该目标的 IOU 值
删除所有 IOU 值大于给定阈值的目标
重复 3->1 到最后，该列剩下几个目标，说明图像中包含几个该类目标（比如图中的人）

但是这种 NMS 有一个问题，当两个 ground truth 的目标的确重叠度很高时,NMS 会将具有较置信度的框去掉(置信度改成 0),参见下图所示

有很多改进的 NMS 变体：
soft NMS、softer NMS、adaptive NMS、fast NMS 、Cluster NMS
回归器精细修正候选框位置
对 NMS 处理后的剩余的建议框进一步的筛选。分别用 20 个回归器对上述 20 个类别中剩余的议框进行回归操作，最终得到每个类别的修正后的得分最高的 bounding box。
resize 方案
作者提出了多种 resize 的方法：

等比例缩放

保留长宽比，连带临近像素
保留长宽比，不带临近像素

非等比例缩放 （不保留长宽比例，不带临近像素）

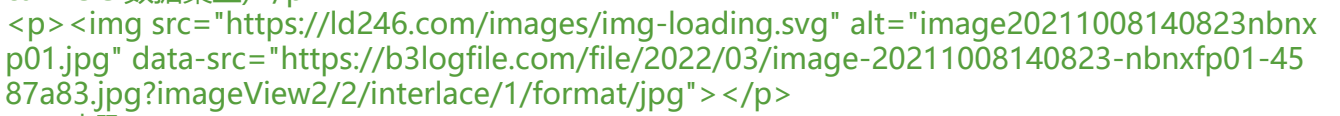
<blockquote>
可视化能够使得某个 feature map 的某个值 最大化 的 原始候选框
可以看到某个值对应的感受野的位置和所获得的特征。
论文的附录里面有好多这样的图，可以详细的看一下。
</blockquote>
<h3 id="对比-消融-实验">对比（消融）实验</h3>
这里是论文中的一种重要方法，要解决目标检测问题，但是目标检测的数据集比较少，所以就先 imagenet 这个大规模的分类任务上预训练一个模型，再用这个预训练的模型迁移泛化微调到目标检域上。
这里展示了 fine tuning 的重要性。

上图可以看到，没有加 fine tuning 的时候，直接用在 imagenet 训练好的模型的各个层的权重效果不是很好。而且，多加了全连接层后 mAP 还下降了。
加了 fine tuning 之后，可以看到网络越深效果越好。而且加了 bbox 回归之后，效果更好。
而传统的方法就很一般了。
为什么用 SVMs 而不用 softmax ？
同济子豪兄 RNN 视频 https://ww.bilibili.com/video/BV1d64y1W74E 的 29 分钟。
bbox 回归：

让 黄色 去 拟合 蓝色。

<h3 id="RCNN网络时间耗时-">RCNN 网络时间耗时：</h3>

可以看出,最耗时的是每个区域通过 CNN 提取特征的步骤，但是其他的步骤所花费的时间也不短这张图里面后面两个步骤虽然耗时较少，但是与类别 N 有关，时间复杂度是 O(N)级别的。（这里作想突出的可能是 虽然耗时，但是可以扩展。再次增加多各类别后，总体时间不会有很大的变化）
整体可以看出全部过程耗时长，程序繁琐臃肿。训练速度慢，而且不适合实时监测。为非端到端训练网络。
所需要的空间比较大，对于 SVM 和 bbox 回归训练，需要从每个图像中每个候选框提取特征，并写入磁盘。对于非常深的网络，如 VGG16，从 VOC 07 训练集上的 5k 图上提取的特征需要数百 GB 的存储空间。
<h3 id="R-CNN改进">R-CNN 改进</h3>

提取候选框：Edge Boxes、RPN 网络
共享卷积运算：SPPNet、Fast R-CNN
兼容任意尺寸图像：SPPNet、ROI Pooling
网络结构：端到端网络
融合各层特征：FPN
提取候选框可以通过 EdgeBoxes,直到后面的 faster rcnn 提出的 RPN 网络。
之前的 RCNN 是把所有的候选框都喂进 CNN 网络，重复的区域会进行重复的运算。这里通过一张图喂入 CNN 网络，所有区域共享得到的特征图，可以极大地缩减耗费时间
因为卷积层后面连着全连接层，而全连接层的特征数是固定的。原方案是将所有提取的区域 resize 到 227 * 227 一个正方形的形状，如果有的区域比如人类，等会产生严重的畸变，影响结果。所以在 PP-Net 中用到了空间金字塔池化，在 Fast R-CNN 里面用到了 ROI Pooling，这样就可以兼容任意寸的输入了。
在 Fast R-CNN 里面加了长宽比，可以预设一些固定的比例的框，用于检测不同的物体。比如矮的可以检测汽车等，瘦高的检行人，路灯等。后面在 YOLOv2 单阶段方法中也用了 anchor。

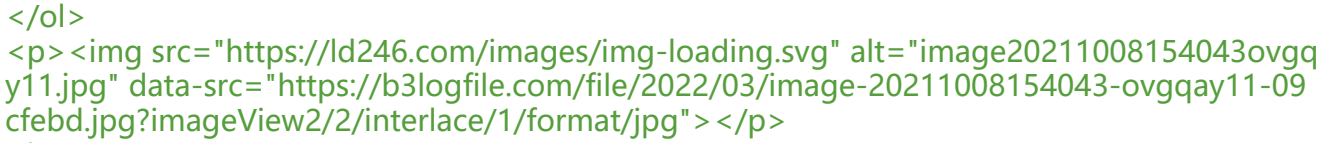
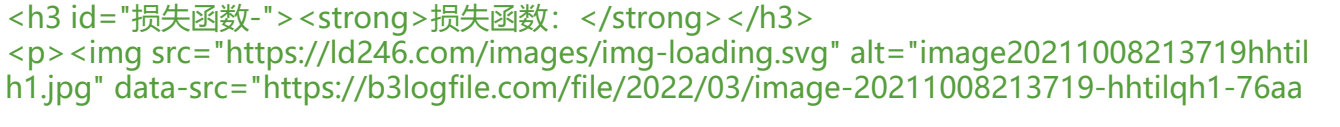
<blockquote>
共享卷积运算
先选出候选框，不像 RCNN 那样把每个候选区域给深度网络提取特征，而是整个图提取一次特，再把候选框映射到特征图上，从而获得该候选框的所属的特征图。
空间金字塔池化 (Spatial Pyramid Pooling,SPP) RCNN 学习笔记(3):Spatial Pyramid Pooling 站巨人的肩膀上-CSDN 博客 【*#优秀#* 】
在那个时候，还没有 FCN 的思想，那如何去做才能使得网络不受输入尺寸的限制呢？Kaiming He 大神就想出，用不同尺度的 pooling 来 pooling 出固定尺度大小的 feature map，这样就可不受全链接层约束任意更改输入尺度了。
SPPNet 将任意大小的图像池化生成固定长度的图像表示，一般代替最后一个卷积层的池化层，过对特征图进行相应尺度的 pooling，是的能 pooling 出 4 x 4，2 x 2, 1 x 1 的特征图，再将这些特图 concat 成列向量与下一层的全连接层相连。
特征图金字塔网络 FPN （Feature Pyramid Networks） (22 条消息) FPN 网络详_kk123k 的博客-CSDN 博客_fpn 网络
2017 年提出的一种网络，FPN(论文) 主要解决的物体检测中的多尺度问题，通过简单的网络连接改变，在基本不增加原有模型计算量的情况下，大幅提升了小物体检测的性能。
低层的特征语义信息较少，但是目标位置准确，而且可以检测小尺度目标；高层的特征语信息比较丰富，但是目标位置比较粗略，适合检测大尺度目标。
另外，也有别的算法从业多尺度特征融合的方法，但是一般是采用融合后的特征做预测，而 FPN

不一样的地方在于预测是在不同的特征层内独立进行的。
</blockquote>
RCNN 是将 CNN 方法引入目标检测领域， 大大提高了目标检测效果，可以说改变了目标检测域的主要研究思路。R-CNN 的提出，使目标检测性能相对于 2012 年的先前最佳结果，平均精度（mP）提高了 30%以上，达到 53.3%的 mAP。
<hr>
<h2 id="Fast-R-CNN">Fast R-CNN</h2>
<h3 id="Fast-R-CNN流程">Fast R-CNN 流程</h3>
Fast R-CNN 是作者 Ross Girshick 继 R-CNN 后的又一力作。同样使用 VGG16 作为网络的 backbone，与 R-CNN 相比训练时间快 9 倍，测试推理时间快 213 倍，准确率从 62% 提升至 66%(在 Pacal VOC 数据集上)

步骤：

继续使用 selective search 方法生成 2k 个候选区域
总图像输入网络，得到相应的特征图，将上面的候选区域映射到特征图 上这样可以经过一次 CNN 获得特征图，重复复用就可以了。获得相应的特征矩阵
将每个特征矩阵通过 ROI pooling 层缩放到 7 * 7 特征图，接着将特征图平通过一系列的全连接层（softmax 分类 和 bbox regressor 回归）

<blockquote>
这里的提取的 2000 个候选框，并没有都参与运算。训练过程中只使用一小部分就可以了，而且于采样的数据分为正样本和负样本。正样本就是候选框中确实存在所需检测目标的样本，负样本可以单地理解为背景，就是里面没有所检测的目标。
为什么需要区分正负样本 ： 正负样本的分配，采样策略以及正负样本的数量和比的设置等，对算法的精度有着显著的影响。比如在猫狗分类中，极端一点样本只有猫，没有狗。那网的训练结果肯定不大理想。
论文 ： 每张图选 64 个样本进行训练
<hr>
这个是参考的 SPPNet，不用像 R-CNN 一样再进行 2000 多次正向传播，就是这个操作可以大减少网络训练和测试时间。
<hr>
在 R-CNN 中，是专门训练了 SVM 来做分类和用边界框回归器调整 box 的位置，而在 fast R-CN 中都已经结合在一个网络当中了
</blockquote>
<h3 id="ROI-Pooling">ROI Pooling</h3>
<blockquote>
可以先看下下面的图，这里忽略了 channel。
具体操作步骤：
</blockquote>

根据输入 image，将 ROI 映射到 feature map 对应的位置
将映射后的区域按照输出的维度划分，比如这个需要 49 维度，所以就划分出 7 * 7 个格子
对每个格子进行 max pooling 操作。


<hr>
<h3 id="损失函数-">损失函数：</h3>


d88.jpg?imageView2/2/interlace/1/format/jpg">
由于需要预测分类概率和边界框回归参数，所以需要计算两个损失。

这里的 p 是分类器预测的 softma 概率分布 p=(p0,p1...pk) k 表示有 k 类
u 对应目标真实类别标签
t^u^ 对应边界框回归器预测的对应类别 u 的回归参数（tx^u^ ,ty^u^ ,tw^u^ ,th^u^ ）
v 对应真实目标的边界框回归参数。这个真是的回归参数怎么来的 比如 dx 就是用

这个分类损失，其实就是个交叉熵损失函数。
这个边界框回归损失，这里的[u>=1]是艾弗森括号，就是 n>=1 为 1，其他为 0.
主要是说 差值小于 1 的话 就是 0.5*差值的平方，其他就是差值-0.5 然后两两相减的值的 smoot L1 结果求和。
<blockquote>
回归损失函数 1：1 loss, L2 loss 以及 Smooth L1 Loss 的对比 - Brook_icv - 博客园
Smooth L1 (这里为什么不用均方误差(MSE) L2 及 均绝对误差(MAE L1) 为什么用这个
</blockquote>

相比于 L1 损失函数，可以收敛得更快。
相比于 L2 损失函数，对离群点、异常值 不那么敏感，梯度变化相对较小，训练时不容易跑飞。

主要解决 网络模块分散，解决 2K 张候选图都过 CNN 拉低速度的问题,这时，网络的大头时间主是 cpu 上 ss 算法提取 2000 个左右的候选框。所以下一个
<hr>
<h2 id="Faster-R-CNN">Faster R-CNN</h2>
Faster R-CNN 是作者 Ross Girshick 继 Fast R-CNN 后的又一力作。同样使用 VGG16 作为网的 backbone，推理速度在 GPU 上达到 5fps(包括候选区域的生成)，而之前的 fast R-CNN 的 ss 算处理一张图都需要 2 点多秒。准确率也有进一步的提升。在 2015 年的 ILSVRC 以及 COCO 竞赛中得多个项目的第一名。
<hr>
<h3 id="架构和流程-">架构和流程：</h3>
其实 Faster R-CNN 相当于 RPN + Fast R-CNN 的合体
步骤：

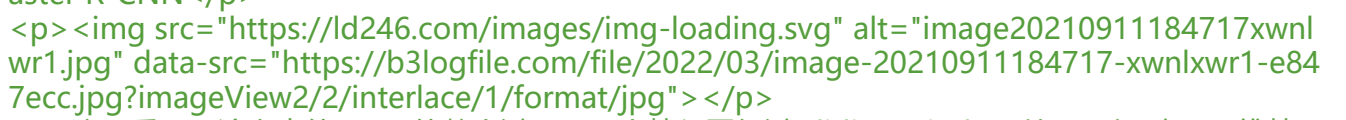
将图像输入网络得到相应的特征图
使用 RPN 结构生成候选框，将 RPN 生成的候选框投影到特征图上获得相应的特征矩阵
将每个特征矩阵通过 ROI Pooling 层缩放到 7x7 大小的特征图，接着将特征图展平通过一系列连接层得到预测结果。

<h3 id="anchors">anchors</h3>

<blockquote>
其实这个操作可以看成看作是对特征图进行了 3 * 3 的卷积操作，最后得到一个 channel 为 256 的特征图。尺寸和公共特征图相同。可以看成是这样的张量形式 <code>(56,H,W)</code>。
对于 k 个 anchor boxes 通过并联两个 1x1 的卷积层 生成 2k 个分数和 4k 个坐标位置
</blockquote>
<blockquote>

2k scores :
这个向量是每 2 个为一组，对应的是一个 anchor box，分别表示前景和背景的概率。并没有进分类。
4k coordinates
每四个为一组，对应一个 anchor box，表示这个 anchor box 的位置。
dx dy 是针对 anchor box 的中心坐标的偏移量，dw dh 是对 anchor box 的宽度和高度的调整
</blockquote>
回归到图上，可以看到，这里设为一个锚点产生 9 个锚框 三种比例 1:1、1:2、2:1，三种大小 12 平方，256 平方 512 平方。（这个也是经验所得）[至于为什么会有 512x512 面积大小 都已经超过锚点的感受野，作者解释是：类似看到目标的局部一小部分，也可以大概的猜出物体的完整区域。“一斑而见全豹”]
可以发现有很多的 anchor box，而对于 RPN 生成的候选框 <code>(anchor box经过RPN回归数调整后的产物)</code> 之间存在大量的重叠的处理方法是 基于候选框的 cls 得分，采用非极大值制，IoU 设为.7，这样每张图只剩下大约 2k 个候选框。正好和 ss 算法提供的候选框的个数差不多了
RPN 损失函数
前面的公式 使用的是多类别交叉熵损失，虽然只有背景和前景两个类别；
边界框回归损失函数这里也同样的采用的是 smooth L1 函数。
至于后面的 fast rcnn 的损失函数前面也讲到了。
Faster R-CNN 训练
现在所使用 的 faster rcnn 网络 是直接采用 RPN loss 和 fast rcnn 相加联合训练方法。就是按权加在一起，直接进行反向传播。在 pytorch 中官方实现 faster rcnn 的代码也是使用的联合训练方法训练的。
论文中：

利用 ImageNet 预训练分类模型初始化前置卷积网络层参数，并开始单独训练 RPN 网络参数</l>
固定 RPN 网络独有的卷积层以及全连接层参数，再利用 ImageNet 预训练分类模型初始化前置积网络参数，并利用 RPN 网络生成的目标建议框去训练 Fast RCNN 网络参数
固定利用 Fast RCNN 训练好的前置卷积网络层参数，去微调 RPN 网络独有的卷积层以及全连层参数。
同样保持固定前置卷积网络层参数，去微调 Fast RCNN 网络的全连接层参数。最后 RPN 网络与ast RCNN 网络共享前置卷积网络层参数，构成一个统一网络。

<h3 id="RPN-">RPN:</h3>
RPN 的引入，可以说真正意义上把物体检测整个流程融入到一个神经网络中，这个网络结构就是aster R-CNN


这里看一下论文中的 RPN 的整个过程。一个特征图经过 sliding window 处理，得到 256 维特，然后通过两次全连接得到结果 2k 个分数和 4k 个坐标，那么，是怎么得到的呢？

RPN 的 input 特征图指的是哪个特征图？
为什么是用 sliding window？文中不是说用 CNN 么？
256 维特征向量如何获得的？
2k 和 4k 中的 k 指的是什么？
图右侧不同形状的矩形和 Anchors 又是如何得到的？

回答：

输入特征图就是公共的 Feature map,主要用来 RPN 和 RoI Pooling；
可以把 3 * 3 的 sliding window 看作是对特征图进行了 3 * 3 的卷积操作，最后得到一个 chann

l 为 256 的特征图。尺寸和公共 Feature map 相同可以看成是这样的张量形式 <code>(256,H,W)</ode>
上面的张量可以看成是一堆水管堆在一起，有 <code>H * W</code> 个 256 维向量。对每一向量都做两次全连接操作：一个得到 2 个分数(前景，背景)；一个得到 4 个坐标。由于每个向量(每个管)都要做全连接操作，等同于对特征图做了两次 1 * 1 卷积操作。得到一个 <code>(2,H,W)</code> 和 一个 <code>(4,H,W)</code> 的特征图。换句话说就是有 <code>H * W </code> 个结果，每结果包含 2 个分数，4 个坐标。

这里 说一下为什么是 2 个分数，因为这里只是判断一下是背景图还是前景，并不进行类别判断四个图标也是针对原图 坐标的偏移。
<ol start="4">
& 5 ：

上面得到了 <code>H * W</code> 个结果，就是相等于每个长长的管子变成了 <code>2分数4坐标</code> 。每个结果，也就是每个点映射到原图都是个框框，一般框框的大小就是原图对应特图的缩小比例。那么，这个框框是不是我们需要的框呢，可以把框的左上角或中心点作为锚点（ancho），其实就是找到不动点坐参考坐标系圆点。那么每个映射到原图的锚点上面该画出多少框呢？ 就是 个。就是图中的 K anchor boxes。（每个锚点产生 K 个框）。所以原图上会有多少框呢？ 就是 <coe>H * W * 9</code> 个框。所以综上所述，RPN 就是判断这些框是不是物体，以及判断框的偏移。 个框到底多大，长宽比多少，也都是预先设定好的。原文定的是 9 种组合。所以有 <code>H * W * </code> 个结果，也就是 18 个分数和 36 个坐标。
<img src="https://ld246.com/images/img-loading.svg" alt="image20210911192301eryysd1.jpg" data-src="https://b3logfile.com/file/2022/03/image-20210911192301-eryysfd1-28333c6.jpg?imageView2/2/interlace/1/format/jpg">
RPN 回顾：
最后我们再把 RPN 整个流程走一遍，首先通过一系列卷积得到公共特征图，假设他的大小是 N x 16 x 16，然后我们进入 RPN 阶段，首先经过一个 3 x 3 的卷积，得到一个 256 x 16 x 16 的特征图也可以看作 16 x 16 个 256 维特征向量，然后经过两次 1 x 1 的卷积，分别得到一个 18 x 16 x 16 特征图，和一个 36 x 16 x 16 的特征图，也就是 16 x 16 x 9 个结果，每个结果包含 2 个分数和 4 个标，再结合预先定义的 Anchors，经过后处理，就得到候选框；整个流程如图!
<img src="https://ld246.com/images/img-loading.svg" alt="image20210911192717bnwmc21.jpg" data-src="https://b3logfile.com/file/2022/03/image-20210911192717-bnwomc21-0bd4894.jpg?imageView2/2/interlace/1/format/jpg">
<hr>
<h2 id="对比总结">对比总结</h2>
图片在笔记中。。
【深度学习】目标检测算法结（R-CNN、Fast R-CNN、Faster R-CNN、FPN、YOLO、SSD、RetinaNet） - 郭耀华 - 博客园 (nblogs.com)
https://blog.cdn.net/weixin_44474718/article/details/89414127
<blockquote>
R-CNN
分为四块。都是单独训练的
先通过 ss 算法提取候选框，然后作为 CNN 网络的输入，然后将提取到的特征送入每一类的分器 同时使用回归器精细修正候选框的位置。
Fast R-CNN
分为两块
首先 单独使用 ss 算法提取候选框
然后 这三个部分融合在在 CNN 网络中
Faster R-CNN
可以看到这个的四个部分 全部融合到了 CNN 网络当中。这是一个整体，实现了端对端的训练过

在递进的过程中，网络是越来越简洁的，速度和效果也是越来越好的
</blockquote>
<h2 id="代码讲解-">代码讲解：</h2>
<h3 id="数据集">数据集</h3>
<blockquote>
标准数据集，voc-2007 是衡量图像分类识别能力的基准。
faster-rcnn，yolo -v1, yolo-v2 都以此数据集为最为演示样例，因此，有必要了解一下本数据集的成架构。
</blockquote>
包含约 10,000 张带有边界框的图片用于训练和验证。含有 20 个类别。具体包括:

Person: person
Animal: bird, cat, cow, dog, horse,sheep
Vehicle: aeroplane, bicycle, boat,bus, car, motorbike, train
Indoor: bottle, chair, dining table,potted plant, sofa, tv/monito

训练集：
<pre><code class="language-python highlight-chroma"><spa
 class="highlight-cl">aeroplane <span class="highlight-
i">238
<span class="high
ight-n">bicycle 243
<span class="high
ight-n">bird 330
<span class="high
ight-n">boat 181
<span class="high
ight-n">bottle 244
<span class="high
ight-n">bus 186
<span class="high
ight-n">car 713
<span class="high
ight-n">cat 337
<span class="high
ight-n">chair 445
<span class="high
ight-n">cow 141
<span class="high
ight-n">diningtable 200
<span class="high
ight-n">dog 421
<span class="high
ight-n">horse 287
<span class="high
ight-n">motorbike 245
<span class="high
ight-n">person 2008
<span class="high
ight-n">pottedplant 245
<span class="high
ight-n">sheep 96
<span class="high

ight-n">sofa 229
<span class="high
ight-n">train 261
<span class="high
ight-n">tvmonitor 256
</code></pre>
测试集
<pre><code class="language-python highlight-chroma"><spa
class="highlight-cl">aeroplane <span class="highlight-
i">204
<span class="high
ight-n">bicycle 239
<span class="high
ight-n">bird 282
<span class="high
ight-n">boat 172
<span class="high
ight-n">bottle 212
<span class="high
ight-n">bus 174
<span class="high
ight-n">car 721
<span class="high
ight-n">cat 322
<span class="high
ight-n">chair 417
<span class="high
ight-n">cow 127
<span class="high
ight-n">diningtable 190
<span class="high
ight-n">dog 418
<span class="high
ight-n">horse 274
<span class="high
ight-n">motorbike 222
<span class="high
ight-n">person 2007
<span class="high
ight-n">pottedplant 224
<span class="high
ight-n">sheep 97
<span class="high
ight-n">sofa 223
<span class="high
ight-n">train 259
<span class="high
ight-n">tvmonitor 229
</code></pre>
可以看出，除了 person 数量较多，其他类别样本个数不算多，在如此小的数据集上，深度学习获得较高的分类识别结果，足以说明深度学习的强大性能.
<blockquote>
imagesets 为数据集文件，JPEGImages 存放图片数据，Annotations 是对应的标记文件。
Annotations 文件夹存放的是 xml 格式的文件，每个文件对应 JPEGImages 文件夹里的一张图<

p>
ImageSets 存放的是每一种类型 challenge 对应的图像数据
</blockquote>

Action 下存放的是人的动作（running、jumping 等等，这也是 VOC challenge 的一部分）</l
>
Layout 下存放的是人体部位（head、hand、feet 等等，这也是 VOC challenge 的一部分）</l
>
Main 下存放的是图像物体识别的数据，总共分为 20 类

train 里面放的是训练集的图片编号 （前面表示图像的 name，后面 1 表示正样本，-1 代表负样
）
val 存放的是验证集的图片编号 。 与上面没有重合
trainval 是两者合并的集合

<blockquote>
图片的像素尺寸大小不一，但是横向图的尺寸大约在 500 * 375 左右，纵向图的尺寸大约在 375
 500 左右，基本不会偏差超过 100。
</blockquote>