Report

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun

Students:
Phạm Nguyễn Hoàng
Trần Văn Liên
Nguyễn Phạm Thiện Dũng

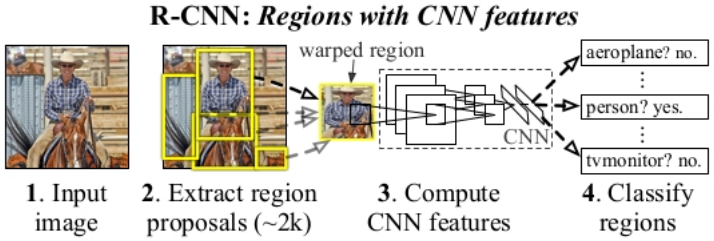
1 Problem

Object detection is a computer technology related to computer vision and image processing. The task of object detection is to detect instances of objects in certain classes such as humans, cars or animals [10]. Object detection has many applications and are widely used in many industries nowadays.

Input of the problem is the images which contain some objects to detect. Output is the labels of those objects and the bounding boxes for each object.

2 R-CNN

R-CNN or Slow R-CNN is the first architecture in the R-CNN serie. We review R-CNN shortly in order to focus on Fast R-CNN and Faster R-CNN. The architecture of R-CNN is created from region proposals and CNNs [2].

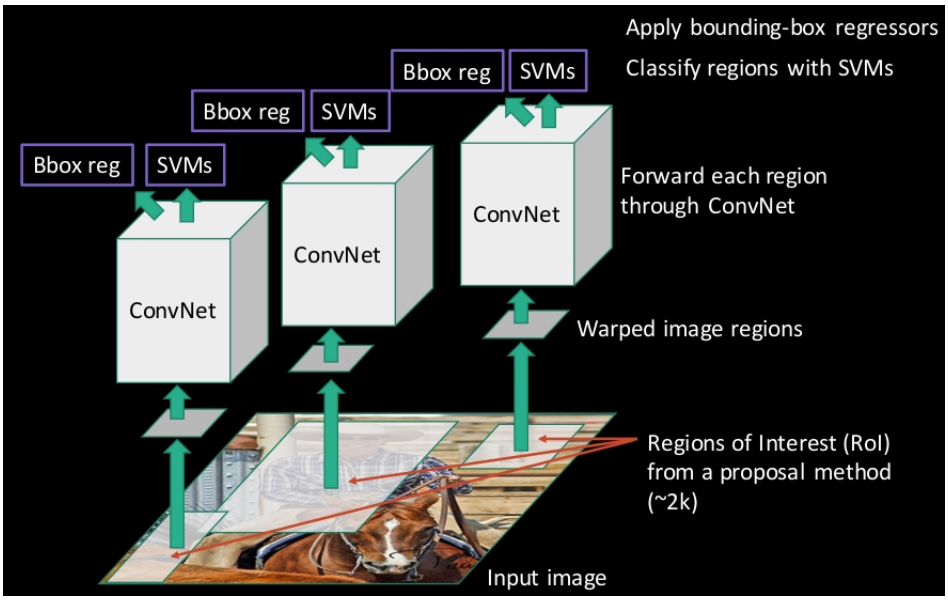


R-CNN **region proposals** is selective search. A feature vector is extracted from each region proposal by a CNN.

In test time, the selective search extract about 2000 region proposals per image. For each class, R-CNN extracts feature vector using a SVM which is trained for that class. After that, a greedy non-maximum suppression is applied to reject a region if it has an intersection-over-union (IoU) overlap with a higher scoring selected region than a learned threshold [2].

R-CNN is trained using the following steps [4]:

1. Pre-train a convolutional neural network for ImageNet classification.
2. Fine-tune the model from step 1 for object detection (using softmax classifier + log loss).
3. Cache those feature vectors from step 2 to disk.
4. Train linear SVMs on those feature vectors from step 3 (hinge loss).
5. Train bounding box regressors on those feature vectors from step 3 (squared loss).



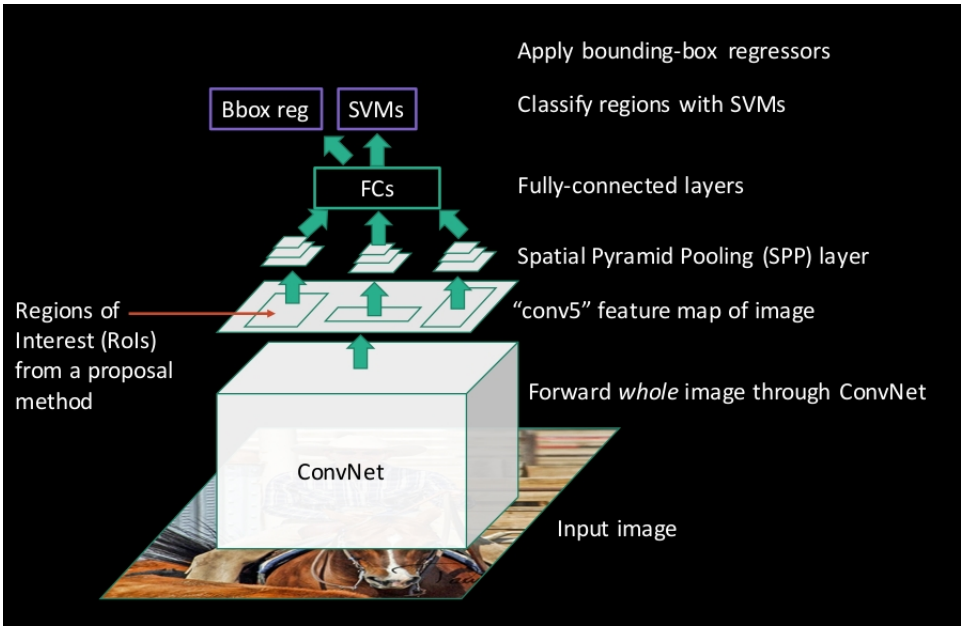
Each region proposal is processed through a convolution neural network and then classified by a SVM. This is considered as one of the reason that make R-CNN slow and inferior to Fast R-CNN.

R-CNN has some disadvantages [1, 3]:

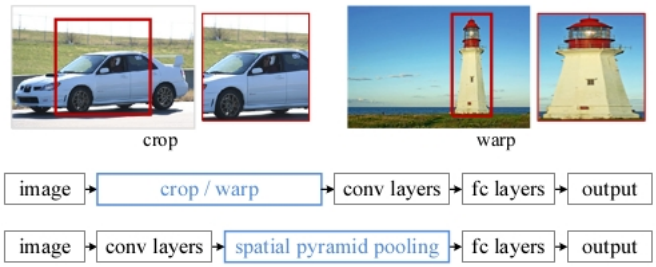
1. There are a lot of training procedures:
   - Fine-tune the network with softmax classifier.
   - Train linear SVMs for the features from the previous convolution neural network.
   - Train bounding box regressors.
2. Training is very slow and takes a lot of disk space.
3. Detection is slow (47s per image with VGG16).

3 SPPnet

Spatial Pyramid Pooling Net (SPPnet) fixes the previous drawback of R-CNN. R-CNN has to forward each region proposal to a separate CNN. SPPnet makes testing time faster by computing a convolutional feature map for the entire image and classify each object proposal using this feature map. Therefore, we only need to forward this feature map to only a single CNN.



SPPNet also introduce **spatial pyramid pooling** layer which remove teh fixed-size constraint of the network. A SPP layer is added on top of the last convolutional layer to pool the features and generate fixed-length output [5].



SPPnet is trained by these following steps with 2 slightly different steps when training R-CNN [4]:

1. Pre-train a convolutional neural network for ImageNet classification.
2. **Cache SPP features to disk using the model from step 1.**
3. **Train a new model by using the model from step 1 and fine-tune a 3-layer network from the features in step 2 (log loss).**
4. Cache those feature vectors from step 3 to disk.
5. Train linear SVMs on those feature vectors from step 3 (hinge loss).
6. Train bounding box regressors on those feature vectors from step 3 (squared loss).

The detail implementation of SPPnet can be see in [5]. SPPnet is a big improvement for R-CNN and got an amazing result and beat a lot of opponents at that time.

However, SPPnet is unable to update weights below the spatial pyramid pooling layer. Back-propagation through SPP layer is inefficient when each training sample comes from a different image. each RoI may have a very large field (the size is usually approximately the same as the entire image). Therefore, the training inputs are large and the layer has to process a lot of data [6].

4 Fast R-CNN

Fast R-CNN is an improvement to R-CNN and SPPnet in object classification. Fast R-CNN is 9 times faster than R-CNN and 3 times faster SPPnet. The detection network processes images in 0.3s (excluding object proposal time) and get a mAP of 66% (R-CNN has a mAP of 62%) [1].

R-CNN and SPPnet have certain drawbacks which are stated in previous sections.

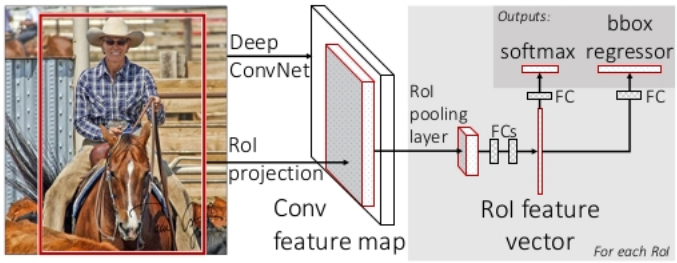
- R-CNN is slow because it has to perform a ConvNet forward for each object proposal [1].
- SPPnet fixes this drawback of R-CNN by computing a convolutional feature map for the entire image and classify each object proposal using this feature map. The training and testing speed are much faster than R-CNN. However, SPPnet has the following disadvantages:
  - There are also a lot of training procedures.
  - Training also takes lots of disk space.
  - Fine-tuning algorithm cannot update the convolutional layers that precede the spatial pyramid pooling.

Fast R-CNN has the following advantages over these 2 architectures:

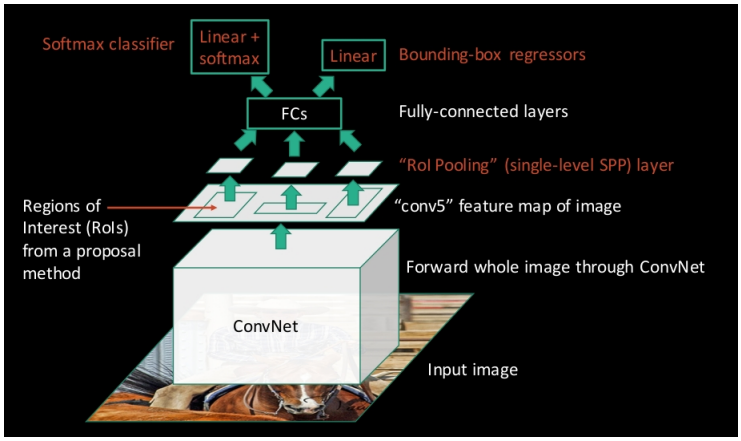
1. Higher detection quality.
2. Faster training and testing time.
3. Training can update all network layers.
4. No disk storage is required.

Fast R-CNN architecture contains 4 steps:

1. Input is an image and a set of object proposals (output from **Selective Search**).
2. The image is forwarded through some convolutional and max pooling layers to produce a feature map.
3. A region of interest (RoI) pooling layer extracts a fixed-length feature vector from the feature map for each object proposal.
4. Each feature vector is fed into some fully connected layers and finally 2 different layers:
   - One layer produces the sofmax probability that estimates over K object classes and a background class.
   - One Layer gives the bounding boxes for each of K object classes (each bounding box contains 4 numbers).



The RoI pooling layer use max pooling to convert any feature into a smaller feature map with a fixed size $H \times W$ (usually 7×7). Each RoI is a rectangular window (r, c, h, w) with the top-left corner (r, c), the height h and the width w.



RoI max pooling layer divides the $h \times w$ window into $H \times W$ sub-windows with approximate size $\frac{h}{H} \times \frac{w}{W}$ and max pools each sub-window to get the feature vector.

Fast R-CNN, mini-batches for stochastic gradient descent are sampled hierarchically by sampling N images and then sampling $\frac{R}{N}$ regions of interest (RoI) from each images. Normally $N = 2$ and $R = 128$. Fast R-CNN uses a streamlined training process with one fine-tuning stage that optimizes a softmax classifier and bounding box regressor simultaneously rather than different stages [1].

Fast R-CNN has 2 output layers:

1. One layer outputs the probability distribution $p = (p_0, ..., p_K)$ over $K+1$ classes (K classes and the background class).
2. One layer outputs the bounding boxes for K classes $t^k = (t^k_x, t^k_y, t^k_w, t^k_h)$

Each RoI is labeled with a ground truth class u and a ground-truth bounding box regression target v. We have the following multi-task loss:

$$L(p, u, t_u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v)$$

$L_{cls}(p, u) = -\log p_u$ is the log loss for the true class u.
$[u \geq 1] = 1$ when $u \geq 1$ and 0 otherwise.
L_{loc} is defined over the bounding-box regression target for class u with $v = (v_x, v_y, v_w, v_h)$ and the predicted bounding box $t^u = (t^u_x, t^u_y, t^u_w, t^u_h)$.

$$L_{loc}(t^u, v) = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}(t^u_i - v_i)$$

with

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if} |x| < 0 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

The mini-batches are sampled with 64 RoIs from each image. 25% of the RoIs are from object proposals that have the intersection over union (IoU) overlap with a ground truth bounding box of at least 0.5. The remaining RoIs are sampled from object proposals that have a maximum IoU with ground truth in the interval [0.1, 0.5).

For the **detection** phase:

1. Fast R-CNN take an image as input and a list of R object proposals.
2. For each RoI r, the foward pass outputs a probability distribution p and a set of predicted bounding box for r.
3. Assign a detection confidence to r for each object class k using the estimated probability.
4. Perform non-maxiumum supression independently for each class.

The results are outstanding for Fast R-CNN [3].

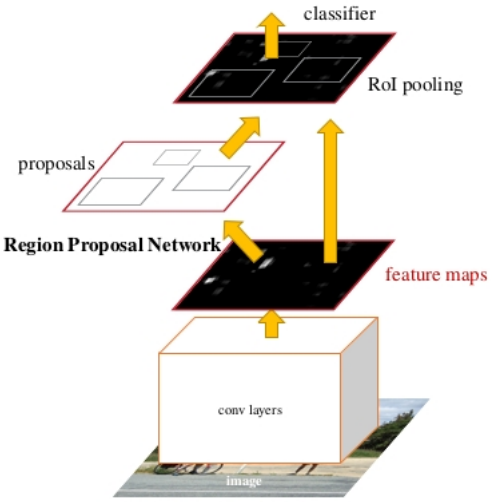
	Fast R-CNN	**R-CNN**	**SPPnet**
Training time	9.5h	84h	25h
Testing time / image	0.32s	47s	2.3s
mAP	66.9%	66.0%	63.1%

5 Faster R-CNN

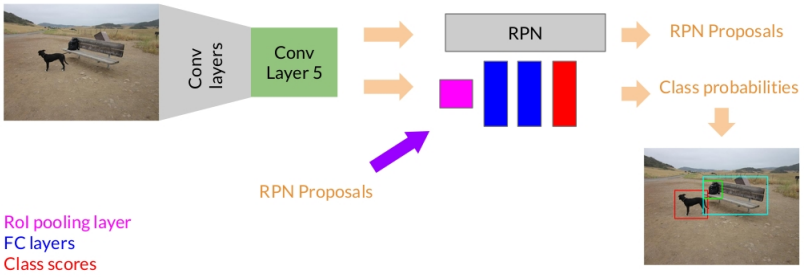
Faster R-CNN includes 2 modules:

1. The first module is a deep fully convolutional network proposed regions.
2. The second one is a Fast R-CNN detector which uses the proposed regions from the previous module.

The entire system is a single and unified network for object detection. The Region Proposal Network gives the Fast R-CNN modules the regions to find the objects [6].



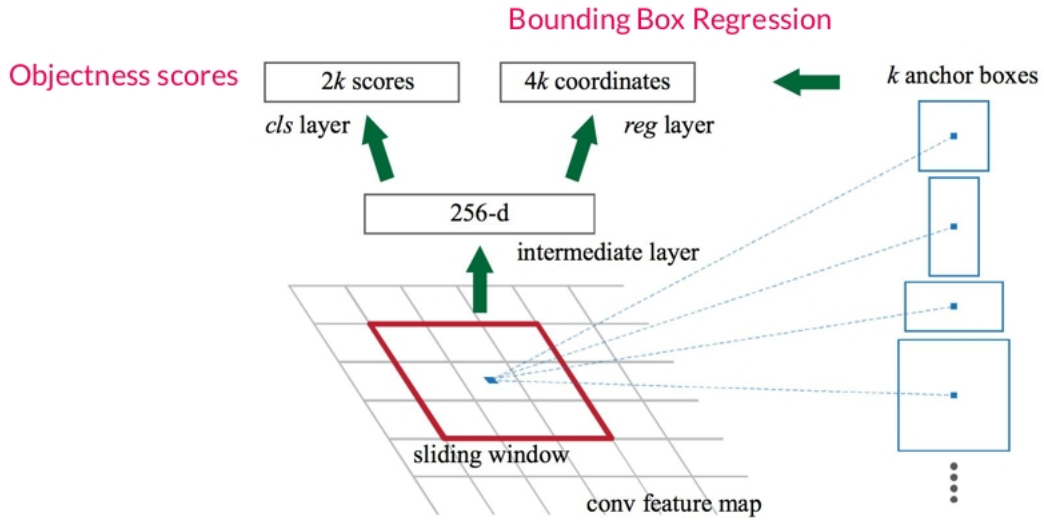
Here is the detailed structure of Faster R-CNN [7].



A Region Proposal Network takes:

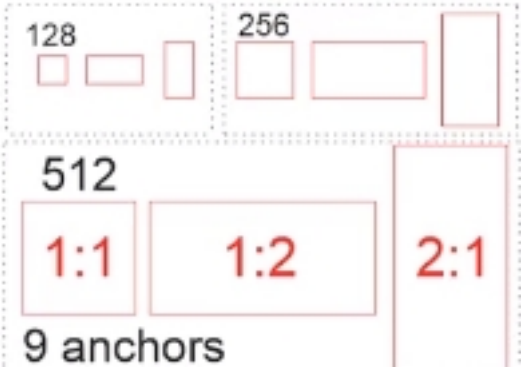
1. **Input:** An image (any size)
2. **Output:** A set of rectangular object proposals with objectness score. In the paper [6], the authors tested 2 architectures: ZF Net [11] and VGG-16 [8].

In order to generate region proposals, a small network is slided over the convolutional feature map output by the last shared convolutional layer. This feature map is fed into 2 connected layers - a box-regression layer *(reg)* and a box-classification layer *(cls)*.



At each sliding location, the network simultaneously predict multiple region proposals with the maximum possible proposals for each location is k. Therefore, the *reg* layer will have $4k$ outputs which encodes the rectangular coordinate for k bounding boxes and $2k$ scores which

estimate the probability of object or not object for each proposal [6]. The k proposals are called *anchors*. Each anchor is centered at the sliding window and has different ratio. For example, if $k = 9$, we would have those anchors [9].



In order to train RPNs, a binary class label (being an object or not) is assigned to each anchor. A **positive label** is assigned when:

1. The anchor has the highest IoU (Intersection over Union) overlap with the ground truth box.
2. The anchor has an IoU overlap higher than 0.7 with any ground truth box.

A **negative label** is assigned when the IoU ratio is lower than 0.3 for all ground truth boxes. The loss function for an image is defined:

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

i is the anchor.
p_i is the predicted probability of being an object for anchor i.
t_i is the coordinates of the predicted bounding box for anchor i.
p_i^* is the ground truth object label, equals to 1 if the anchor is positive and 0 if the anchor is negative.
t_i^* is the ground truth bounding box.
L_{cls} is the log loss over 2 classes (object and not object).
N_{cls} is the number of anchors in a mini-batch (set to 256 in general)
N_{reg} is the number of anchor locations (about 2400).

For the bounding box regression, we calculate the paramaterers of the 4 coordinates:

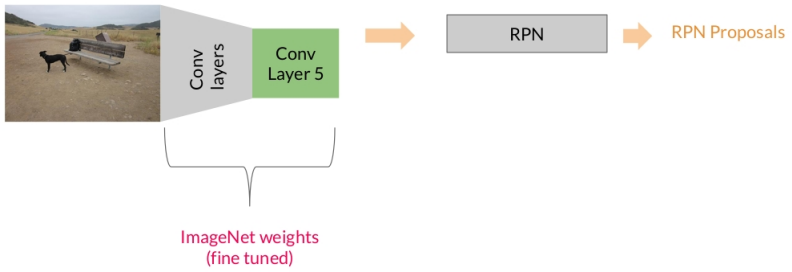
$$t_x = \frac{x - x_a}{w_a}, t_y = \frac{y - y_a}{h_a}$$
$$t_w = \log \frac{w}{w_a}, t_h = \log \frac{h}{h_a}$$
$$t_x^* = \frac{x^* - x_a}{w_a}, t_y^* = \frac{y^* - y_a}{h_a}$$
$$t_w^* = \log \frac{w^*}{w_a}, t_h^* = \log \frac{h^*}{h_a}$$

where x, y, w, h is the predicted box's center coordinates, width and height respectively. x_a, y_a, w_a, h_a is for the anchor box and x^*, y^*, w^*, h^* is for the ground truth box. The feature used for regression are of the same spatial size on the feature maps. k bounding-box regressors are learned for each ratio and do not share weights [6].

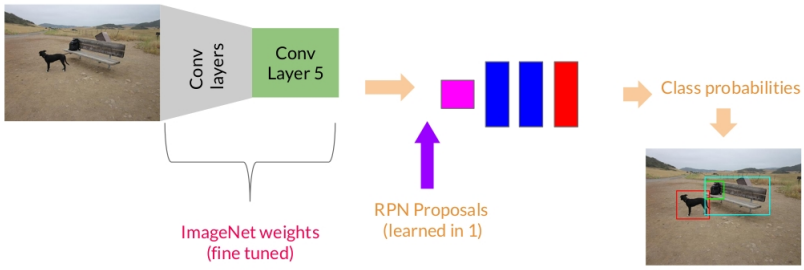
The RPN is trained by back propagation and stochastic gradient descent. Each mini-batch is selected from 256 random anchors of an image which contains 128 positive samples and 128 positive negative samples. If there are fewer than 128 positive samples, negative samples will be added. Weights are randomly initialized from Gaussian distribution ($\mu = 0, \sigma = 0.001$). The learning rate is 0.001 for the first 60,000 mini-batches and 0.0001 for the next 20,000 mini-batches when training on PASCAL VOC dataset [6].

For the second phase (detection), Fast R-CNN is used. However, a unified network compose of RPN and Fast R-CNN with shared convolutional layers are learned. There are 4 steps for training [7]:

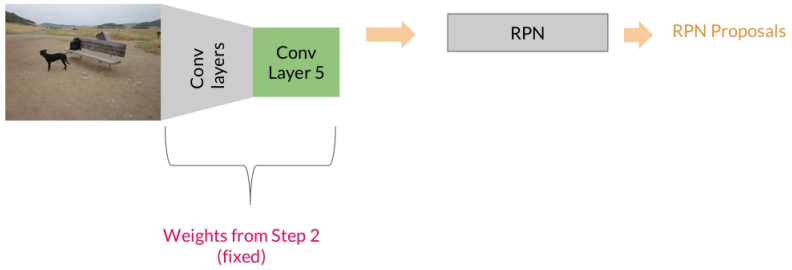
1. Train RPN initialized with an ImageNet pre-trained model.



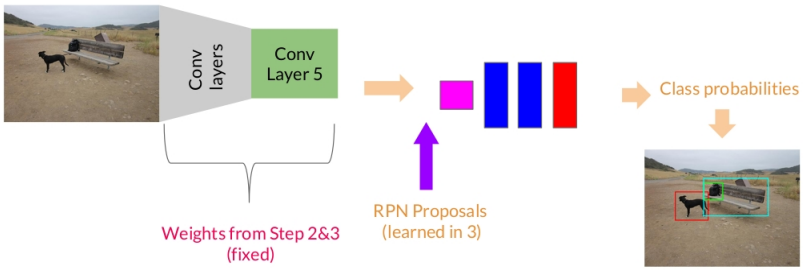
2. Train Fast R-CNN with learned RPN proposals



3. The model from step 2 is used to initialize RPN and re-train.



4. Fine tune the fully connected layer of Fast R-CNN using the shared convolutional layers as in step 3.



The results are much better in speed when comparing Faster R-CNN with R-CNN and Fast R-CNN. Here is the comparison of these 3 architectures on VOC 2007 [9].

	R-CNN	**Fast R-CNN**	**Faster R-CNN**
Test time per image	50s	2s	0.2s
mAP	66.0%	66.9%	66.9%

References

[1] Ross Girshick. "Fast R-CNN". In: (Apr. 30, 2015). URL: https://arxiv.org/abs/1504.08083.

[2] Ross Girshick. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: (Nov. 11, 2013). URL: https://arxiv.org/abs/1311.2524.

[3] Ross Girshick. *Slides Fast R-CNN*. Tech. rep. URL: http://www.robots.ox.ac.uk/~tvg/publications/talks/fast-rcnn-slides.pdf.

[4] Ross Girshick. *Slides Training R-CNNs of various velocities (slow, fast, and faster)*. Tech. rep. URL: https://www.dropbox.com/s/xtr4yd4i5e0vw8g/iccv15_tutorial_training_rbg.pdf?dl=0.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: (June 18, 2014). URL: https://arxiv.org/abs/1406.4729.

[6] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: (June 4, 2015). URL: https://arxiv.org/abs/1506.01497.

[7] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. *Slides Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. Tech. rep. Jan. 3, 2016. URL: https://www.slideshare.net/xavigiro/faster-rcnn-towards-realtime-object-detection-with-region-proposal-networks.

[8] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: (Sept. 4, 2014). URL: https://arxiv.org/abs/1409.1556.

[9] Ardian Uman. *Paper Review: Faster R-CNN for Real-time Object*. URL: https://www.youtube.com/playlist?list=PLkRkKTC6HZMzp28TxR_fJYZ-K8Yu3EQw0.

[10] Wikipedia. *Object Detection*. URL: https://en.wikipedia.org/wiki/Object_detection.

[11] Matthew Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: (Nov. 12, 2013). URL: https://arxiv.org/abs/1311.2901.