Inception

The inception network was an important milestone in the development of CNN classifier. Prior to its inception, most popular CNNs just stacked convolution layers deeper and deeper, hoping to get better performance.

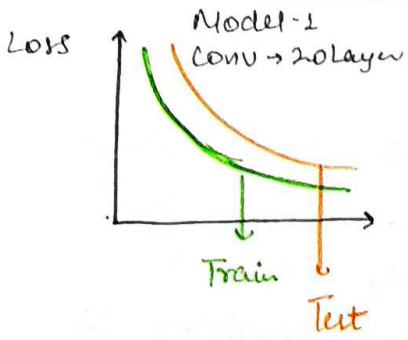
The Inception network on the other hand, was complex (heavily engineered). It used a lot of tricks to push performance; both in terms of speed and accuracy. Its constant evolution lead to the creation of several versions of the network. The popular versions are as follows:

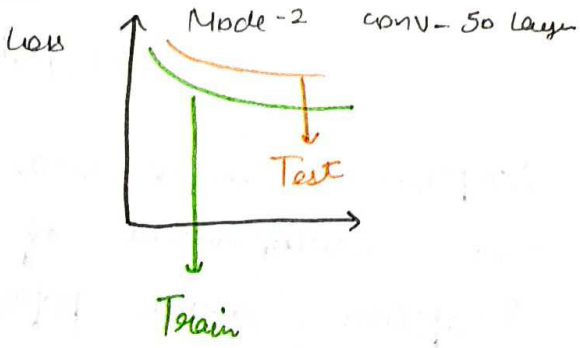
Inception V1

Inception V_2 and Inception V_3

Inception V_4 and Inception-ResNet.

Each version is an iterative improvement over the previous one. Understanding the upgrades can help us to build custom classifiers that are optimized both in speed and accuracy. Also





At a certain conv layer → loss decrease
After that increase the layer → loss increase

conv layer increasing → Loss start decreasing

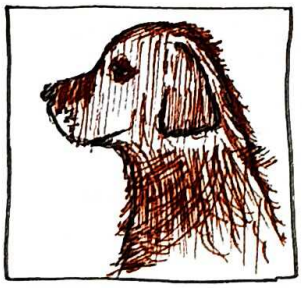
conv layer increasing → Loss steady

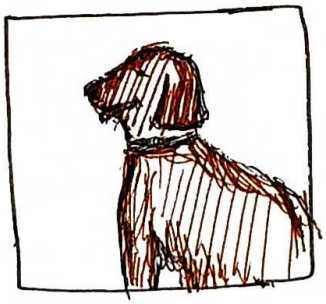
conv layer increasing → Loss start increasing

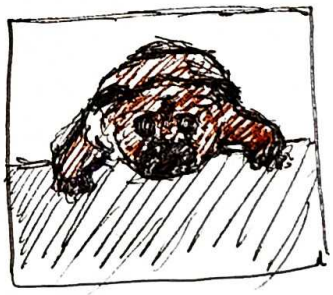
Inception V1

The Premise:

Salient parts in the image can have extremely large variation in size. For instance, an image with a dog can be either of the following, as shown below. The area occupied by the dog is different in each image.







→ Because of this huge variation in the location of the information choosing the right kernel size for the convolution operation becomes tough. A larger Kernel is preferred for information that is distributed more globally and a smaller Kernel is preferred for information that is distributed more locally.

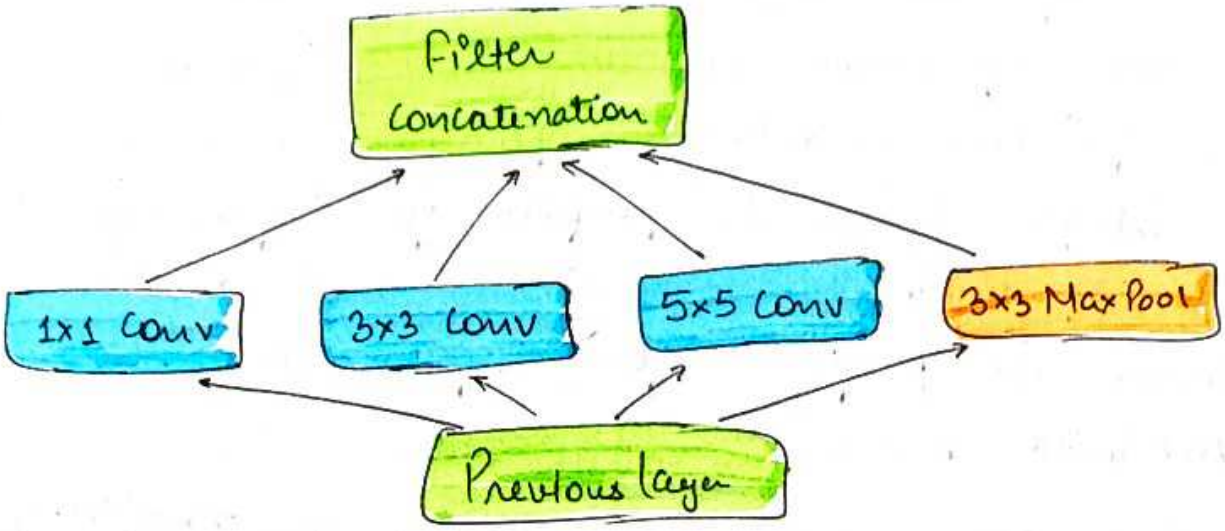
→ Very deep networks are prone to overfitting. It also hard to pass gradient update through the entire network.

→ Naively stacking large convolutional operation is computationally expensive.

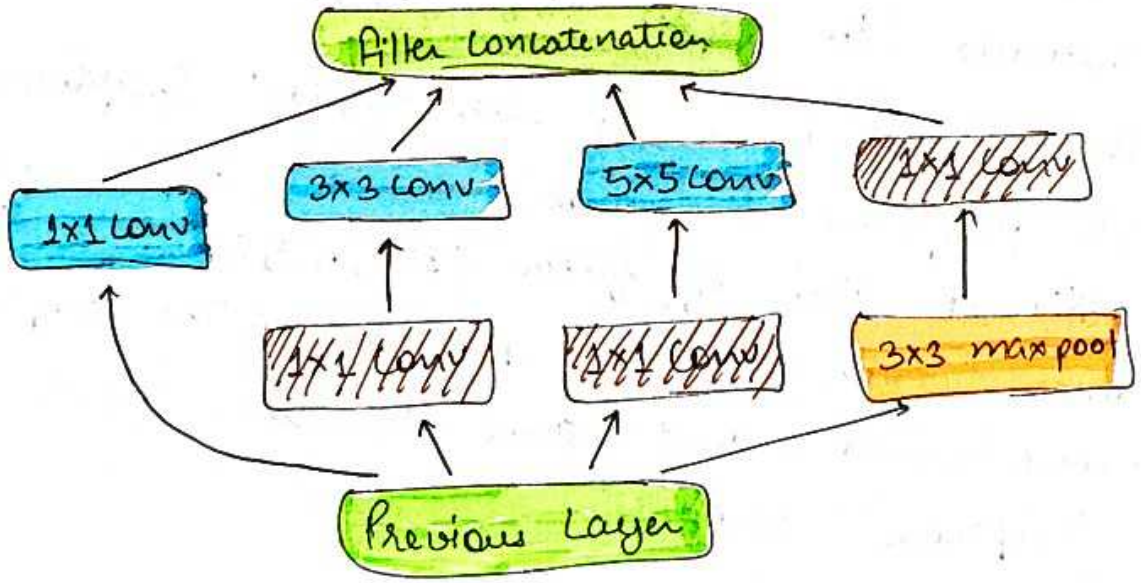
The Solution?

Why not have filter with multiple sizes operate on the same level? The network essentially would get a bit "wider" rather than "deeper". The authors designed the inception module to reflect the same.

The below image is the "naive" inception module. It performs convolution on an input, with 3 different sizes of filters (1×1, 3×3, 5×5). Additionally, max pooling is also performed. The outputs are concatenated and sent to the next inception module.



As stated before, deep neural networks are computationally expensive. To make it cheaper, the authors limit the number of input channels by adding an extra 1x1 convolutional before the 3x3 and 5x5 conv. Though adding an extra operation may seem counterintuitive, 1x1 conv are far more cheaper than 5x5 conv and the reduced number of input channel also help. Do not that however, the 1x1 conv is introduced after the max pooling layer, rather than before.



* Can also apply stride = 2 $\Longleftrightarrow$ Conv (1x1)

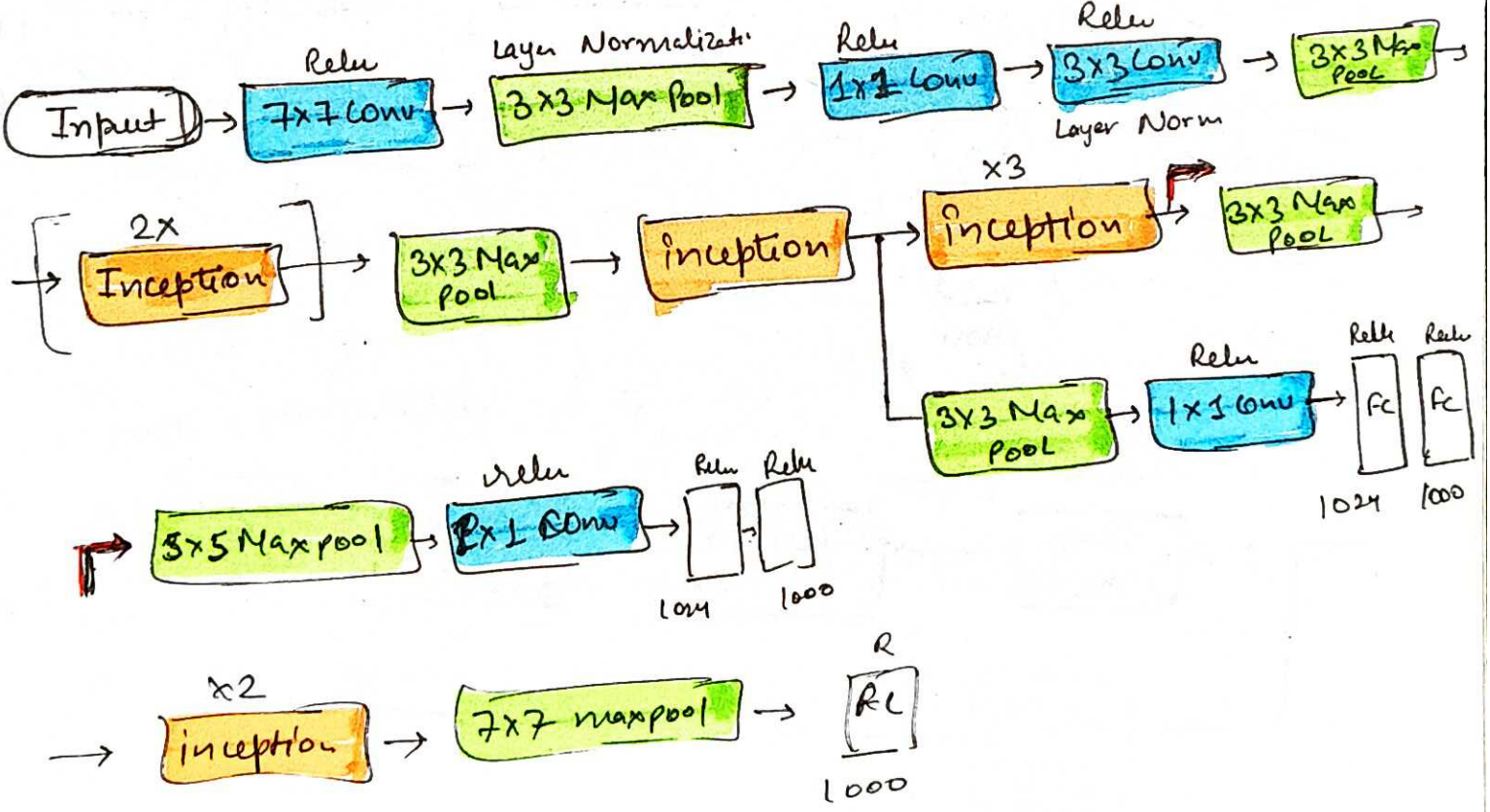
↳ work same as conv in inception.

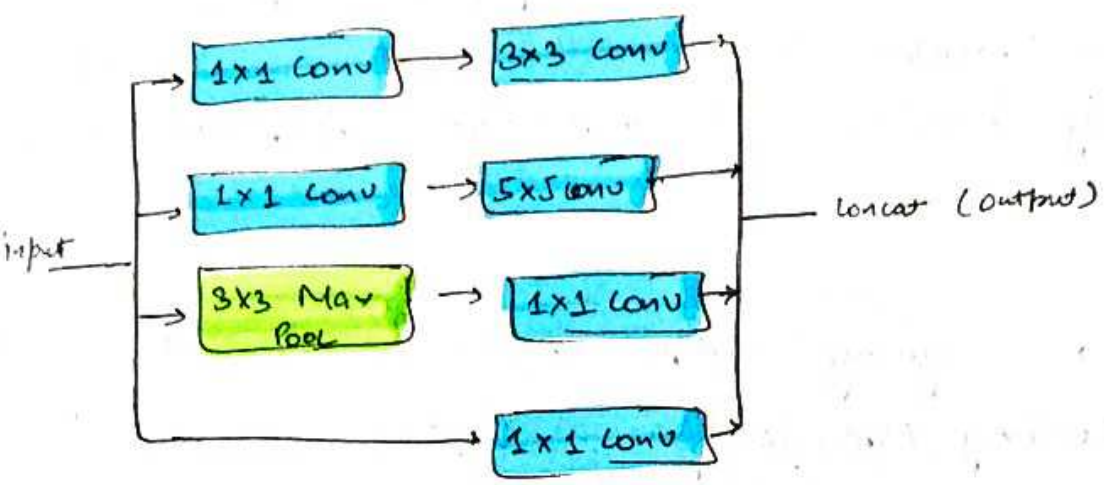
Using the dimension reduced inception module a neural networks architecture was built. This was popularly known as google Net (Inception V1)

Inception V1 → going deeper has a caveat: exploding / vanishing gradients:

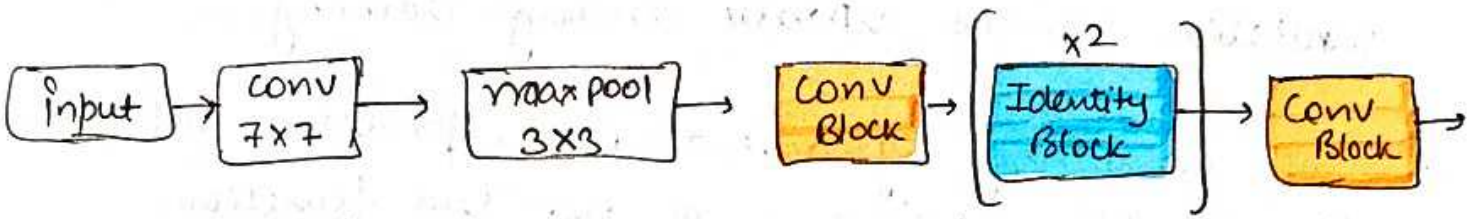
1. The exploding gradient is a problem when large error gradients accumulate and result in unstable weight updates during training.

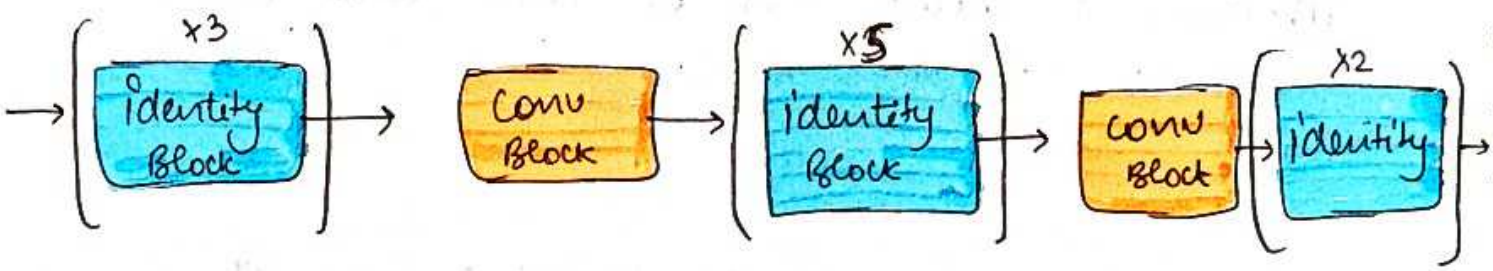
2. The vanishing gradient is a problem when the partial derivative of the loss function approaches a value close to zero and the network couldn't train

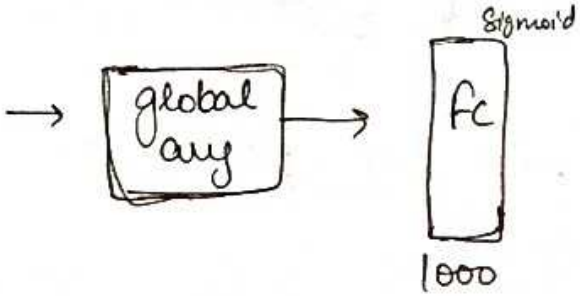


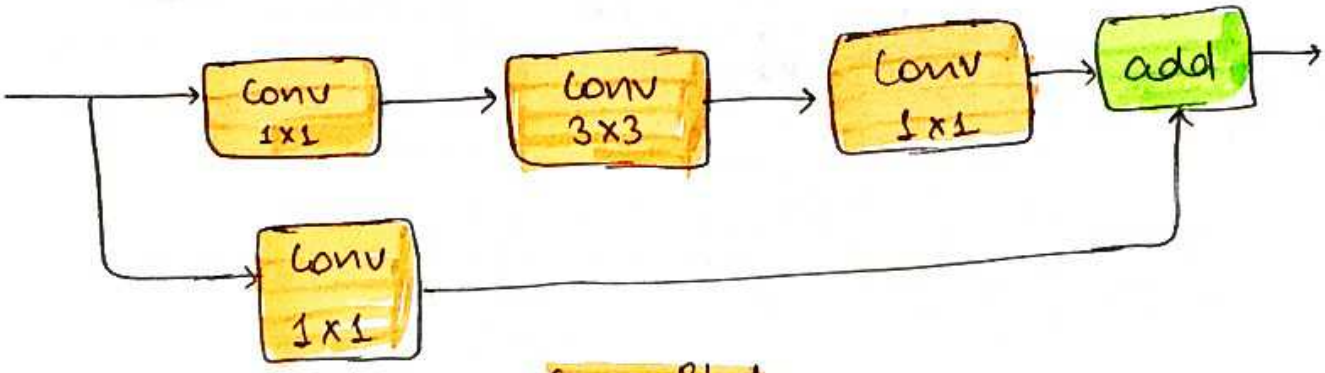


Resnet

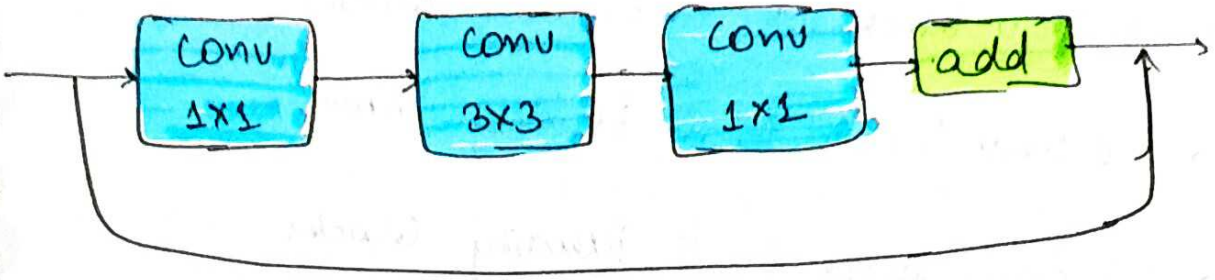








Conv Block



Identity block

The degradation problem is adressed by introducing bottleneck residual blocks. There are 2 kind of residual blocks:

Identity block: consists of 3 convolution layer with 1x1, 3x3, and 1x1 kernel sizes, all of which are equipped with BN. The Relu activation function is applied to the first two layers, while the input of the identity block is added to the last layer before applying Relu.

Convolution block: same as identity block but the input of the convolution block is first passed through a convolution layer with 1x1 kernel size and BN before adding to the last convolution layer of the main series.

Notice that both residual blocks have 3 layer In total resnet-50 has 26 million parameters an 50 layers:

1 conv layer with BN then Relu is applied

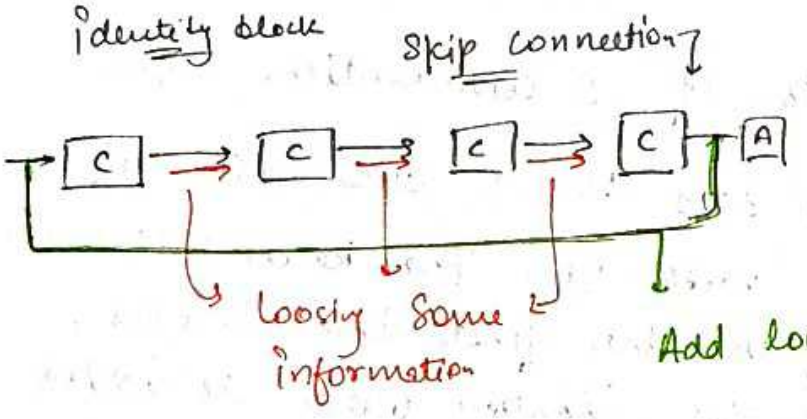
9 layers → 1 conv block & 2 identity blocks

12 layers → 1 conv block & 3 identity blocks

18 layer → 1 conv block & 5 identity blocks

9 layer → 1 conv block & 2 identity blocks

1 fully connected layer with softmax



CNN Architecture	Default input	Default output	No. of layers	No. of Param	New addition Perk
LeNet-5	32x32x1	10	5	60k	Conv layer
AlexNet	224x224x3	1000	8	60M	Local Response Norm
VGG-16	224x224x3	1000	16	138M	Very deep but still single thread
Inception V1	224x224x3	1000	22	7M	Auxilary classif & inception Module
ResNet	224x224x3	1000	50	26M	Batch Norm & Residual block

Resnet solve the problem of vanishing gradients, which occurs in deep neural networks. As layers increase, the gradients become too small during backpropagation, making it hard for the network to learn effectively. Resnet introduces skip connections (or residual connections) that allow the model to bypass some layers, enabling better gradient flow and improving training in deep architecture.

Resnet Architecture

The ResNet architecture is built around the concept of residual learning, where layers learn the residual (or difference) betn the input and the input and the output rather than learning the full transformation. This is achieved using skip connection.

Here's a breakdown of the ResNet architecture:

1. Input layer: Initial convolutional layer with a large filter (e.g 7x7) and stride 2, followed by max pooling.

2. Residual Block:
   - → Each block contains a few convolutional layers (typically two or three).
   - → A skip connection (shortcut) is added, bypassing these layers and directly connecting the input to the output of the block.
   - → The output of the convolutional layers is added to the input before passing through the next activation function.

The residual block can be either:

→ Identity block: If the input and output dimension are the same.

→ Convolutional blocks: If dimensions are the same.

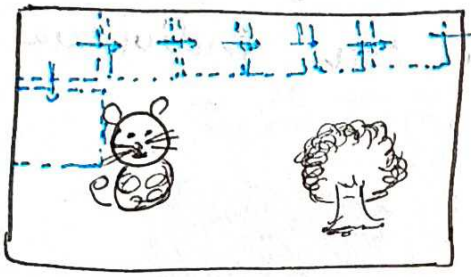
3. Stacked Residual Blocks: Multiple residual blocks are stacked, increasing the depth of the network without losing gradient flow.

4. Global Average Pooling: After all residual Blocks, a global average pooling layer reduces the spatial dimensions.

5. Fully connected layer: Finally, a fully connected layer produces the classification output.

R-CNN

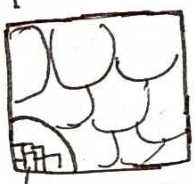
Object Detection → R-CNN
↳ Fast RCNN
↳ Faster RCNN
↳ YOLO



→ filters are moving and finding images. (Basic approach)
iteration of this process is ↔↑↑ more
This process is called exhaustive search.

RCNN → Region based CNN

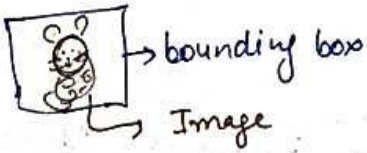
* Regions
↳ Exhaustive Search (Basic)
↳ Selective Search → greedy algorithm →
↳ Region proposal Method
used in RCNN is selective search
~2000 propose region.



Region Proposal

The first stage of the R-CNN pipeline is the generation of "region proposals" or regions in an image that could belong to a particular object. The authors use the selective search algo. The selective search algo works by generating sub-segmentations of the image that could belong to one object-

based on color, texture, size and shape - and iteratively combining similar regions to form objects. This gives 'object proposal' of different scales. Note the R-CNN pipeline is agnostic to the region proposal algorithm. The authors use the selective search algo to generate 2000 category independent region proposals (usually indicated by rectangular regions or bounding boxes) for each individual images.



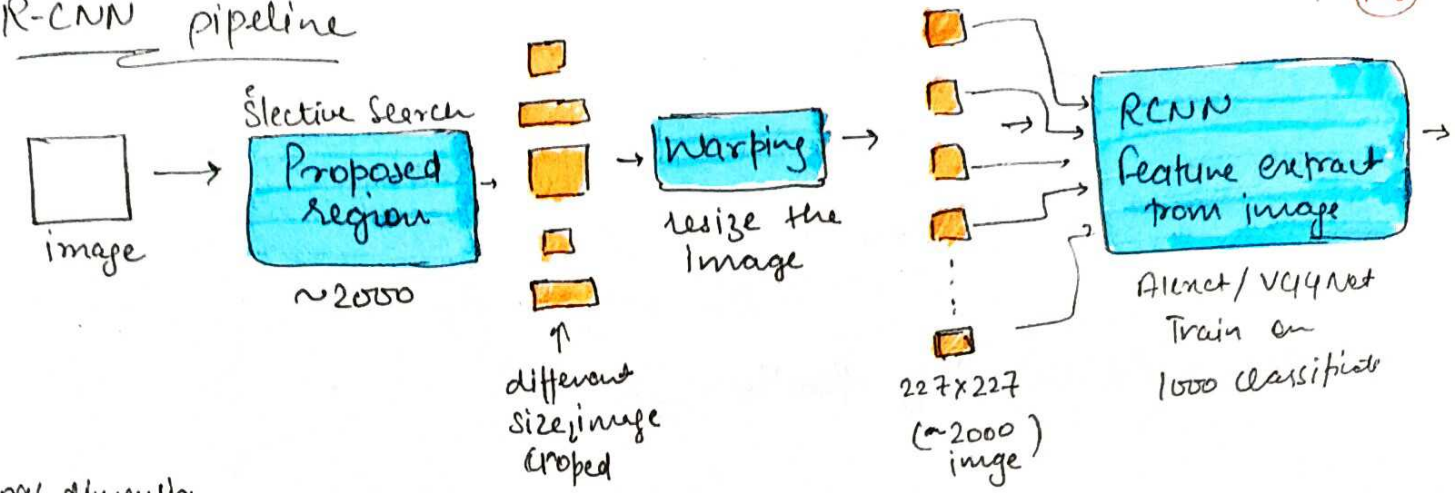
R-CNN Pipeline Stages

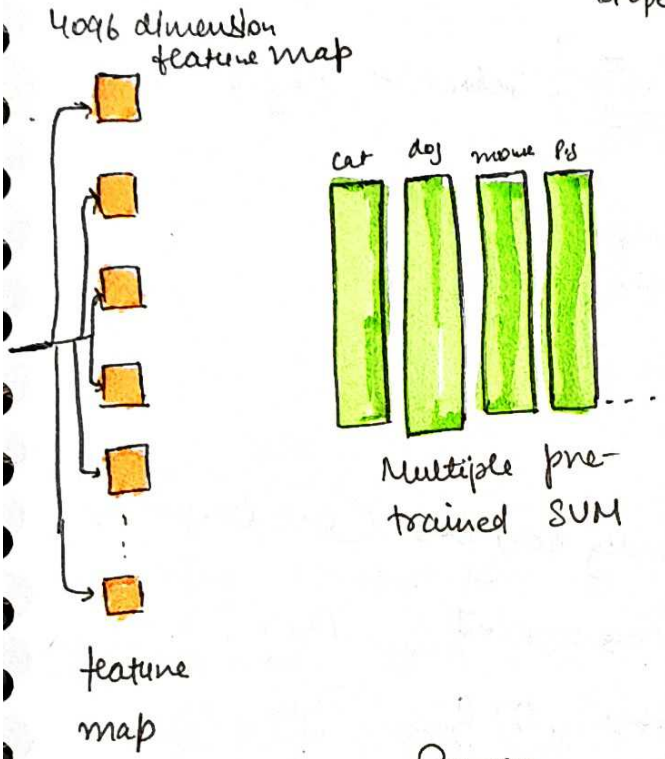
Stage 1: Feature extraction from the Region Proposals

Stage 2: SVM for object classification

Stage 3: Bounding box regression.

R-CNN pipeline





RCNN →

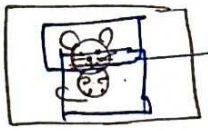
(i) Regions Proposal → Exhaustive / Selective Search

(ii) Warping and resize

(iii) Pretrained CNN → Alex Net / VGG Net

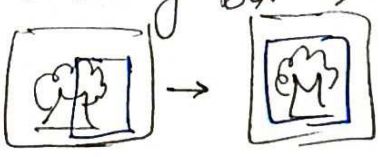
(iv) Pretrained → SVM_{cat}, SVM_{dog}, SVM_{person} —

(v) Clean up → Maybe same image come many time

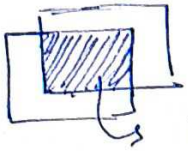


→ Same Image → 2 time

(vi) Bounding Box → fine tune

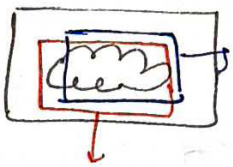


IOU → Intersection of union

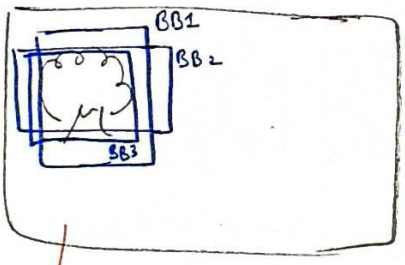


Metric → Mean Avg Position

Bounding box regressor →



Clean up

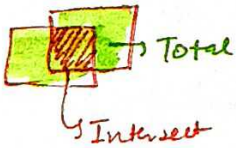


BB (Boundig box) 1 → 0.9 (confidence) ← select Tree

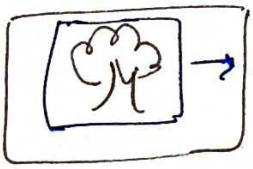
BB 2 → 0.7 Tree

BB 3 → 0.8 Tree

Find IOU →

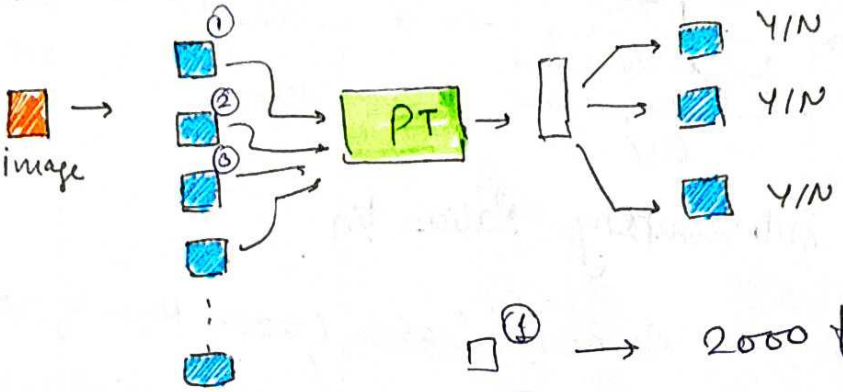


→ If Total divide by intersect is greater than 50 then we drop BB2 and BB3



Fast RCNN

In RCNN (2014)



① → 2000 feature map

② → 2000 feature map

③ → 2000 feature map

2000 → " " "

Every image contain 2000 feature image

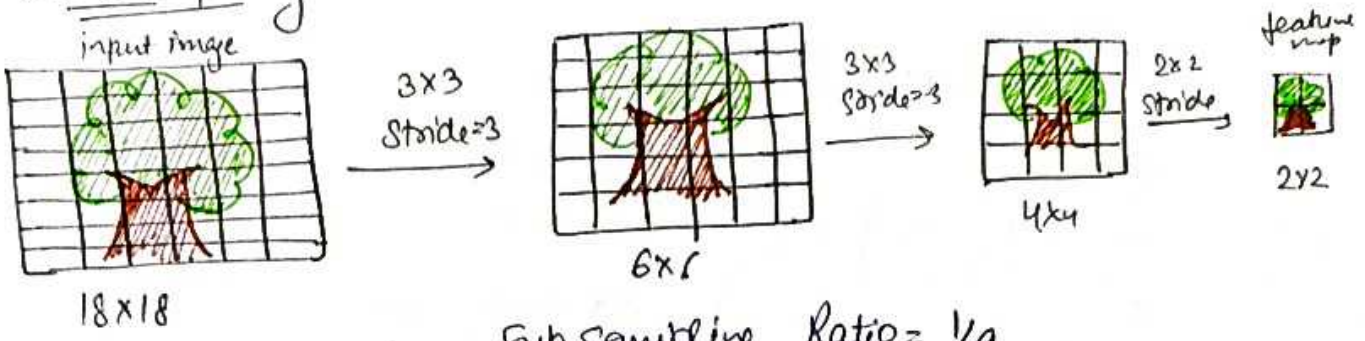
Problem:

1. Very computation
2. Extreme time consuming
   One image took → 47 second
3. It was a complex multi stage training pipeline.

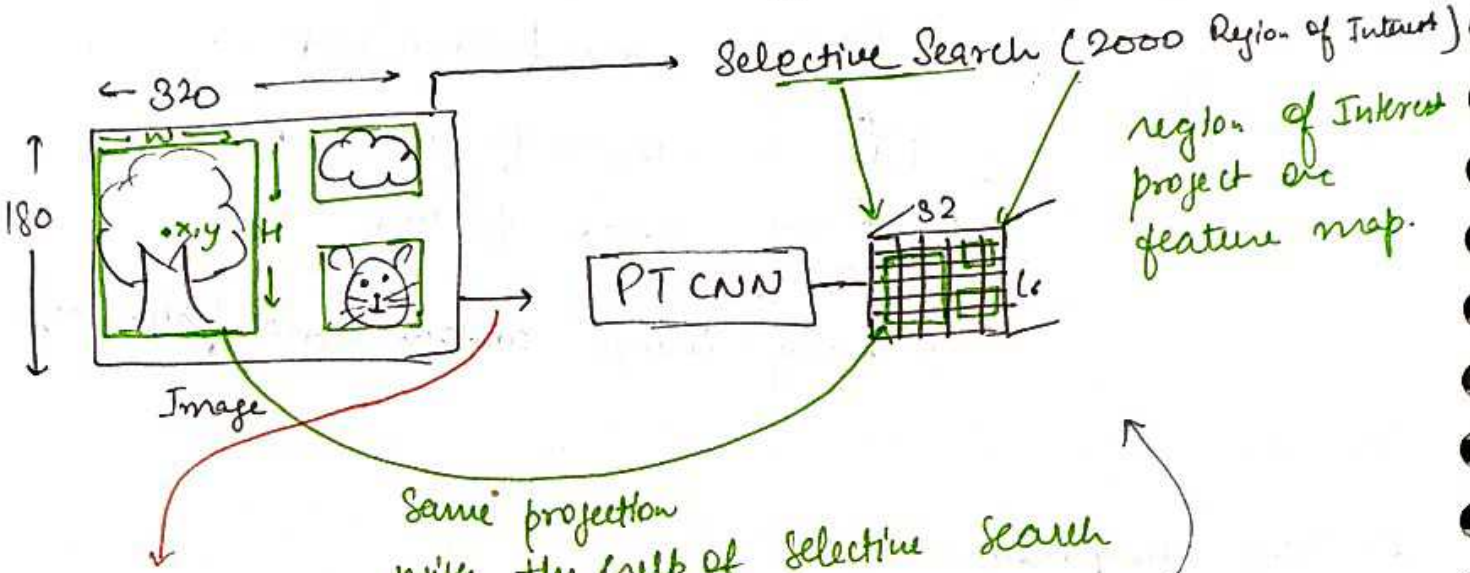
Fast - R-CNN

1. Sub Sampling
2. ROI Protection (ROI → Region of Interest)
3. ROI Pooling

Sub Sampling



Sub sampling Ratio = 1/9



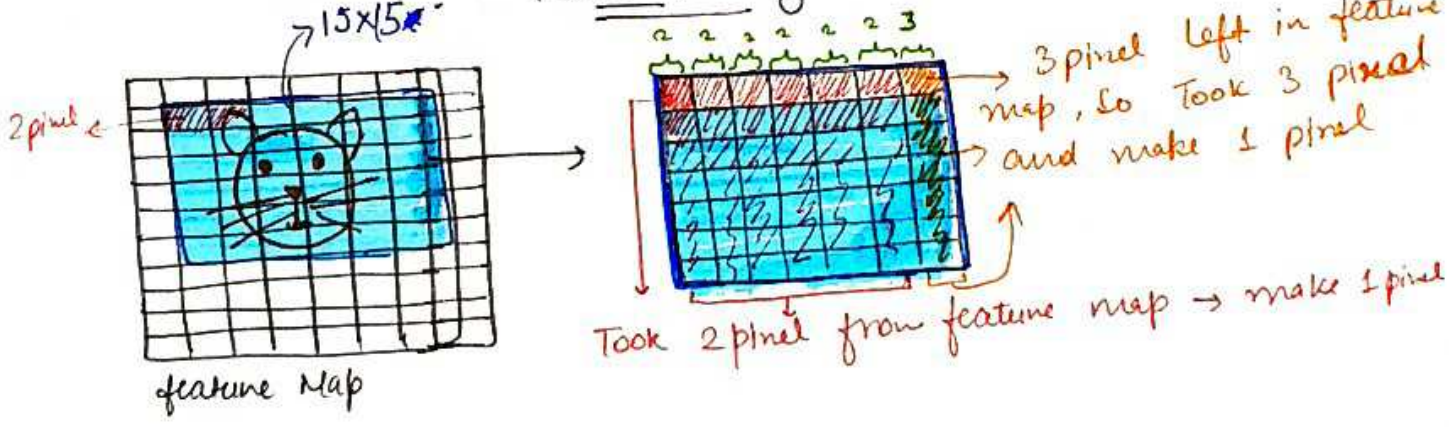
Same projection with the help of selective search

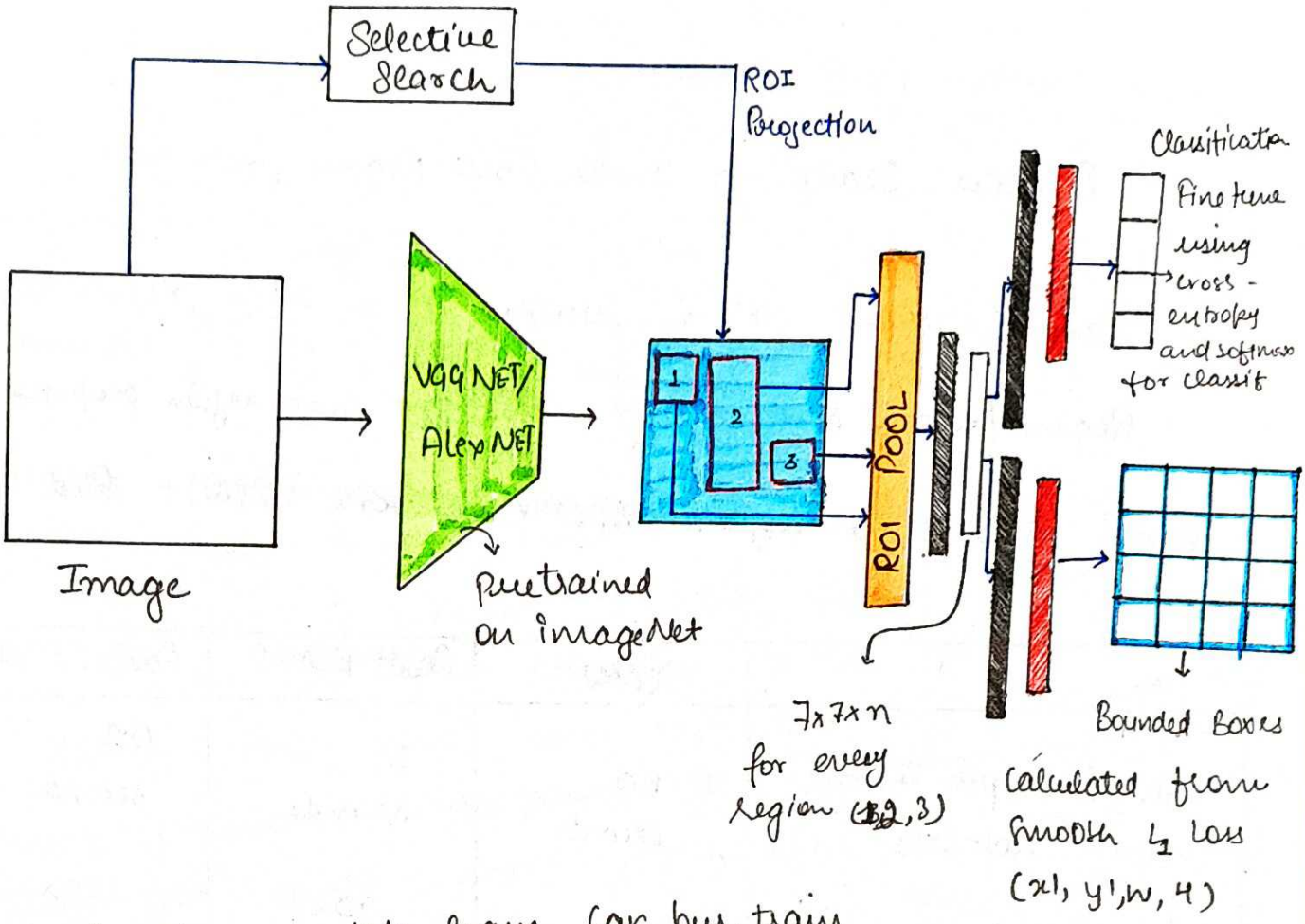
ROI Porjection

In RCNN we transfer multi size croped image (2000) but in fast RNN → Transfer a single Image

~~ROI Pooling~~ And PTCNN helps to reduce dimension and selective search help to project ROI on feature map. ROI size also reduce accordly to feature map.

ROI Pooling





In classification → We have car, bus, train
So Total class is 4 because one class is for background

In Bounded box → find L1 loss between initial check bounded box is perfect or not.

	RCNN	Fast RCNN
Training Time (Speedup)	84 hours 1x	9.5 hours 8.8x
Test time per image (Speedup)	47 second 1x	0.32 seconds 146x
Test time per image with selective search (Speedup)	50 second 1x	2 second 25x

Faster RCNN

Problem with Fast RCNN

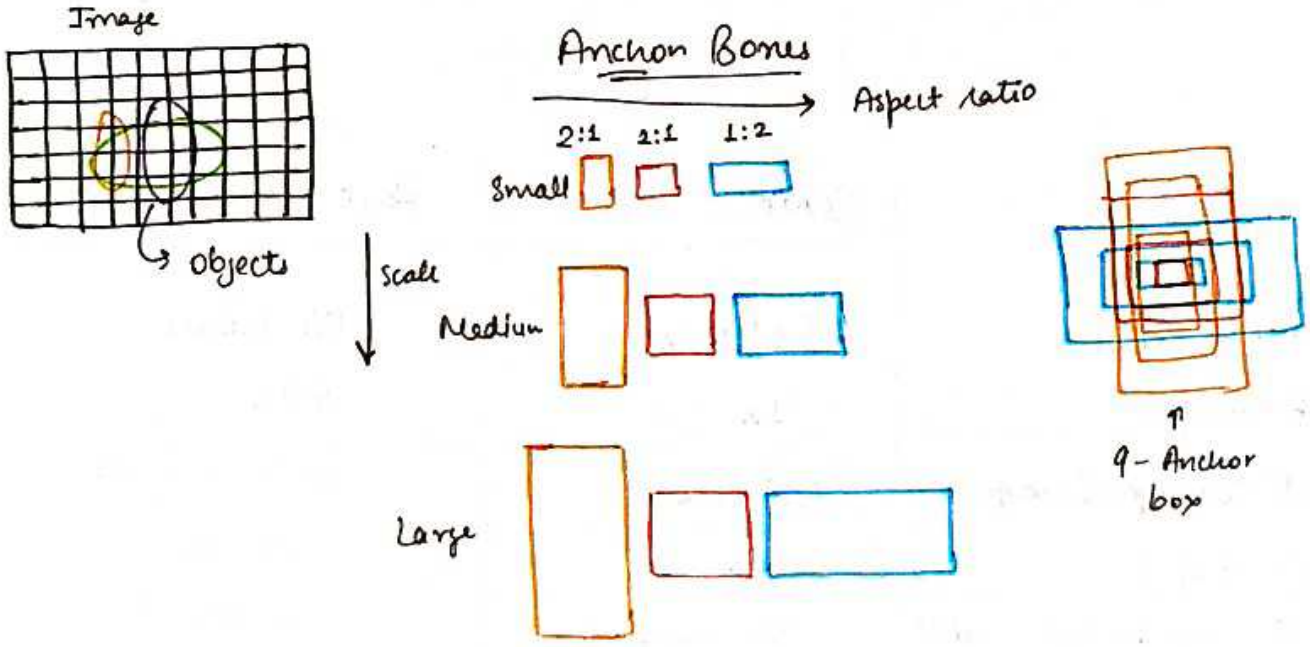
Selective Search → create 2000 Region proposed

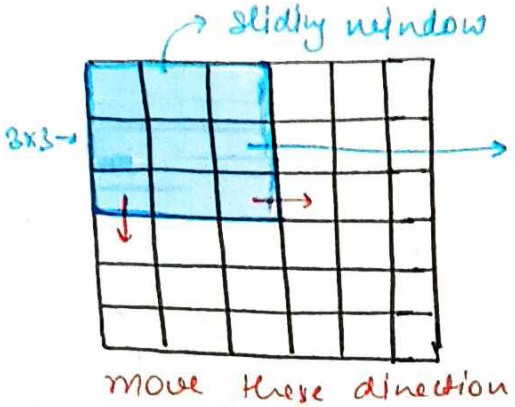
In Faster RCNN → replace Selective Search with Region Proposal Network

Region Proposed Network → less than 2000 region proposed

Faster R-CNN → **Region Proposal Network (RPN)** + **Fast RCNN**

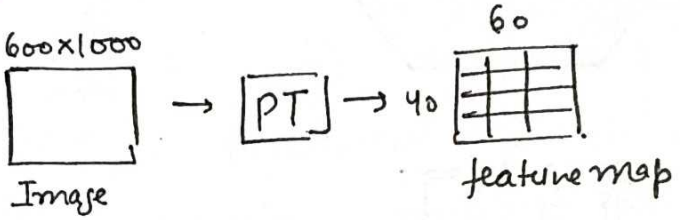
	RCNN	Fast RCNN	Faster RCNN
Test time per image with Proposals	50 seconds	2 seconds	0.2 seconds
(Speedup)	1x	25x	250x
mAP (Pascal VOC 07)	66.0	66.9	66.9





9 region of Interest for one sliding window.

No. of slide ↑↑ = region of Interest ↑↑



600 x 1000, we would have a feature map of size - 40 x 60. And using a diff sliding window at each position for all the 40 x 60 values in the feature Map, we end up having 40 x 60 x 9 = ~20000 proposals. Compared to selective Search which give just 2000 proposals, we have almost 10 times more proposals. This would be computationally more expensive and also would have more false positives

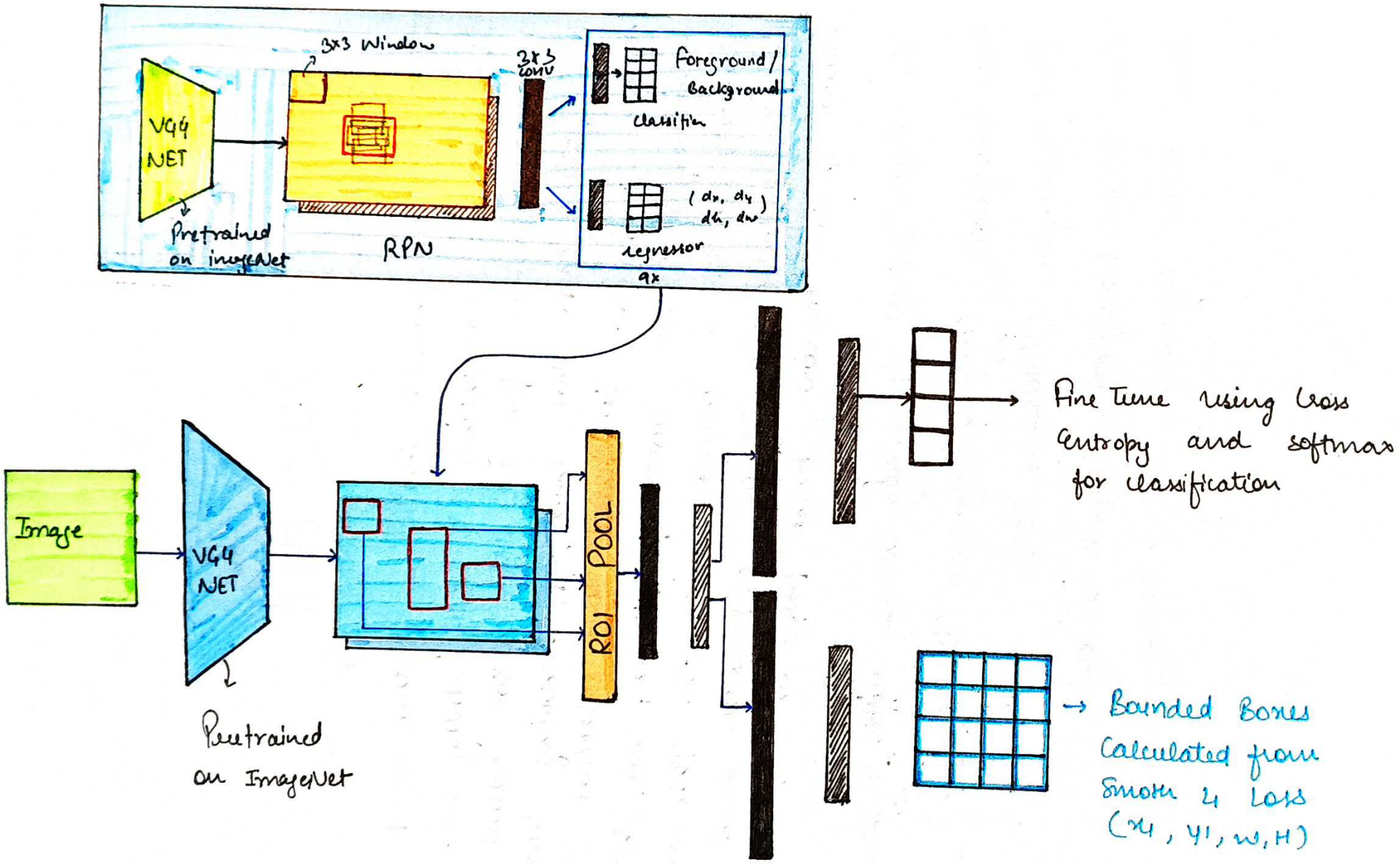
The Faster R-CNN is jointly trained with 4 losses

RPN classification (Object foreground / background)

RPN Regression (Anchor → ROI)

Fast RCNN Classification (Object classes)

Fast RCNN Regression (ROI → Bounding Box)

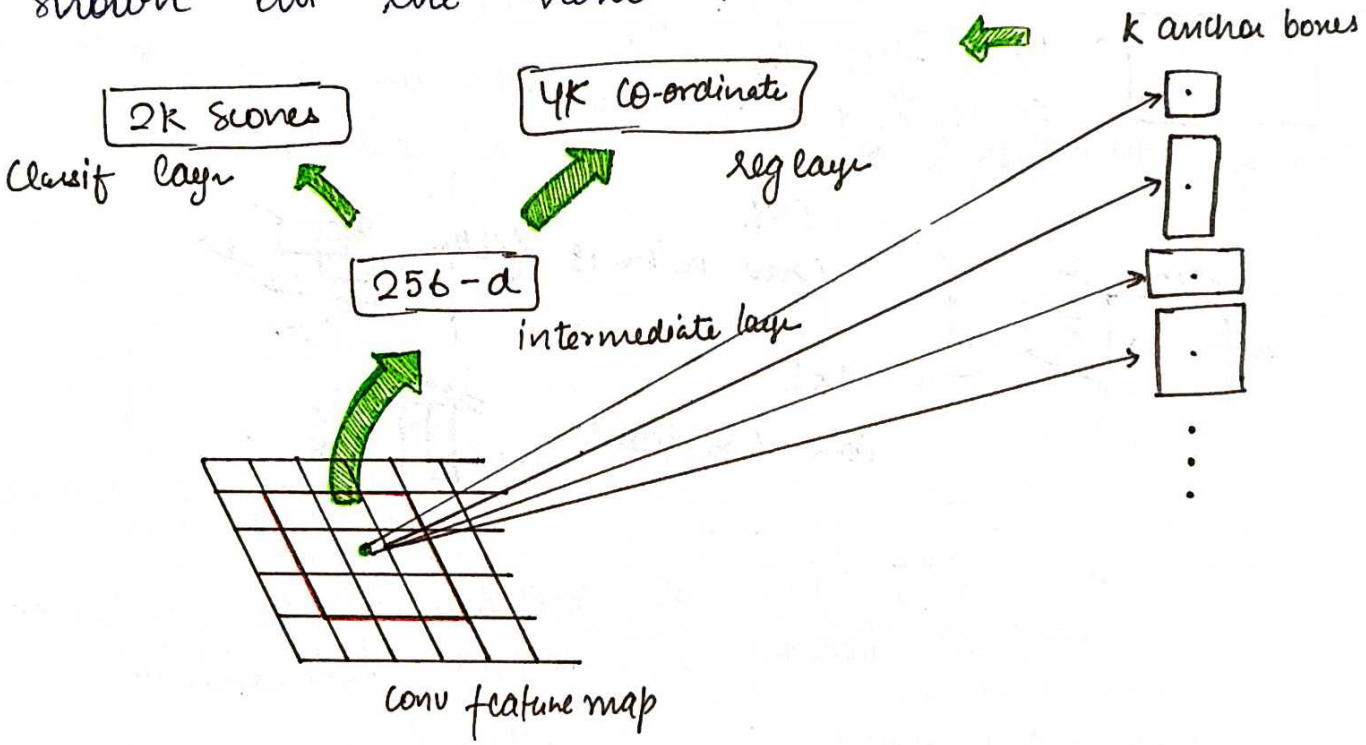

3x3 Window
3x3 conv
VGG NET
Pretrained on imageNet
RPN
foreground / background
classifier
(dx, dy) dh, dw
regressor
9x
Image
VGG NET
Pretrained on ImageNet
ROI POOL
Fine Tune using cross entropy and softmax for classification
→ Bounded Boxes Calculated from Smooth L1 Loss (x1, y1, w, H)

Faster R-CNN

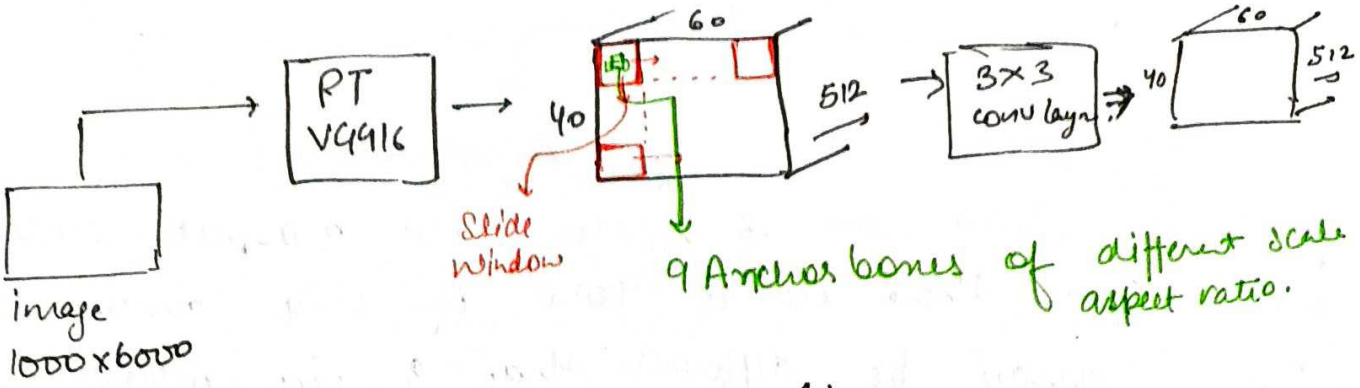
Anchor → depend on

- Scale
- Aspect Ratio

Generally, there are 3 scales and 3 aspect ratios and thus there is a total of k=9 anchor boxes. But k may be different than 9 in other words, k regions are produced from each region proposal, where each of the k regions varies in either the scale or the scale or the aspect ratio. Some of the anchor variations are shown in the next figure.



Faster R-CNN Detailed

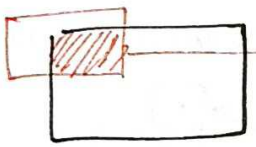


$60 \times 40 \times 9 = \sim 21k$

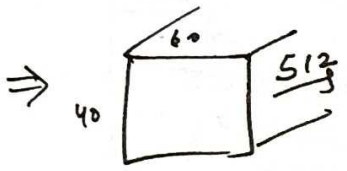
→ Crossboarder ~ 6k

→ Non Max Supression → IOU → Intersection of union. → If < 0.5 not accept; If > 0.5 accept → confidence

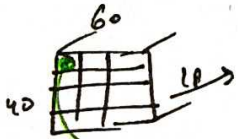
Crossboarder



→ ROI (Region of Interest) crossing boarder. So, decrease from 21k to 6k



1×1 conv with 18 filter



1×1 conv (36 filter)

9 Anchor box → define co-ordinate dx, dy, dw, dh of Anchor box

9 Anchor Box → 1, 2, 3, 4, 5, 6, 7, 8, 9

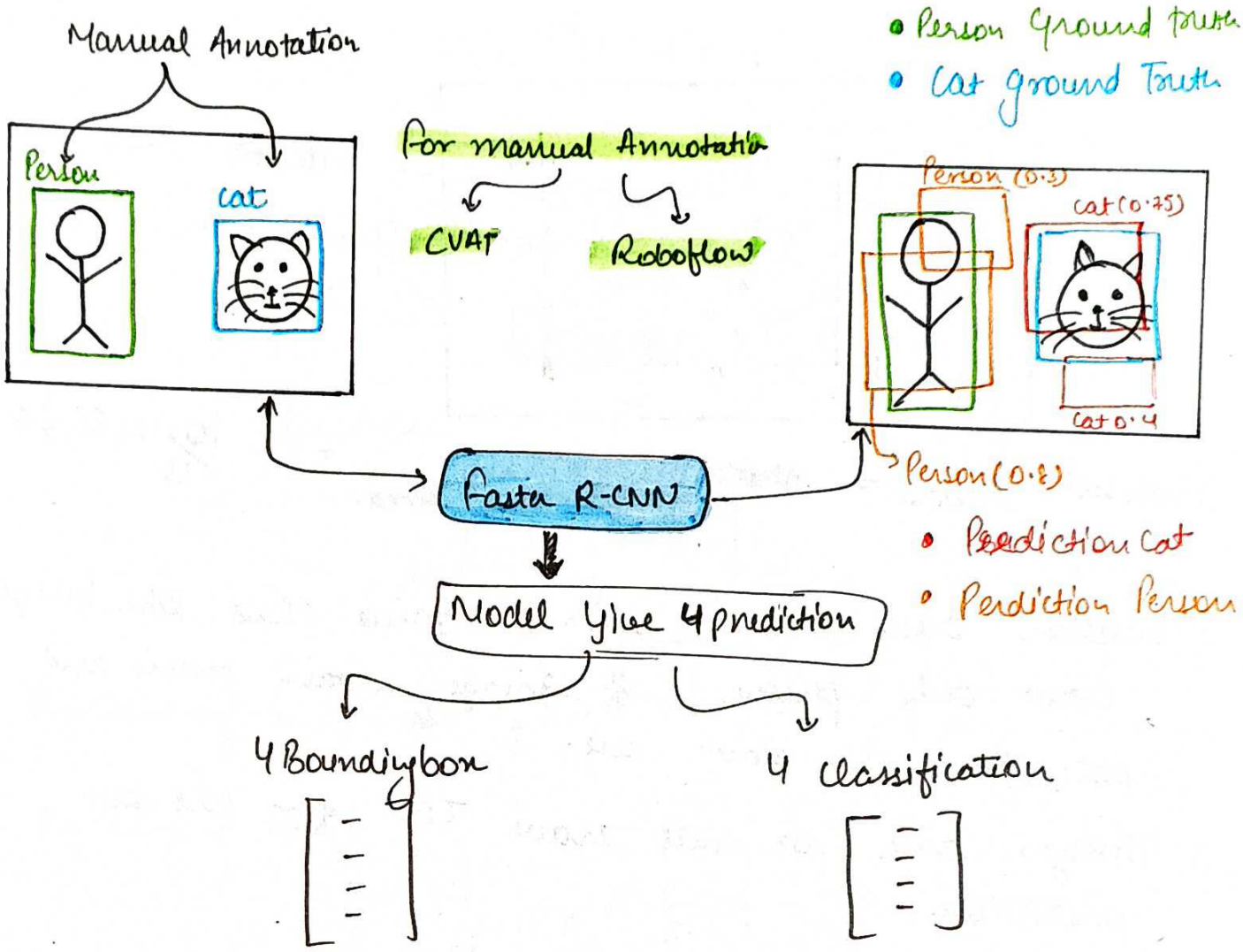
1 or 0 → 0 or 1

0 → background

1 → foreground

We have prob not 0 or 1

IOU, Non Max Supression And MaP



Model Performance

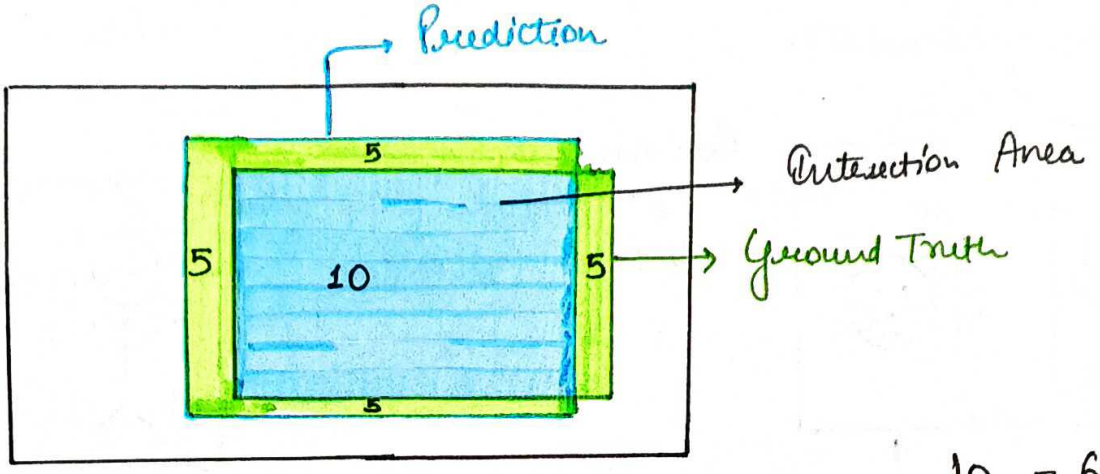
(1) We usually detect a lot more objects than we usually have in that image.
- → Selective
- → Anchor

[confidence vary]

(2) We need a way to decide → Threshold
Evaluate model → Mean Avg, Precision.

(3) Because of Point 1, we also need a clean up process → No. of prediction limited and Precision and Recall also increase

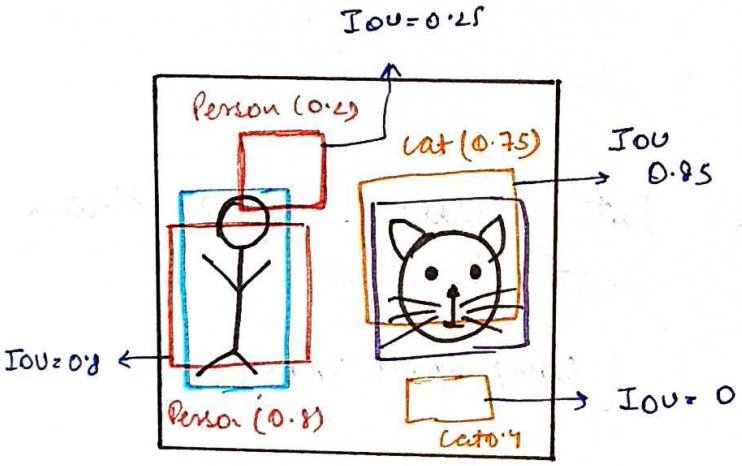
Intersection over union (IOU)



Intersection/union = blue Area / blue+green Area = $10/15 = 66.66$

Intersection over union use → Same class like image have only person. If image contain person and cat then we don't use.

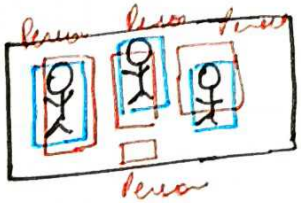
* Through this we will have IOU for all our predictions.



- ground truth cat
- ground truth person

Mean Average Precision

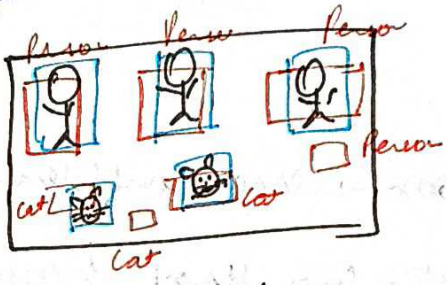
$$\text{Precision} = \frac{TP}{TP + FP}$$



Object detection class of Person

$$= \frac{3}{3+1} = \frac{3}{4} = 75\%$$

Let say, Image → both cat and dog



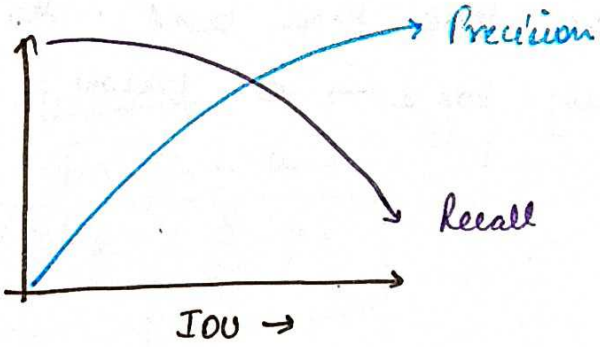
Precision of Person = $\frac{3}{4} = 75\%$.

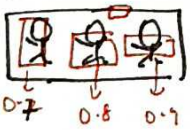
Precision of cat = $\frac{2}{3}$

Precision = $\frac{TP}{\text{All detects}}$

Recall = $\frac{TP}{\text{Actual Ground Truth}}$

Mean Average Precision → $\frac{1}{k} \sum_{i=1}^{n} AP_k$





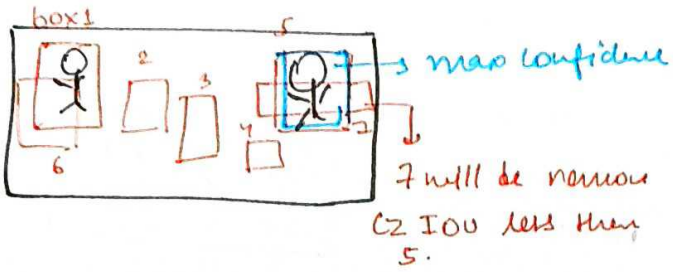
My threshold = 0.5
all prediction are above threshold that's why our prediction is true.

Avg Precision (Person) = 0.75 = $\frac{3}{4}$

Let say, My threshold = 0.85
Now, our two prediction are false

Avg Precision = 0.25 = $\frac{1}{4}$

Non Max Supression



box		Confidence level
1	→	0.9
2	→	0.4
3	→	0.7
4	→	0.8
5	→	0.95
6	→	0.85
7	→	0.9

Step 1 → To remove all bound box with confidence level less than x (when x = 0.75)

removed = 2, 3

Person class

↳ Boundig box → max confidence (Box Max)

↳ All boundig box that have IOU > 0.4 → (any value) with max box will be remove.

→ We remove 7 cz IOU less than 5 bound box. So we cleaned second Image of person (only 5 bound box there)

→ Now, same process apply on first person → box 1 have higher IOU then box 6. So, box 6 will remove. Only box 1 → on person (cleaned person 1 too)

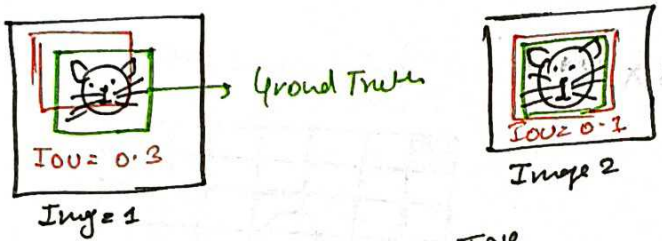
Let discuss more about Precision and Recall

$$\text{Precision} = \frac{TP}{\text{All detection}}$$

$$\text{Recall} = \frac{TP}{\text{Actual Ground Truth}}$$

True Positives

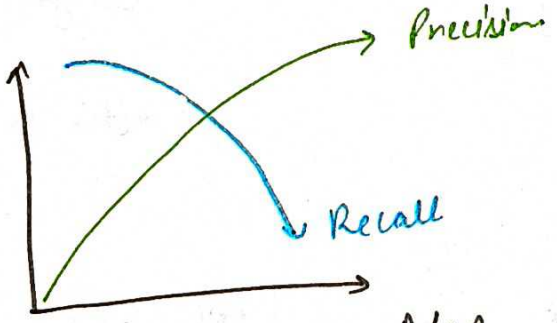
↳ Prediction = IOU > Threshold



Our Threshold = 0.5

→ Image 1 reject cz $\overset{IOU}{\wedge}$ less than threshold (0.3 < 0.5)

→ Image 1 accepted as TP cz IOU greater than threshold (0.7 ≥ 0.5)



Area Under Curve = MaP

$$MaP = \frac{1}{K} \sum_{i=1}^{K} AP_k$$

* We find this for every class