Task: 08

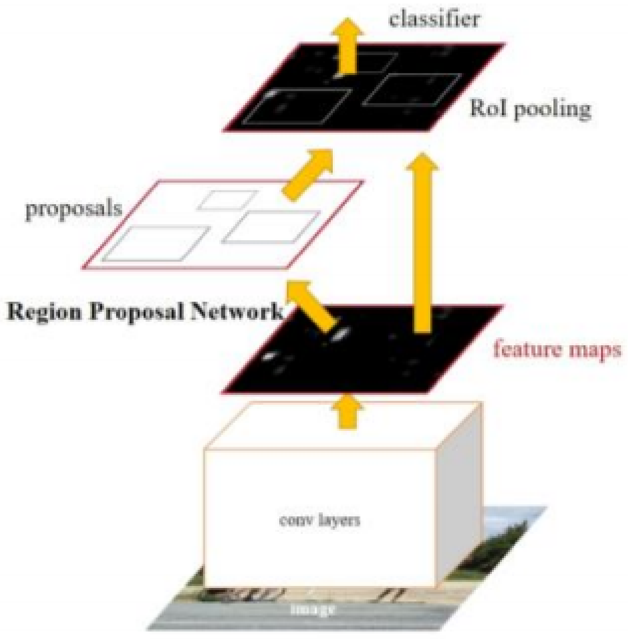
Faster R-CNN

Region based convolutional neural network

Faster R-CNN has two networks:

Region proposal network (RPN) for generating region proposals and a network using these proposals to detect objects. The main difference here with Fast R-CNN is that the later uses selective search to generate region proposals. The time cost of generating region proposals is much smaller in RPN than selective search, when RPN shares the most computation with the object detection network. Briefly, RPN ranks region boxes (called anchors) and proposes the ones most likely containing objects.

The architecture is as follows.



1. Region proposal network:

RPN takes an image as input and outputs a set of rectangular object proposals along with their objectness score. This is achieved by a fully convolutional network. The ultimate goal is to share computation with a Fast R-CNN object detection network. It’s assumed that both networks share a common set of convolutional layers. The paper on faster R-CNN uses the Zeiler and Fergus model with 5 shareable convolutional layers and the Simonyan and Zisserman model often called VGG-16, that consists of 13 shareable convolutional layers. For generation of region proposals, the small network is slided over the convolutional feature map which is obtained by the last shared convolutional layer.

2. Anchors:

Each sliding window simultaneously predicts multiple region proposals, where the number of maximum possible proposals for each location is denoted as. So the reg layer has 4k outputs encoding the coordinates of k boxes, and the cls layer outputs 2k scores that estimate probability of object or not object for each proposal4. The k proposals are parameterized relative to k reference boxes, which we call anchors. An anchor is centered at the sliding window in question, and is associated with a scale and aspect ratio

Loss function:

We assign a positive label to two kinds of anchors:

(i) The anchor/anchors with the highest Intersection-over-Union (IoU) overlap with a ground-truth box.

(ii)An anchor that has an IoU overlap higher than 0.7 with any ground-truth box. A single ground truth box may assign positive labels to multiple anchors.Usually the second condition is sufficient to determine the positive samples but we still adopt the first condition for the reason that in some rare cases the second condition may find no positive sample. We assign a negative label to a non-positive anchor if its IoU ratio is lower than 0.3 for all ground-truth boxes .Anchors that are neither positive nor negative do not contribute to the training objective.

$$\begin{aligned} L(\{p_i\},\{t_i\}) &= \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) \\ &+ \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*). \end{aligned} \qquad (1)$$

Here,

I: index of an anchor in a mini-batch

Pi: predicted probability of anchor I being an object.

P*: Ground truth label (1 if the anchor is positive, 0 otherwise).

Ti: a vector representing the 4 parameterized coordinates of the predicted bounding box
T* the ground truth vector box associated with a positive anchor.

Lcls: classification loss over two classes.

Lreg: regression loss

Feature sharing by RPN and R-CNN:

1. Alternating training:

In this solution, we first train RPN, and use the proposals to train Fast R- CNN. The network tuned by Fast R-CNN is then used to initialize RPN, and this process is iterated.

2. Approximate joint training:

In this solution, the RPN and Fast R-CNN networks are merged into one network during training as in Figure 2. In each SGD iteration, the forward pass generates region proposals which are treated just like fixed, precomputed proposals when training a Fast R-CNN detector. The backward propagation takes place as usual, where for the shared layers the backward propagated signals from both the RPN

Loss and the Fast R-CNN loss are combined. This solution is easy to implement. But this solution ignores the derivative w.r.t. the proposal boxes coordinates that are also network responses, so is approximate.

3. Non-approximate joint training:

The bounding boxes predicted by RPN are also functions of the input. The RoI pooling layer in Fast R-CNN accepts the convolutional features and also the predicted bounding boxes as input, so a theoretically valid back-propagation solver should also involve gradients w.r.t. the box coordinates. These gradients are ignored in the above approximate joint training. In a non-approximate joint training solution, we need an RoI pooling layer that is differentiable w.r.t. the box coordinates.

4. Step Alternating Training:

In the first step,we train the RPN. This network is initialized with an ImageNet pre-trained model and fine-tuned end to end for the region proposal task. In the second step, we train a separate detection network by Fast R-CNN using the proposals generated by the step-1 RPN. This detection net-work is also initialized by the ImageNet-pretrained model. At this point the two networks do not share convolutional layers. In the third step, we use the detector network to initialize RPN training, but we fix the shared convolutional layers and only fine tune the layers unique to RPN. Now the two networks share convolutional layers. Finally, keeping the shared convolutional layers fixed, we fine-tune the unique layers of Fast R-CNN.

Training

Faster RCNN architecture is a unified network composed of RPN and Fast RCNN and the CNN layers are shared by both architectures. We cannot train RPN and Fast RCNN separately(it will give different weights and thus we will need to pass CNN twice for each of them).

The authors used a 4-step training algorithm(some others were also discussed which you can see in the paper) discussed below:

1. RPN trained initially and pre-trained weights from image-net is used.

2. Fast RCNN is trained using proposals generated by step-1 RPN. (Two networks don’t share convolution layers at this point).

3. RPN is trained using convolution layers from step 2 and only layers unique to RPN are updated(weights of convolution layer is not updated)