# R-FCN详解



## 论文题目：R-FCN: Object Detection via Region-based Fully Convolutional Networks

**论文链接：论文链接**

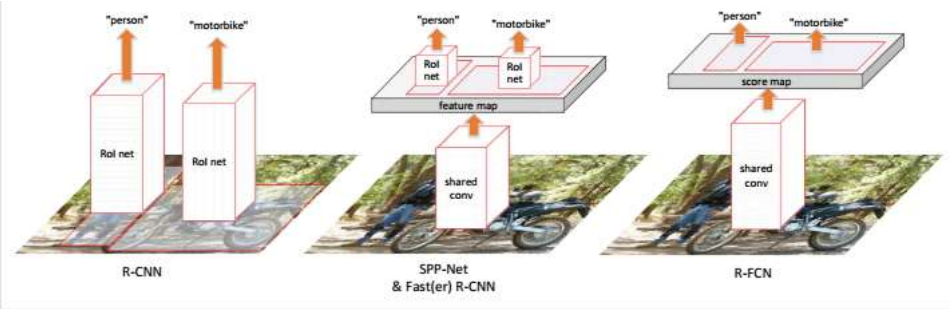
**论文代码：**Caffe版本链接地址；Python 版本链接地址；Deformable R-FCN版本链接地址

### 一、R-FCN初探

#### 1. R-FCN贡献

- 提出Position-sensitive score maps来解决目标检测的**位置敏感性**问题；
- 区域为基础的，**全卷积**网络的二阶段目标检测框架；
- 比Faster-RCNN快**2.5-20**倍（在K40GPU上面使用ResNet-101网络可以达到 0.17 sec/image）；

#### 2. R-FCN与传统二阶段网络的异同点




**Methodologies of *region-based* detectors using ResNet-101**

| | R-CNN | Faster R-CNN | R-FCN [ours] |
|---|---|---|---|
| depth of shared conv subnetwork | 0 | 91 | 101 |
| depth of RoI-wise subnetwork | 101 | 10 | **0** |

图1 R-FCN与传统二阶段网络的异同点

**相同点**：首先，两者二阶段的检测框架（全卷积子网络+RoI-wise subnetwork）；其次两者最终输出的结果都是相应的类别和对应的BB；

**不同点：**

如上图所示，我们可以看到和Faster R-CNN 相比，R-FCN具有更深的共享卷积网络层，这样可以获得更加抽象的特征；同时，它没有RoI-wise subnetw
不像Faster R-CNN的feature map左右都有对应的网络层，它是真正的全卷积网络架构；从图中的表格可以看出Faster R-CNN的共享卷积子网络是91层，
wise子网络是10层，而R-FCN只有共享卷积子网络，深度为101层。与R-CNN相比，最大的不同就是直接获得整幅图像的feature map，再提取对应的ROI
是直接在不同的ROI上面获得相应的feature map。

#### 3. 分类网络的位置不敏感性和检测网络的位置敏感性

**我在很多相关的检测论文中都看到这两个概念，但是一直都没有理解其真正的含义，相信很多朋友们也有同样的困惑，所以我在这里解释一下。**

- Increasing translation invariance for image classification
  - Shift of an object inside an image should be indiscriminative
  - Leading deep (fully) convolutional architectures are translation-invariant
- Respecting translation variance for object detection
  - Responses should reflect how candidate boxes overlap with objects
  - A considerable deep per-ROI subnet in Faster-RCNN using ResNet-101

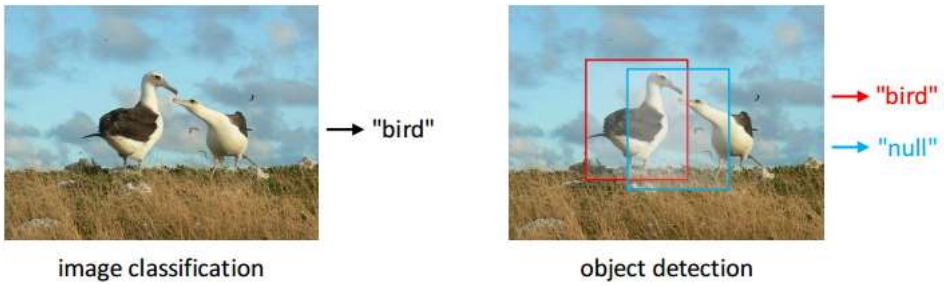



图2 分类网络的位置不敏感性和检测网络的位置敏感性

**分类网络的位置不敏感性**：简单来讲，对于分类任务而言，我希望我的网络有一个很好地分类性能，随着某个目标在图片中不断的移动，我的网络仍然可
的将你区分为对应的类别。如上图左边所示，不管你这只鸟在图片中如何移动，我的分类网络都想要准确的将你分类为鸟。即我的网络有很好地区分能力。
表明，深的全卷积网络能够具备这个特性，如ResNet-101等。

**检测网络的位置敏感性：**简单来讲，对于检测任务而言，我希望我的网络有一个好的检测性能，可以准确的输出目标所在的位置值。随着某个目标的移动，
网络希望能够和它一起移动，仍然能够准确的检测到它，即我对目标位置的移动很敏感。我需要计算对应的偏差值，我需要计算我的预测和GT的重合率等
是，深的全卷积网路不具备这样的一个特征。

**总之，分类网络的位置不敏感性和检测网络的位置敏感性的一个矛盾问题，而我们的目标检测中不仅要分类也要定位，那么如何解决这个问题呢，R-FCN
Position-sensitive score maps来解决这个问题；**

#### 4. R-FCN网络的设计动机

Faster R-CNN是首个利用CNN来完成proposals预测的，从此之后很多的目标检测网络都开始使用Faster R-CNN的思想。而Faster R-CNN系列的网络都可
成2个部分：ROI Pooling之前的共享全卷积网络和ROI Pooling之后的ROI-wise子网络（用来对每个ROI进行特征提出，并进行回归和分类）。第1部分就是
用普通分类网络的卷积层，用来提取共享特征，然后利用ROI Pooling在最后一层网络形成的feature map上面提取针对各个RoIs的特征向量，然后将所有R
特征向量都交给第2部分来处理（即所谓的分类和回归），而第二部分一般都是一些全连接层，在最后有2个并行的loss函数：softmax和smoothL1，分别用
每一个RoI进行分类和回归，这样就可以得到每个RoI的真实类别和较为精确的坐标信息啦（x, y, w, h）。

需要注意的是第1部分通常使用的都是像VGG、GoogleNet、ResNet之类的基础分类网络，**这些网络的计算都是所有RoIs共享的，在一张图片上面进行测
候只需要进行一次前向计算即可。而对于第2部分的RoI-wise subnetwork，它却不是所有RoIs共享的，主要的原因是因为这一部分的作用是"对每个RoI
类和回归"，所以不能进行共享计算。**那么问题就处在这里，首先第1部分的网络具有"位置不敏感性"，而如果我们将一个分类网络比如ResNet的所有卷积
置在第1部分用来提取特征，而第2部分则只剩下全连接层，这样的目标检测网络是**位置不敏感的translation-invariance，**所以其检测精度会较低，而且这
也会浪费掉分类网络强大的分类能力（does not match the network's superior classification accuracy）。而ResNet论文中为了解决这个问题，做出了一点
**即将RoI Pooling层不再放置在ResNet-101网络的最后一层卷积层之后而是放置在了"卷积层之间"**，这样RoI Pooling Layer 之前和之后都有卷积层，并
Pooling Layer之后的卷积层不是共享计算的，它们是针对每个RoI进行特征提取的，所以这种网络设计，其RoI Pooling层之后就具有了**位置敏感性transla
variance**，但是这样做会牺牲测试速度，因为所有的RoIs都需要经过若干层卷积计算，这样会导致测试速度很慢。R-FCN就是针对这个问题提出了自己的
案，在速度和精度之间进行折中。

### 二、R-FCN架构分析

#### 1. R-FCN算法步骤

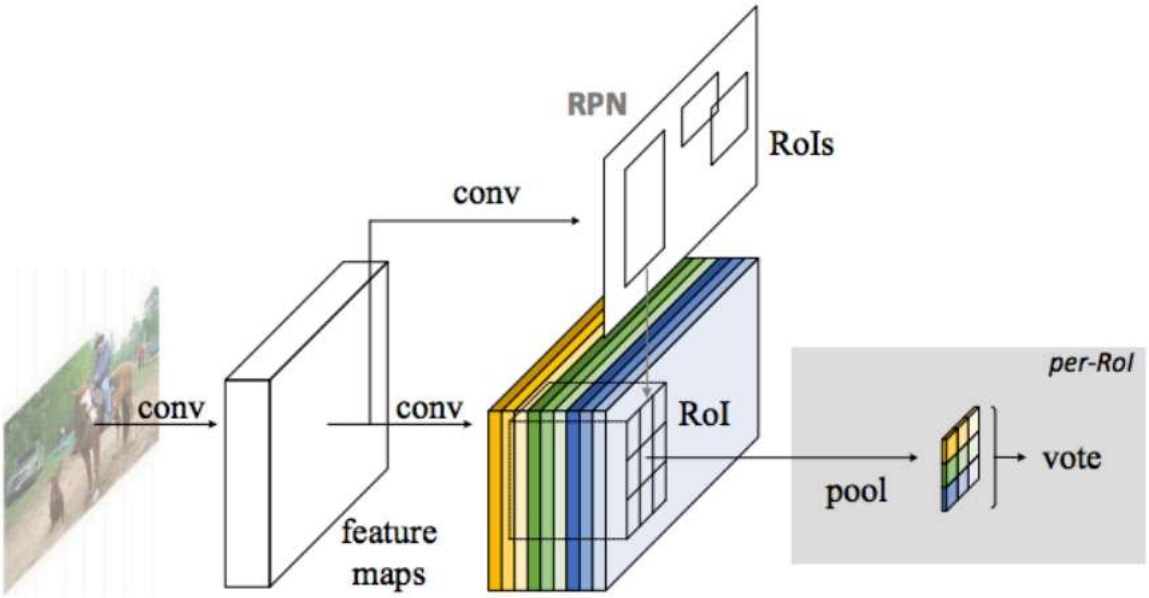



图3 R-FCN算法　步骤

如图所示，我们先来分析一下R-FCN算法的整个运行步骤，使得我们对整个算法有一个宏观的理解，接下来再对不同的细节进行详细的分析。

- 首先，我们选择一张需要处理的图片，并对这张图片进行相应的预处理操作；
- 接着，我们将预处理后的图片送入一个预训练好的分类网络中（这里使用了ResNet-101网络的Conv4之前的网络），固定其对应的网络参数；
- 接着，在预训练网络的最后一个卷积层获得的feature map上存在3个分支，第1个分支就是在该feature map上面进行RPN操作，获得相应的ROI；第2就是在该feature map上获得一个K*K*（C+1）维的位置敏感得分映射（position-sensitive score map），用来进行分类；第3个分支就是在该feature r获得一个4*K*K维的位置敏感得分映射，用来进行回归；
- 最后，在K*K*（C+1）维的位置敏感得分映射和4*K*K维的位置敏感得分映射上面分别执行位置敏感的ROI池化操作（Position-Sensitive RoI Pooling，使用的是平均池化操作），获得对应的类别和位置信息。

这样，我们就可以在测试图片中获得我们想要的类别信息和位置信息啦。

**2. Position-Sensitive Score Map解析**

图3是R-FCN的网络结构图，其主要设计思想就是**“位置敏感得分图position-sensitive score map”**。现在我们来解释一下其设计思路。如果一个RoI中含有类别C的物体，我们将该RoI划分为K*K 个区域，其分别表示该物体的各个部位，比如假设该RoI中含有的目标是人，K=3，那么就将“人”划分成了9个子区域top-center区域毫无疑问应该是人的头部，而bottom-center应该是人的脚部，我们将RoI划分为K*K个子区域是希望这个RoI在其中的每一个子区域都应该含类别C的物体的各个部位，即如果是人，那么RoI的top-center区域就应该含有人的头部。当所有的子区域都含有各自对应的该物体的相应部位后，那么分类会将该RoI判断为该类别。**也就是说物体的各个部位和RoI的这些子区域是“一一映射”的对应关系。**

OK，**现在我们知道了一个RoI必须是K*K个子区域都含有该物体的相应部位，我们才能判断该RoI属于该物体，如果该物体的很多部位都没有出现在相应的中，那么就该RoI判断为背景类别。**那么现在的问题就是**网络如何判断一个RoI的 K*K个子区域都含有相应部位呢？**前面我们是假设知道每个子区域是否含的相应部位，那么我们就能判断该RoI是否属于该物体还是属于背景。那么现在我们的任务就是**判断RoI子区域是否含有物体的相应部位。**

这其实就是position-sensitive score map设计的核心思想了。R-FCN会在共享卷积层的最后一层网络上接上一个卷积层，而该卷积层就是**位置敏感得分图position-sensitive score map**，该score map的含义如下所述，**首先它就是一层卷积层，它的height和width和共享卷积层的一样（即具有同样的感受野），是它的通道个数为K*K*(C+1)** 。其中C表示物体类别种数，再加上1个背景类别，所以共有(C+1)类，而每个类别都有 K*K个score maps。现在我们只针对其一个类别来进行说明，假设我们的目标属于人这个类别，那么其有 K*K 个score maps，**每一个score map表示原始图像中的哪些位置含有人的某个部位，score map会在含有对应的人体的某个部位的位置有高的响应值**，也就是说**每一个score map都是用来描述人体的其中一个部位出现在该score map的何处在出现的地方就有高响应值”。**既然是这样，那么我们只要**将RoI的各个子区域对应到属于人的每一个score map上然后获取它的响应值就好了。**但是要注意是，**由于一个score map都是只属于一个类别的一个部位的，所以RoI的第 i个子区域一定要到第i张score map上去寻找对应区域的响应值，因为RoI的第i域需要的部位和第i张score map关注的部位是对应的。那么现在该RoI的K*K个子区域都已经分别在属于人的K*K个score maps上找到其响应值了，那么这些响应值都很高，那么就证明该RoI是人呀。**当然这有点不严谨，因为我们只是在属于人的 K*K个score maps上找响应值，我们还没有到属于其它类别的score maps上找响应值呢，万一该RoI的各个子区域在属于其它类别的上的score maps的响应值也很高，那么该RoI就也有可能属于其它类别呢？是吧，如果2个物体本身就长的很像呢？这就会涉及到一个比较的问题，那个类别的响应值高，我就将它判断为哪一类目标。它们的响应值同样高这个情况发生的几率很小我们不做讨论。

OK，这就是position-sensitive score map的全部思想了，应该很容易理解了吧。

**3. Position-Sensitive RoI Pooling解析**

上面我们只是简单的讲解了一下**ROI的K*K个子区域在各个类别的score maps上找到其每个子区域的响应值，我们并没有详细的解释这个“找到”是如何找**就是位置敏感RoI池化操作（Position-sensitive RoI pooling），其字面意思是池化操作是位置敏感的，下来我们对它进行解释说明。

如图3所示，通过RPN提取出来的RoI区域，其是包含了x,y,w,h的4个值，也就是说不同的RoI区域能够对应到score map的不同位置上，而一个RoI会被划分个bins（也就是子区域。每个子区域bin的长宽分别是 h/k 和 w/k ），每个bin都对应到score map上的某一个区域。既然该RoI的每个bin都对应到score ma某一个子区域，那么**池化操作就是在该bin对应的score map上的子区域执行，且执行的是平均池化。**我们在前面已经讲了，第i个bin应该在第i个score ma找响应值，那么也就是在第i个score map上的第i个bin对应的位置上进行平均池化操作。由于我们有(C+1)个类别，所以每个类别都要进行相同方式的池化操

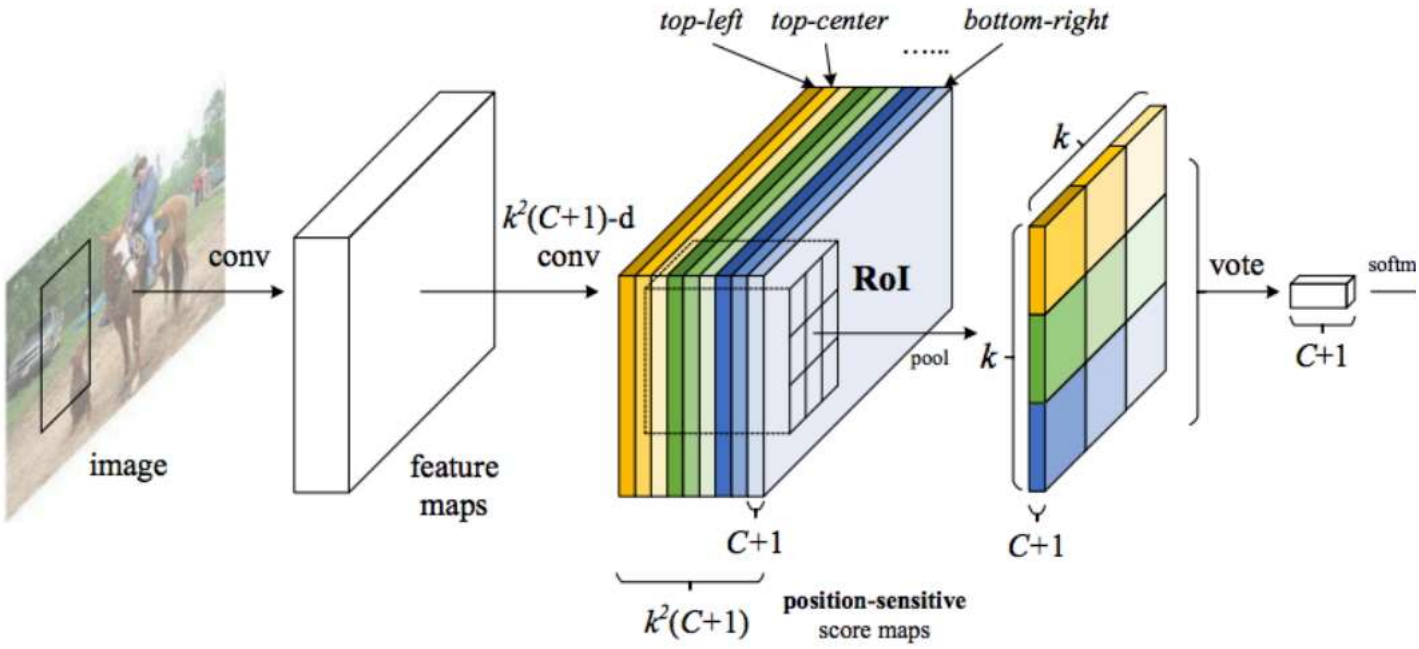



图4 Position-Sensitive RoI Pooling解析

图4已经很明显的画出了池化的方式，对于每个类别，它都有K*K个score maps，那么按照上述的池化方式，ROI可以针对该类别可以获得K*K个值，那么(C+1)个类别，那么一个RoI就可以得到K*K*(C+1)个值，就是上图的特征图。**那么对于每个类别，该类别的K*K个值都表示该RoI属于该类别的响应值，那K*K个数相加就得到该类别的score**，那么一共有(C+1)个scores，那么在这(C+1)个数上面使用简单的**softmax函数**就可以得到各个类别的概率了（注意，需要使softmax分类器了，只需要使用简答的softmax函数，因为这里就是通过简单的比大小来判断最终的类别的）。

**4. Position-Sensitive Regression解析**

前面的**position-sensitive score map和Position-sensitive RoI pooling**得到的值是用来分类的，那么自然需要相应的操作得到对应的值来进行回归操作position-sensitive score map和Position-sensitive RoI pooling思路，其会让每一个RoI得到(C+1)个数作为每个类别的score，那么现在每个RoI还需要 4个数回归偏移量，也就是x,y,w,h的偏移量，所以仿照分类设计的思想，我们还需要一个类似于position-sensitive score map的用于回归的score map。那么应该置这个score map呢，论文中给出了说明：**即在ResNet的共享卷积层的最后一层上面连接一个与position-sensitive score map并行的score maps，该sc maps用来进行regression操作，我们将其命名为regression score map，而该regression score map的维度应当是 4*K*K ，然后经过Position-sensiti pooling操作后，每一个RoI就能得到4个值作为该RoI的x,y,w,h的偏移量了，其思路和分类完全相同。**

**5. 为什么position-sensitive score map能够在含有某个类别的物体的某个部位的区域上具有高响应值?**

这种有高响应值现在只是作者自己设想的啊，如果网络不满足这一点的话，那么我们前面的所有分析都不成立啦。现在我们就大致解释一下为什么**训练该RoI够让网络最终满足这一点。**首先根据网络的loss计算公式，如果一个RoI含有人这个物体，那么该RoI通过position-sensitive score map和Position-sensitive pooling得到的(C+1)个值中属于人的那个值必然会在softmax损失函数的驱动下变得尽量的大，那么如何才能使得属于人的这个值尽量的大呢？那么我们需属于人的这个预测值是怎么来的？经过前面的分析，我们已经知道它是通过Position-sensitive RoI pooling这种池化操作获得的，那么也就是说**使得(C+1)个属于人的那个值尽量大，必然会使得position-sensitive score map中属于人的那个score map上的RoI对应的位置区域的平均值尽量大，从而会使得该sc map上在该区域上的响应值尽量大，因为只有该区域的响应值大了，才能使得预测为人的概率大，才会降低softmax的loss，整个训练过程才能进行下去。**

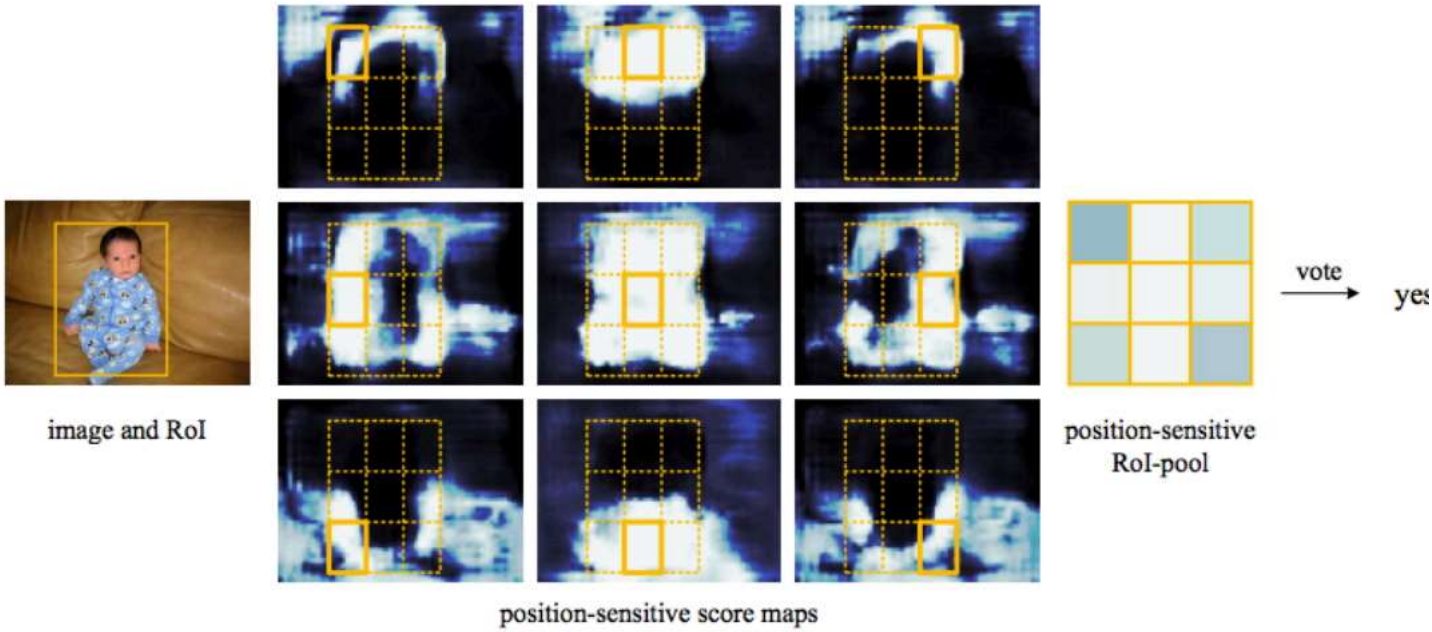



Figure 3: Visualization of R-FCN ($k \times k = 3 \times 3$) for the *person* category.

图5 位置敏感得分映射表现1

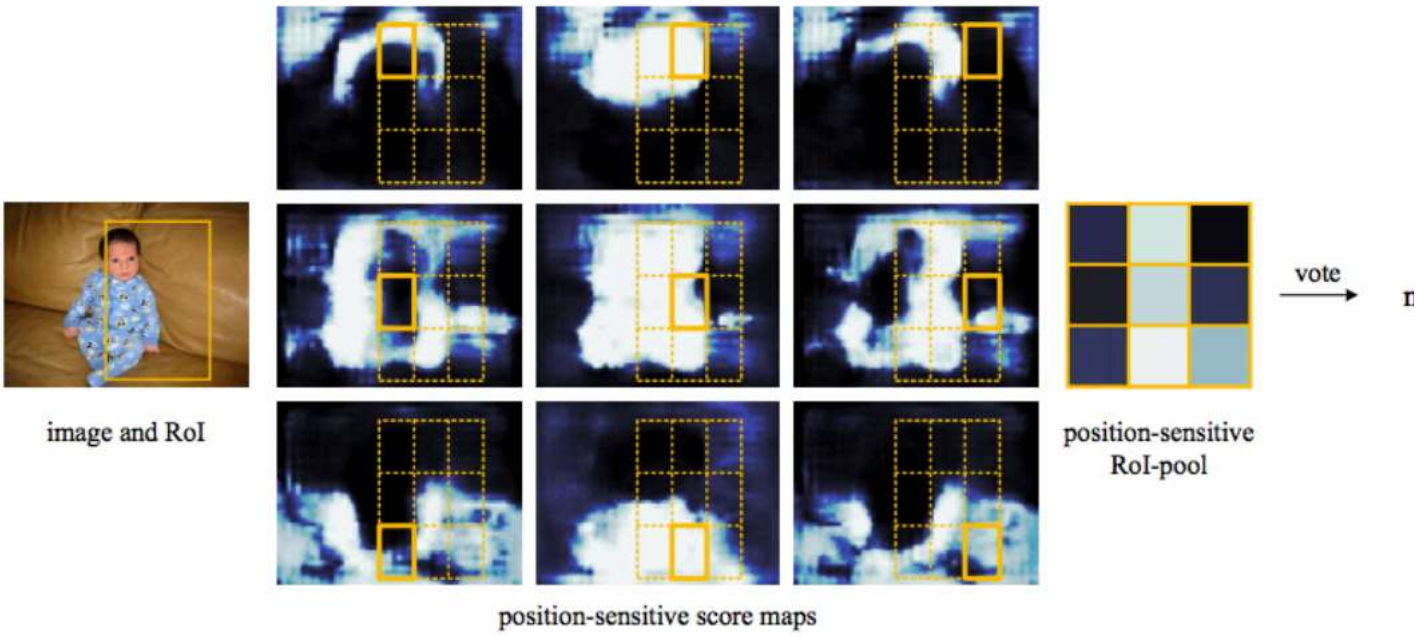



Figure 4: Visualization when an RoI does not correctly overlap the object.

图6 位置敏感得分映射表现2

如图5和图6所示，我们同样可以得出以上的结论。如图5所示，我们输入了一张含有一个小孩的图片，图中黄色的BB表示我们的检测到的目标，也就是我
个ROI，接下来是9张位置敏感的得分映射图（在这里使用的是3x3的特征映射），这9张图分别表示对人这个目标的top-left、top-center、... bottom-right不
域敏感的得分映射。对应到图中就是将这个ROI分为9个子区域，每一个子区域其实大致上对应到了小孩的不同部位，而不同的部位一般都会有其独特的特
在，9个区域敏感得分映射图对不同的区域比较敏感（所谓的敏感就是说如果这个子区域中存在该目标的某个部位特征时，其才会输出较大的响应值，否则
我会输出较小的响应值）。图5中的9个得分映射对ROI中划分的对应子区域都比较敏感（都有很强的响应值，越白表示响应越大，越黑表示响应越小），
中的9个子区域都有较大的响应值。然后进行位置敏感池化操作，最后进行Vote操作，由于9个区域中基本上都有很高的响应值，最后投票通过，认为这个
的对象是一个person。同理，可以得出图6是一个背景类。（图6的位置敏感ROI池化中有5个区域是黑色的，即表示具有较低的响应值，只有4个区域比较
表示具有较高的响应值，根据Vote机制，就将其分类为背景类）。

**6. Loss计算及其分析**

$$L(s, t_{x,y,w,h}) = L_{cls}(s_{c^*}) + \lambda[c^* > 0]L_{reg}(t, t^*).$$

这个Loss就是两阶段目标检测框架常用的形式。包括一个分类Loss和一个回归Loss。lamdy用来平衡两者的重要性。**对于任意一个RoI，我们需要计算它的softmax损失，和当其不属于背景时的回归损失。**这很简单，因为每个RoI都被指定属于某一个GT box或者属于背景，即先选择和GT box具有最大重叠率（IOU）的RoI，然后在剩余的RoI中选择与GT box的重叠率值大于0.5RoI进行匹配操作，最后将剩余的RoI都归为背景类。即每个RoI都有了对应的标签，我可以根据监督学习常用的方法来训练它啦。

**7. online hard example mining**

这个方法是目标检测框架中经常会用到的一个tricks，其主要的思路如下所示：首先对RPN获得的候选ROI（正负样本分别进行排序）进行排序操作；然后正样本（目标）的ROI中选择前N个ROI，将正负样本的比例维持在1:3的范围内，基本上保证每次抽取的样本中都会含有一定的正样本，都可以通过训练来网络的分类能力。如果不进行此操作的话，很可能会出现抽取的所有样本都是负样本（背景）的情况，这样让网络学习这些负样本，会影响网络的性能。全是我个人的理解，哈哈哈）

**8. Atrous algorithm（Dilated Convolutions或者膨胀卷积）**

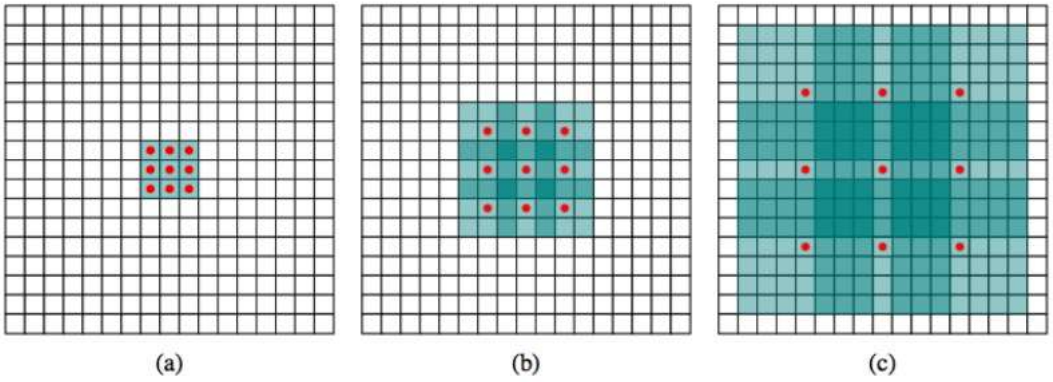



Figure 1: Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage. (a) $F_1$ is produced from $F_0$ by a 1-dilated convolution; each element in $F_1$ has a receptive field of $3 \times 3$. (b) $F_2$ is produced from $F_1$ by a 2-dilated convolution; each element in $F_2$ has a receptive field of $7 \times 7$. (c) $F_3$ is produced from $F_2$ by a 4-dilated convolution; each element in $F_3$ has a receptive field of $15 \times 15$. The number of parameters associated with each layer is identical. The receptive field grows exponentially while the number of parameters grows linearly.

图7 膨胀卷积

这个方法同样也是目标检测中常用的一个tricks，其最主要的目的是可以在减小卷积步长的同时扩大feature map的大小，即同等情况下，通过这个操作，我以获得一个更大的feature map，而实验表明，大的feature map会提升检测的性能。具体的解释可以去看这个链接。上图是一个膨胀卷积的操作，通过几次作，我们可以看到我们的接收场在不断的扩大，具体的解释请看英文吧。

**9. 为了过滤背景RoIs使用的方法**

在测试的时候，为了减少RoIs的数量，作者在RPN提取阶段就对RPN提取的大约2W个proposals进行了过滤，方法如下所示，

- 去除超过图像边界的proposals；
- 使用基于类别概率且阈值IoU=0.7的NMS过滤；
- 按照类别概率选择top-N个proposals；

所以在测试的时候，最后一般只剩下300左右个RoIs，当然这个数量是一个超参数。并且在R-FCN的输出300个预测框之后，仍然要对其使用NMS去除冗余测框。

**10. 训练细节**

R-FCN和Faster R-CNN采取了同样的训练策略，具体的训练策略可以参考这篇博客。

**11. 图片中的ROI和特征上的ROI之间的映射关系**

如果你不清楚它们是如何映射的，请查看这个链接。

**三、R-FCN性能分析**

**1. 定量结果分析**

## Comparisons among "almost" fully convolutional strategies using ResNet-101

| method | RoI output size ($k \times k$) | mAP on VOC 07 (%) |
|---|---|---|
| naïve Faster R-CNN | $1 \times 1$ | 61.7 |
| | $7 \times 7$ | 68.9 |
| class-specific RPN | - | 67.6 |
| R-FCN (w/o position-sensitivity) | $1 \times 1$ | *fail* |
| R-FCN | $3 \times 3$ | 75.5 |
| | $7 \times 7$ | **76.6** |

表1 使用ResNet-101全卷积策略

如上表所示，作者测试了不同大小的ROI对性能的影响（我们使用了预训练的ResNet-101网络，在VOC 07数据集上面进行测试），我们可以看到如果使用ROI，显示输出失败，具体原因不得而知。当使用7x7的ROI时，能够获得最好的结果，这也是论文中最终使用7x7大小的ROI的原因吧，作者应该是做了很验证工作。

## Comparisons between Faster R-CNN and R-FCN using ResNet-101

| | depth of per-RoI subnetwork | training w/ OHEM? | train time (sec/img) | test time (sec/img) | mAP (%) on VOC07 |
|---|---|---|---|---|---|
| Faster R-CNN | 10 | | 1.2 | 0.42 | 76.4 |
| **R-FCN** | 0 | | 0.45 | 0.17 | 76.6 |
| Faster R-CNN | 10 | ✓ (300 RoIs) | 1.5 | 0.42 | 79.3 |
| **R-FCN** | 0 | ✓ (300 RoIs) | 0.45 | 0.17 | **79.5** |
| Faster R-CNN | 10 | ✓ (2000 RoIs) | 2.9 | 0.42 | *N/A* |
| **R-FCN** | 0 | ✓ (2000 RoIs) | 0.46 | 0.17 | 79.3 |

表2 Faster R-CNN与R-FCN性能比较

如上表所示，我们比较了Faster R-CNN和R-FCN的性能，从表中我们可以看出与Faster R-CNN相比，R-FCN有更快的运行速度，大概是2.5倍以上。另外可以发现性能稍微有一点点提升，当调整ROI的个数时，我们发现300个ROI时能够获得最好的性能。

## On the Impact of Depth

| | training data | test data | VGG-16 | ResNet-50 | ResNet-101 | ResNet-152 |
|---|---|---|---|---|---|---|
| R-FCN | 07+12 | 07 | 75.6 | 77.0 | 79.5 | 79.6 |
| R-FCN multi-sc train | 07+12 | 07 | 76.5 | 78.7 | 80.5 | 80.4 |

表3 预训练网络的深度对性能的影响

如上表所示，随着预训练网络层数的加深，我们的检测性能在不断的得到提高，使用VGG和ResNet网络还是有很大的性能差异，但是过深的网络并没有提性能，可能的原因是我们的网络发生了过拟合情况。

## Experiments on COCO using ResNet-101

| | training data | test data | AP@0.5 | AP | test time (sec/img) |
|---|---|---|---|---|---|
| Faster R-CNN | train | val | 48.4 | 27.2 | 0.42 |
| **R-FCN** | train | val | **50.2** | **29.0** | 0.17 |

表4 COCO数据集的训练结果

如上表所示，我们采用了COCO数据集进行性能验证，与Faster R-CNN相比，R-FCN可以实现3倍的加速，准确率可以提升2个百分点。

**2. 定性结果分析**

图8 VOC 2007检测结果

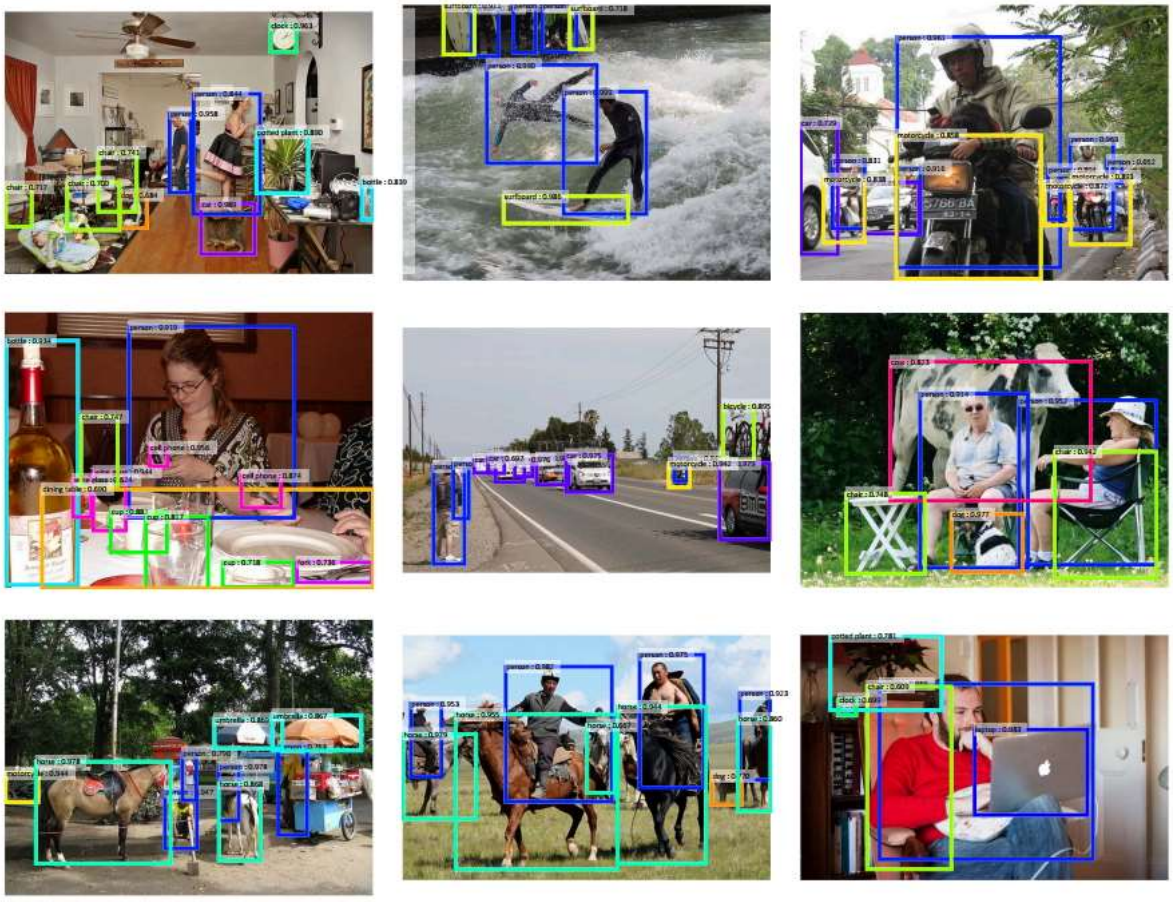


图9 COCO检测结果

以上是R-FCN算法在VOC2007和COCO数据集上面的性能表现，总体上看效果还是挺不错的，具体的效果需要你自己去尝试，根据自己的需求去选择合适的算法。

## 四、总结

**总的来讲，和Faster R-CNN相比，R-FCN具有更快的运行速度（2.5倍以上），稍微提高了一点检测精度，在速度和准确率之间进行了折中，提出position sensitive score map来解决检测的位置敏感性问题。算法中的很多细节值得我们进行深入的研究和分析，希望你从中学到了很多有用的东西。**

## 参考文献：

[1] R-FCN对应的poster，相关链接；

[2] VGG Reading Group - Sam Albanie ， 参考链接，密码：hby1；

[3] 详解R-FCN，博客链接；

py-R-FCN的预训练模型
**Py-R-FCN预训练模型详解** Py-R-FCN（Python实现的Region-based Fully Convolutional Networks）是一种用于目标检测的深度学习框架，它基于FCN（全卷积网络）并结合

R-FCN源代码
**R-FCN源代码详解** R-FCN（Region-based Fully Convolutional Networks）是一种在目标检测领域广泛应用的深度学习模型，它结合了区域提议网络（Region Proposal Netw

R-FCN检测原理
R-FCN目的是减小Faster-RCNN中detection head的检测时间:Faster-RCNN使用RoI pooling得到RoI features后,还需要使用由多层全连接层构成的detection head做分类和回归,而

目标检测——R-FCN算法解读
代码:https://github.com/daijifeng001/r-fcn 文章目录 1、算法概述 2、R-FCN细节 3、实验结果 1、算法概述 之前基于区域的目标检测方法,像Fast/Faster R-CNN虽然在提取区域

Face Paper: R-FCN论文详解 BigCowPeking
本篇博客一方面介绍R-FCN算法（NISP2016文章），该算法改进了Faster RCNN，另一方面介绍其Caffe代码，这样对算法的认识会更加深入。论文：R-FCN：object detection

R-FCN：基于区域的全卷积网络来检测物体 **热门推荐** 灰巧克力爱松露
CVPR 2016 阅读~ 原文标题为"R-FCN: Object Detection via Region-based Fully Convolutional Networks "。

R-FCN Duino的工具箱
R-FCN

Mask R-CNN详解：实例分割与FCN、FPN、ROIAlign的应用
"Mask R-CNN是一种用于实例分割的深度学习模型，由FCN发展而来，结合了ResNet和特征金字塔网络（FPN），并引入了ROIAlign技术以提高小目标分割的准确性。该模型在