Technology Study Report

Faster R-CNN: Implementation Details and Difficulties

冯浩哲

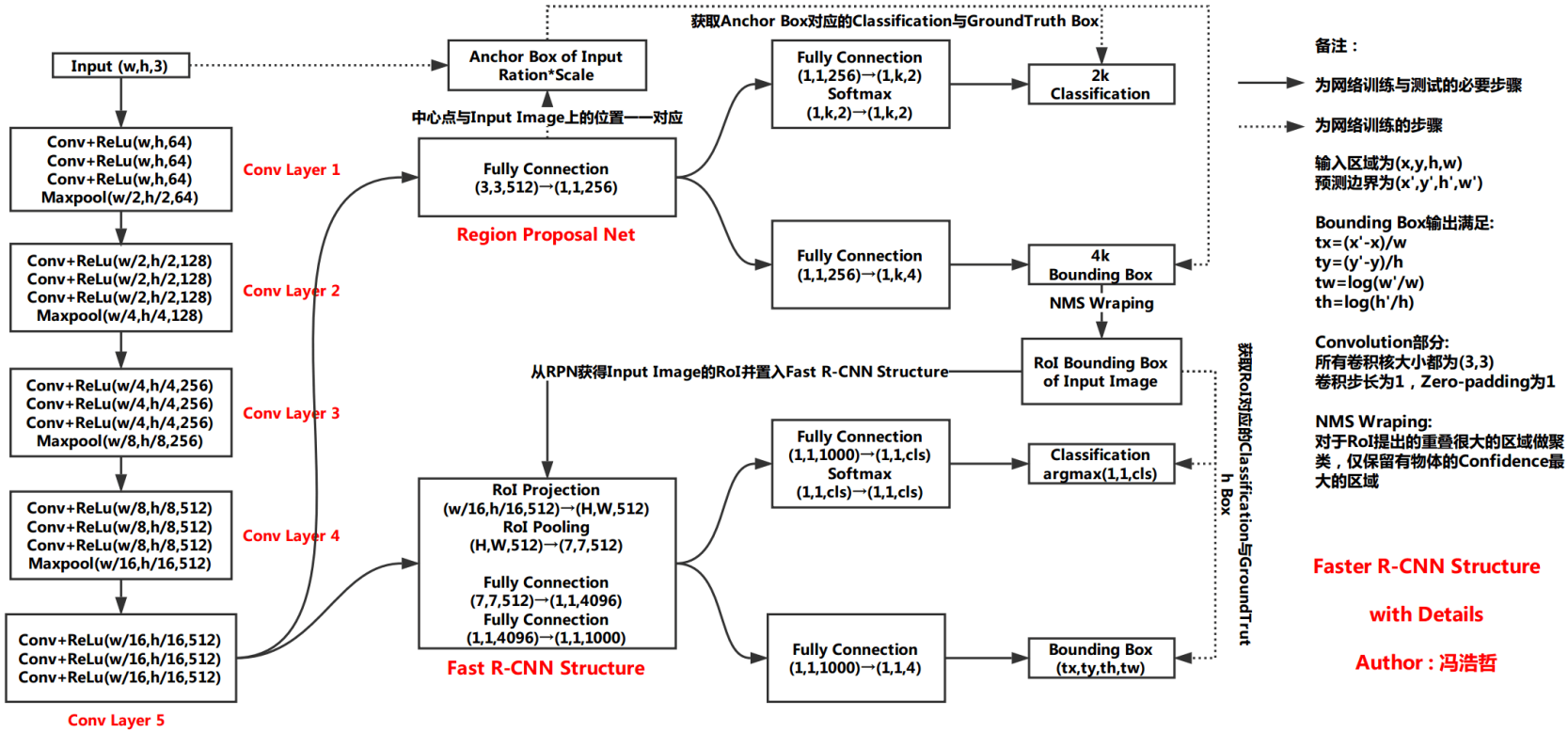
2017 年 11 月 10 日

提纲

Faster R-CNN是Object Detection与Instance Segmentation的基础框架，因此搞懂这个框架的实现细节非常重要。本次分享我准备先对Faster R-CNN的框架与具体细节做一次清楚的叙述，然后分享一下这个框架的实现难点与可行的解决方案，最后我会推荐一些Github上可以学习的工程与一个可以迅速上手Faster R-CNN的教程，提纲如下：

- Faster R-CNN 的实现细节与Framework(以VGG为主体框架)
- Pascal VOC数据集介绍
- 难点1：Region Proposal Net 训练过程
- 难点2：RPN与Fast R-CNN交互问题
- 难点3：RoI Pooling的实现
- 难点4：RPN与Fast R-CNN交叉训练过程

Faster R-CNN 的实现细节与Framework(以VGG为主体框架)



Object Detection训练数据集是什么样的

我们以PASCAL VOC数据集为例。

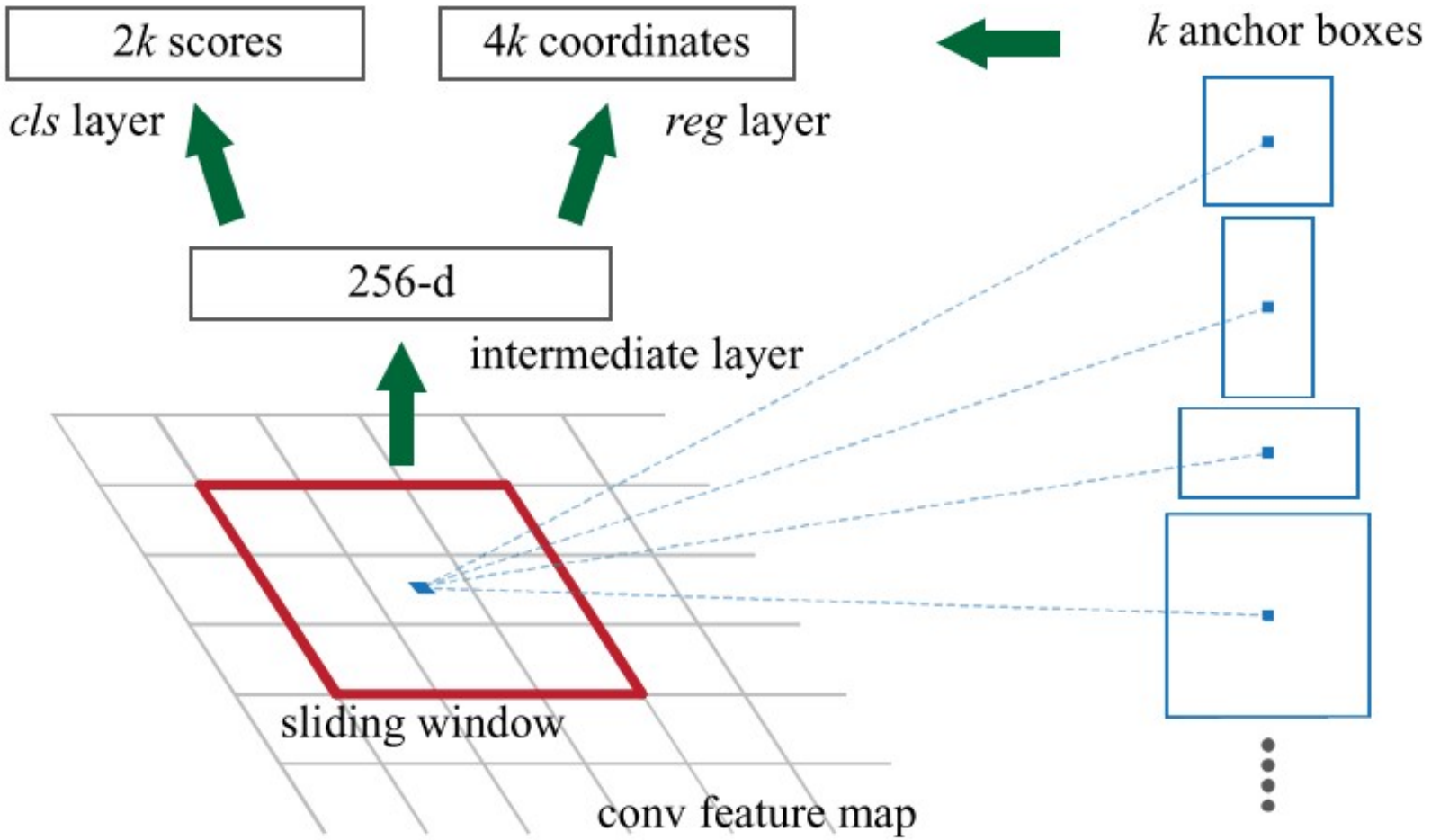
PASCAL VOC数据集提供了不同目标检测任务(如Bicycle,Bird,Boat)的训练数据集与测试数据集。每一张图片有一个对应的xml，xml记录了该图片中的目标种类以及每一种目标相应的Bounding Box。

图片都是统一的jpg格式的2D图片，但是大小不一。

具体我们可以直接看数据集。

难点1：Region Proposal Net 训练过程

Region Proposal Net的关键部分如下图:



难点1：Region Proposal Net 训练过程

我们最后一层 feature map尺寸为$(\frac{w}{16}, \frac{h}{16}, 512)$，因此对于feature map上的某个点$(x_f, y_f)$，其对应输入图像的坐标为$(16 * x_f, 16 * y_f)$，根据这个坐标生成9个锚盒，并对这9个锚盒做预测。

在预测的过程中，对Feature map上的每一个点采样周围3*3的区域做全连接，输出9组对应的Anchor Box中是否有物体,以及如果有物体的话对Bounding Box进行精确预测。这个过程中有2个实现困难:

- Anchor Box是否有目标，以及相应的Ground Truth Label预测
- 边缘Anchor Box的问题

Anchor Box是否有目标，以及相应的Ground Truth Label预测

在具体实现中，label应该这样标记:

对于给定的图像，我们需要对每个Anchor Box对图像的Ground Truth Box进行IoU的计算，选IoU最大且大于0.7的Ground Truth作为这个Anchor Box 的Ground Truth.如果这个Anchor Box对于所有的Ground Truth的Iou都小于0.3，标记为负样本，其它情况不做任何标记。

Loss函数是$L(\{p_i\},\{t_i\}) = \frac{\sum L_{cls}(p_i,p_i^*)}{N_{cls}} + \lambda * \frac{\sum_i p_i^* L_{reg}(t_i,t_i^*)}{N_{reg}}$

因此在实现的时候对于每一个锚盒应该有3个标签$(-1,0,1)$,0,1为Object是否存在的标签，而-1表示锚盒没有任何标记，对损失函数不起任何影响。这样实现我们就可以在生成256维特征向量的同时计算锚盒的标签，并进行损失函数计算了。我们的数据交互要达到这个目的，也就是在进行处理的同时生成相应的标签。

我们可以选择先对图片处理，计算特征图的每一个Localization(其实就是每一个点)对应原图的9个anchor的标签(因为这实际上并不涉及到神经网络的运算)，然后用神经网络再处理图片，或者是采用并行的方法，在用神经网络处理图片的同时得到标签。

边缘Anchor Box的问题

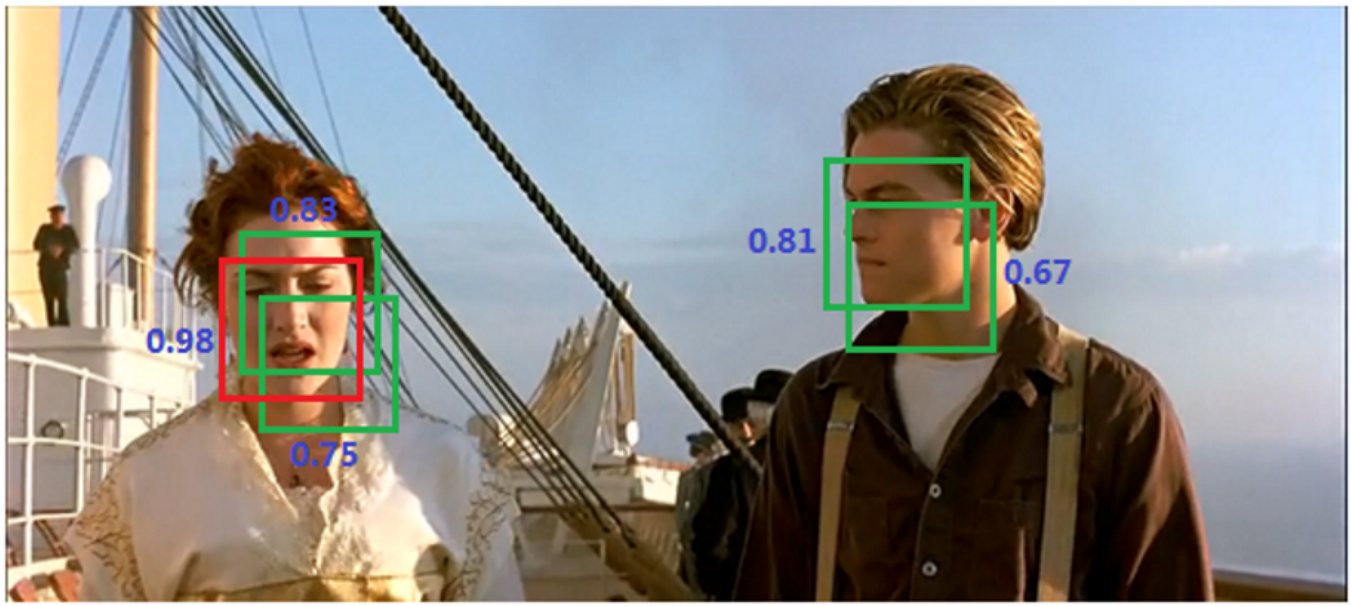
注意到我们feature map上的点与输入图像是16倍的比例关系，因此如果取feature map上较为边缘的点的时候，在Ratio * Scale的作用下，会导致在原图中生成的anchor box超过图片边界。比如说如果我们以原图的(8，8)点为中心画anchor，然后scale为(8,16,32),ratio为(0.5,1,2)，那么就会生成如下的9个anchor:

```
# anchors =
# -83 -39 100 56
# -175 -87 192 104
# -359 -183 376 200
# -55 -55 72 72
# -119 -119 136 136
# -247 -247 264 264
# -35 -79 52 96
# -79 -167 96 184
# -167 -343 184 360
```

负值部分与过大的正值部分都会超出图片边缘,依照论文我们应该在代码中采用两种方式来解决这个问题。
一方面，在练Region Proposal Net的过程中，我们会直接忽略这些锚盒，而在最终的Region Proposal如果出现了这些锚盒就进行边缘裁剪，然后将裁减过的区域作为RoI进行特征图映射。
另一方面，为了保证在训练过程中不至于忽略太多的锚盒，我们将送入图片的短边统一先进行缩放为608(因为能被16整除)个像素，然后再输入(注意长边可以插值使得也能被16整除)。忽略的操作可以和1中一样，对对应的锚盒标记为-1就可以了。

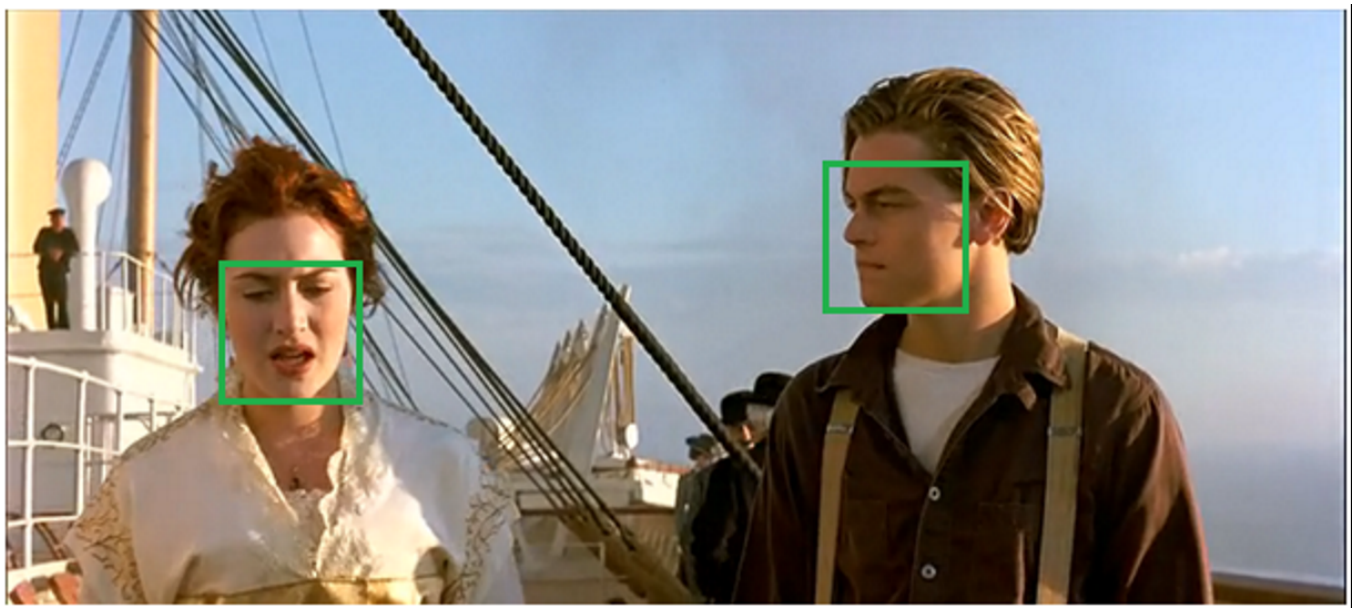
难点2：RPN与Fast R-CNN交互问题

RPN的还有一部分问题在于与Fast R-CNN进行交互的问题。假设我们输入一张图，然后针对这个图我们会以每组2k,4k的形式输出若干组数据，然后对那些物体预测为是的部分作为ROI，这就会导致以下问题:



难点2：RPN与Fast R-CNN交互问题

我们得到的RoI中有可能对于一个物体有很多框将其框了出来，这就会造成冗余运算，因此我们需要一个NMS算法来对这些区域筛选得到下面的区域:



这个算法的基本原理是对于那些有物体的框框按IoU的相似度进行层次聚类，保证类间距要大于一个阈值(也就是说类内框框互相重合度都很高)。
然后我们在每一类中选Confidence最大的作为输出，这就是对RPN的输出进行精简而减少冗余运算的小细节。

难点3:RoIpooling的实现

我们在得到RoI之后，获得Roi的左上与右下坐标，然后做RoI Projection。

RoI pooling是将映射到最后一层的Feature Map的RoI变成固定的大小(论文中是7*7的大小)，这种操作因为其尺度不一样导致了它是一个动态的操作，即对不同的RoI映射要进行单独研究。一般来说假设RoI的特征映射尺寸是$(H, W, 512)$，那么我们的maxkernel就是$(H/7, W/7)$ stride也是$(H/7, W/7)$,这样依照公式就是$\frac{H-(H/7)}{(H/7)}+1=7$

但是RoI Pooling的实现有2个问题，第一个问题就是Pytorch与Tensorflow都没有RoI Pooling这个函数，不过现在网上已经有了RoI Pooling的C扩展，以及Pytorch可以用torch.autograd.Function模块来自己写这个函数。第二个问题是很现实的，就是假设此时$H/7$不是整数怎么办？

这个我查看了RoI Pooling的C扩展代码，然后给出一个大致的框架如下:

难点3:RoIpooling的实现

这个我查看了RoI Pooling的C扩展代码，然后给出一个大致的框架如下:

假设我们现在的Featuer Map为$(15 * 13)$，然后我要映射到$(7 * 7)$那么我的步长就是$(15/7, 13/7)$ 因此我可以把区间分为:

$$(0, (15/7), 2 * (15/7), 3 * (15/7), ..., 6 * (15/7), 15)$$

分为7段:

$$(0, \frac{15}{7}) = (0, 2.14), (\frac{15}{7}, 2 * \frac{15}{7}) = (2.14, 4.28), ..., (6 * \frac{15}{7}, 15) = (12.8, 15)$$

我们对其的每个节点做round操作，就得到了一组整数区间，同样对另一边做操作就可以了,然后反向传播的时候只需要记录max节点并将其导数置为1就可以了。

难点4：RPN与Fast R-CNN交叉训练过程

Region Proposal Net与Fast R-CNN是两个共用VGG层参数的网络，因此训练是关键问题。文章给出了以下有效的训练策略：

- 首先，将所有参数初始化训练Region Proposal Net。训练中标定锚盒与Ground Truth的$IoU > 0.7$的为前景，标定$IoU < 0.3$的为背景，依次训练分类与Bounding Box Regression部分。
- 其次，利用Region Proposal Net所提供的RoI训练Fast R-CNN部分网络，依次训练分类与Bounding Box Regression部分。
- 最后，固定Fast R-CNN部分网络的VGG卷积层参数，再训练Region Proposal Net。
- 依次进行训练直到收敛。接下来每次训练Region Proposal Net的时候都固定卷积层参数。

Github源码与可迅速上手的教程

Github上有一些基于Python-Caffe的Faster R-CNN代码，Tensorflow与Pytorch也有相关版本，依次如下：

- https://github.com/rbgirshick/py-faster-rcnn
- https://github.com/endernewton/tf-faster-rcnn
- https://github.com/ruotianluo/pytorch-faster-rcnn

TF与Pytorch版本都是工业级的代码，对于C扩展，数据集整合交互以及模型部分都有应用级的封装，源码也非常值得看。如果暂时没有时间阅读源码，可以参考以下非常易上手的使用教程：

- http://blog.csdn.net/sinat_30071459/article/details/51332084