Summary of Object Detection papers:

RCNN:
Selective search (localization) + CNN (feature extraction) + SVM (classification) + bounding box regressor (refinement)

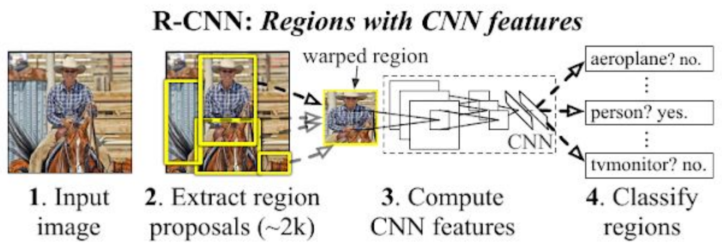


Figure 1: Object detection system overview. Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. R-CNN achieves a mean average precision (mAP) of **53.7% on PASCAL VOC 2010**. For

SPP-Net:
RCNN has two main shortcomings, and SPP-Net improves them:

1. RCNN need to compute 2000 convolutions for one image, which is time-consuming. SPP-Net computes only one convolution for one image, then use SPP pooling for feature map to extract features.
2. RCNN need to warp and dilate roi of different sizes. SPP-Net use SPP layer to make all features of roi is fixed size before fully connected layer.

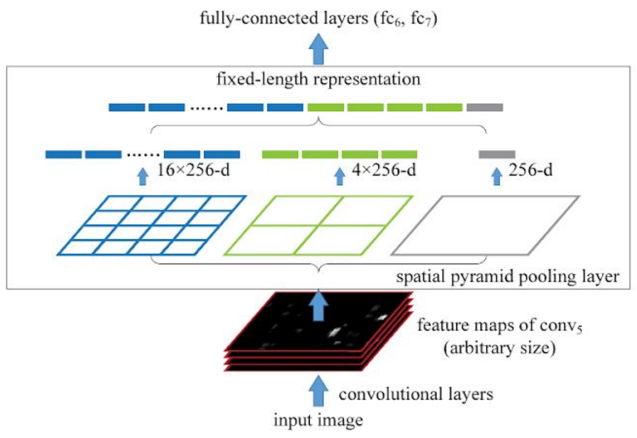

fully-connected layers (fc_6, fc_7)
fixed-length representation
16×256-d
4×256-d
256-d
spatial pyramid pooling layer
feature maps of $conv_5$ (arbitrary size)
convolutional layers
input image

Figure 3: A network structure with a **spatial pyramid pooling layer**. Here 256 is the filter number of the $conv_5$ layer, and $conv_5$ is the last convolutional layer.

Fast-RCNN:
Architecture:

1. Use one SPP pooling layers instead of multiple ones.
2. Combine CNN + classification (softmax) + box regression
3. Use Truncated SVD to accelerate speed.

Training Method:

1. Hierarchical sampling
2. Use two-task loss
3. Fine tune convolutional layers, unlike SPP net.

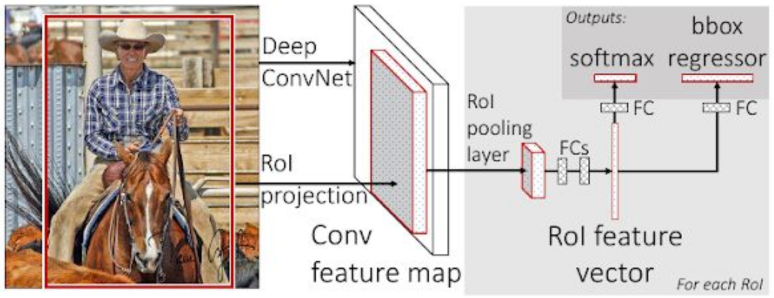


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

Faster-RCNN:
Proposals generation is the bottleneck for accelerating the object detection algorithms. Faster-RCNN merges proposal generation and detection together into a whole network, but it is still two-stage detection algorithm, compared to Overfeat, which is one-stage alg.
Architecture:

1. Use the pre-trained deep neural network to generate a feature map.
2. RPN generates anchors based on the feature map, then outputs the class (object vs. background) and proposals after refinement. In this way can it produce more accurate and effective proposals for the detector.
3. Fast-RCNN part: based on the same feature map, pool from the roi, which is the proposals generated on the same feature map by the RPN, to the fix-length feature vector. Then feed it into fully connected layers. Last, we use two fully connected layers for classification and bounding box regression.

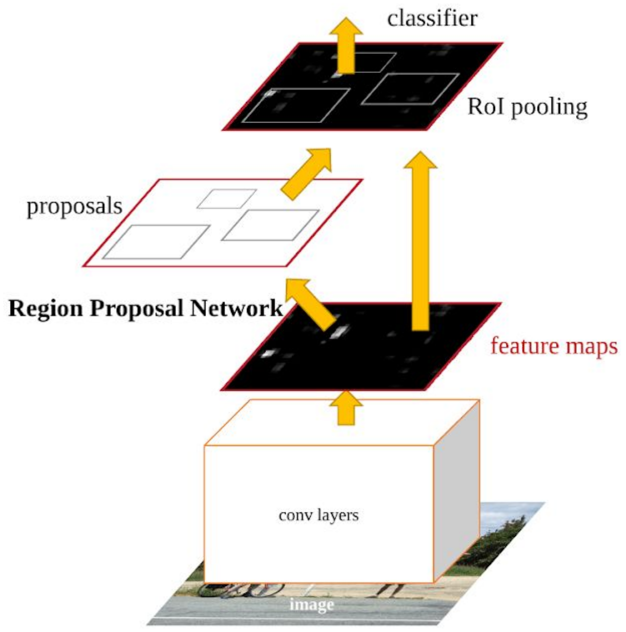
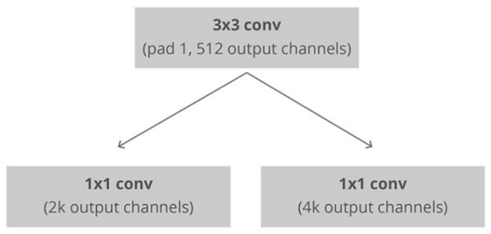


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

The architecture of RPN network:



Convolutional implementation of an RPN architecture, where k is the number of anchors.

Why use 3*3 convolutional layer? This is the n*n dimension for the sliding window. Since the authors wanted to detect objects using anchors in the every 3*3 region (Note this is on the feature map instead of the raw pixels), so 3*3 is how big the region is. Note 1) this 3*3 is different from k=9, which is the product of the 3 aspect ratios and 3 scales. 2) the stride of 3*3 conv is 1!!
Why use 1*1*channel conv layers for the final scores of 1) objectiveness and 2) box regression? Since like Fast-RCNN part, they use Fully connected layers for doing these two different things, but Fast-RCNN only does this for a single area, which is the input roi of the feature map (image). Comparatively, for the RPN part, the 3*3 sliding window is moving, so the fully connected layer is shared for all different regions which are slided by the 3*3 window. So in general, we use 1*1 conv layer to implement this shared fully connected layer.

Training Method:

Using the alternating training to train both RPN and Fast-RCNN back and forth, iteratively. At the same time, making them share the same feature map and convolutional layers.

R-FCN:

Why R-FCN:

1.Fast/Faster RCNN do not share computation when detect for each RoI, they just pool separately for each RoI. R-FCN uses the position layers which are shared on the whole image and pool layers can be learnt, both of which are not RoI-specific computation.

2. Solve the dilemma between translation invariance for image classification and translation variance for object detection. Fully convolutional layers are translation invariant, so they are more suitable for image classification, proved by all models for ImageNet. R-FCN adds position-sensitive fully convolutional layers to after fully convolutional layers, which are translation invariant, to encode the position information.

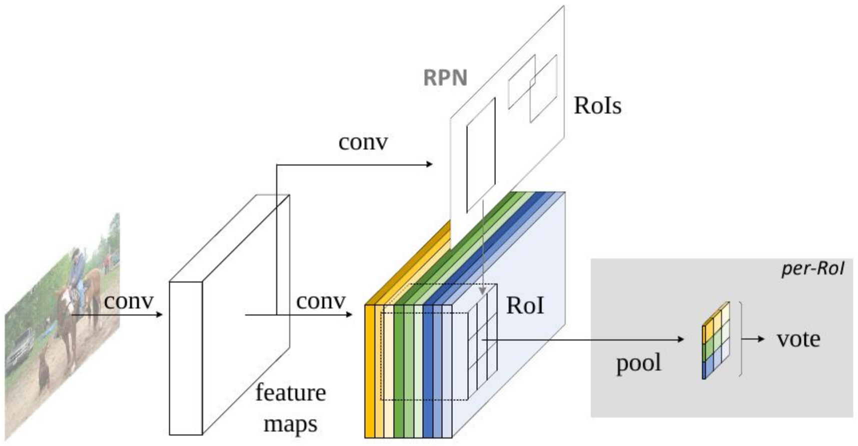


Figure 2: Overall architecture of R-FCN. A Region Proposal Network (RPN) [18] proposes candidate RoIs, which are then applied on the score maps. All learnable weight layers are convolutional and are computed on the entire image; the per-RoI computational cost is negligible.

Architecture:

1. Like Fast/Faster RCNN, first uses all convolutional layers (fully convolutional layers) from Res-101, to extract the feature map on the image. Note after the conv layers, RFCN use a 1*1*1024 fully convolutional layer to reduce the dimension.
2. Then like Faster R-CNN, RPN and position-sensitive layers share the feature maps.
3. RPN subnetwork has the same architecture with Faster RCNN, which is the fully convolutional layers.
4. For position-sensitive layers, take a look at Figure 1 as follows:

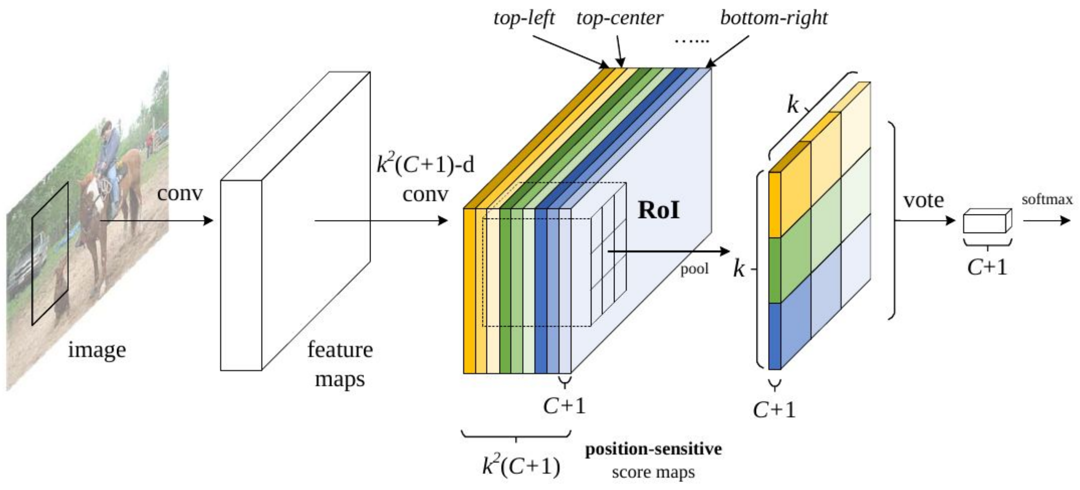


Figure 1: Key idea of **R-FCN** for object detection. In this illustration, there are $k \times k = 3 \times 3$ position-sensitive score maps generated by a fully convolutional network. For each of the $k \times k$ bins in an RoI, pooling is only performed on one of the k^2 maps (marked by different colors).

There are k^{2}(C+1) convolutional layers to compute on the entire image to get k^{2}(C+1) position-sensitive score maps. Each score map encodes the information of one of k positions for one of C+1 categories (C object classes + 1 background). As shown in the Figure 1, different color stands for different position, and each position, the depth is the kinds of classes.

5. After score maps, RFCN uses the pooling layers aggregates score information for each k^{2} position for each categories for RoI. It divides each RoI into k^{2} bins, and use average pooling for each bin. For the vote part, RFCN simply add for each class all the pooled score on each bin and get C+1 scores, then uses softmax to determine which category has the highest probability for this RoI.

6. For the box regression part, RFCN adds 4k^{2} for each RoI. Then uses average pooling to get 4-d vector to parameterize a box as t = (t_{x}, t_{y},t_{w}, t_{h}).

To visualize the result and better understand the position-sensitive score map, see the following figures:

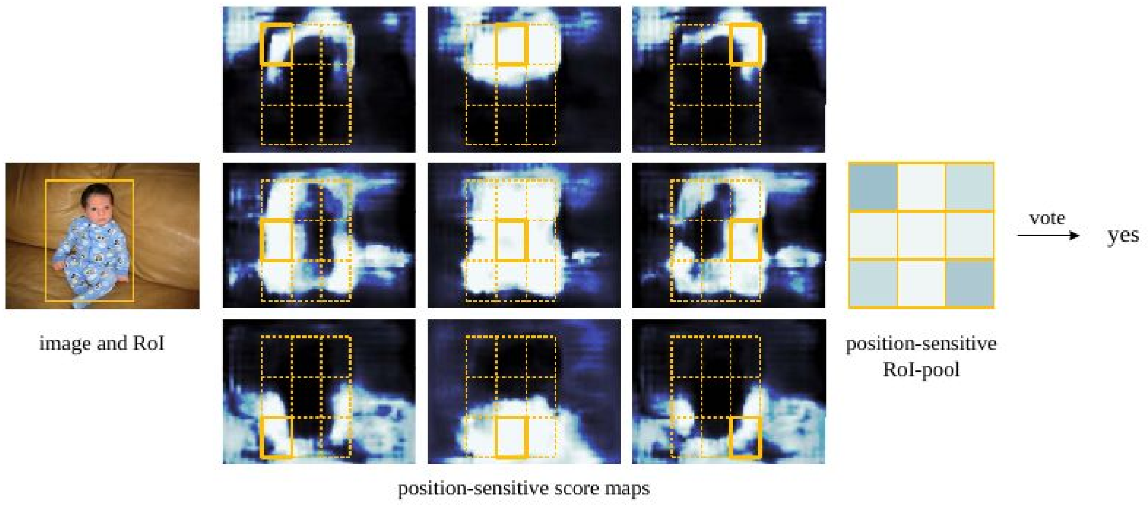


Figure 3: Visualization of R-FCN ($k \times k = 3 \times 3$) for the *person* category.

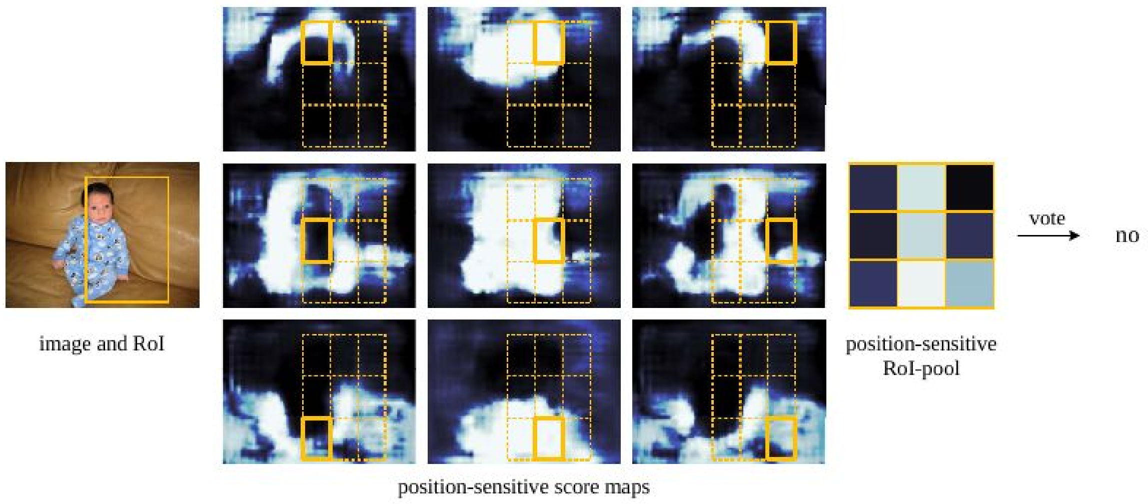


Figure 4: Visualization when an RoI does not correctly overlap the object.

In Figure 3, there are k^{2} = 9 different score maps extract from different position maps for the object category person. If the RoI exactly bounds on the person, we can see that after RoI pooling, the average score is high, and after voting, RFCN can detect this is a person. On the contrary, if RoI does not overlap the object, the average score is low. Note that in Fig 3 and 4, the score maps are the same for the same position, but the RoI is different, so the average score is different.

Questions about RCNN:

1. What is the positive and negative classes for box?? I know positive means the objects, and negative means the background, but how to assign the exact and specific number of it?? 0,+1,-1?? (2.3)
2. Is it possible that a box with the IoU more than 0.3 for one object while less than 0.3 for another one?? If it is, should we assign positive or negative to it?? (solved)
   in Faster RCNN paper, it says that the negative one is assigned to those boxes or anchors whose IoU overlaps with any ground-truth box lower than 0.3.
3. What is hard negative mining method? (solved)
   https://www.zhihu.com/question/46292829
https://www.quora.com/What-is-hard-negative-mining?redirected_qid=21542872

Questions about SPP-net:

1. In practice, the GPU implementations are preferably run on fixed input images, so how to tackle the problem that input different size images during the training? When use the same network, how to implement that we can change the input sizes?
2. How to analyze the table such like Table 9? How to get the result of a certain layer or without a certain layer(ablation analysis) ??
3. In page 9, how they implement the empirical strategy??(bottom right)
4. Why in mapping a window to feature maps, x' = x/S+1, why +1??

Question about Fast RCNN:

1. Why the bounding box regression is for every class, instead of every object??
2. Fast RCNN still need to rely on other algorithms to localize objects first?
3. What is special for smooth L1 loss?
4. What is the global learning rate (vs. weights and bias learning rate)
5. In Fast RCNN it also fine-tune the conv layers, I think it violates the definition of fine-tuning. (solved)
   Q:take a look at this blog:
https://flyyufelix.github.io/2016/10/03/fine-tuning-in-keras-part1.html

Question about Faster RCNN:

1. For RPN, there is a classifier and bounding box regressor, but there are also a classifier and bounding box regressor in later Fast RCNN part. Can we remove the background using RPN and remove the background class in Fast RCNN part? Why are there two bounding box regressor?
   Q: For RPN, we use classes, which are object vs. background, to do the non-maximum suppression in order to reduce the proposals fed into the detection network. For both box regression and classification in RPN, the author did the experiments in the ablation part, showing that the network without any of them will decrease the mAP.
2. Can CNN learn about the offset of so many proposals? Because the location of proposals are arbitrary, and I don't think the offset has a fixed model and features like classification for a CNN to learn.

3. Is there an intuition why they used 1*1 convolutional NN for RPN to classify the back and fore ground and as the bounding box regressor?
4. RPN is a fully convolutional net, so what is the difference between fully convolutional net and the original one? (solved: take a look at this answer in quora: https://www.quora.com/How-is-Fully-Convolutional-Network-FCN-different-from-the-original-Convolutional-Neural-Network-CNN)
5. For FCN, by what intuition or under what circumstance should we use FCN rather than the usual convolutional net??
6. How does 1*1 convolutional network in RPN work ?? https://stats.stackexchange.com/questions/194142/what-does-1x1-convolution-mean-in-a-neural-network
7. In RPN, the output of 1*1 conv still has the spatio size which is not 1*1 unlike fully convolutional layer, so how to compute the final score for each anchor??
   (Maybe uses the same method stated in VGG paper that compute the spatial averaged across the score map, then uses the softmax to compute the final probability.)

Common Concepts:

1. Object Proposals: https://www.quora.com/What-is-the-definition-of-Object-proposal-in-object-detection
2. RoI: region of interest. It can be any region in the image which is very important for the task.
3. Precision, Recall, IoU and mAP: see https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173