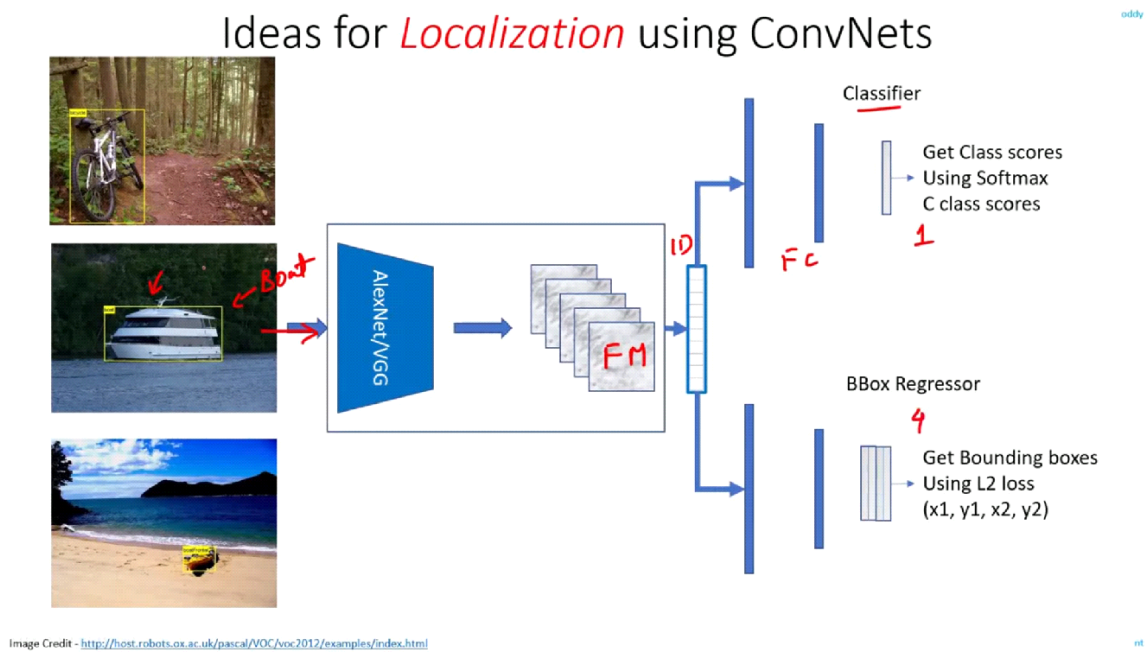
Family of RCNN

Vibhav Joshi

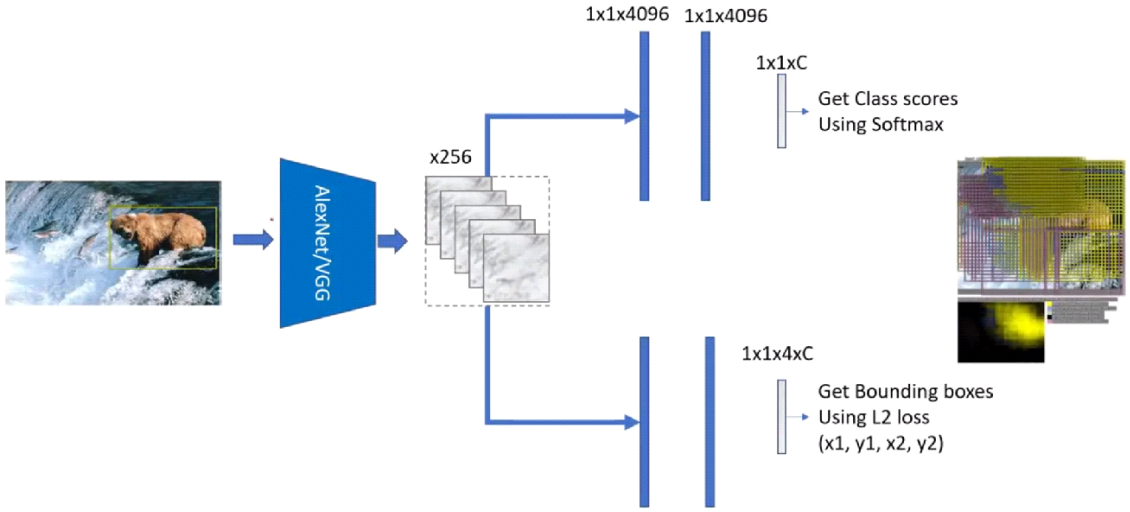
13th MAy, 2020

RCNN

- Prior to RCNN sliding window were used on different sizes of images to get class scores and bounding boxes
- But the problem was extra computation that was done on areas where the object was not present



Ideas for *Detection* using ConvNets



OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks – Sermanet et al
Images Credit – Overfeat paper & https://towardsdatascience.com/object-localization-in-overfeat-5bb2f7328b62

Overfeat

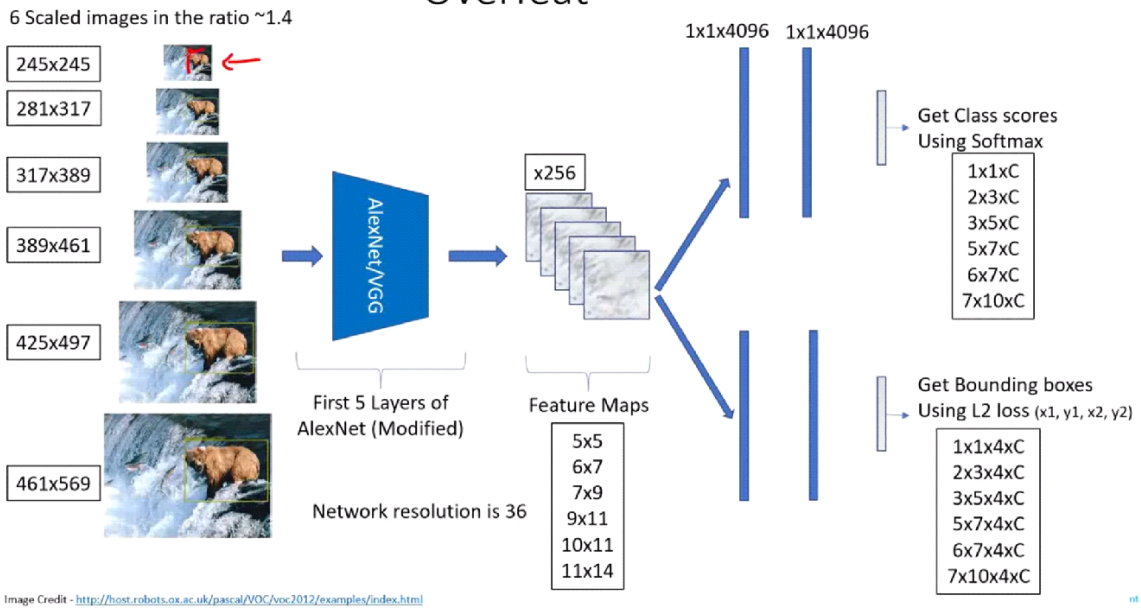
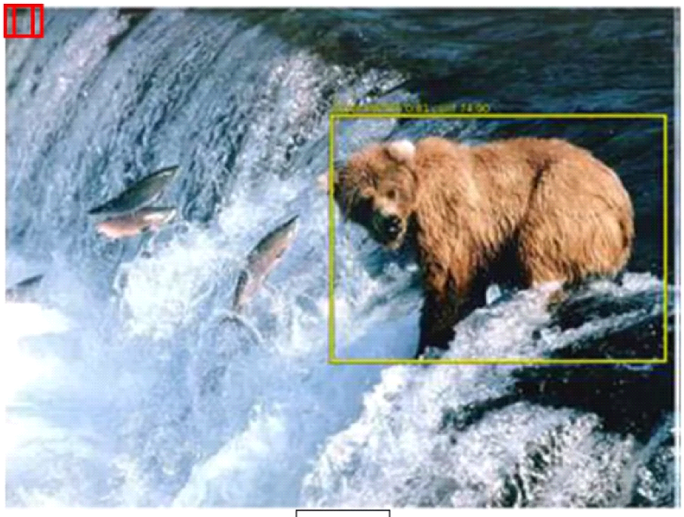


Image Credit - http://host.robots.ox.ac.uk/pascal/VOC/voc2012/examples/index.html

Dense Sampling – Sliding Window

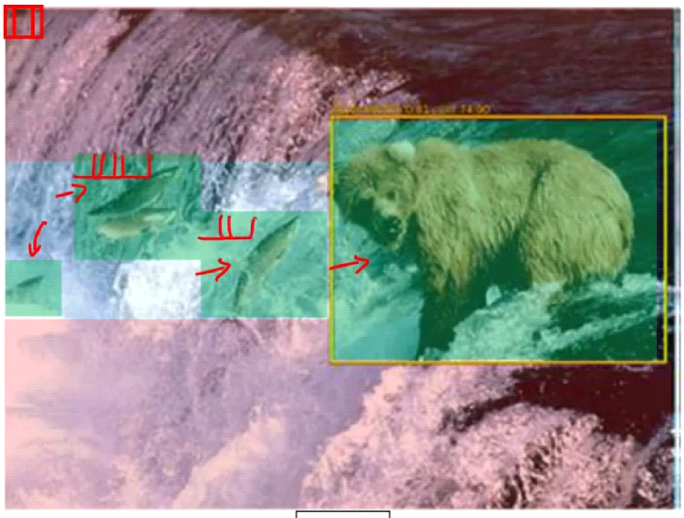
11x11, s= 4



245x245

Dense Sampling – Sliding Window

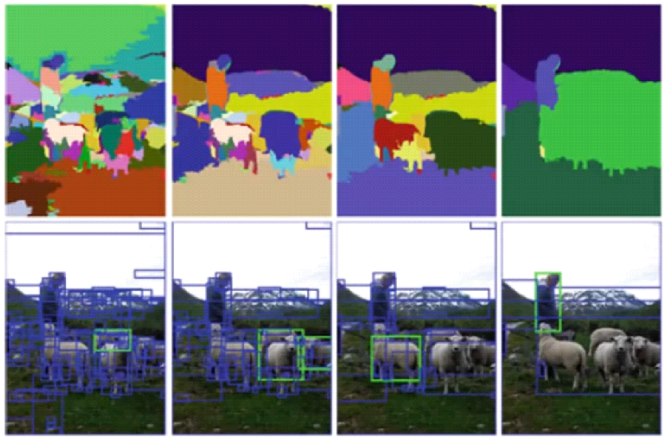
11x11, s= 4



245x245

- Thus we come up with region proposals which will give regions which have object
- The technique is selective search in which superpixel stradding is used to give regions which have same pisel intensity

Region Proposal Techniques

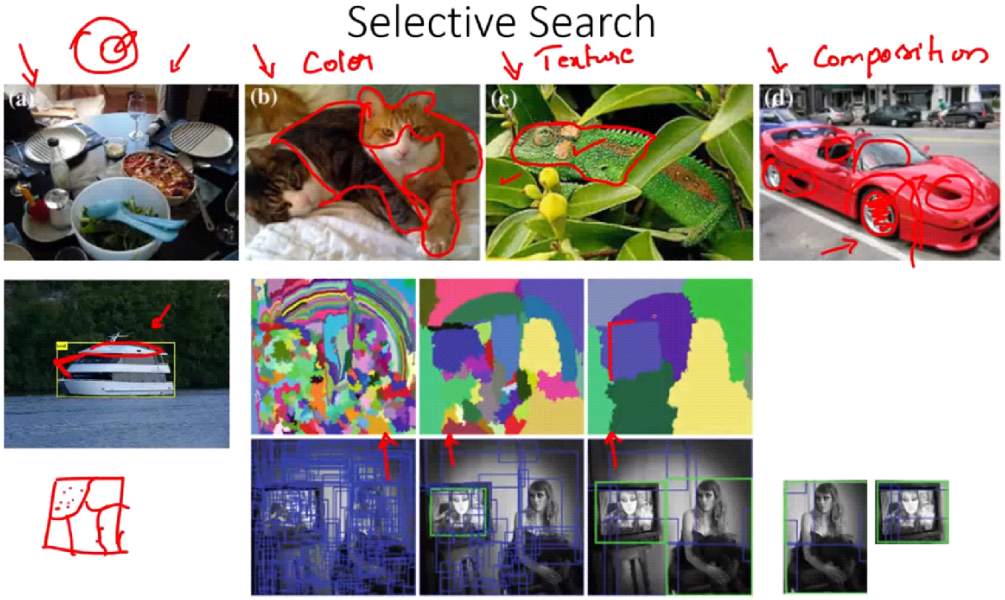


0	1	4	5	6	0	235	255
1	2	3	3	0	0	240	255
1	2	3	4	0	0	255	255
3	3	4	67	130	120	245	255
5	4	50	120	115	226	250	255
56	45	60	115	120	223	255	255
57	48	58	110	111	215	255	255
59	150	125	255	225	220	255	255

Superpixels Straddling

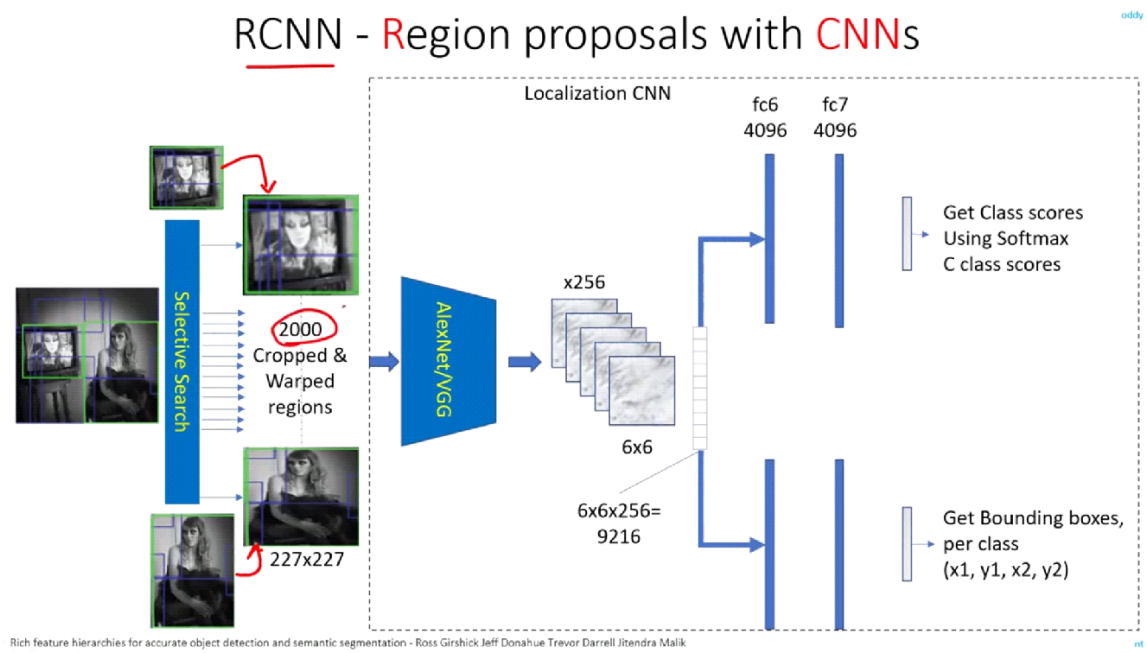
Selective Search for Object Recognition J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, A. W. M. Smeulders

Selective Search

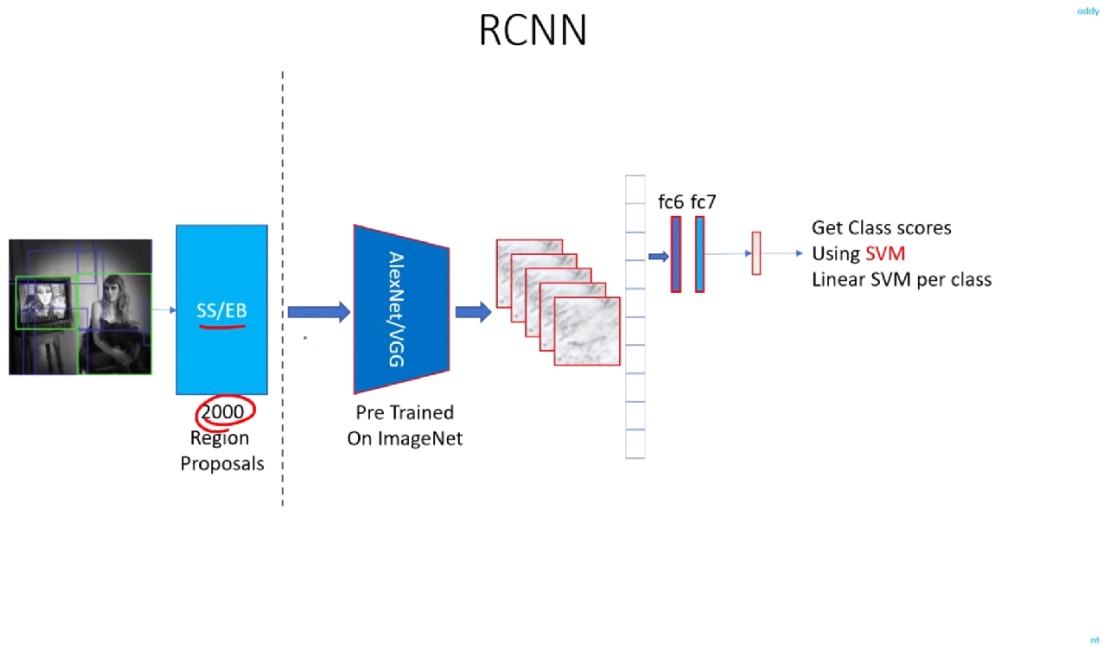


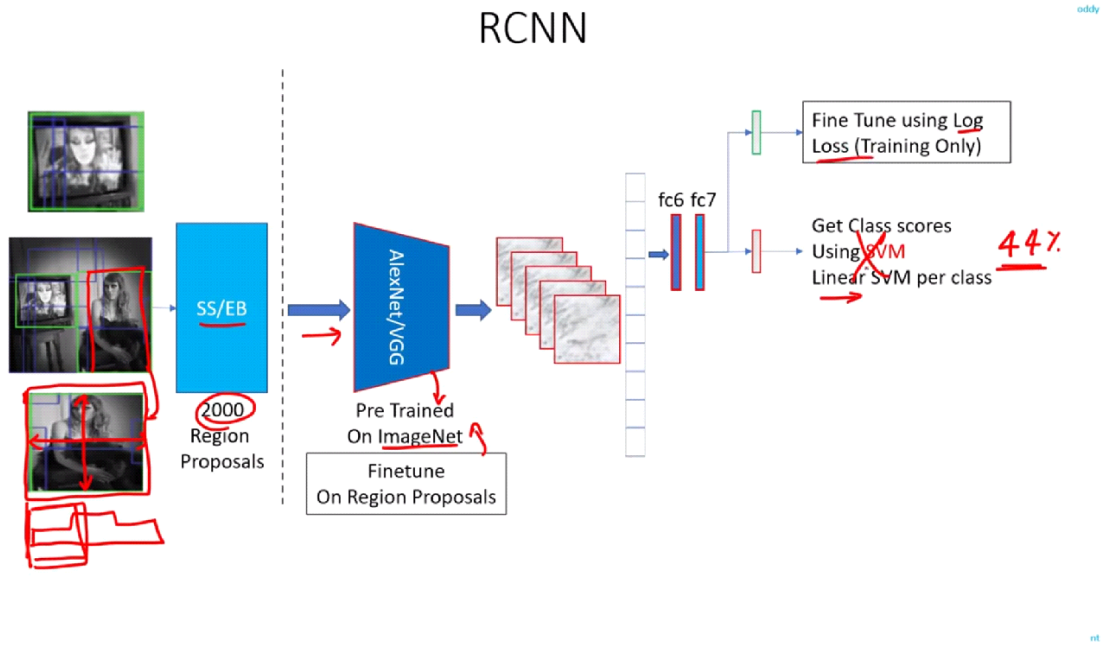
Selective Search for Object Recognition J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, A. W. M. Smeulders

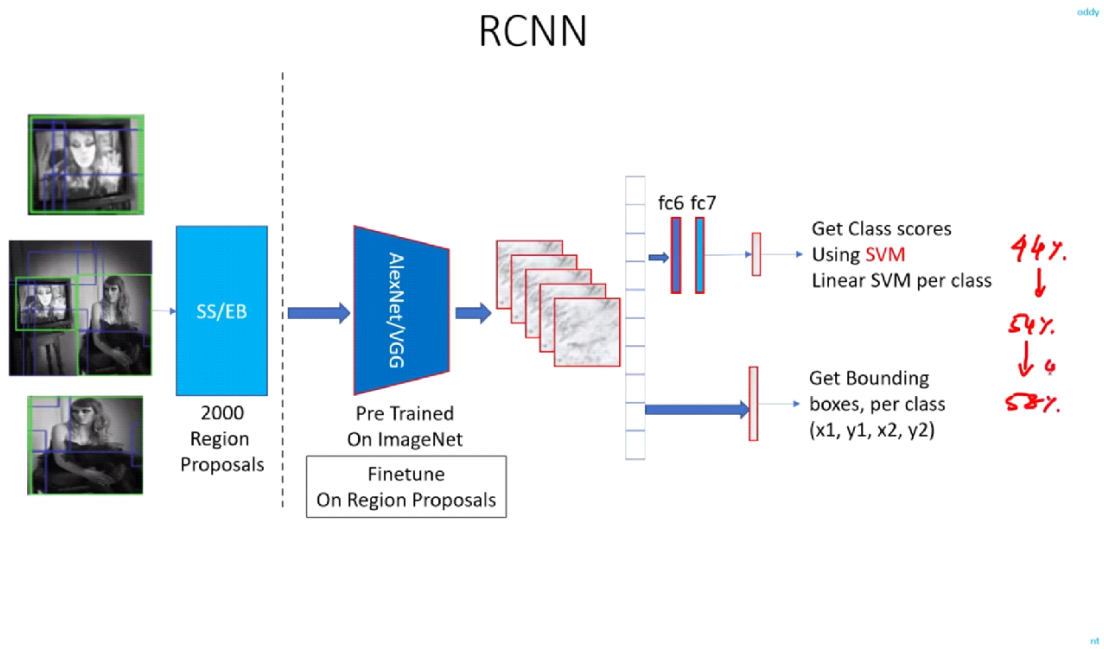
- Thus we get the Rcnn architecture in which first we propose regions and then perform localisation on the network



- We use SVM instead of softmax function and train backbone ConvNets on the warped regions of different sizes

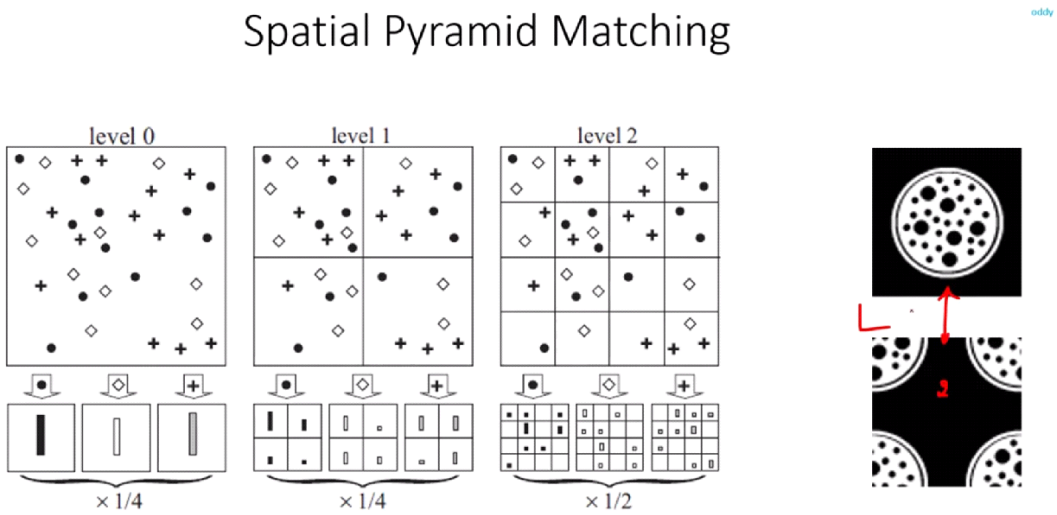



RCNN
SS/EB
2000
Region
Proposals
AlexNet/VGG
Pre Trained
On ImageNet
Finetune
On Region Proposals
fc6 fc7
Fine Tune using Log
Loss (Training Only)
Get Class scores
Using SVM
Linear SVM per class
44%


RCNN
SS/EB
2000
Region
Proposals
AlexNet/VGG
Pre Trained
On ImageNet
Finetune
On Region Proposals
fc6 fc7
Get Class scores
Using SVM
Linear SVM per class
Get Bounding
boxes, per class
(x1, y1, x2, y2)
44%
54%
58%

Fast Rcnn

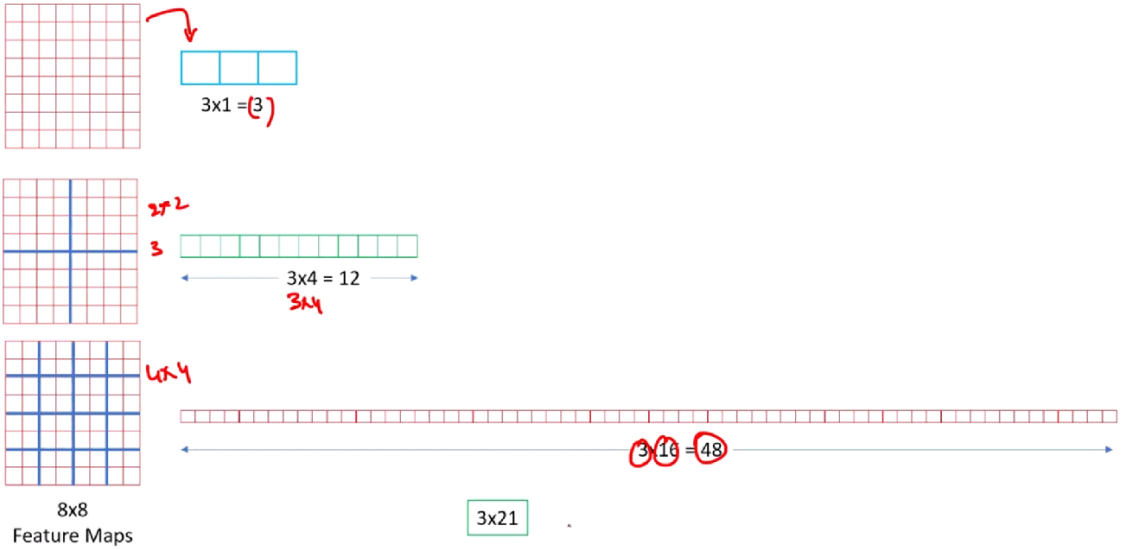
- In Rcnn the backbone ConvNets took a lot of time on processing on the regions and produce feature maps for them
- Thus in SPPNet they introduce spatial Pyramid Matching in which the image was divided in feature vectors which will then have the feature of every part of image



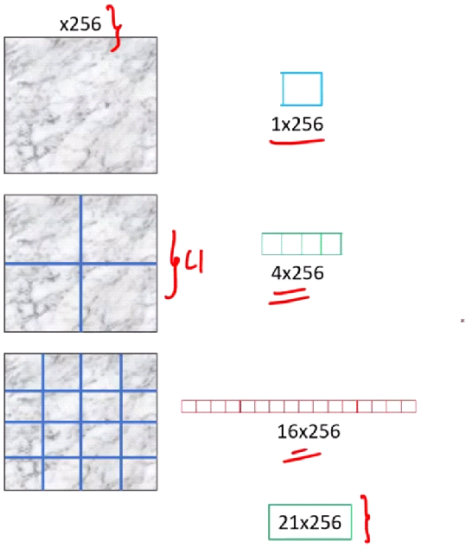
Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories - Svetlana Lazebnik, Cordelia Schmid, Jean Ponce

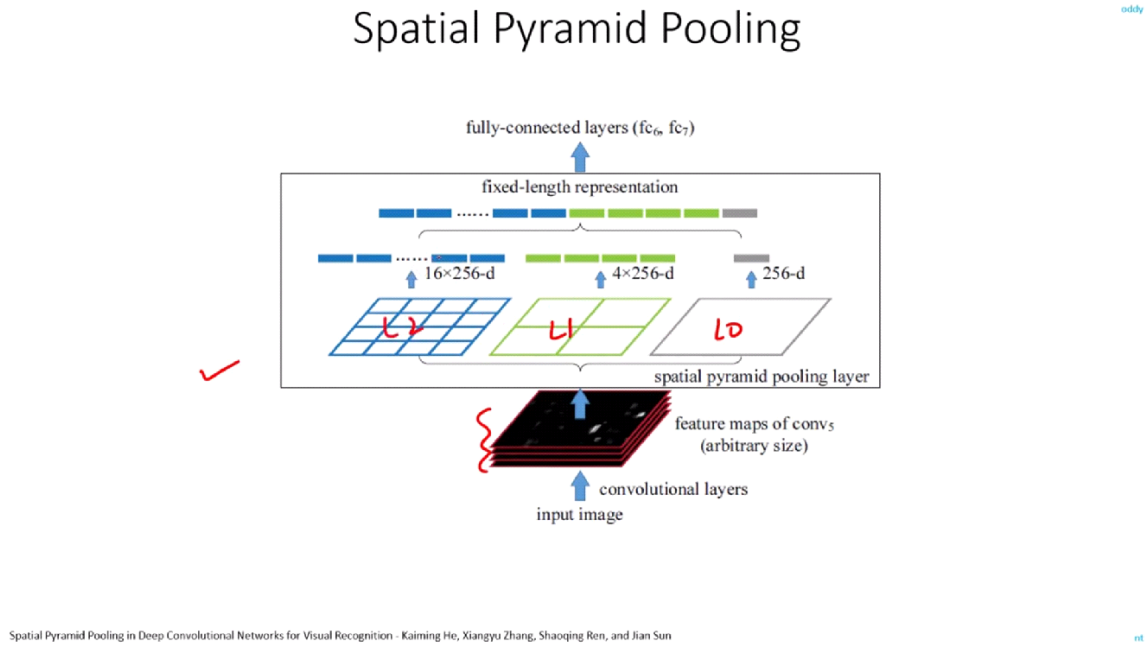
Image Credit - https://homepages.inf.ed.ac.uk/rbf/HIPR2/translte.htm

Spatial Pyramid Matching

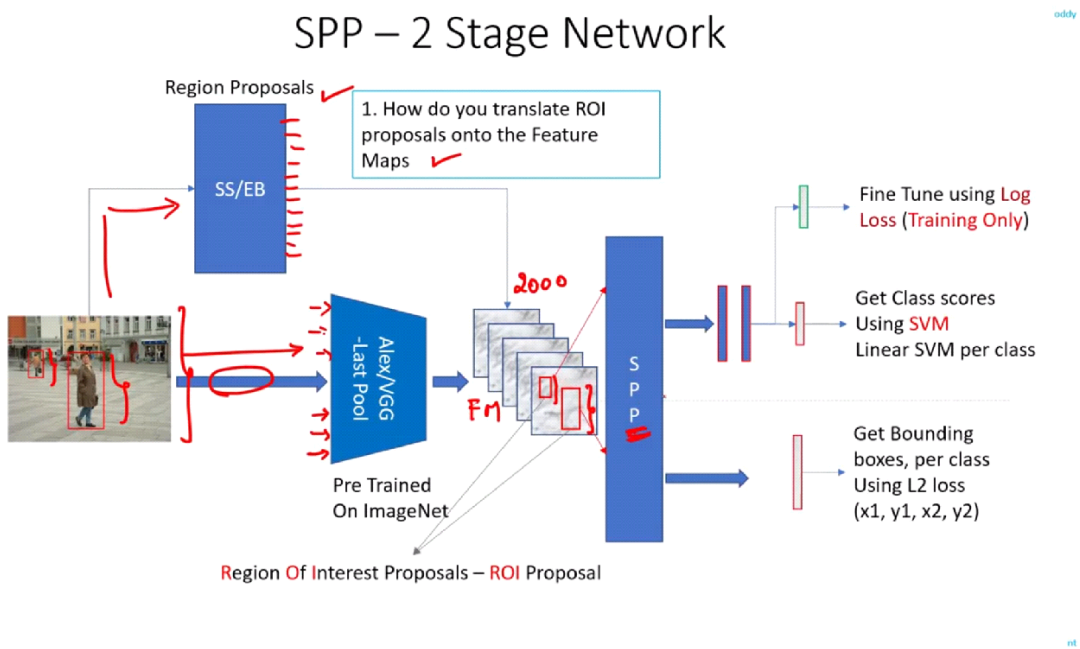


SPP for Feature Maps outputs

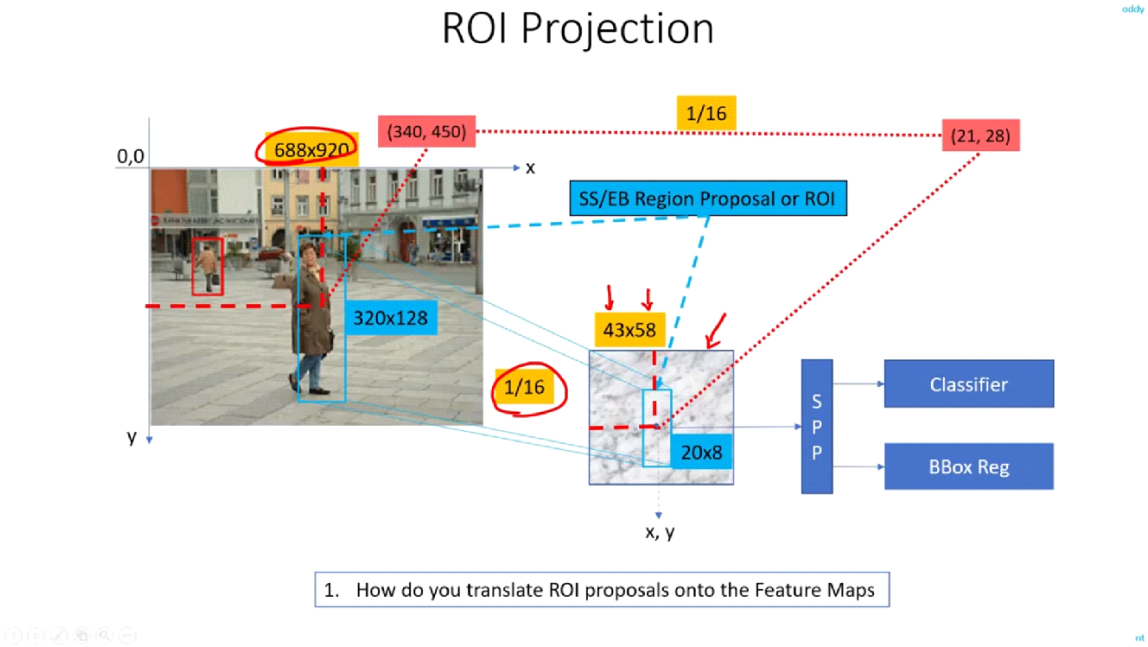




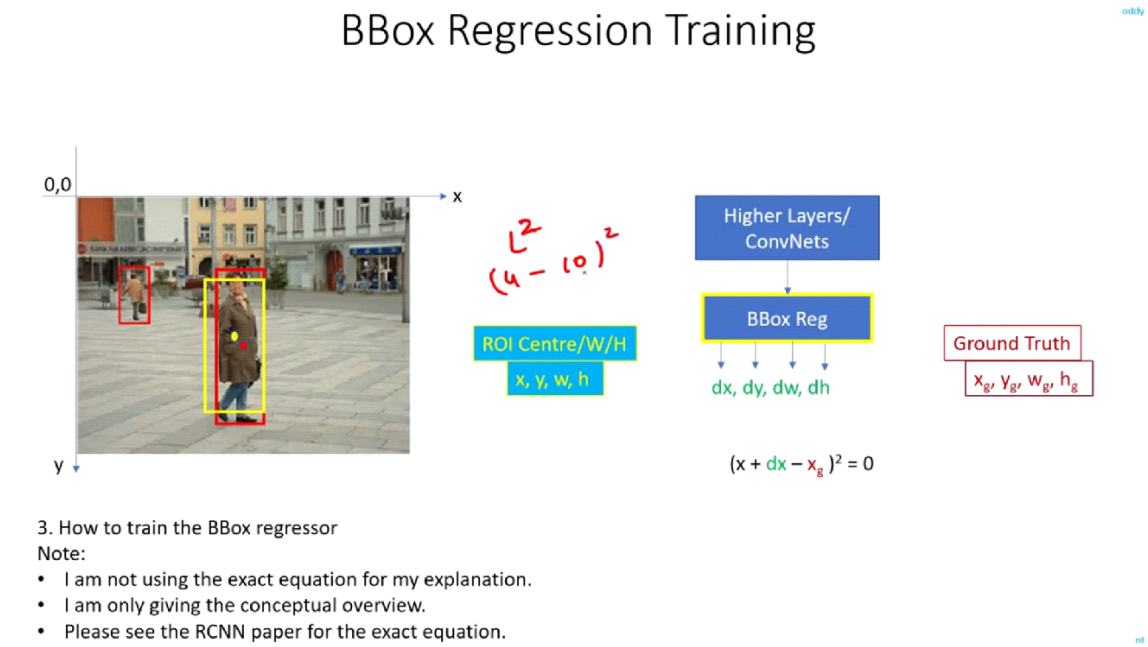
- Thus in SPP we take region proposals out and perform it and then map it on our feature vectors
- Region of Interest pooling took over SPP in Fast Rcnn in which only 7*7 size was considered



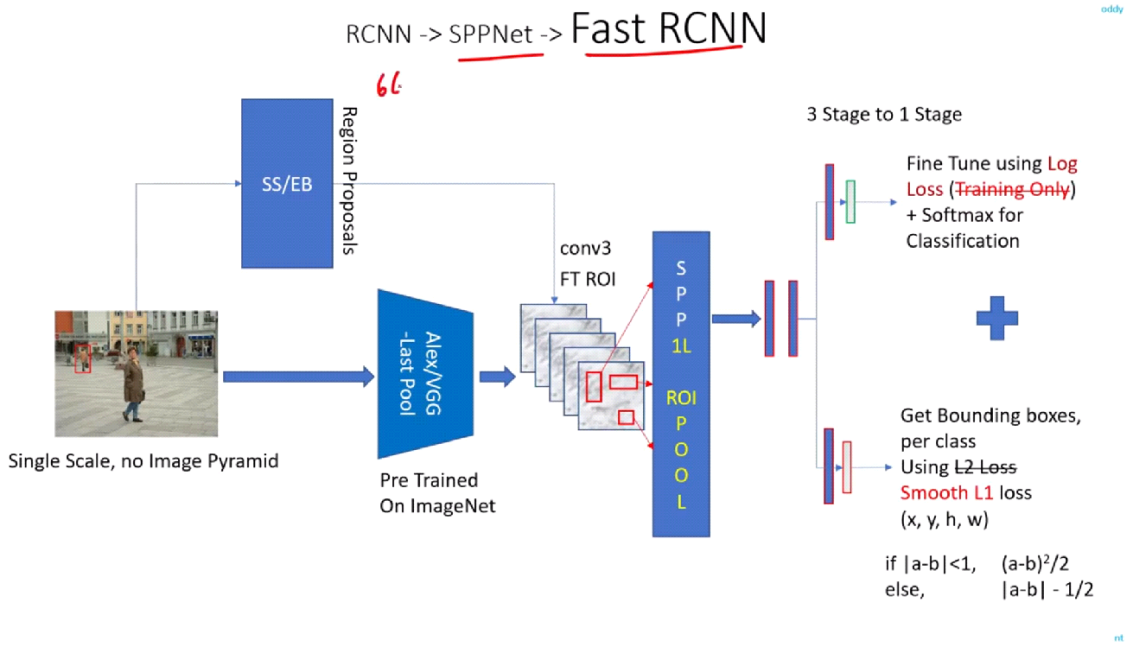
- Thus the regions we get from selective search were mapped on Feature maps from ConvNets
- The subsampling ratio of ConvNets were used to project them on Feature maps



- As we already had bounding box we only needed to refine them and only adjust delta dx,dy,dw,dh by regression

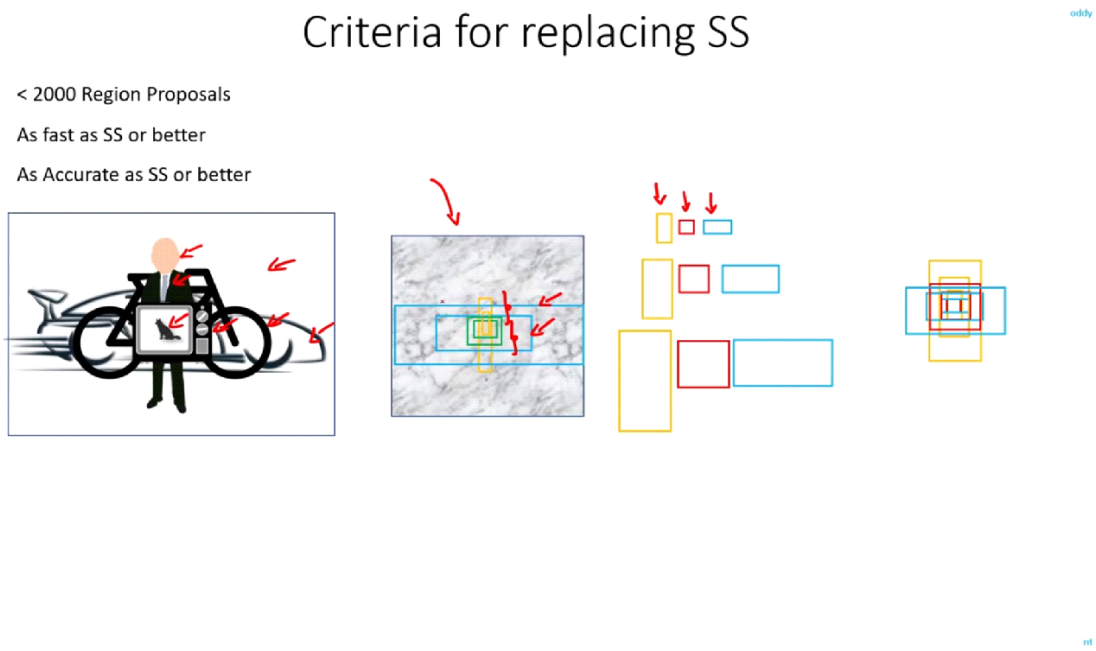


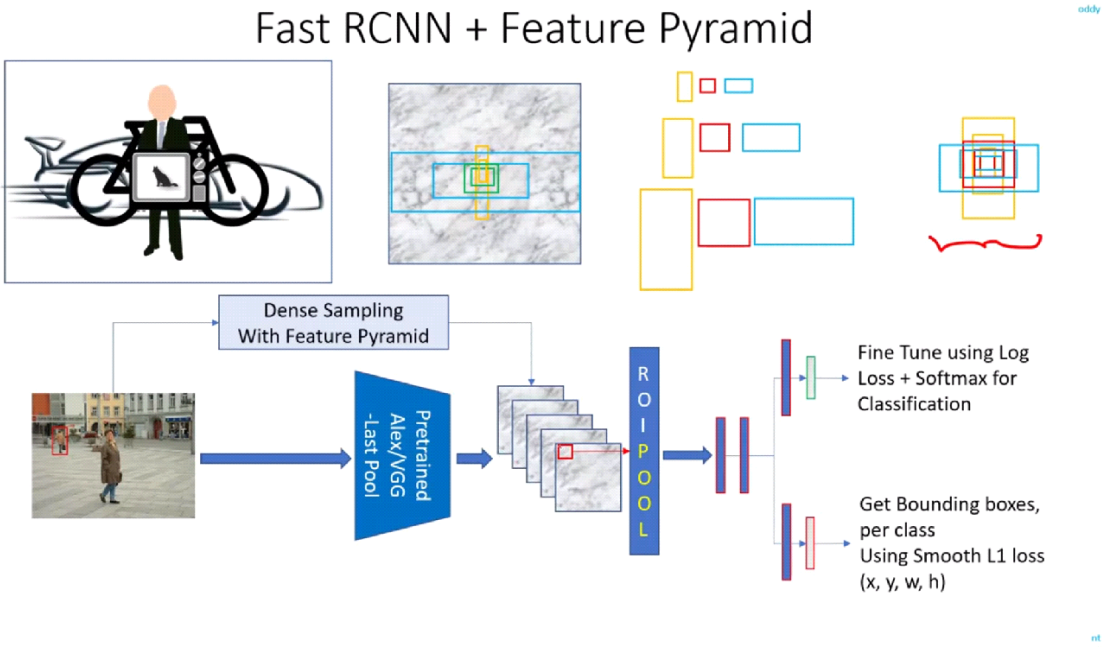
- Thus in Fast Rcnn we used training in we used ROI which decreased our time

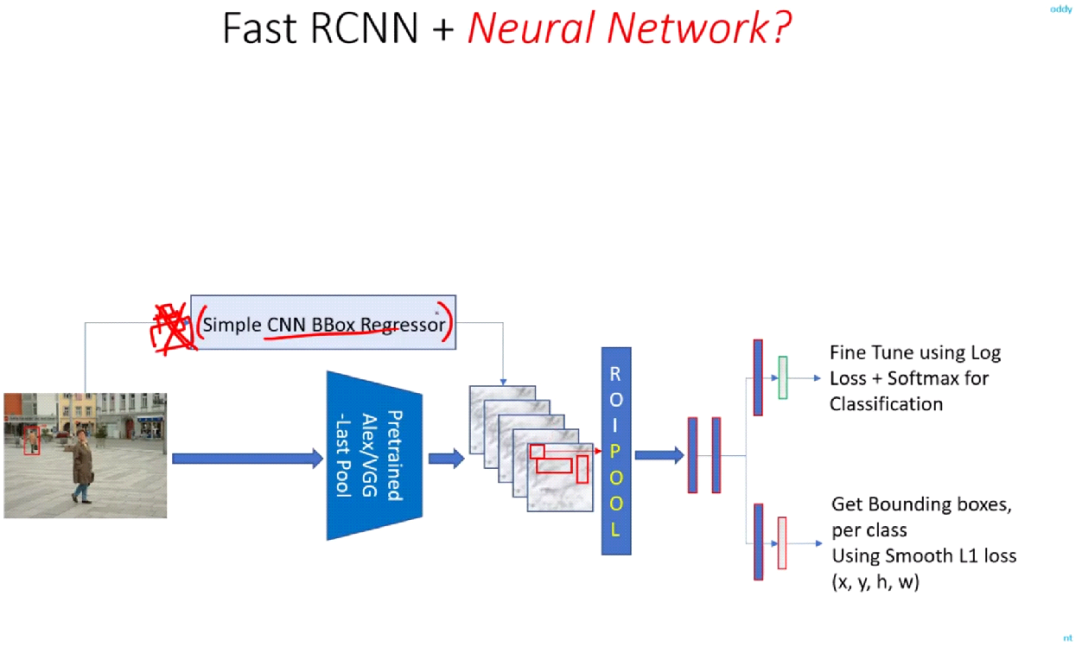

RCNN -> SPPNet -> Fast RCNN
SS/EB
Region Proposals
3 Stage to 1 Stage
Fine Tune using Log Loss (Training Only) + Softmax for Classification
conv3
FT ROI
S P P 1L
ROI P O O L
Alex/VGG -Last Pool
Single Scale, no Image Pyramid
Pre Trained On ImageNet
Get Bounding boxes, per class Using L2 Loss Smooth L1 loss (x, y, h, w)
if |a-b|<1, (a-b)²/2
else, |a-b| - 1/2

Faster Rcnn

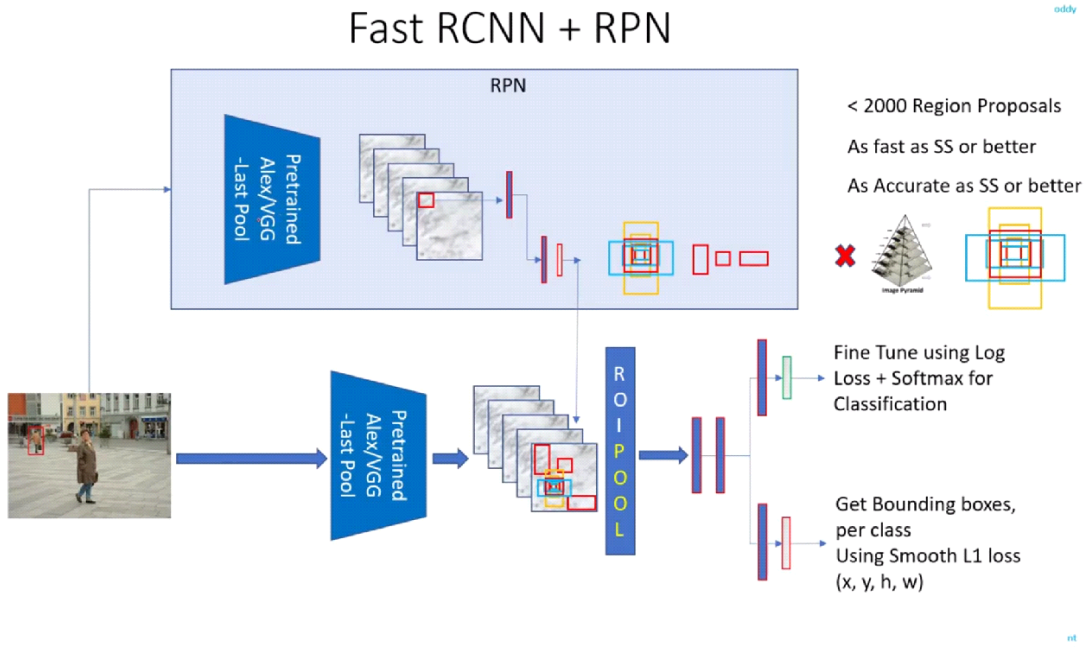
- In Faster Rcnn we replace selective search methods with dense sampling that is sliding window technique
- How do we get bounding box of different size objects?By using sliding windows of different sizes and then using relative bounding box regressor



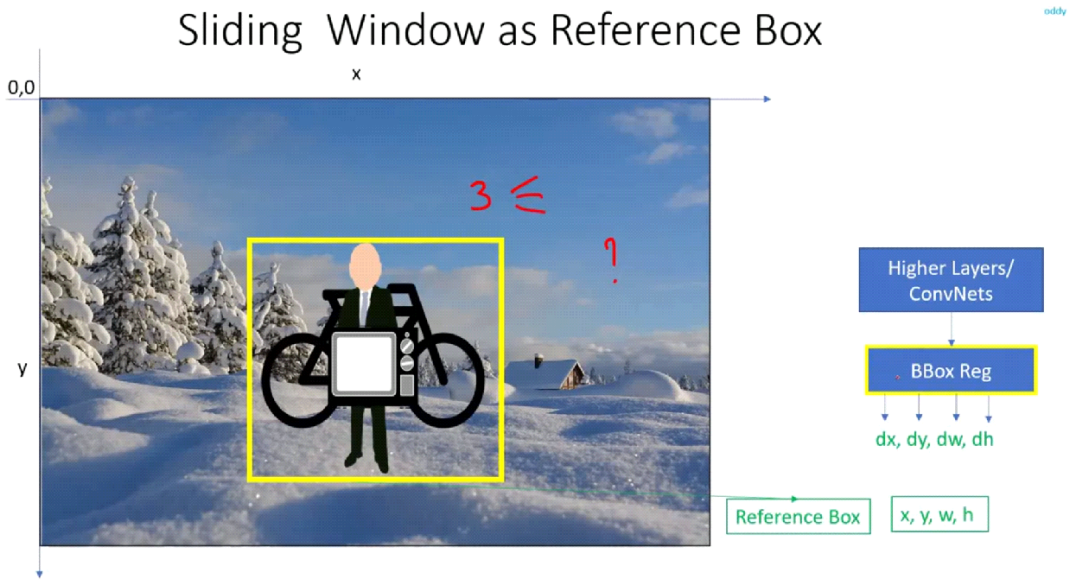




- Thus we introduce RPN region proposal network that is just a ConvNets to predict bounding Box

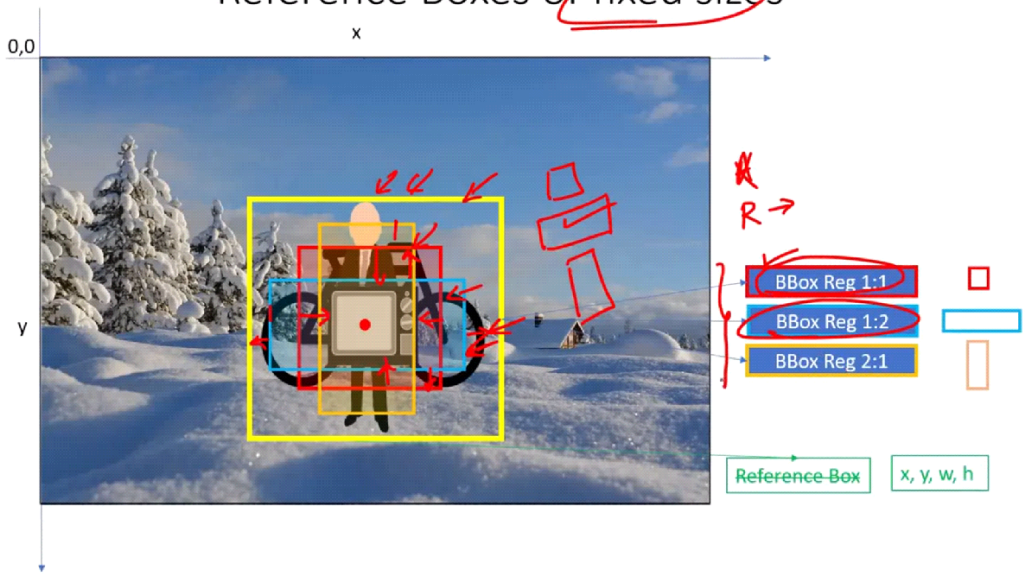


- Thus we use 3 types of bounding box of 3 different type total is 9 which which then by relative BBox give us regions which are called <u>Anchors.</u>



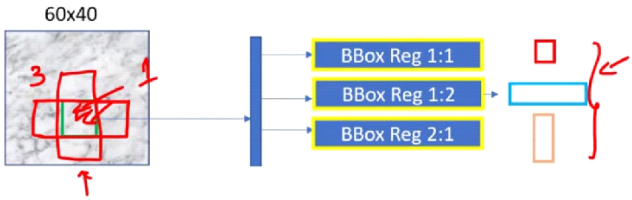
Note: This is not the exact formula used in the Faster RCNN paper. I just used this for simpler explanation. Please see the paper for details.

Reference Boxes of fixed sizes



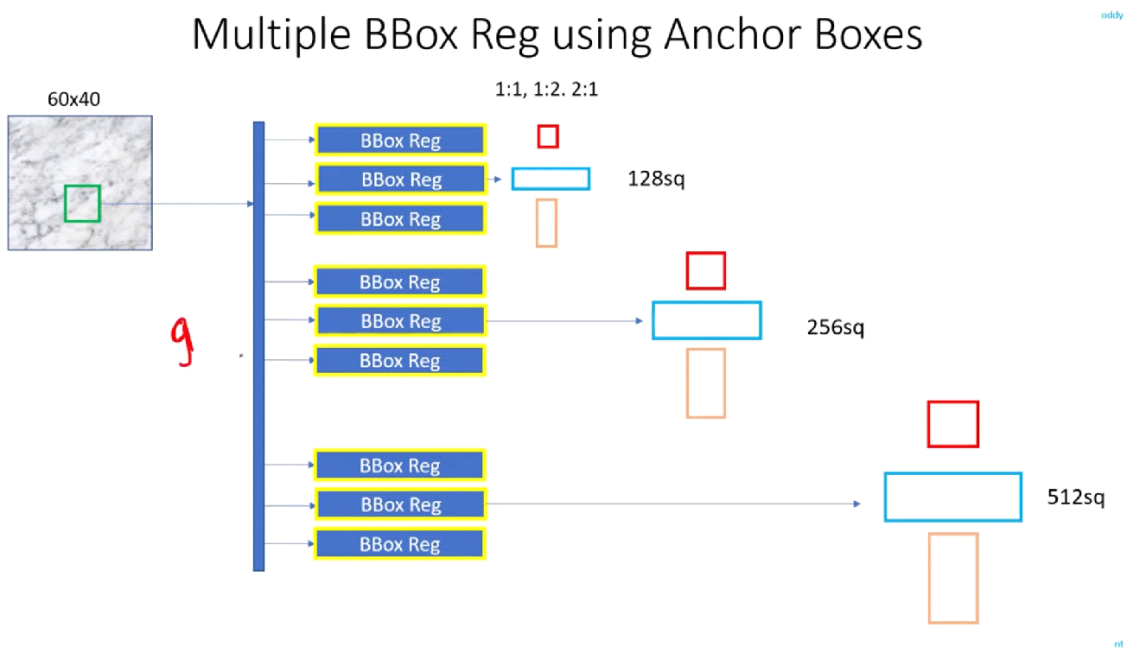
Note: This is not the exact formula used in the Faster RCNN paper. I just used this for simpler explanation. Please see the paper for details.

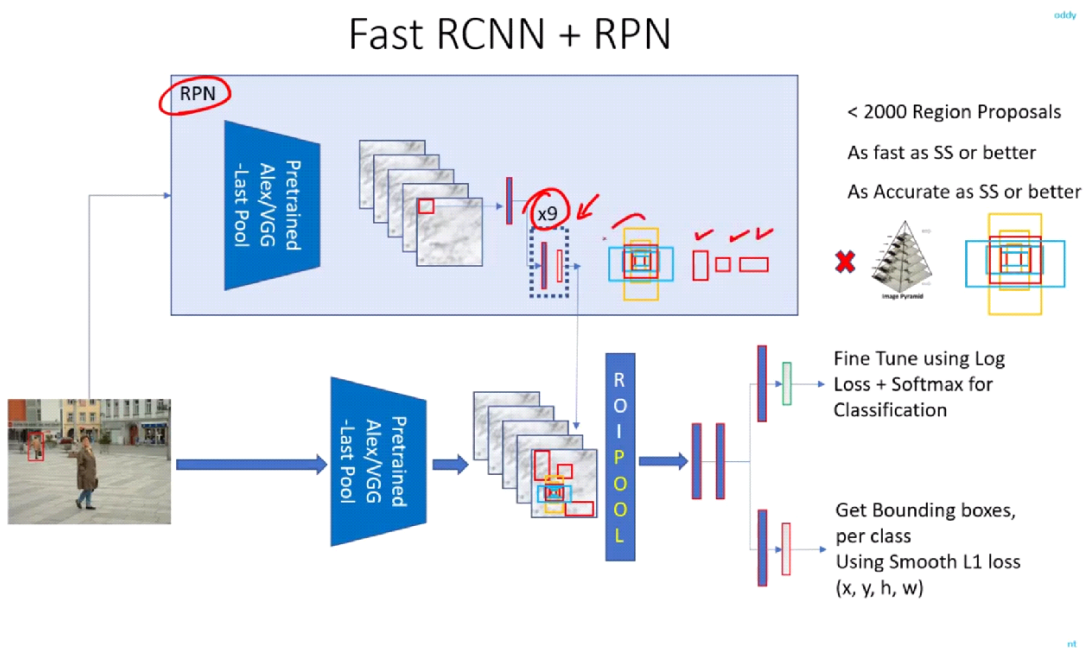
Multiple BBox Reg using Reference Boxes



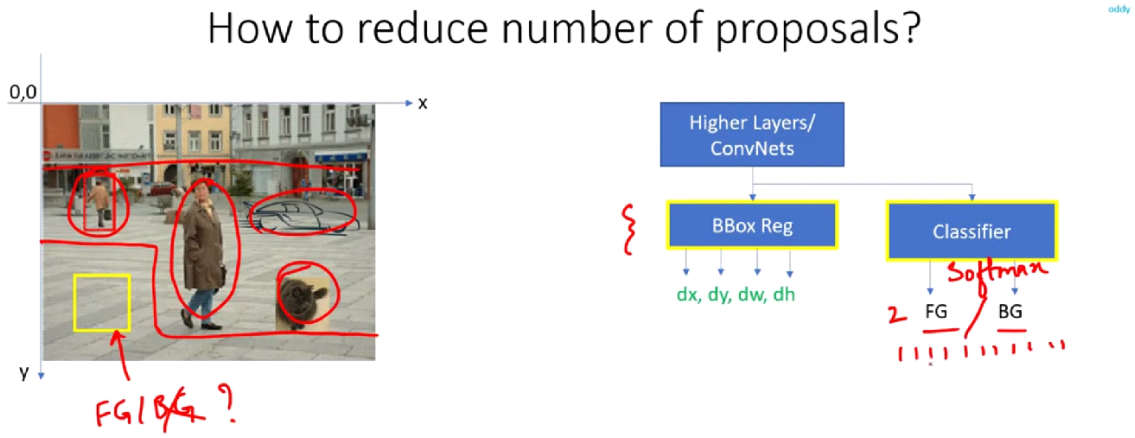
This is different from Feature Pyramid

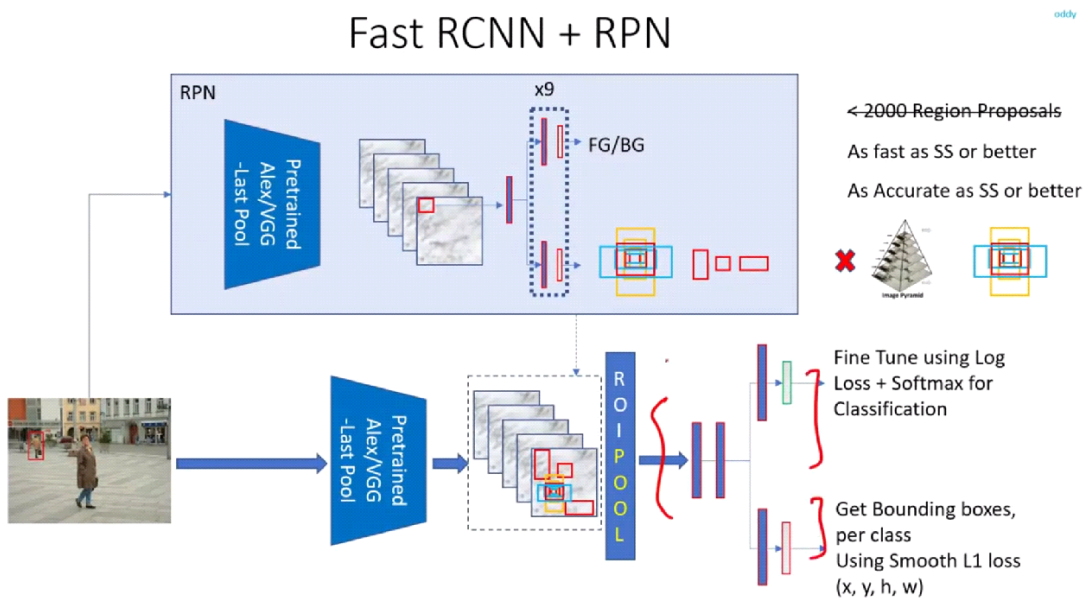
These boxes are called Anchor Boxes



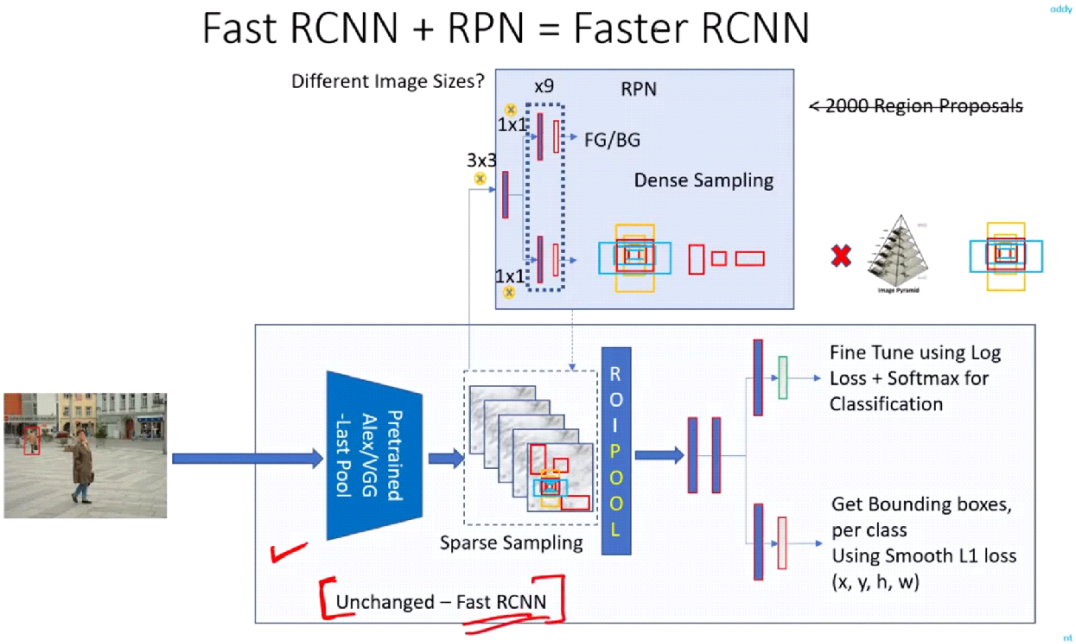


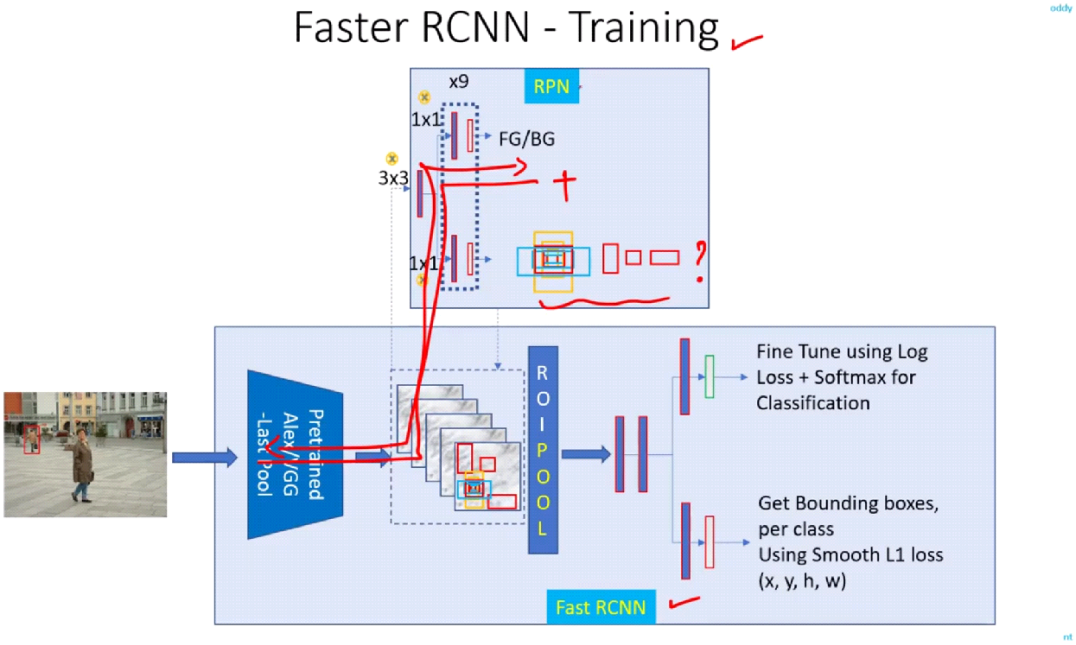
- To reduce number of classifier we introduce the Foreground and Background prediction and use a classifier to tell if region is goreground or background

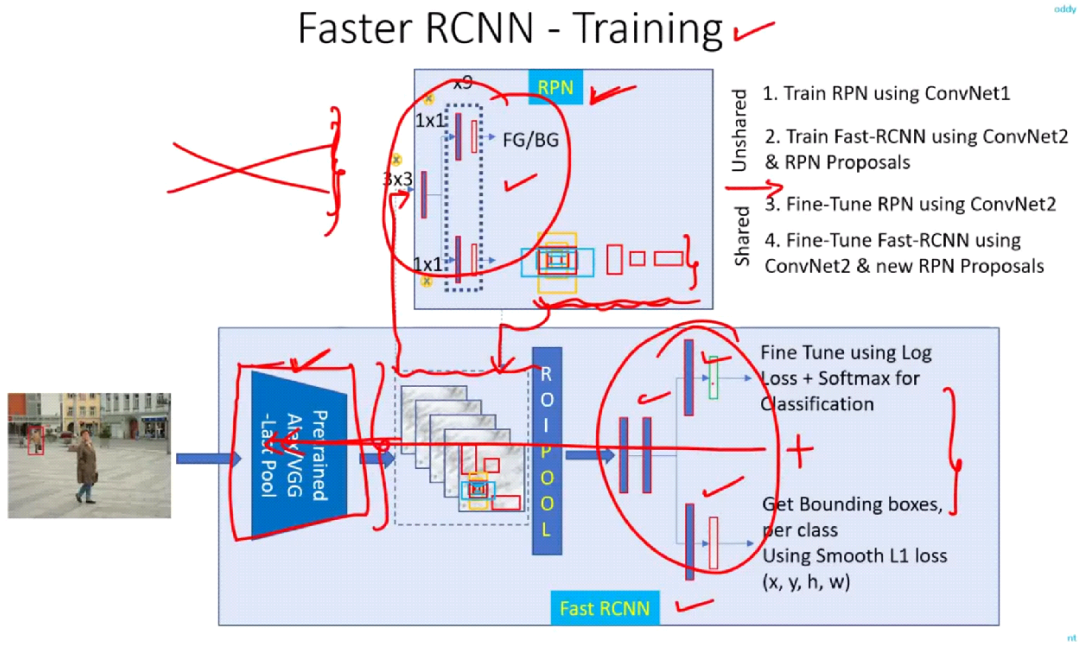




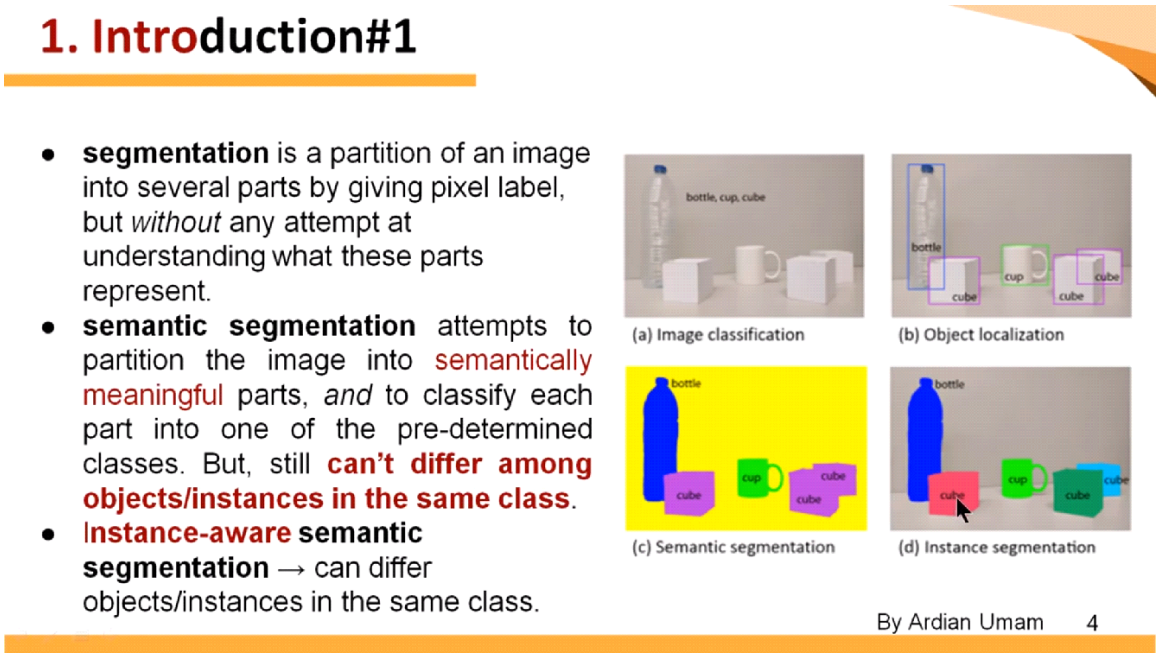
- Thus we get a Faster Rcnn and which is trained in 4 stages given below


Fast RCNN + RPN = Faster RCNN
Different Image Sizes?
x9
RPN
1x1
FG/BG
3x3
Dense Sampling
1x1
< 2000 Region Proposals
Pretrained Alex/VGG -Last Pool
ROI POOL
Fine Tune using Log Loss + Softmax for Classification
Get Bounding boxes, per class Using Smooth L1 loss (x, y, h, w)
Sparse Sampling
Unchanged – Fast RCNN


Faster RCNN - Training
x9
RPN
1x1
FG/BG
3x3
1x1
Pretrained Alex/VGG -Last Pool
ROI POOL
Fine Tune using Log Loss + Softmax for Classification
Get Bounding boxes, per class Using Smooth L1 loss (x, y, h, w)
Fast RCNN



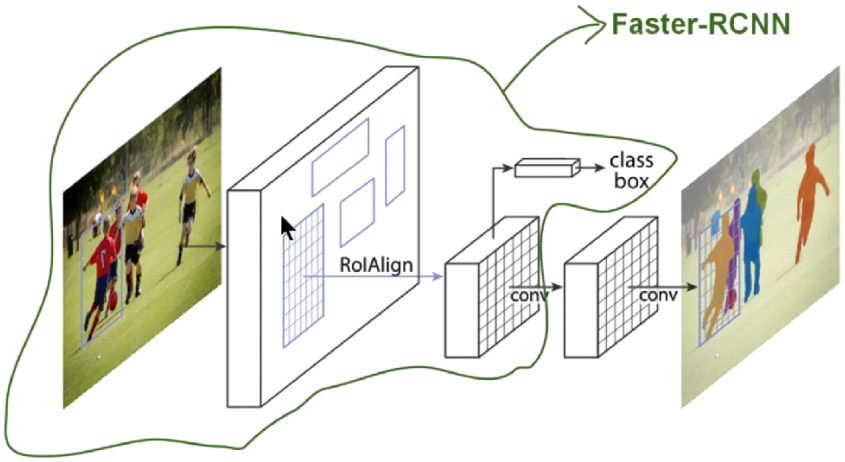
Mask Rcnn



Mask Rcnn is faster rcnn plus fully connected Network to give mask by upsampling

3. Architecture#01

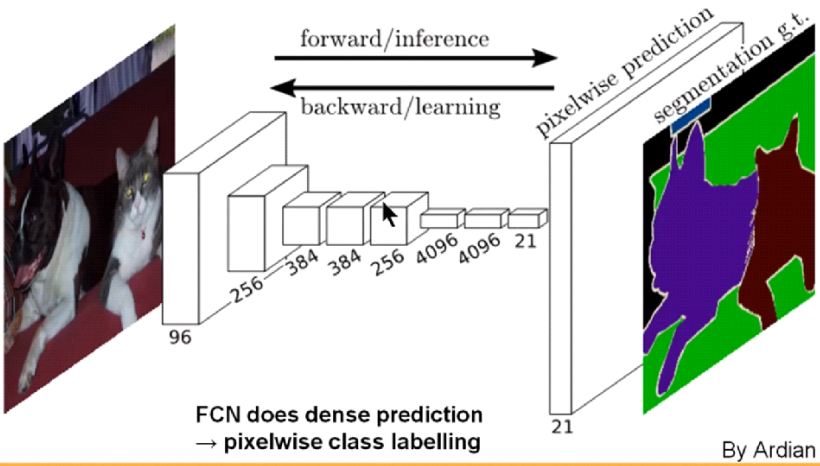
Mask R-CNN → Faster R-CNN + FCN



By Ardian Umam 9

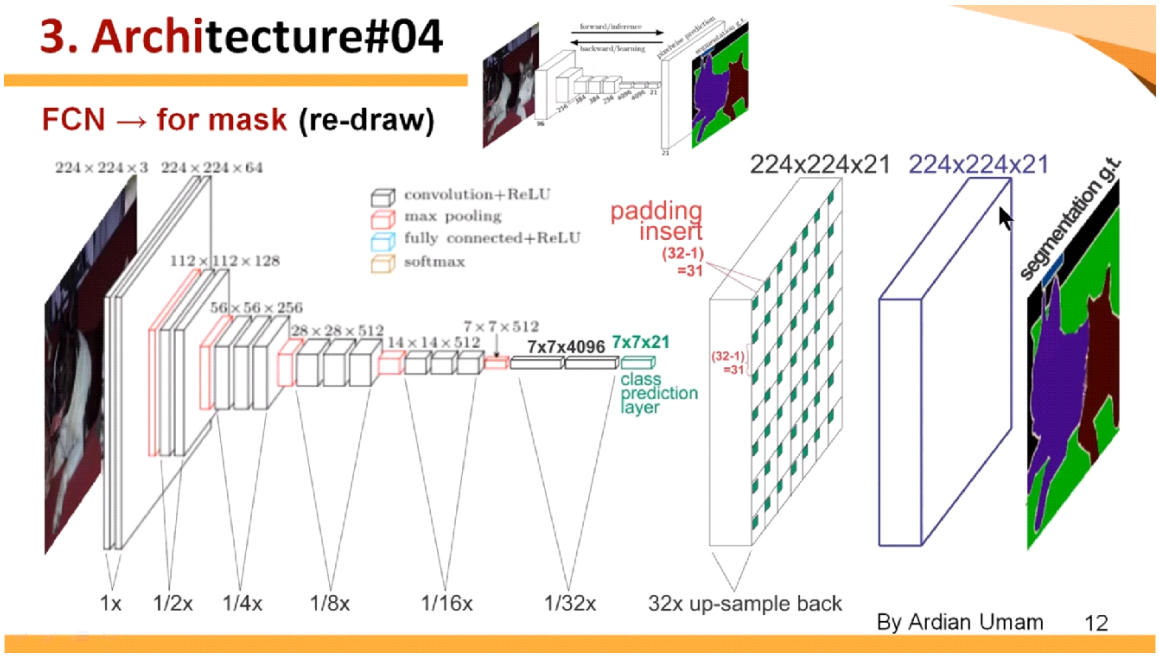
3. Architecture#03

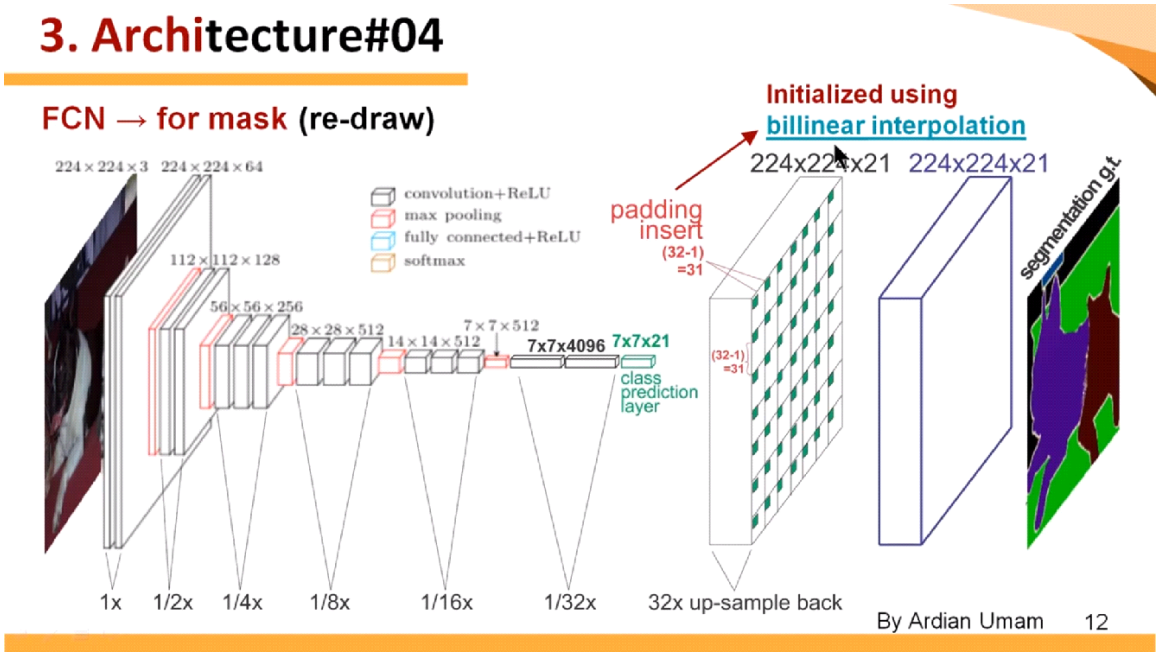
FCN → for mask



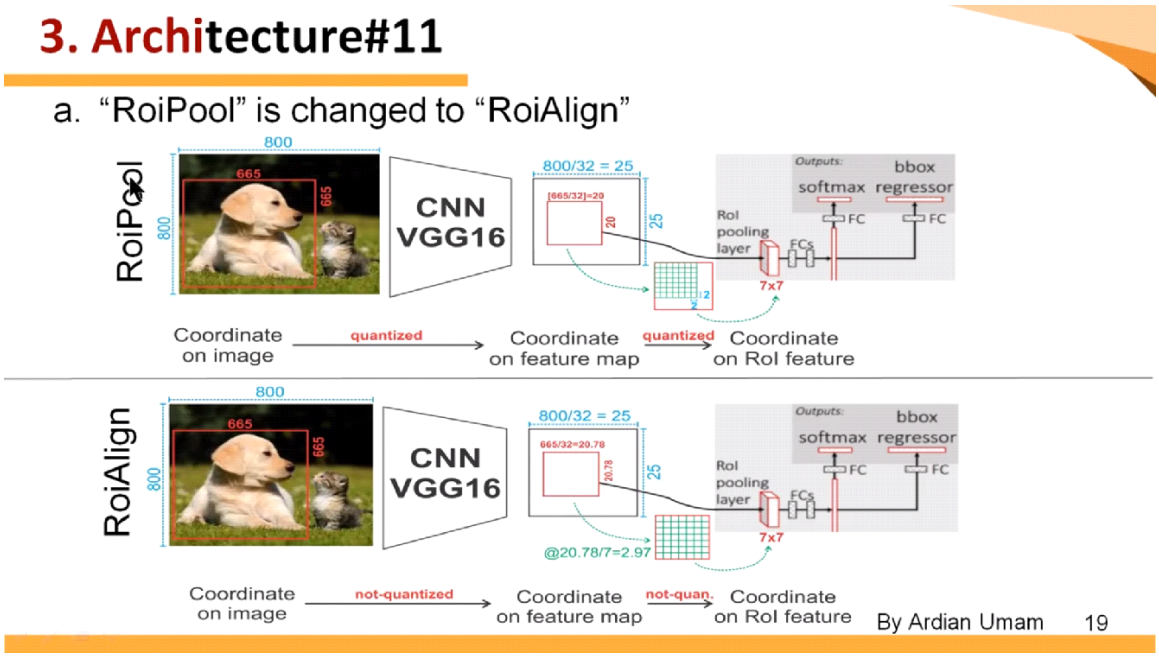
By Ardian Umam 11

- Thus we use bilinear interpolation to upsample and give mask





We use Roi Align instead of RoiPool in mask rcnn



Mask RCNN has been the new state of art in terms of instance segmentation. There are rigorous papers, easy to understand tutorials with good quality open source codes around for your reference. Here I want to share some simple understanding of it to give you a first look.

Mask RCNN is a deep neural network aimed to solve instance segmentation problem in machine learning or computer vision. In other words, it can separate different objects in a image or a video. You give it a image, it gives you the object bounding boxes, classes and masks.

There are two stages of Mask RCNN. First, it generates proposals about the regions where there might be an object based on the input image. Second, it predicts the class of the object, refines the bounding box and generates a mask in pixel level of the object based on the first stage proposal. Both stages are connected to the backbone structure.

What is backbone? Backbone is a FPN style deep neural network. It consists of a bottom-up pathway , a top-bottom pathway and lateral connections. Bottom-up pathway can be any ConvNet, usually ResNet or VGG, which extracts features from raw images. Top-bottom pathway generates feature pyramid map which is similar in size to bottom-up pathway. Lateral connections are convolution and adding operations between two corresponding levels of the two pathways. FPN outperforms other single ConvNets mainly for the reason that it maintains strong semantically features at various resolution scales.

Now let's look at the first stage. A light weight neural network called RPN scans all FPN top-bottom pathway(hereinafter referred to feature map) and proposes regions which may contain objects. That's all it is. While sccaning feature map is an efficient way, we need a method to bind features to its raw image location. Here come the anchors. Anchors are a set of boxes with predefined locations and scales relative to images. Ground-truth classes(only object or background binary classified at this stage) and bounding boxes are assigned to individual anchors according to some IoU value. As anchors with different scales bind to different levels of feature map, RPN uses these anchors to figure out where of the feature map 'should' get an object and what size of its bounding box is. Here we may agree that

convolving, downsampling and upsampling would keep features staying the same relative locations as the objects in original image, and wouldn't mess them around.

At the second stage, another neural network takes proposed regions by the first stage and assign them to several specific areas of a feature map level, scans these areas, and generates objects classes(multi-categorical classified), bounding boxes and masks. The procedure looks similar to RPN. Differences are that without the help of anchors, stage-two used a trick called ROIAlign to locate the relevant areas of feature map, and there is a branch generating masks for each objects in pixel level. Work completed.

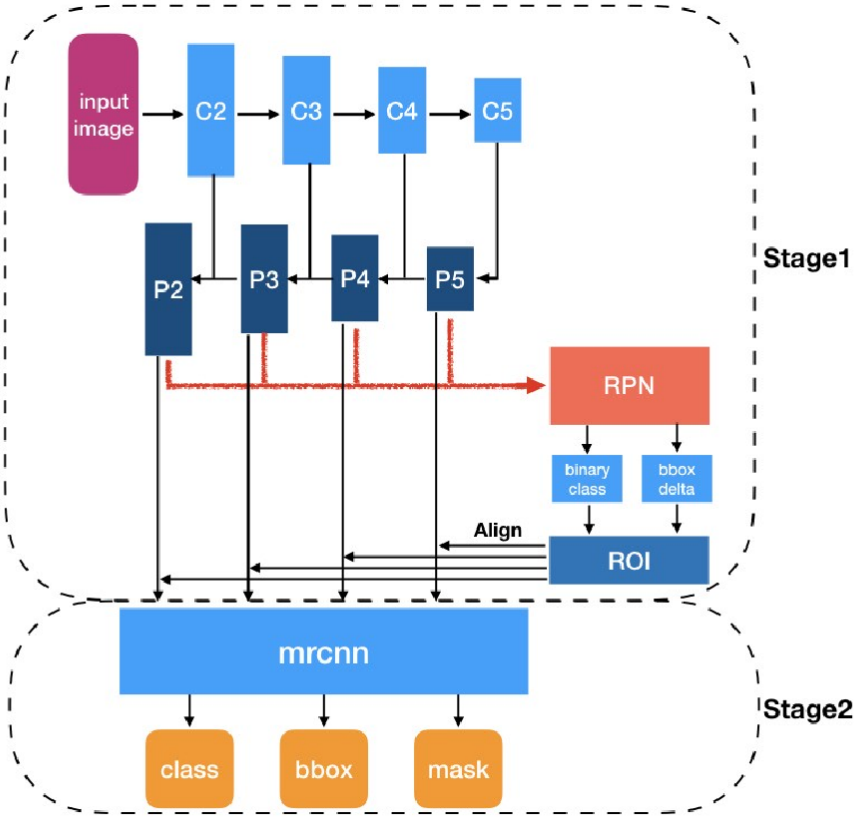
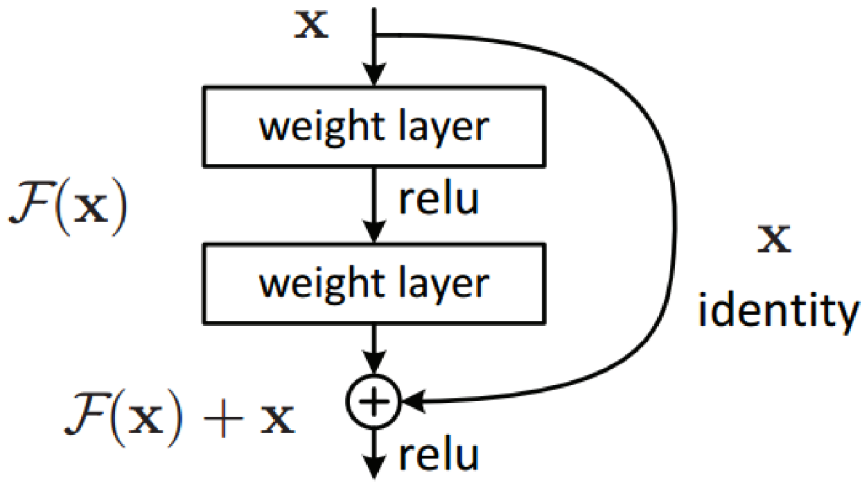


Illustration of Mask RCNN structure

The most inspiring things I found about Mask RCNN is that we could actually force different layers in neural network to learn features with different scales, just like the anchors and ROIAlign, instead of treating layers as black box.

Residual Network



Why we need residual blocks?

- As we build deep network our accuracy drops and we dont get accuracy related to shallow network
- Hence We use skip connections to learn The residual or the Identity block that should be added or subtracted to input to get desired output

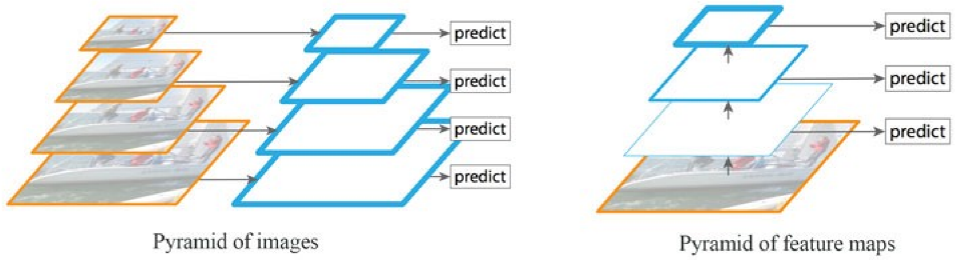
R(x) = Output — Input = H(x) — x

Rearranging it, we get, H(x) = R(x) + x

Our residual block is overall trying to learn the true output, H(x) and if you look closely at the image above, you will realize that since we have an identity connection coming due to x, the layers are actually trying to learn the residual, R(x). So to summarize, the layers in a traditional network are learning the true output (H(x))whereas the layers in a residual network are learning the residual (R(x)). Hence, the name: Residual Block.

FPN (Feature Pyramid Network)

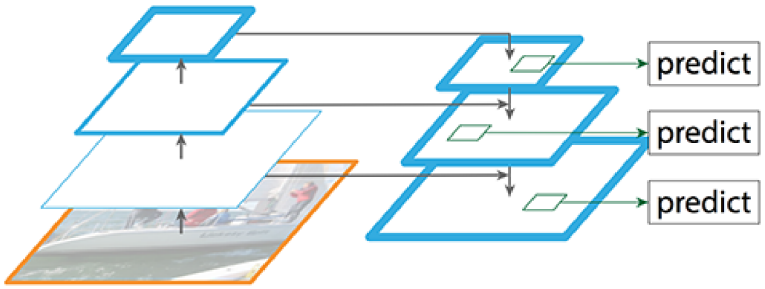
Detecting objects in different scales is challenging in particular for small objects. We can use a pyramid of the same image at different scale to detect objects (the left diagram below). However, processing multiple scale images is time consuming and the memory demand is too high to be trained end-to-end simultaneously. Hence, we may only use it in inference to push accuracy as high as possible, in particular for competitions, when speed is not a concern. Alternatively, we create a pyramid of feature and use them for object detection (the right diagram). However, feature maps closer to the image layer composed of low-level structures that are not effective for accurate object detection.



Source

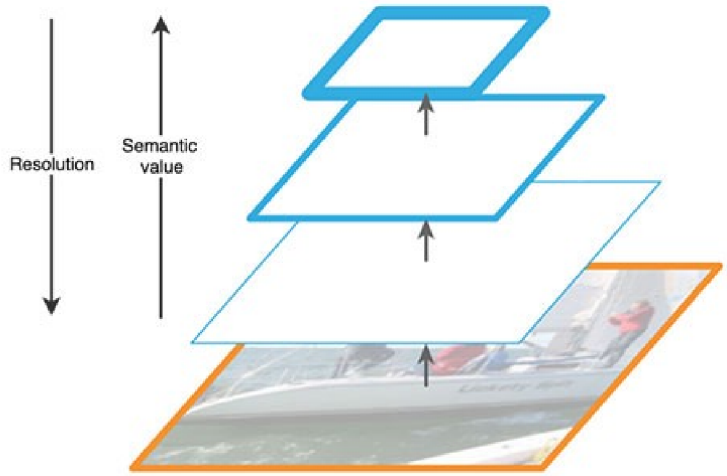
Feature Pyramid Network (FPN) is a feature extractor designed for such pyramid concept with accuracy and speed in mind. It replaces the feature extractor of detectors like Faster R-CNN and generates multiple feature map layers (multi-scale feature maps) with better quality information than the regular feature pyramid for object detection.

Data Flow



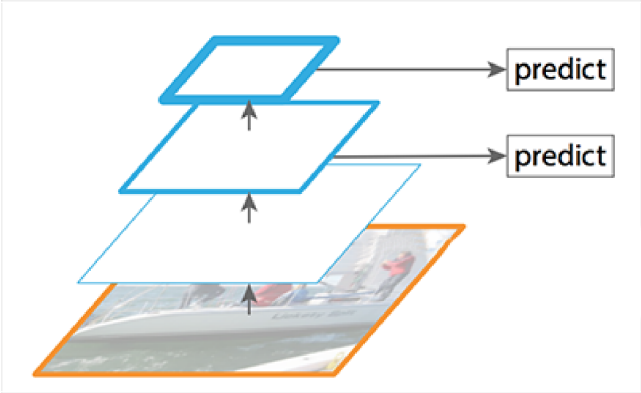
FPN (Source)

FPN composes of a bottom-up and a top-down pathway. The bottom-up pathway is the usual convolutional network for feature extraction. As we go up, the spatial resolution decreases. With more high-level structures detected, the semantic value for each layer increases.



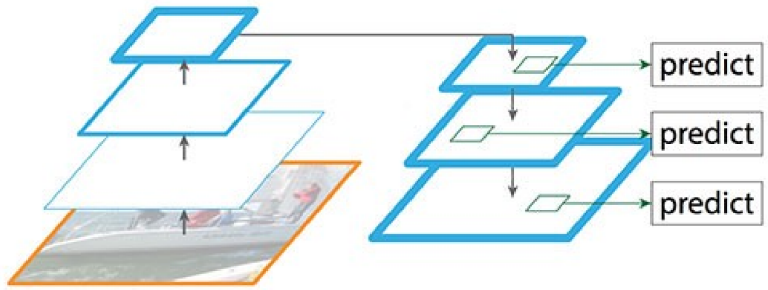
Feature extraction in FPN (Modified from source)

SSD makes detection from multiple feature maps. However, the bottom layers are not selected for object detection. They are in high resolution but the semantic value is not high enough to justify its use as the speed slow-down is significant. So SSD only uses upper layers for detection and therefore performs much worse for small objects.



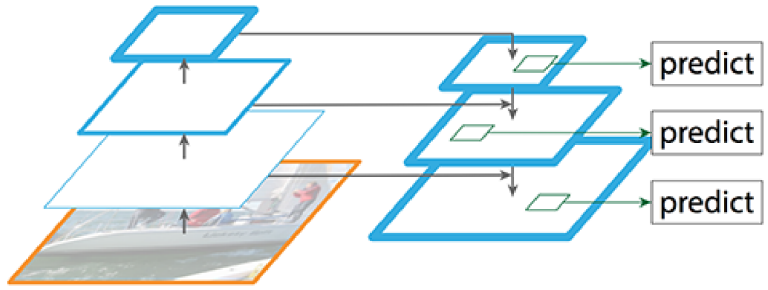
Modified from source

FPN provides a top-down pathway to construct higher resolution layers from a semantic rich layer.



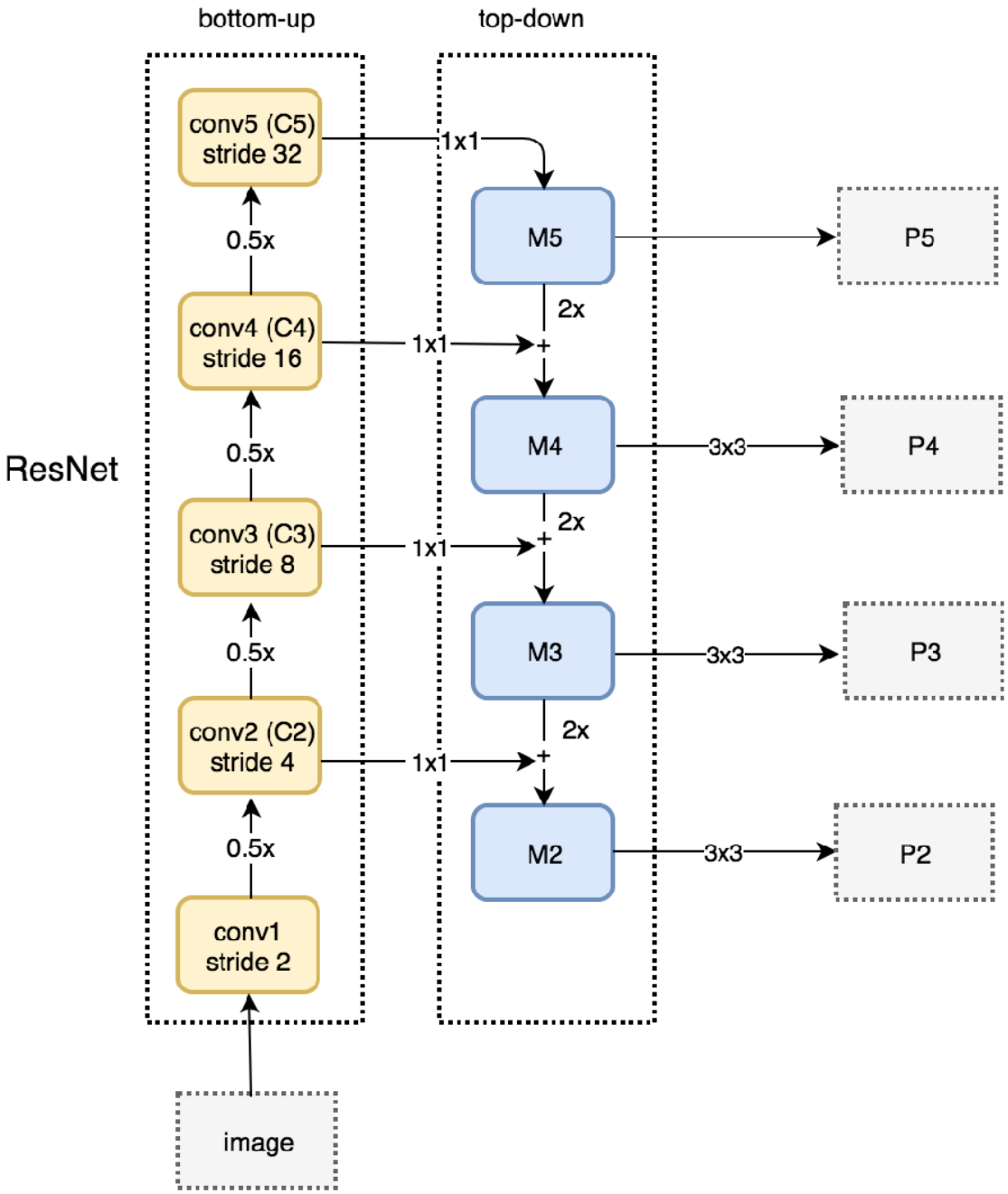
Reconstruct spatial resolution in the top-down pathway. (Modified from source)

While the reconstructed layers are semantic strong but the locations of objects are not precise after all the downsampling and upsampling. We add lateral connections between reconstructed layers and the corresponding feature maps to help the detector to predict the location betters. It also acts as skip connections to make training easir (similar to what ResNet does).



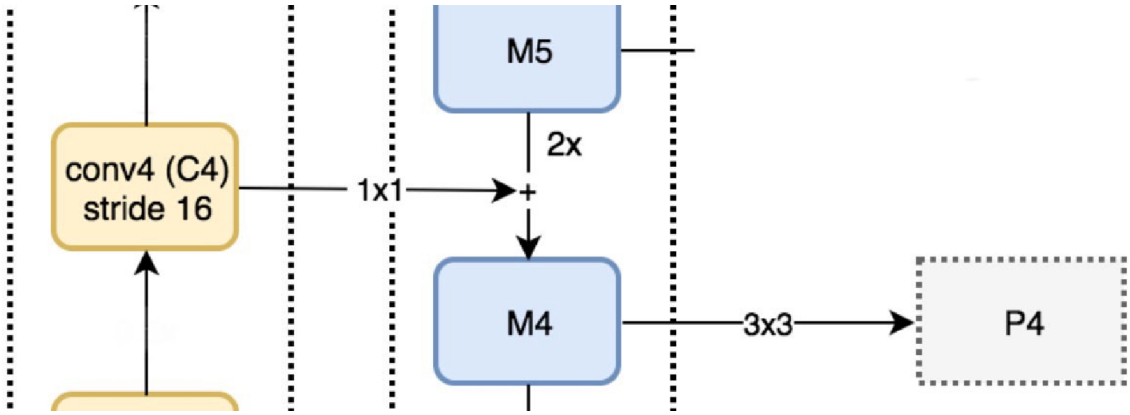
Add skip connections (Source)

Bottom-up pathway



The bottom-up pathway uses ResNet to construct the bottom-up pathway. It composes of many convolution modules (convi for i equals 1 to 5) each has many convolution layers. As we move up, the spatial dimension is reduced by 1/2 (i.e. double the stride). The output of each convolution module is labeled as Ci and later used in the top-down pathway.

Top-down pathway



We apply a 1 × 1 convolution filter to reduce C5 channel depth to 256-d to create M5. This becomes the first feature map layer used for object prediction.

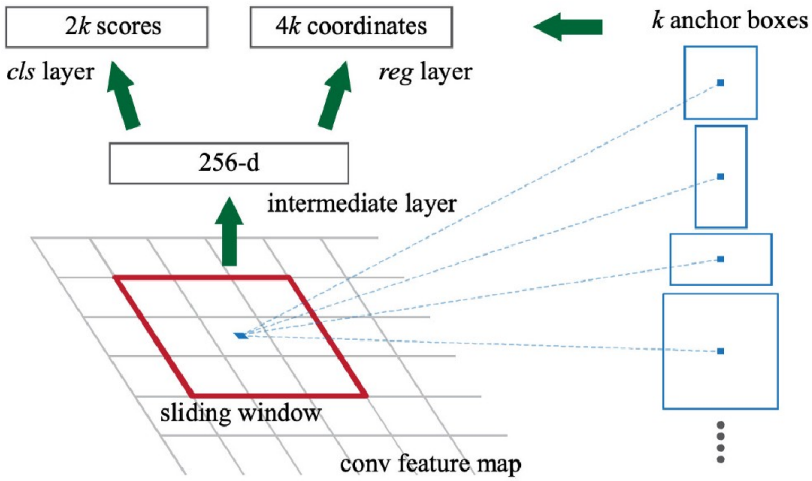
As we go down the top-down path, we upsample the previous layer by 2 using nearest neighbors upsampling. We again apply a 1 × 1 convolution to the corresponding feature maps in the bottom-up pathway. Then we add them element-wise. We apply a 3 × 3 convolution to all merged layers. This filter reduces the aliasing effect when merged with the upsampled layer.

We repeat the same process for P3 and P2. However, we stop at P2 because the spatial dimension of C1 is too large. Otherwise, it will slow down the process too much. Because we share the same classifier and box regressor of every output feature maps, all pyramid feature maps (P5, P4, P3 and P2) have 256-d output channels.

FPN with RPN (Region Proposal Network)

FPN is not an object detector by itself. It is a feature extractor that works with object detectors.

FPN extracts feature maps and later feeds into a detector, says RPN, for object detection. RPN applies a sliding window over the feature maps to make predictions on the objectness (has an object or not) and the object boundary box at each location.

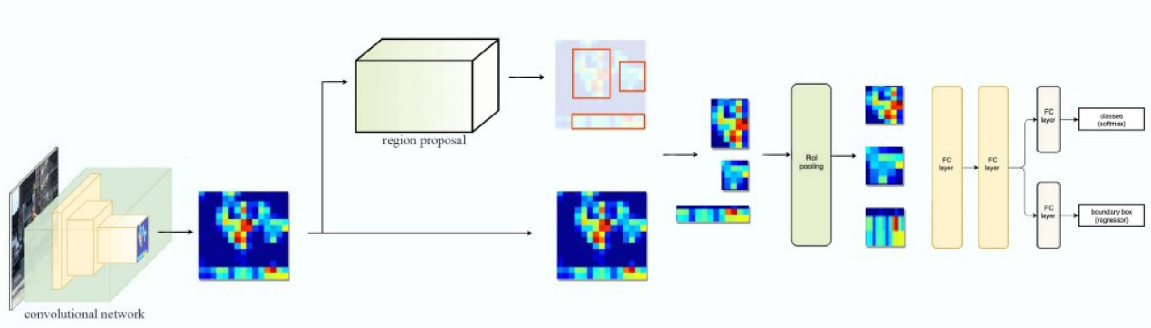


Source

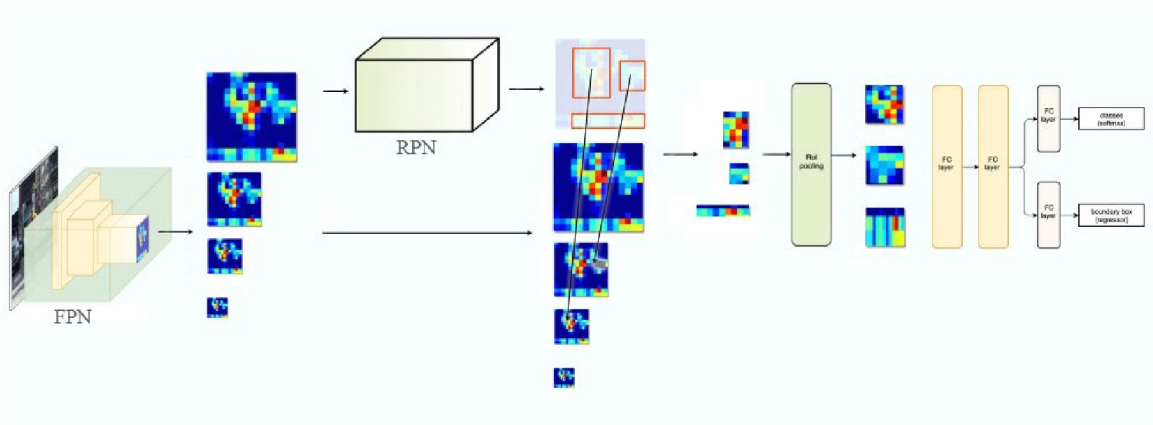
In the FPN framework, for each scale level (say P4), a 3 × 3 convolution filter is applied over the feature maps followed by separate 1 × 1 convolution for objectness predictions and boundary box regression. These 3 × 3 and 1 × 1 convolutional layers are called the RPN head. The same head is applied to all different scale levels of feature maps.

FPN with Fast R-CNN or Faster R-CNN

Let's take a quick look at the Fast R-CNN and Faster R-CNN data flow below. It works with one feature map layer to create ROIs. We use the ROIs and the feature map layer to create feature patches to be fed into the ROI pooling.



In FPN, we generate a pyramid of feature maps. We apply the RPN (described in the previous section) to generate ROIs. Based on the size of the ROI, we select the feature map layer in the most proper scale to extract the feature patches.

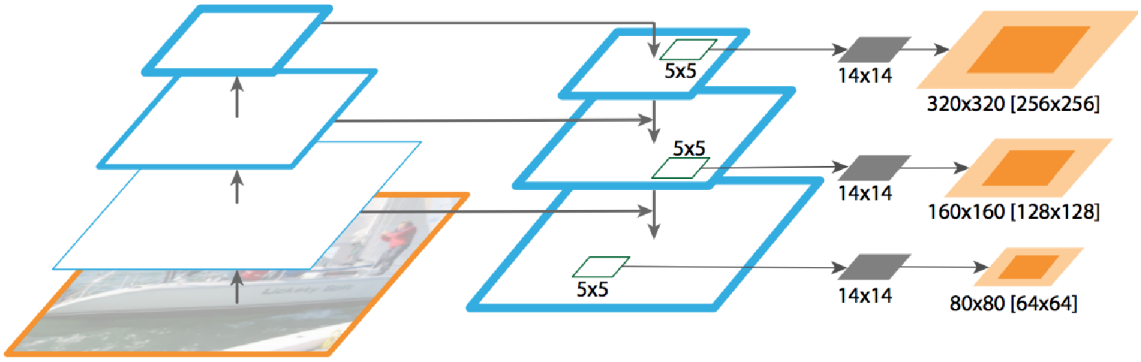


The formula to pick the feature maps is based on the width w and height h of the ROI.

So if k = 3, we select P3 as our feature maps. We apply the ROI pooling and feed the result to the Fast R-CNN head (Fast R-CNN and Faster R-CNN have the same head) to finish the prediction.

Segmentation

Just like Mask R-CNN, FPN is also good at extracting masks for image segmentation. Using MLP, a 5 × 5 window is slide over the feature maps to generate an object segment of dimension 14 × 14 segments. Later, we merge masks at a different scale to form our final mask predictions.


5x5
5x5
5x5
14x14
14x14
14x14
320x320 [256x256]
160x160 [128x128]
80x80 [64x64]