# 第一次工作总结[[1]](#footnote-1)

赵淼

本周的工作是围绕物体检测任务展开的，相对于图片分类而言，物体检测更具有实际应用价值，也更具有前沿性，同时也是后续一系列工作的基础，以物体检测任务作为开端，可以更顺利的进入实际课题中。

在物体检测任务中，深度学习是极其有力的工具，近些年物体检测比赛中，排名靠前的队伍使用的方法基本都是使用深度学习。深度学习的方法在物体检测中既具有极高的检测准确率，同时也具有极高的实时性，其中Faster RCNN最先做到了这一点，它在VOC数据集中取得了60%的准确率，而且达到1秒5帧的速度，在这之后，一些相关的方法被提出，通过对Faster RCNN的学习，可以使我更快的了解后面一系列方法中的思想。

本周工作主要分为两部分，一部分是对论文的解读，另一部分是对代码的调试和理解。

## 1 论文解读

### 1.1 简介

作者在论文的开始直接提出了Faster RCNN相对于以前方法的优越性，在Faster RCNN之前，使用神经网络解决物体检测问题，都要涉及到候选区域提取的算法，这部分通常需要一个独立模块进行处理，无法纳入到神经网络中，只能在CPU中进行运算，这直接拖慢物体检测的速度，作者在这里创新性的提出了使用RPN层进行候选区域的提出，将候选区域提取的模块直接纳入到神经网络中，极大的提高了物体检测的速度。

通过RPN层的提出，作者达到了特征共享的目的。特征共享这个概念在这篇论文提出之前就已经得到了大家的关注。通过特征共享，作者实现了真正意义上的端到端的物体检测网络。这里的重点是实现候选区域提取算法所使用的特征要与后面的候选区域分类与回归网络使用的特征相同，只有通过特征的共享，才可以将候选区域检测模块纳入到神经网络架构中。

作者提出的网络结构如图1.1所示，这里作者将整个网络分为两个部分，一部分是全卷积的候选框的提取模块，一部分是物体检测模块，整体形成了一个两阶段的物体检测网络，相对于两阶段的物体检测网络，这两年单阶段的网络架构也很流行，这种架构速度更快，但检测精度不足，这部分是我后面要进行学习的部分。

这一网络在基础的物体检测任务表现出极佳的效果，于是又有一些人将其应用到一些具体的实际应用中，如3D物体检测，局部检测，物体分割，图像字幕等，这部分有待后面的继续学习。



图1-1 Faster RCNN架构

### 1.2 候选区域提取模块

#### 1.2.1 网络结构

这一部分的前面是特征提取网络，作者直接将ZF和VGG-16的网络放到了这里，对原始图像进行卷积分别形成了256维的特征图和512维的特征图。

在特征提取网络后面就是RPN网络，这个网络是整个Faster RCNN的关键，作者通过3\*3的滑动窗口在特征图上进行滑动，针对每个滑动窗口获得了k个anchor boxes，然后进行3\*3卷积，将卷积后的特征图输入到2个1\*1的卷积层得到分类的结果和回归的结果。这里分类的结果表示对应的anchor boxes是否是物体，回归的结果表示anchor boxes的4个顶点坐标如何变化才能将物体框住，具体结构如图1.2所示。



图1-2 RPN层结构

#### 1.2.2 anchor

这里作者提出了anchor的概念，anchor指的是每个滑动窗口的中心，作者将每个anchor所在的区域都映射到原图像的对应区域中去，然后采用不同的长宽比例和缩放比例获得不同的anchor boxes，这里我的理解是每个特征在原图都对应着一定范围的感受野，而且特征不仅可以代表感受野内的部分，还可以代表以感受野为中心，不同范围内的图像内容，这里举例假设在感受野内存在人体的躯干，那么就可以预测是否存在人和人的区域范围大体在哪里，作者通过这种anchor box的形式可以大大降低候选区域提取算法提出候选区域的数目。

同时，anchor具有平移不变性，这里使用卷积共享的形式，当候选区域平移后，依然能够找到候选区域，同时大幅度减少了输出层（1×1卷积层）参数数目（512 × (4 + 2) × 9）。

#### 1.2.3 RPN层训练

在实际训练中作者将两种anchor设为正例，一类是对每个ground\_truth IOU最大的的box，一类是IOU大于0.7的box，这里可能对同一个ground\_truth产生多个正例框。对于对于所有ground\_truth的IOU小于0.3设为反例，既不是正例也不是反例的anchor不会对训练目标给予贡献。

这里损失函数定义为两部分，一部分是分类，一部分是回归，本文里k个anchor box的回归不共享参数。

作者在训练时随机采样了256个anchors，正反例大概1：1,如果正例不足，则用反例补充，进而解决训练数据不均衡问题，这部分是解决训练数据分布不均衡的一个方案，对于解决训练数据不均衡具有借鉴意义。

#### 1.2.4 总结

可以说RPN层是整个Faster RCNN的最关键的部分，前面的部分是VGG-16的网络结构，后面直接使用了Fast RCNN的结构，可以说只有这一层是作者这篇论文中真正的创新点，这里RPN层的任务就是提出可能含有感兴趣物体的区域，由于作者这里采用的是经过ImageNet数据集训练的VGG-16网络进行特征提取，实际上已经基本具备提取物体特征的能力，这里作者认为这是一种注意力的机制，先确定物体可能存在的区域，再判断物体的具体类型和范围，进而达到更好的图体检测效果。但我认为这里还包含着流形学习的思想，即在整个图像所能表示的空间中只有一部分可以表示现实中的物体，而这部分的分布呈现出流形分布的特点，作者的网络前部分是要寻找物体流形的分布，后半部再进行各特定物体流形分布的寻找，逐渐缩小范围，进而达到更好的检测效果，这里提示我们可以在执行具体任务时以更大规模的数据集或者是相似任务的数据集先对神经网络进行训练，减少数据规模不足对训练任务的影响，提高训练效果。

### 1.3 Fast RCNN

这部分对候选框进行分类与回归的任务，作者在这部分并没进行深入的探讨，主要注意的是作者输入的特征提取网络提取的特征图和RPN提出的候选框信息，也就是实现了候选框提取与物体检测部分的特征共享。

### 1.4 训练

作者针对Faster RCNN的训练主要提出了两种训练方案，一是交替训练，二是近似联合训练，后面在实际实验中采用的是近似联合训练，当然作者也认为还有其他的训练方案，不过这部分不在论文探讨的范围内，也是我可以继续研究的方向。

#### 1.4.1 交替训练

这部分的训练主要分为以下4步，下面的model代表的是特征提取网络，unique代表这部分特有的参数。

第一步：用model初始化RPN网络，然后训练RPN，在训练后，model以及RPN的unique会被更新

第二步：用model初始化Fast-RCNN网络，注意这个model和第一步一样。然后使用训练过的RPN来计算proposal，再将proposal给予Fast-RCNN网络。接着训练Fast-RCNN。训练完以后，model以及Fast-RCNN的unique都会被更新。

说明：第一和第二步，用同样的model初始化RPN网络和Fast-RCNN网络，然后各自独立地进行训练，所以训练后，各自对model的更新一定是不一样的，因此就意味着model是不共享的（论文中的dont share convolution layers）。

第三步：使用第二步训练完成的model来初始化RPN网络，第二次训练RPN网络。但是这次要把model锁定，训练过程中，model始终保持不变，而RPN的unique会被改变。

说明：因为这一次的训练过程中，model始终保持和上一步Fast-RCNN中model一致，所以就称之为着共享

第四步：仍然保持第三步的model不变，初始化Fast-RCNN，第二次训练Fast-RCNN网络。其实就是对其unique进行finetune，训练完毕，得到一个文中所说的unified network。

这部分我的理解是在前两步中更新共享卷积层的参数使之更适应后面的具体任务，也可称为fine\_tune,后面两步在共享卷积特征的基础上训练RPN和Fast RCNN,因为最后的结果是Fast RCNN输出的，这里后训练Fast RCNN,,用第二步中fine\_tune的卷积网络进行后面两步的训练。

#### 1.4.2 近似联合训练

这一部分固定了RPN提取区域，将特征提取区域的损失与Fast RCNN的损失相加构成总的损失，直接对总损失进行反向传播，这种方法相当于忽略了两个阶段的区分，忽略了RPN对后面Fast RCNN的影响，加快了训练速度，在作者的实验中取得了不错的效果。

## 2 代码的调试与理解

这部分首先要做的是按照作者公布的代码将程序调通，使其可以完成基本的任务，然后再逐步调试理解作者编程的思路，同时促进对论文的理解。

### 2.1 代码调试

这部分的具体工作就是按照github上给出的提示完成代码的编译，数据集的下载和示例程序的运行，这部分所遇到的主要问题是对相关的脚本不够了解，这里涉及到编译的脚本和示例程序的运行，包括库的版本的问题，这里很多时候只能通过百度解决，可以借鉴的是作者将整个程序通过训练脚本进行运行，使整个训练实现了自动化，提高了效率。

#### 2.2 代码理解

这部分通过对VOC2007数据集的训练，逐步调试理清了整个程序的整体结构，由于对numpy和tensorflow还不是十分了解，因此一些具体细节还不是很了解，尤其使最关键的RPN部分不是十分理解。

#### 2.2.1 整体架构

整这个工程是按照下面的架构组织的。

1.data 存储训练数据

2.experiment 存储最顶层的实验脚本

3.lib 实验中各层的相关定义以及一些使用的库

4.model 实验中预训练好的模型

5.output 实验输出文件

6.tools 顶层的训练，测试脚本

整个工程作者进行了大量的类的封装，进行了许多扩展性的设计，方便了后面的扩展与维护，但也使整体结构比较复杂，增大了阅读代码的困难。

#### 2.2.2 训练数据的读取

在这部分读取了VOC2007数据集，总共5001幅图像，包括图像的位置，各类物体的位置等信息，这里作者对图像进行了翻转，使数据量增加到了10002幅图像，在工程里没有进行多尺度处理，这部分也是可以探究的一个方向，最终获得了imdb数据对象，这一部分虽然不涉及神经网络的主体部分，但也极为重要，在后面如果需要训练自己的数据，需要进行修改，在这里可以直接读取pkl文件获取box信息，在训练自己数据时需要编写xml文件记录训练数据，同时代码中对box进行了归一化操作，减去均值，除以标准差，并且对数据对象进行了随机化处理。

### 2.2.3 神经网络搭建

在这里主要的部分就是RPN层的构建，这里涉及anchor box的获取，在程序里作者以16\*16的anchor box为基础，分别取长宽比为2：1，1：2，和范围因子为2和0.5构建了9个anchor box，然后对于在原图像中越界anchor box进行了过滤处理，在后面的候选区域提取模块和roi pooling模块中使用了numpy获得了相关函数，然后使用tensorflow进行封装，使之成为了可以直接调用的神经网络层，这部分是整个工程的关键，也是以后我如何构建新层的指导，不过这部分我还没看的很懂，需要后面继续学习。

### 2.2.4 神经网络训练

这部分在工程采用的是端到端的训练方式，直接将RPN层的损失和Fast RCNN的损失加在了一起，进行反向传递，使用的是小批量随机梯度下降，动量取的是0.9，初始学习率是0.001，在经过50000次学习后学习率乘以0.1，经过70000次迭代完成训练，不过这个demo只完成了训练部分，没有完成测试。

## 3 总结

在本周的工作中对Faster RCNN的论文进行了理解和学习，同时对代码完成了调试和初步理解。这里我发现我对numpy和tensorflow还不是十分了解，以至于代码没有完全读懂，这是下一步我要努力的地方，在下面我会对这个网络架构有更深的理解，同时加入新的数据集，进行训练和测试，这里还需要对测试部分的代码有更深的了解。

1. 赵淼 电信研1706班 电子科学与技术 17120036 15650761201 [↑](#footnote-ref-1)