Faster RCNN : Towards Real-Time Object Detection with RPNs

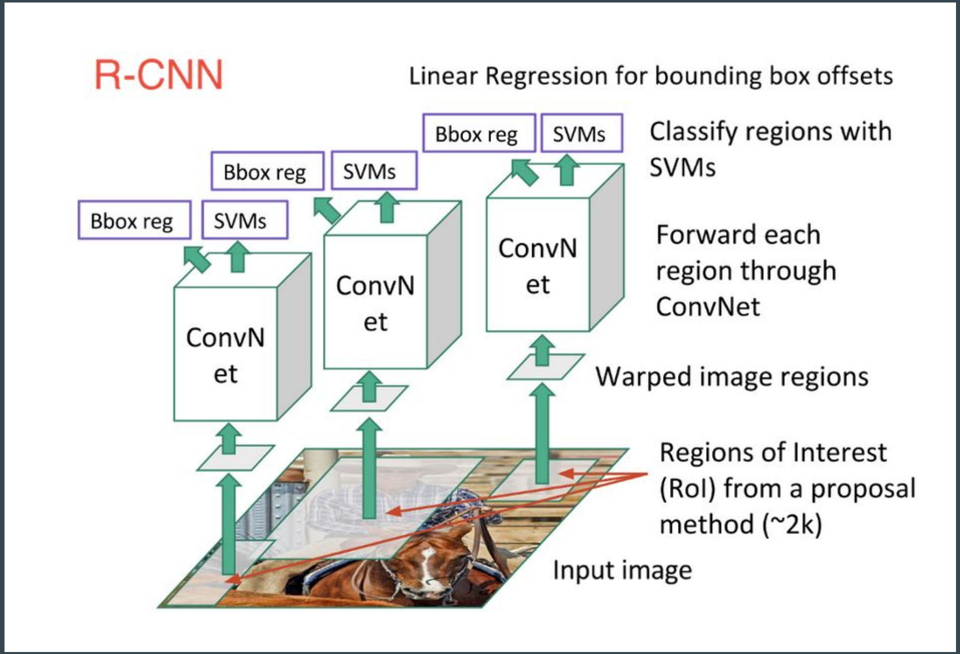
Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun

Presented by
-Pranav S.
- Nikunj Gupta

RCNN

Region-based Convolutional Neural Network

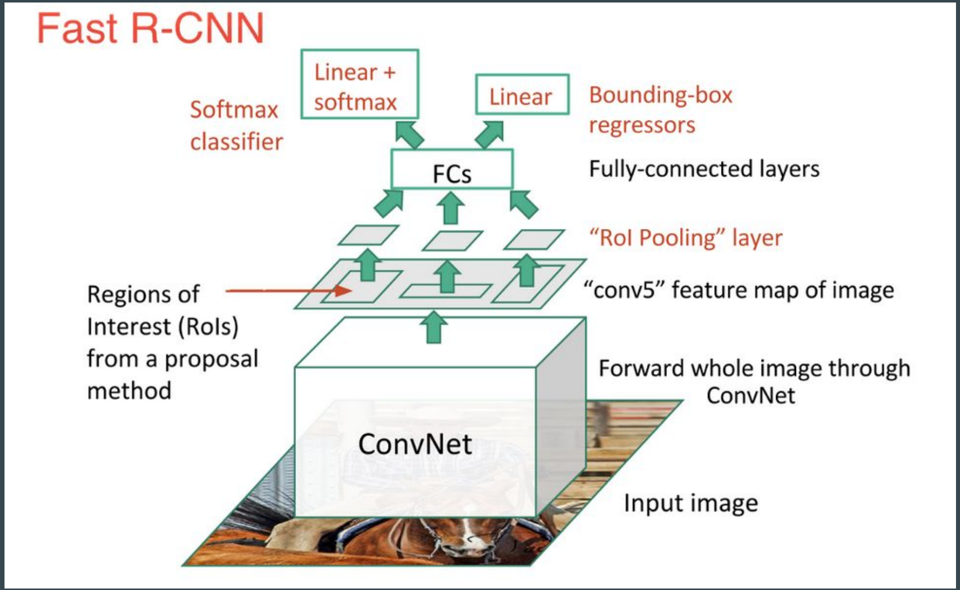
- Region Proposal Generation from input image using Selective Search (~2000)
- Run CNN on each of the proposal
- Feed the output of each CNN into
    - An SVM Classifier
    - A Linear regressor to tighten the boxes on the object



Fast RCNN

Improvement in object detection speed

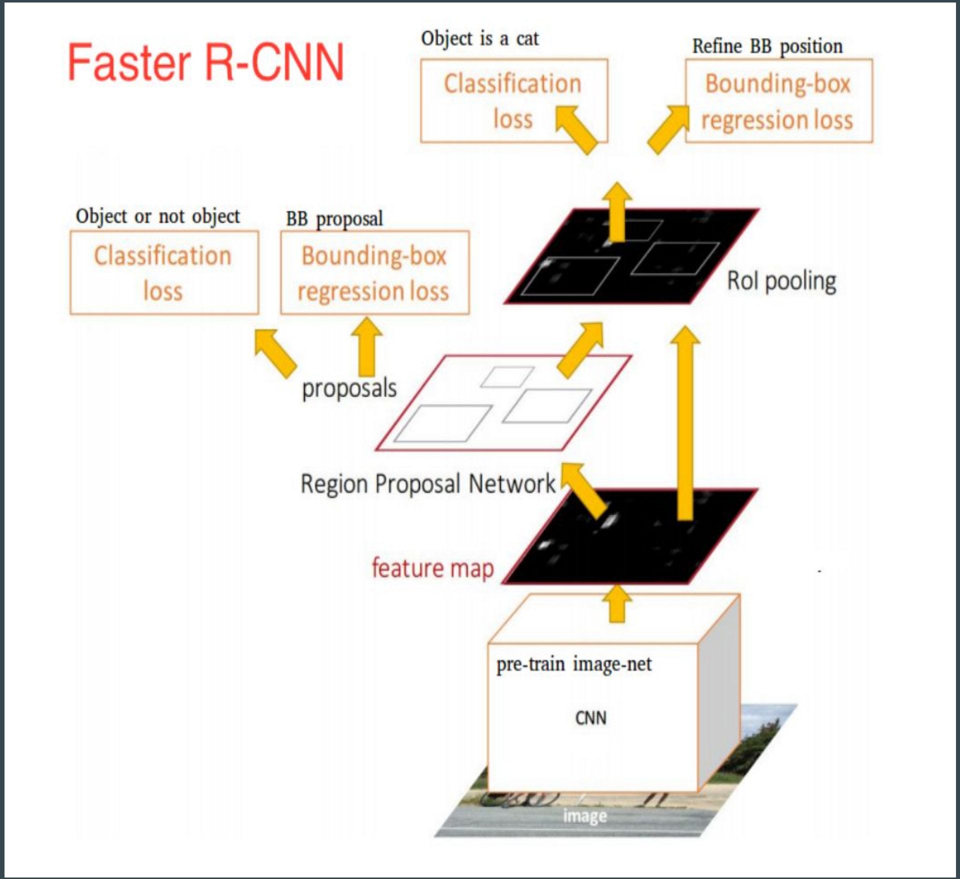
- Feature extraction before region proposals, and uses RoI pooling
- Instead of SVM classifier, a softmax layer is used. Outputs class probabilities directly



Faster RCNN

Use of RPNs (Region Proposal Networks)

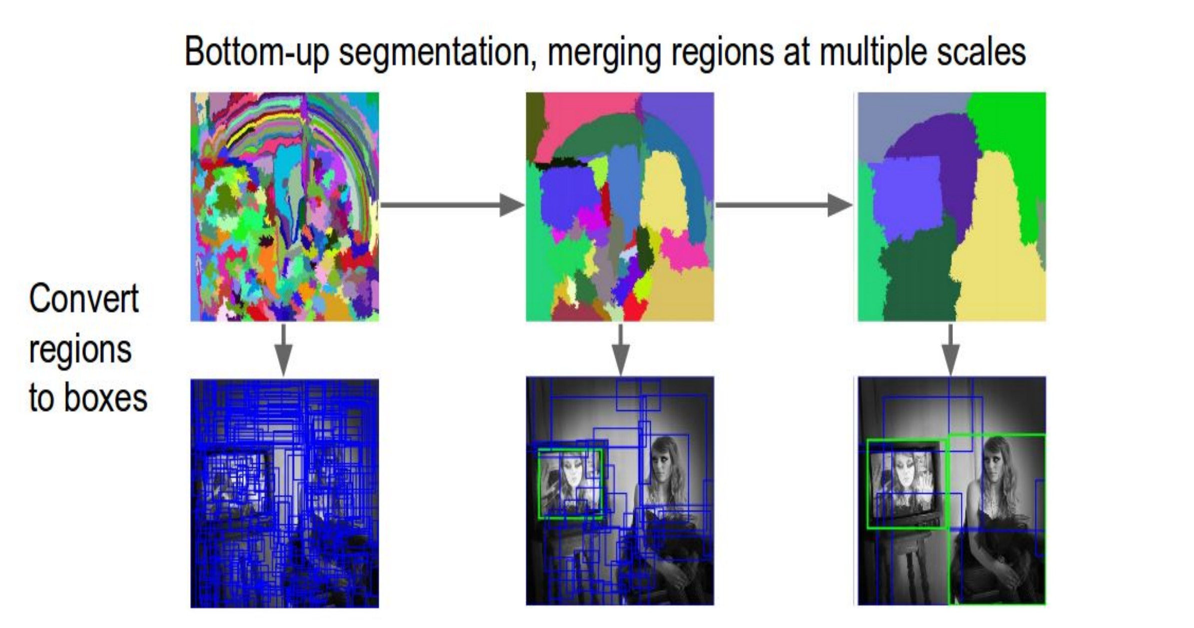
- Used instead of slow algorithms like Selective Search
- Obtained Region Proposals are fed into the Fast RCNN.
- Basically,
  Faster RCNN = RPN + Fast RCNN



Related Work

Object Proposals

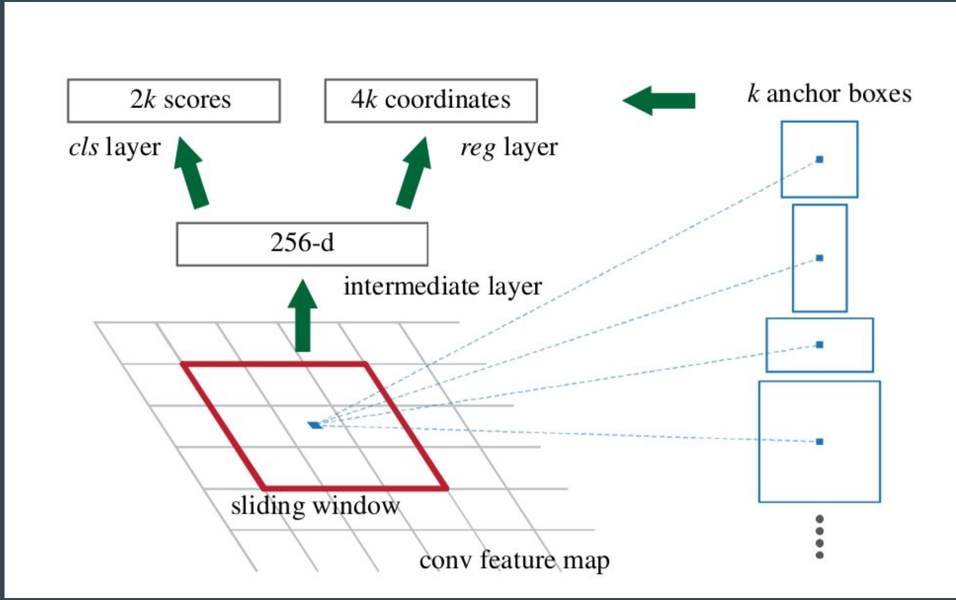
- Grouping super-pixels
  - Selective Search
  - CPMC
- Using sliding Windows
  - EdgeBoxes



Selective Search

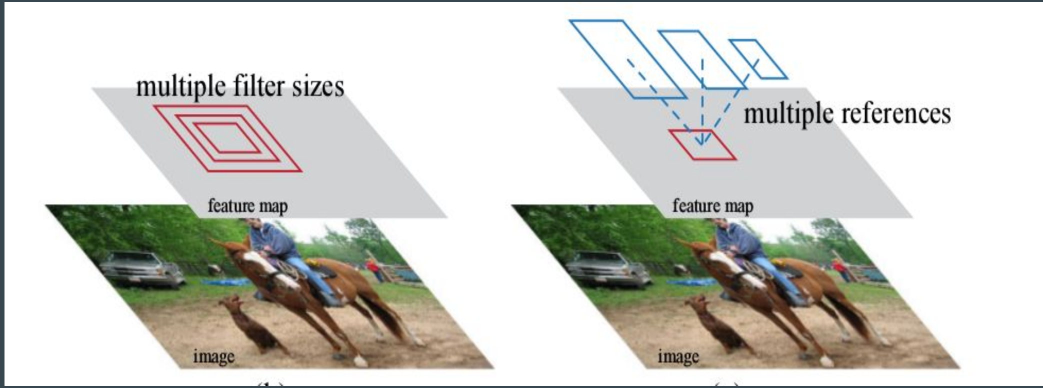
Crux of Faster RCNN: RPN

- Input: Image (of any size)
- Output: Set of rectangular boxes with objectness scores
- Generating regions proposals
    - Sliding window over feature map output of CNN.
    - Predict region proposals using anchor boxes for each sliding window.



Anchors

- Translation-Invariant Anchors
    - Translation of object in image should not affect the proposal function
    - Also, reduces the model size
- Multi-Scale Anchors as Regression References
    - Pyramid of images
    - Pyramid of filters
    - Pyramid of anchors



Loss function

- i = index of an anchor
- p_i = predicted probability of anchor i being an object
- p_i^* = ground-truth label
  - = 1, if anchor is positive,
  - = 0, if anchor is negative
- N_{cls} = size of the mini-batch (2k)
- N_{reg} = number of anchor locations (4k)
- λ = balancing hyperparameter

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

Training RPNs

- Trained end to end, with backpropagation and SGD.

- Sampling 256 anchors from image, where ratio of positive : negative anchors is upto 1 : 1

- Weights from zero mean gaussian distribution with std deviation 0.01 are used to initialize new layers.

- Learning rate =
  - 0.001 for the first 60k mini-batches
  - 0.01 for the next 20k mini-batches

- Momentum = 0.9

- Weight decay = 0.0005

Sharing features between RPN and Fast RCNN

4-Step Alternating Training

- Step 1: Training RPN
- Step 2: Training Fast RCNN detection network (using proposals generated by step 1)
- Step 3: Using detector network to initialize RPN training (keeping the shared convolutional layers fixed) and only fine-tune the layers unique to RPN
- Step 4: Fine-tune the layers unique to Fast RCNN, again keeping the shared ones fixed

Implementation Details

- Resize the images such that shorter side is 600 pixels

- For anchors,
  - 3 scales
    - 128
    - 256
    - 512
  - 3 aspect ratios
    - 1:1
    - 1:2
    - 2:1
- These hyperparameters are decided using *ablation* experiments

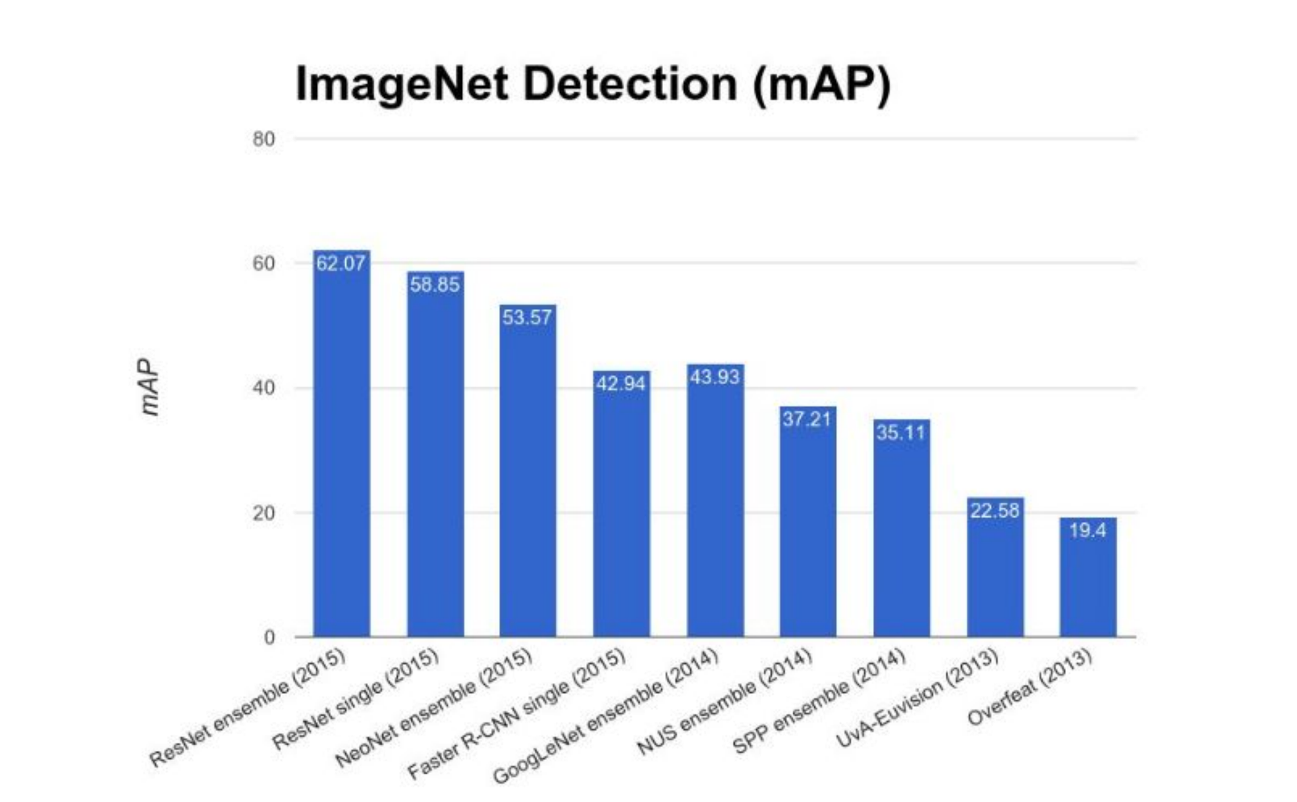
Comparison with daddy and grand daddy!

	R-CNN	**Fast R-CNN**	**Faster R-CNN**
Test time per image (with proposals)	50 seconds	2 seconds	**0.2 seconds**
(Speedup)	1x	25x	**250x**
mAP (VOC 2007)	66.0	**66.9**	**66.9**

Object Detection State-of-the-art: ResNet 101 + Faster R-CNN + some extras

training data	COCO train		COCO trainval	
test data	COCO val		COCO test-dev	
mAP	@.5	@[.5, .95]	@.5	@[.5, .95]
baseline Faster R-CNN (VGG-16)	41.5	21.2		
baseline Faster R-CNN (ResNet-101)	48.4	27.2		
+box refinement	49.9	29.9		
+context	51.1	30.0	53.3	32.2
+multi-scale testing	53.8	32.5	**55.7**	**34.9**
ensemble			**59.0**	**37.4**

ImageNet Detection 2013 - 2015



Thank You