Region Proposal Network — A detailed view

What are anchors? How can RPN learn from feature maps to generate boxes? How does it cover boxes of all shapes?

Sambasivarao. K Follow
Dec 21, 2019 · 4 min read ★

If you are aware of the R-CNN family for object detection, you might have heard the term “RPN”, which is a region proposal network. If you don’t know about the R-CNN family of detectors, I recommend you go through this article before delving deep into RPN.

We all have a vague idea that Region Proposal Network is used to generate proposals for object detection in faster-rcnn. We also heard that it does that by learning from feature maps obtained from a base network (VGG16, ResNet, etc.,). Many people have this vague idea, but very few have a thorough understanding of how it works. I used to have many doubts. How can RPN learn from feature maps to generate boxes? How does it generate boxes at image level with feature maps at different spatial levels? What are anchors? How does it cover boxes of all shapes? Won’t there be any missing box for my object? How much recall does it have? What if my box is very small? These are some of the questions I used to have. Going through paper and many blogs have clarified some of the doubts, then I thought of going through the code implementation to get a much clear idea and it helped! Let’s get into it.

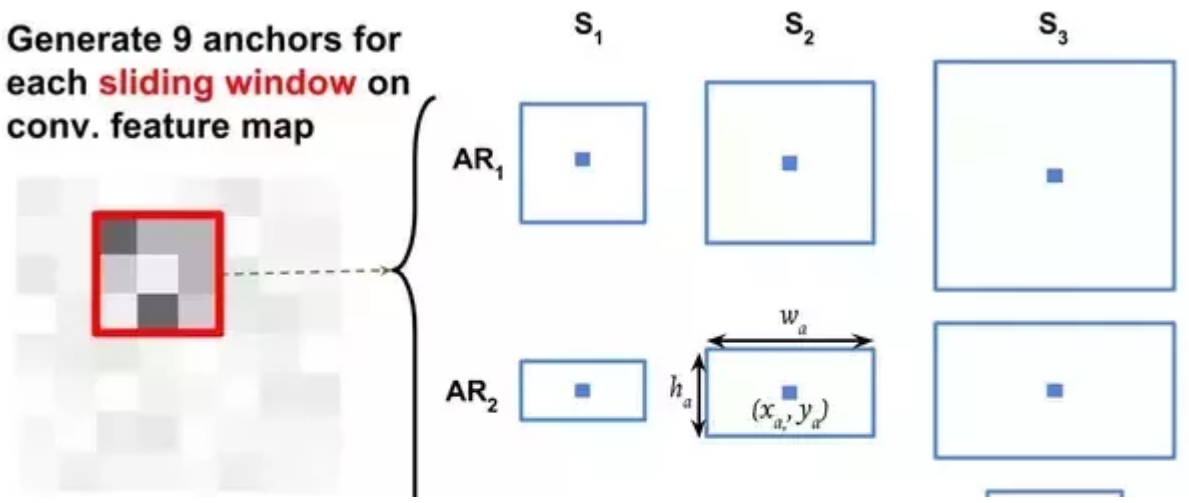
Object Detection approach: We generate the candidate boxes (which might have our objects to detect) and classify those boxes as one of the objects like cat/dog/person etc.. That is classification. At the same time, these boxes shape adjustments learn to properly fit the actual object. That is bounding box regression.

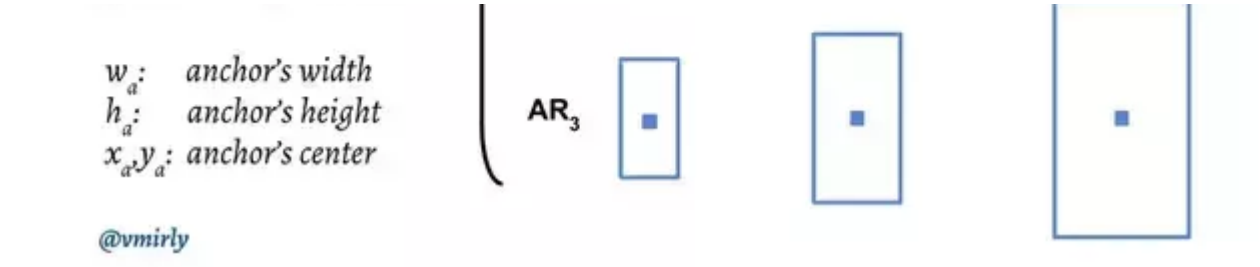
Now the first step, which is the generation of candidate boxes, is done by RPN. In the early versions of object detectors, this proposal generation happens offline by traditional computer vision techniques. One such approach is selective search. The drawbacks of these approaches are computation cost and also offline computation.

RPN came to rescue by doing this in very less time and also it can be merged to any object detection network which makes it useful for end-to-end training. Just like how our CNNs learn classification from feature maps, it also learns the proposals from feature maps. Let me note down the steps in RPN:

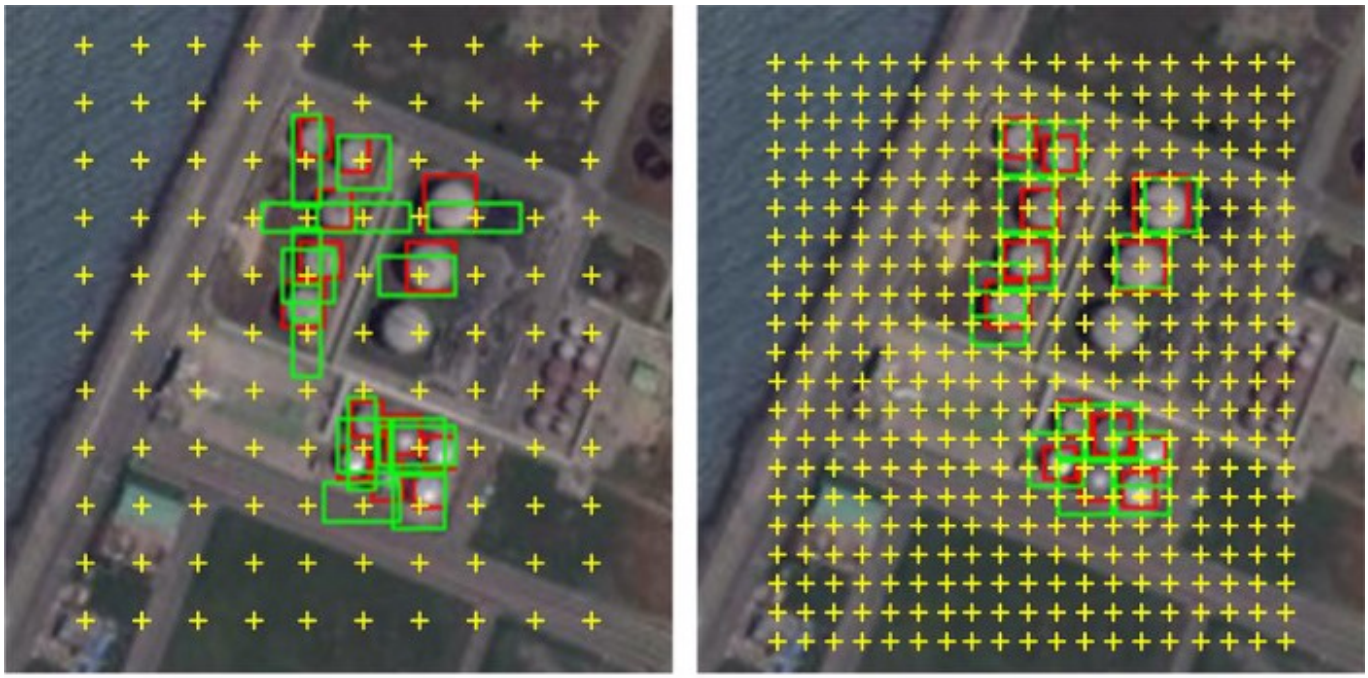
- Generate anchor boxes.
- Classify each anchor box whether it is foreground or background.
- Learn the shape offsets for anchor boxes to fit them for objects.

Anchor point: Every point in the feature map generated by backbone network is an anchor point. We need to generate anchor boxes for every anchor point. We generate candidate boxes using two parameters — scales and aspect ratios. The boxes need to be at image dimensions, whereas the feature map is reduced depending on the backbone. For example, in the case of vgg16, the image is reduced by 16 times by the end of the backbone. So how do we generate boxes at image dimensions? We use this 16 as the stride in generating anchor boxes at image level. (Ex: If anchor scales are [8,16,32] and ratios are [0.5,1,2] and stride is 16, then we use the combination of these scales and ratios to generate 9 anchor boxes for each anchor point and then take a stride of 16 over the image to take the next anchor box.)



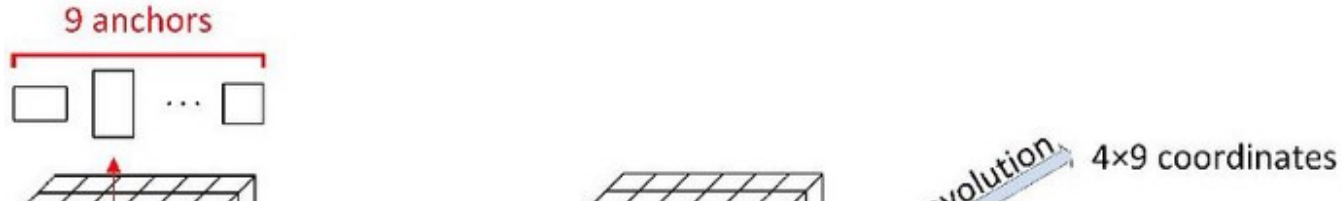


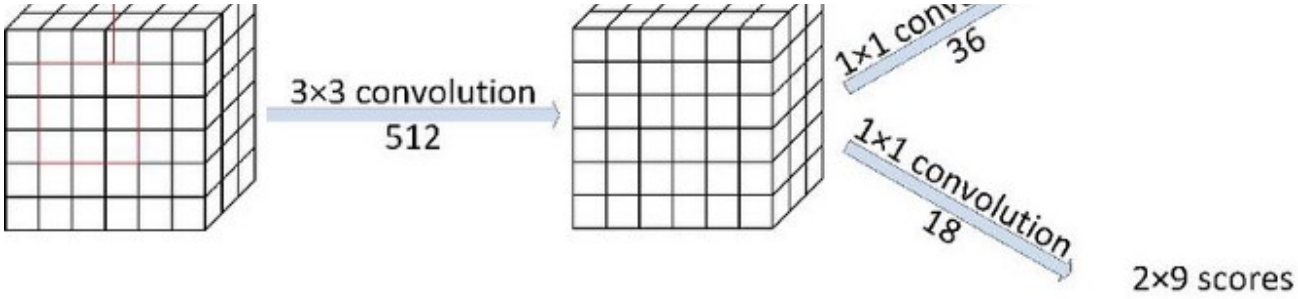
Anchor boxes generation



These correspond to anchor points at image level. 16 and 8 are strides used depending on backbone.

Now we have generated the anchor boxes, but these are dummy boxes which are different from actual object of interest. Also, there might be many boxes which are not having any object in it. So we need to learn whether the given box is foreground or background, at the same time we need to learn the offsets for the foreground boxes to adjust for fitting the objects. These two tasks are achieved by two convolution layers on the feature map obtained from backbone network. Those layers are rpn_cls_score and rpn_bbox_pred and the architecture looks like below.





For 9 anchors at every anchor point, 2 scores (fg and bg) and 4 offsets for coordinates.

We learn offsets for x,y,w,h values, where (x,y) is the center of the box, w and h are width and height. We learn these offsets are regression. For learning these scores and offsets, we need to have targets. These targets are generated by comparing the anchor boxes with ground truth boxes. This process is anchor target generation. In anchor target generation, we calculate the IOU of GT boxes with anchor boxes to check if it is fg/bg and then the difference in the coordinates are calculated as targets to be learned by regressor. Then these targets are used as input for cross-entropy loss and smooth l1 loss.

Once these fg/bg scores and offsets are learned using convolution layers, some portion of fg and bg boxes are considered according to confidence scores. The offsets are applied to those boxes to get the actual ROIs to be processed further. This post-processing of anchor boxes using offsets is called as proposal generation. These final proposals are propagated forward through ROI pooling layer and fc layers. You can refer to my previous posts to learn about ROI pooling and NMS (Non-maximum Suppression).

That's all for this post, hope you got a good understanding of how actually boxes are generated from RPN. You can refer to this code repository for understanding the code. let's meet later in another post soon.

Faster R Cnn Region Proposal Network Rpn Anchor Generation