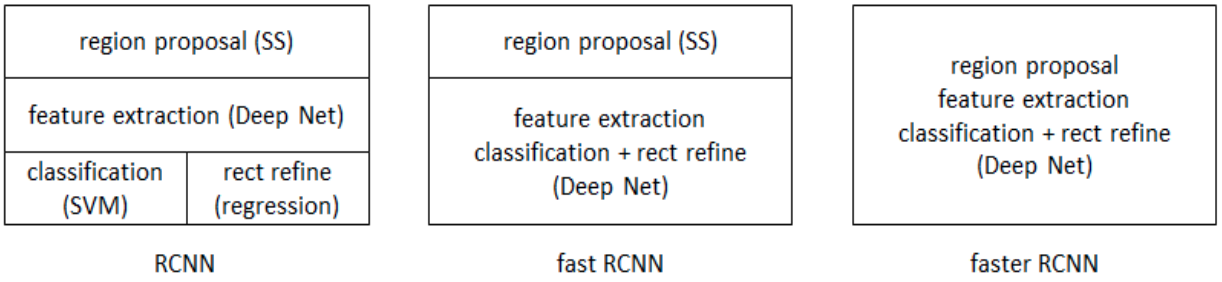
# Faster-RCNN论文阅读

## 前言

fast_rcnn还有一个缺点就是它的roi的提取还是采用的ss方法,需要在网络外独立提取,而Faster-RCNN提出了A Region Proposal Network (RPN) 解决region提取.




---

## 1.Region Proposal Network (RPN)原理

RPN网络用于生成region proposals。该层通过softmax判断anchors属于foreground或者background，再利用bounding box regression修正anchors获得精确的proposals。
网络的整体架构就是在fast的基础上在conv和ROI pooling层之间加了一条Region Proposal Network (RPN)支路.

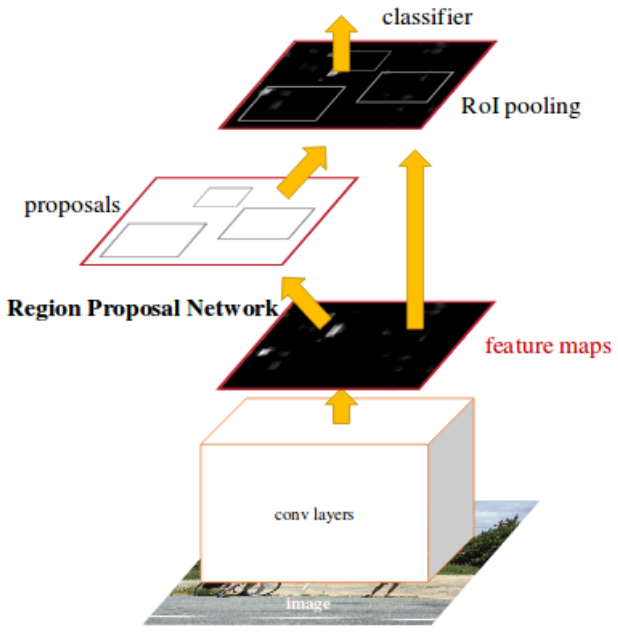



Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

rpn实际上也是一种fully convolutional network.

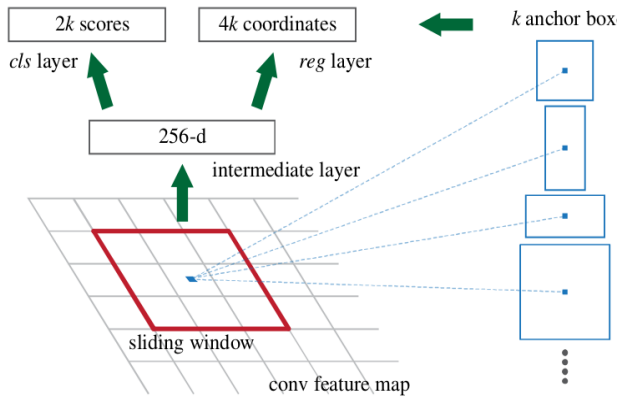
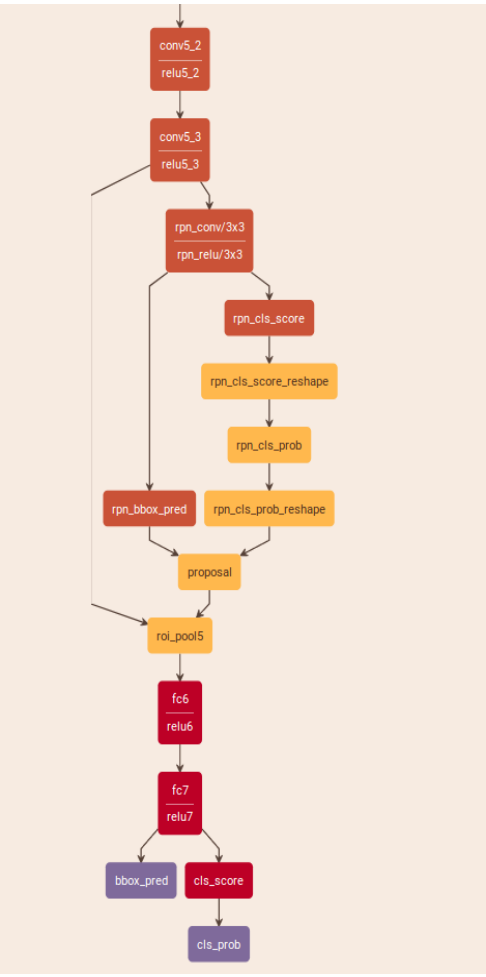



Figure 3: **Left**: Region Proposal Network (RPN). **Right**: E

强烈吐槽paper中的rpn结构图,上图云里雾里描述anchor和3*3卷积核.
直接看[py_faster_rcnn/model/pascal_voc/vgg16/faster_rcnn_alt_opt]网络,
下图是我截取rpn和后续的fc.输入的conv5_3就是vgg16 conv层的feature map.这里的map就是RPN和ROIpooling共享的.




上图中中间的三条支路的右边两条就是rpn网络.

### 1.1 rpn_conv层

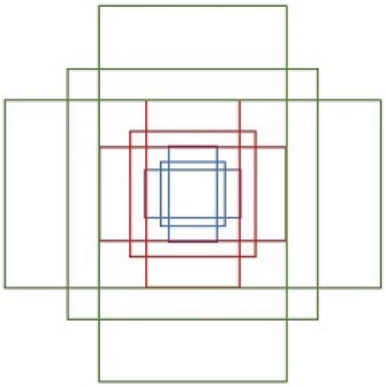
这一层是作者加上去的,采用的3*3的卷积核,固定输出512维的feature map.为了增加感受野,降维和鲁棒性.

```
layer {
  name: "conv5_3"
  type: "Convolution"
  bottom: "conv5_2"
  top: "conv5_3"
  convolution_param {
    num_output: 512
    pad: 1 kernel_size: 3
```

```
  }
}
```

### 1.2 rpn_bbox_pred层

这一层输出中map中的点对应anchor,anchor就是一些不同size和长宽比例的窗口,paper中用了三种size和三种长宽



(1:1,1:2,2:1),那么就有9种anchor如图:

```
layer {
  name: "rpn_bbox_pred"
  type: "Convolution"
  bottom: "rpn/output"
  top: "rpn_bbox_pred"
  convolution_param {
    num_output: 36   # 4 * 9(anchors)
    kernel_size: 1 pad: 0 stride: 1
  }
}
```

rpn_bbox_pred的输出是36维的,每个对应点的是9个anchor的[x,y,w,h],4*9=36,刚好是对应36维的向量.[x,y,w,h]分别是对应anchor的左上角坐标,和长宽.这一层就是学习9个anchor,定位比较粗糙,但是后面的fc层有Bbox回归,所以只需要大概在哪里就可以..mapx中每个点都有9个anchor似乎会很多anchor,其实有些边缘点的anchor大小已经超出了图像的大小的,是会被舍弃的.后面也不会把所有的anchor送入分类.会选取一部分的.

### 1.3rpn_cls层

分类支路是判断每个anchor是前景还是背景.
做了1x1卷积,num_output=18，对应9个anchors，每个anchors又有可能是foreground和background.
卷积完成后来了一个reshape layer,之后softmax分类,然后有reshape layer
reshape layer其实只是为了便于softmax分类，至于具体原因这就要从caffe的实现形式说起了。在caffe基本数据结构blob中以如下形式保存数据：
blob=[batch_size, channel，height，width]
对应至上面的保存bg/fg anchors的矩阵，其在caffe blob中的存储形式为[1, 2*9, H, W]。而在softmax分类时需要进行fg/bg二分类，所以reshape layer会将其变为[1, 2, 9*H, W]大小，即单独“腾空”出来一个维度以便softmax分类，之后再reshape回复原状。

### 1.4Proposal layer

Proposal Layer负责综合所有[dx(A)，dy(A)，dw(A)，dh(A)]变换量和foreground anchors，计算出精准的proposal，送入后续RoI Pooling Layer

```
    layer {
      name: 'proposal'
      type: 'Python'
      bottom: 'rpn_cls_prob_reshape'
      bottom: 'rpn_bbox_pred'
      bottom: 'im_info'
      top: 'rois'
      python_param {
        module: 'rpn.proposal_layer'
        layer: 'ProposalLayer'
        param_str: "'feat_stride': 16"
      }
```

```
}
```

### 1.5 loss 函数

$$L(\{p_i, t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \frac{1}{N_{reg}} \sum_i P_i^* L_{reg}(t_i, t_i^*)$$

Here, i is the index of an anchor in a mini-batch and p i is the predictedprobability of anchor i being an object. The ground-truth label p ∗ i is 1 if the anchor is positive, and is 0 if the anchor is negative. t i is a vector representing the 4 parameterized coordinates of the predicted bounding box, and t ∗ i is that of the ground-truth box associated with a positive anchor. The classification loss L cls is log loss over two classes (object vs. not object). For the regression loss, we use L reg (t i , t ∗ i ) = R(t i − t ∗ i ) where R is the robust loss function (smooth L 1 ) defined in [2]. The term p ∗ i L reg means the regression loss is activated only for positive anchors (p ∗ i = 1) and is disabled otherwise (p ∗ i = 0). The outputs of the cls and reg layers consist of {p i } and {t i } respectively.

## 2 后续fc层

后面的fc层就是fast_rcnn中的bbox回归和sofmax分类.

## 3 训练

Faster CNN的训练，是在已经训练好的model（如VGG_CNN_M_1024，VGG，ZF）的基础上继续进行训练。实际中训练过程分为6个步骤：

- 在已经训练好的model上，训练RPN网络，对应stage1_rpn_train.pt
  - 利用步骤1中训练好的RPN网络，收集proposals，对应rpn_test.pt
  - 第一次训练Fast RCNN网络，对应stage1_fast_rcnn_train.pt
  - 第二训练RPN网络，对应stage2_rpn_train.pt
  - 再次利用步骤4中训练好的RPN网络，收集proposals，对应rpn_test.pt
  - 第二次训练Fast RCNN网络，对应stage2_fast_rcnn_train.pt

paper中是训练两遍,再增加次数好像也没有效果.

## 4 结果

Detection results on PASCAL VOC 2007 test set (trained on VOC 2007 trainval)

| train-time region proposals | | test-time region proposals | | |
|---|---|---|---|---|
| method | # boxes | method | # proposals | mAP (%) |
| SS | 2000 | SS | 2000 | 58.7 |
| EB | 2000 | EB | 2000 | 58.6 |
| RPN+ZF, shared | 2000 | RPN+ZF, shared | 300 | **59.9** |
| *ablation experiments follow below* | | | | |
| RPN+ZF, unshared | 2000 | RPN+ZF, unshared | 300 | 58.7 |
| SS | 2000 | RPN+ZF | 100 | 55.1 |
| SS | 2000 | RPN+ZF | 300 | 56.8 |
| SS | 2000 | RPN+ZF | 1000 | 56.3 |
| SS | 2000 | RPN+ZF (no NMS) | 6000 | 55.2 |
| SS | 2000 | RPN+ZF (no *cls*) | 100 | 44.6 |
| SS | 2000 | RPN+ZF (no *cls*) | 300 | 51.4 |
| SS | 2000 | RPN+ZF (no *cls*) | 1000 | 55.8 |
| SS | 2000 | RPN+ZF (no *reg*) | 300 | 52.1 |
| SS | 2000 | RPN+ZF (no *reg*) | 1000 | 51.3 |
| SS | 2000 | RPN+VGG | 300 | 59.2 |

对比

Table 6: Results on PASCAL VOC 2007 test set with Fast R-CNN detectors and VGG-16. For RPN, the train-time proposals for Fast R-CNN are 2000. RPN* denotes the unsharing feature version.

| method | # box | data | mAP | areo | bike | bird | boat | bottle | bus | car | cat | chair | cow | table | dog | horse | mbike | person | plant | sheep | sofa | train | tv |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SS | 2000 | 07 | 66.9 | 74.5 | 78.3 | 69.2 | 53.2 | 36.6 | 77.3 | 78.2 | 82.0 | 40.7 | 72.7 | 67.9 | 79.6 | 79.2 | 73.0 | 69.0 | 30.1 | 65.4 | 70.2 | 75.8 | 65.8 |
| SS | 2000 | 07+12 | 70.0 | 77.0 | 78.1 | 69.3 | 59.4 | 38.3 | 81.6 | 78.6 | 86.7 | 42.8 | 78.8 | 68.9 | 84.7 | 82.0 | 76.6 | 69.9 | 31.8 | 70.1 | 74.8 | 80.4 | 70.4 |
| RPN* | 300 | 07 | 68.5 | 74.1 | 77.2 | 67.7 | 53.9 | 51.0 | 75.1 | 79.2 | 78.9 | 50.7 | 78.0 | 61.1 | 79.1 | 81.9 | 72.2 | 75.9 | 37.2 | 71.4 | 62.5 | 77.4 | 66.4 |
| RPN | 300 | 07 | 69.9 | 70.0 | 80.6 | 70.1 | 57.3 | 49.9 | 78.2 | 80.4 | 82.0 | 52.2 | 75.3 | 67.2 | 80.3 | 79.8 | 75.0 | 76.3 | 39.1 | 68.3 | 67.3 | 81.1 | 67.6 |
| RPN | 300 | 07+12 | 73.2 | 76.5 | 79.0 | 70.9 | 65.5 | 52.1 | 83.1 | 84.7 | 86.4 | 52.0 | 81.9 | 65.7 | 84.8 | 84.6 | 77.5 | 76.7 | 38.8 | 73.6 | 73.9 | 83.0 | 72.6 |
| RPN | 300 | COCO+07+12 | **78.8** | **84.3** | **82.0** | **77.7** | **68.9** | **65.7** | **88.1** | **88.4** | **88.9** | **63.6** | **86.3** | **70.8** | **85.9** | **87.6** | **80.1** | **82.3** | **53.6** | **80.4** | **75.8** | **86.6** | **78.9** |

Table 7: Results on PASCAL VOC 2012 test set with Fast R-CNN detectors and VGG-16. For RPN, the train-time proposals for Fast R-CNN are 2000.

| method | # box | data | mAP | areo | bike | bird | boat | bottle | bus | car | cat | chair | cow | table | dog | horse | mbike | person | plant | sheep | sofa | train | tv |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SS | 2000 | 12 | 65.7 | 80.3 | 74.7 | 66.9 | 46.9 | 37.7 | 73.9 | 68.6 | 87.7 | 41.7 | 71.1 | 51.1 | 86.0 | 77.8 | 79.8 | 69.8 | 32.1 | 65.5 | 63.8 | 76.4 | 61.7 |
| SS | 2000 | 07++12 | 68.4 | 82.3 | 78.4 | 70.8 | 52.3 | 38.7 | 77.8 | 71.6 | 89.3 | 44.2 | 73.0 | 55.0 | **87.5** | 80.5 | 80.8 | 72.0 | 35.1 | 68.3 | **65.7** | 80.4 | 64.2 |
| RPN | 300 | 12 | 67.0 | 82.3 | 76.4 | 71.0 | 48.4 | 45.2 | 72.1 | 72.3 | 87.3 | 42.2 | 73.7 | 50.0 | 86.8 | 78.7 | 78.4 | 77.4 | 34.5 | 70.1 | 57.1 | 77.1 | 58.9 |
| RPN | 300 | 07++12 | 70.4 | 84.9 | 79.8 | 74.3 | 53.9 | 49.8 | 77.5 | 75.9 | 88.5 | 45.6 | 77.1 | 55.3 | 86.9 | 81.7 | 80.9 | 79.6 | 40.1 | 72.6 | 60.9 | 81.2 | 61.5 |
| RPN | 300 | COCO+07++12 | **75.9** | **87.4** | **83.6** | **76.8** | **62.9** | **59.6** | **81.9** | **82.0** | **91.3** | **54.9** | **82.6** | **59.0** | **89.0** | **85.5** | **84.7** | **84.1** | **52.2** | **78.9** | 65.5 | **85.4** | **70.2** |

---

**目录——-KAM_FANG**




参考资料:
1.Faster-RCNN论文.:https://arxiv.org/abs/1506.01497
2.Faster-RCNN代码:https://github.com/rbgirshick/py-faster-rcnn

2.fasterrcnn详解:http://blog.csdn.net/zy1034092330/article/details/62044941