



# Computer Vision Applications

---

BY QUADEER SHAIKH

# About me

---



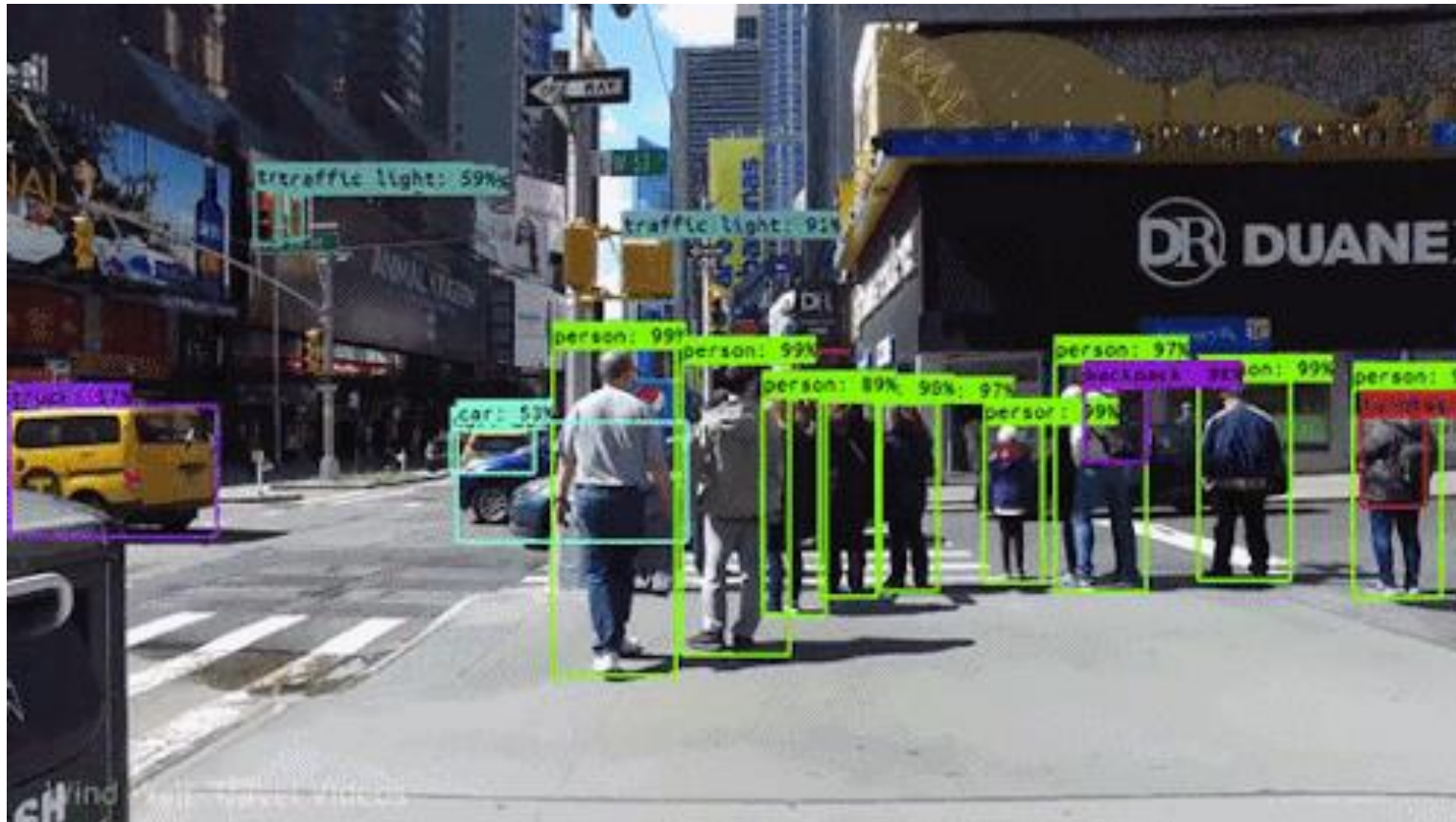
## Work Experience

- Risk Analyst
  - Morgan Stanley (Jan 2023 – Present)
- Data Science Intern
  - AkzoNobel Coatings International B.V. Netherlands (Feb 2022 – Dec 2022)
- Data Science Intern
  - EzeRx Health Tech Pvt. Ltd. (Jan 2022 – July 2022)
- Associate Engineer
  - Tata Communications Ltd. (July 2019 – Aug 2020)
- Network Automation and Analysis Engineer Intern
  - Cisco (June 2018 – July 2018)

## Education

- M.Tech – Artificial Intelligence
  - NMIMS (2021 - 2023, currently pursuing)
- B.E. – Computer Engineering
  - Mumbai University (2015 - 2019)

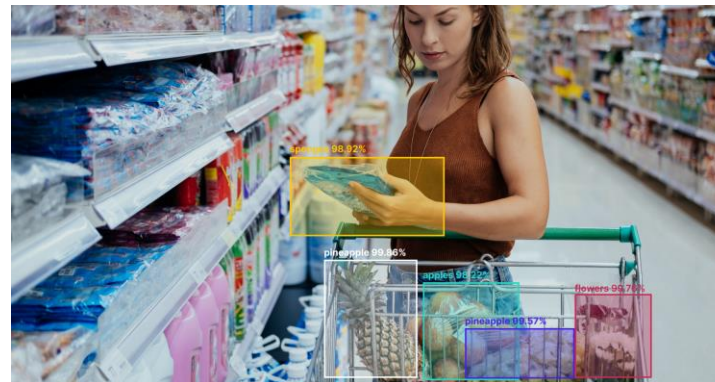
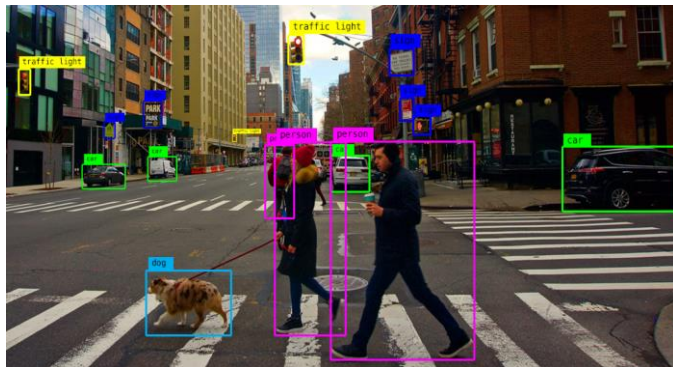
# Object Detection



# What is object detection ?

---

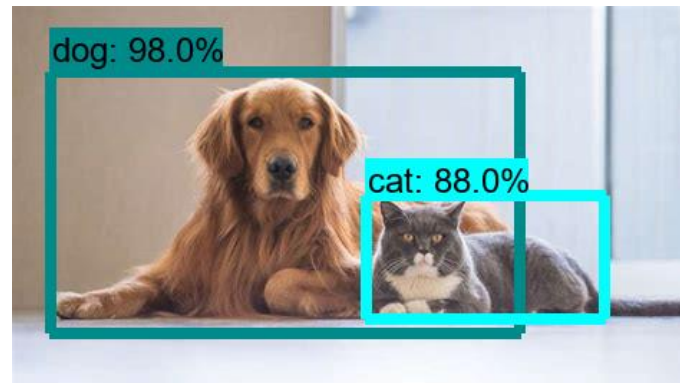
- Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class in digital images and videos.
- It is a computer vision technique for locating instances of objects in images or videos. Object detection algorithms typically leverage machine learning or deep learning to produce meaningful results.
- 17 interesting applications: <https://alwaysai.co/blog/object-detection-for-businesses>



# How do you localize an object in an image ?

---

- Ermm..... Maybe we can plot a rectangle that bounds an object of interest ? And hence, **Bounding Box**
- We know how to predict a class of an image, but how do we go about predicting a bounding box ?
- How do we create a rectangle/bounding box in computer vision using tools like opencv at all ?





# Bounding Box Regression

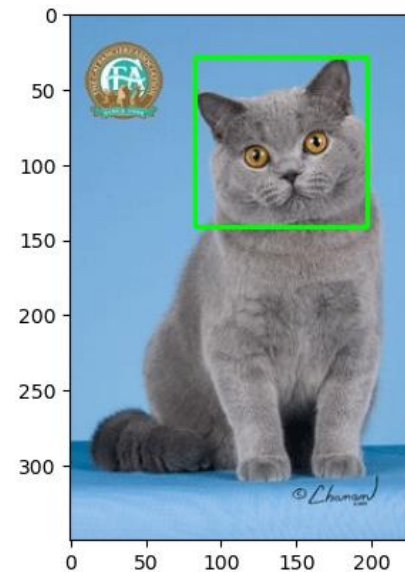
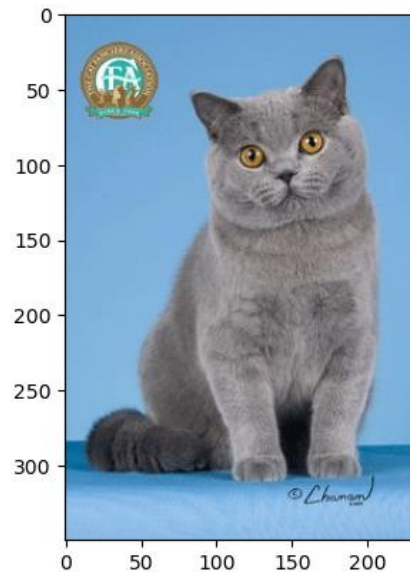
---

- **Syntax:** `cv2.rectangle(image, start_point, end_point, color, thickness)`
- Where, `start_point = (xmin, ymin)` and `end_point = (xmax, ymax)`
- So we perform a regression task to predict 4 values of 2 pairs of coordinates
  - xmin, ymin, xmax, ymax

# Bounding Box Regression

---

- Example: This is an image of Sir. Jim Radcliffe, a British Shorthair
- Image Dimensions: Height is 350, Width is 233, Colored image so Depth is 3
- Rectangle Coords: 83,29,197,142

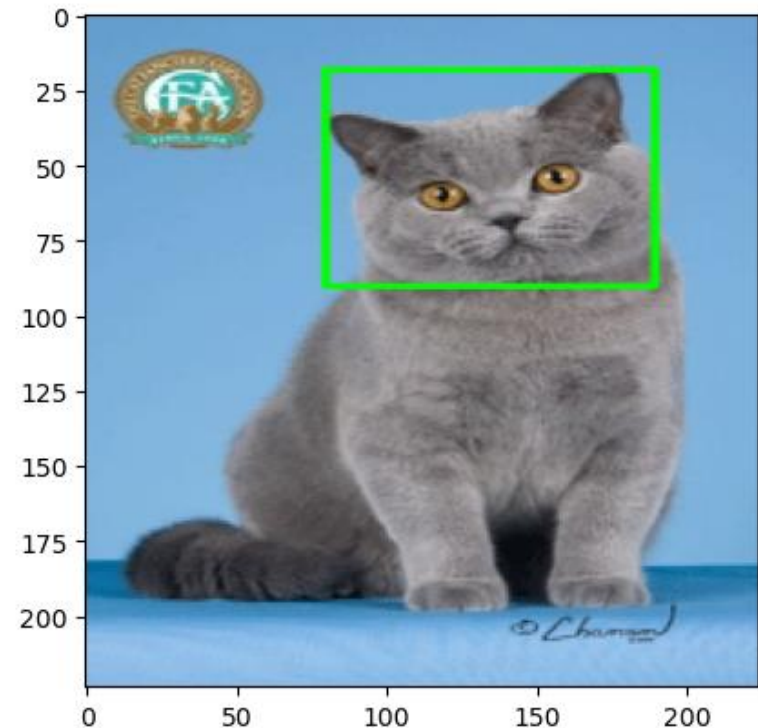


# Bounding Box Regression

---

But don't we feed a fixed size image input to our CNNs ? Maybe something like 224x224x3

- Example: This is an image of Sir. Jim Radcliffe, a British Shorthair
- Image Dimensions: Height is 350, Width is 233, Colored image so Depth is 3
- Rectangle Coords: 83,29,197,142
- That is why we normalize the coordinates
  - $xmin: 83/233$  ( $xmin/width$ )
  - $ymin: 29/350$  ( $ymin/height$ )
  - $xmax: 197/233$  ( $xmax/width$ )
  - $ymax: 142/350$  ( $ymax/height$ )
- If we resize the above image to 224x224x3
  - $xmin = \text{int}(xmin*224)$
  - $ymin = \text{int}(ymin*224)$
  - $xmax = \text{int}(xmax*224)$
  - $ymax = \text{int}(ymax*224)$

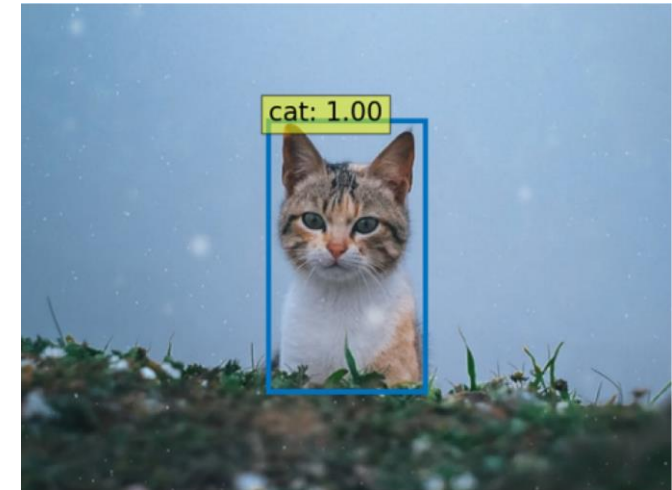
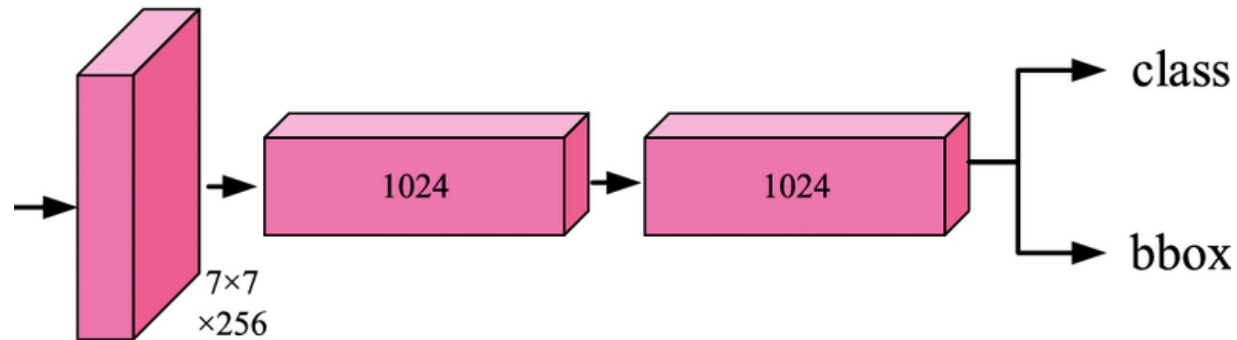




# Classification with Bounding Box Regression

---

Class predictor maybe a sigmoid or a softmax depending on the number of class of objects that we are predicting

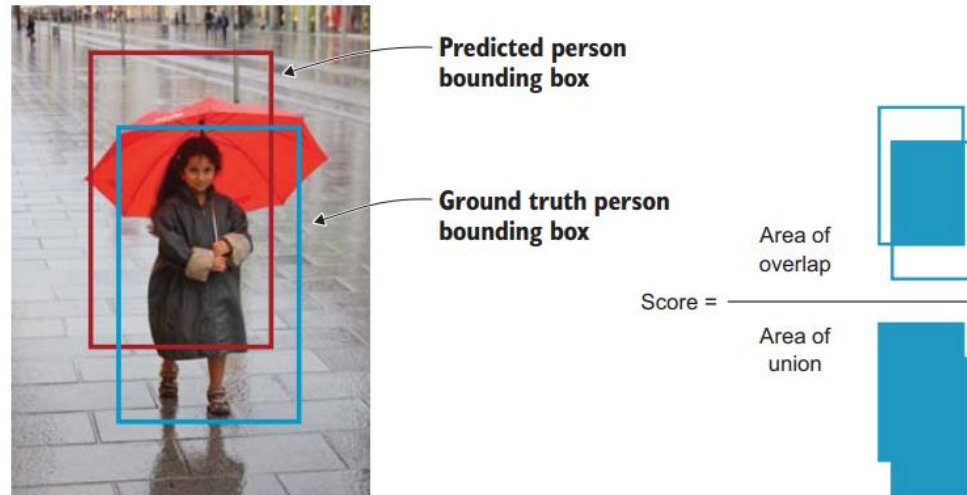


# How to assess the prediction of Bbox ?

---

## Intersection over Union

This measure evaluates the overlap between two bounding boxes: the ground truth bounding box (B<sub>ground truth</sub>) and the predicted bounding box (B<sub>predicted</sub>). By applying the IoU, we can tell whether a detection is valid (True Positive) or not (False Positive)

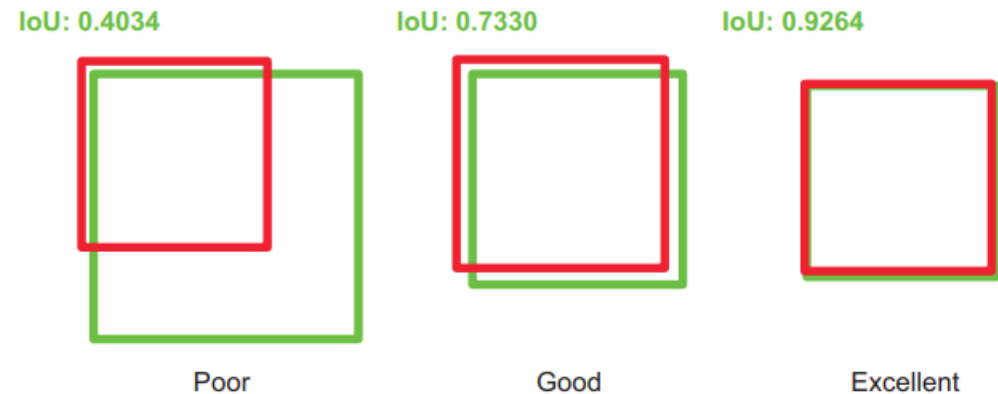


# How to assess the prediction of Bbox ?

---

## Intersection over Union

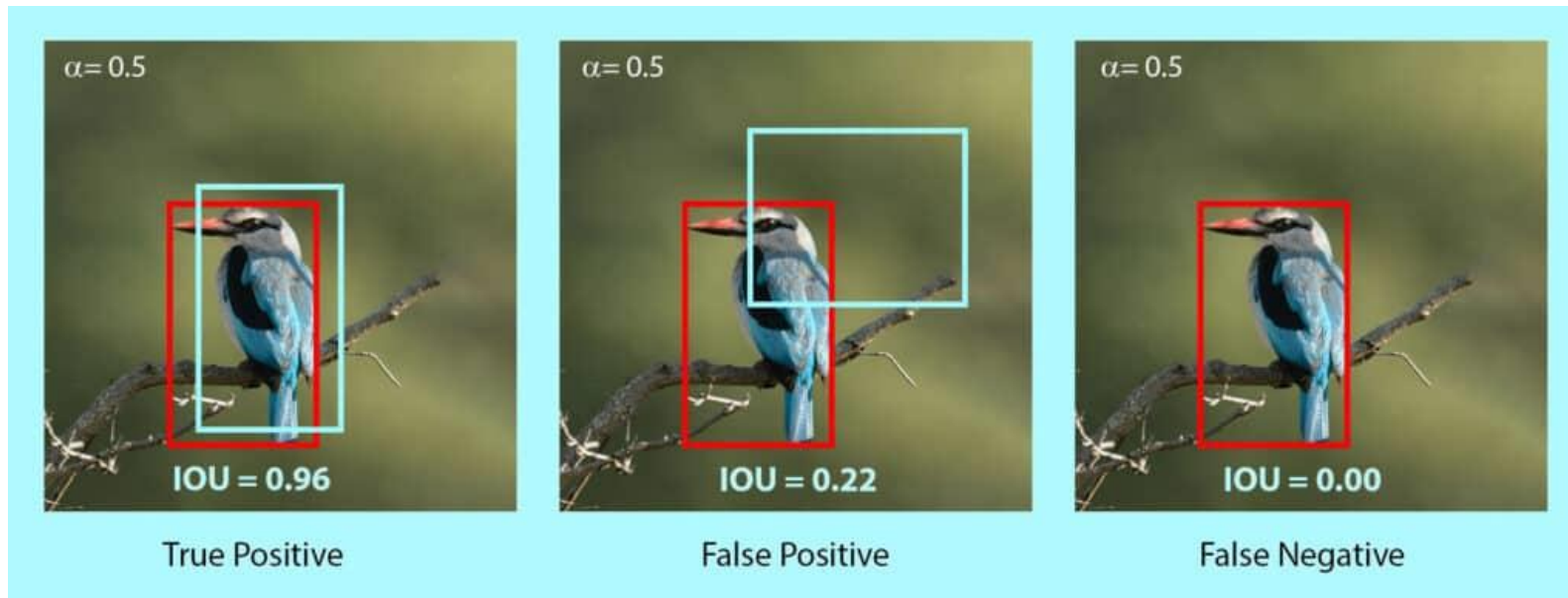
The intersection over the union value ranges from 0 (no overlap at all) to 1 (the two bounding boxes overlap each other 100%). The higher the overlap between the two bounding boxes (IoU value).



# Confusion Matrix for IoU

---

**Note:** True Negative is not applicable to object detection. True Negative means it is correctly detecting the background of an object as background. Which is equivalent to not detecting anything.



# Is object detection really that simple ?

---

- Bounding box regression with classification module works out fine if you have just one object in an image and you want to detect one object
- If you have multiple objects of a similar class in an image it will only detect one of them which it is most confident about
- How do we address this issue ?

Hemk 2 big doggos, but lil nugget gets detected



Wut the floof I is a doggo too

# Take a trip down the memory lane

---

What did we stumble upon in our fundamental computer vision modules that could help us find the objects of interest ?





# Take a trip down the memory lane

---

What did we stumble upon in our fundamental computer vision modules that could help us find the objects of interest ?

Objects of interest.....mmm..... **Regions of Interest (ROI)** ?!



# Region based CNN Models

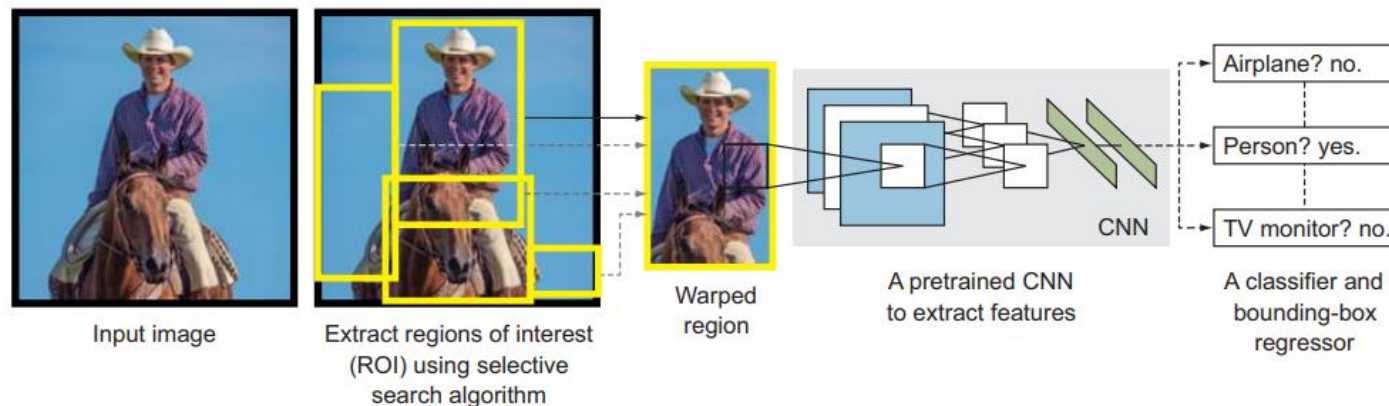
---

1. RCNN (2013)
2. Fast RCNN (2015)
3. Faster RCNN (2015)

# RCNN

---

1. It is the first region based architecture in the region proposal based object detection family
2. It was one of the first large, successful applications of convolutional neural networks to the problem of object detection and localization
3. The approach was demonstrated on benchmark datasets, achieving then-state-of-the-art results on the PASCAL VOC-2012 dataset and the ILSVRC 2013 object detection challenge



# RCNN Components

---

1. Region Proposal: Extract Regions of Interest (ROI)
2. Feature Extraction Module
3. Classification Module
4. Localization Refinement Module (optional component)

# RCNN – Region Proposal

---

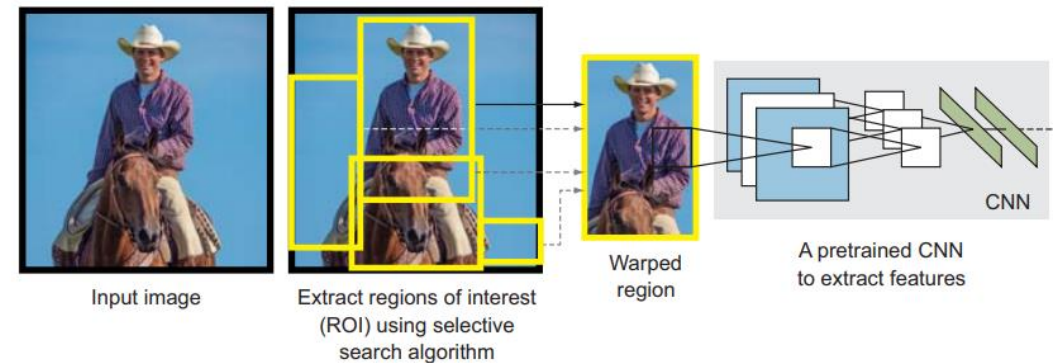
- The regions proposed in this step have a very high probability of containing an object
- An algorithm called *selective search* scans the input image to find regions that contain blobs and proposes Rols (2000 Rols in fast mode, approx. 9000 in regular mode) to be processed by the next components of the RCNN.
- The proposed Rols are of different sizes
- But since a CNN is fed an image of fixed size we warp the Rols to a fixed size
- **Note:** Since the selective search algorithm proposes region proposal bounding boxes, bounding box regression becomes an optional component



# RCNN – Feature Extraction

---

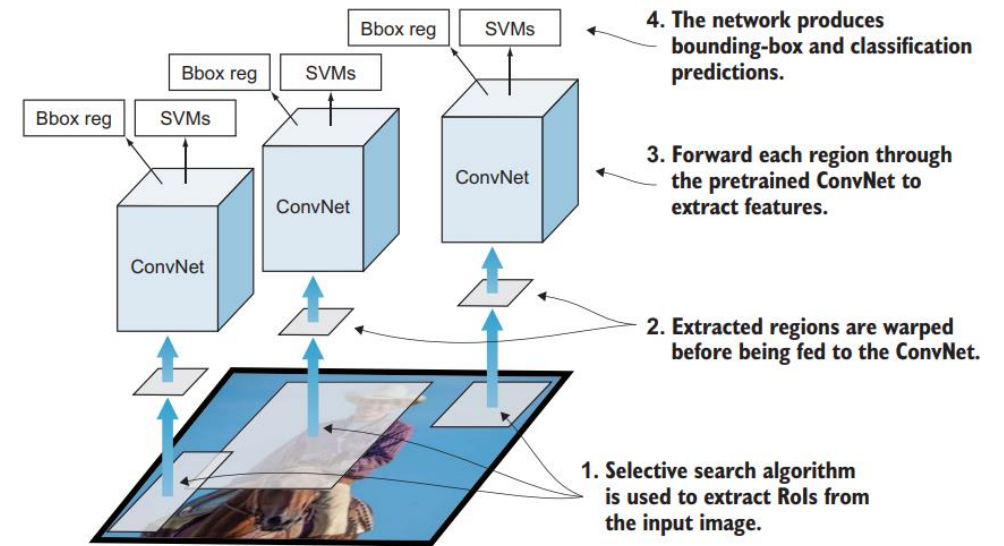
- We run a pretrained CNN (on imagenet) on top of the region proposals to extract features from each of these candidates
- However, since the proposed regions vary in size and are typically warped to fixed size, an imagenet pretrained CNN's feature extraction offers a mediocre/poor performance
- Therefore, the pretrained network is finetuned on the warped region proposals using the softmax classification output layer
- AlexNet/VGG-Net (initial version of VGG-16/19) was used as a pretrained network





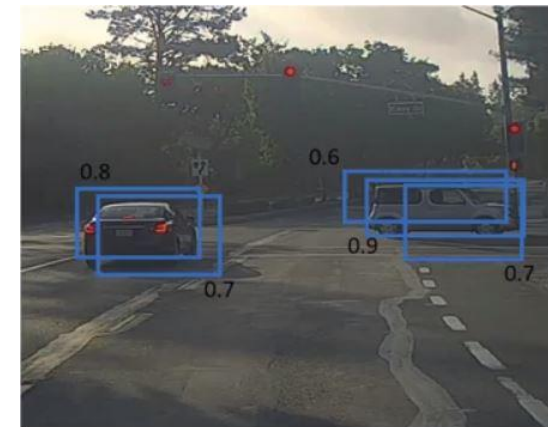
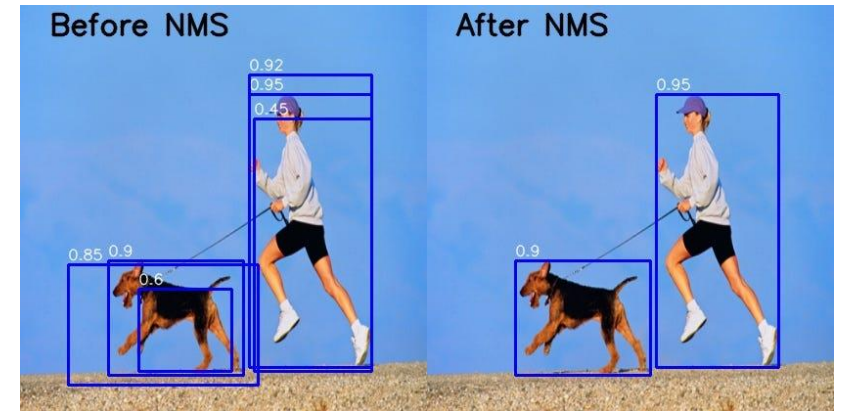
# RCNN – Classification Module & Bbox Refinement

- We train a Linear SVM to perform classification on the features extracted using the pretrained network on the candidate region proposals
- Linear SVM performs a one vs rest classification to classify an object into a particular class
- Since Selective Search takes care of the localization using its own proposed regions bounding box we do not explicitly perform the bounding box regression step
- But to improve the localization step further we include the bounding box regression to give more refined coordinates of the bounding box



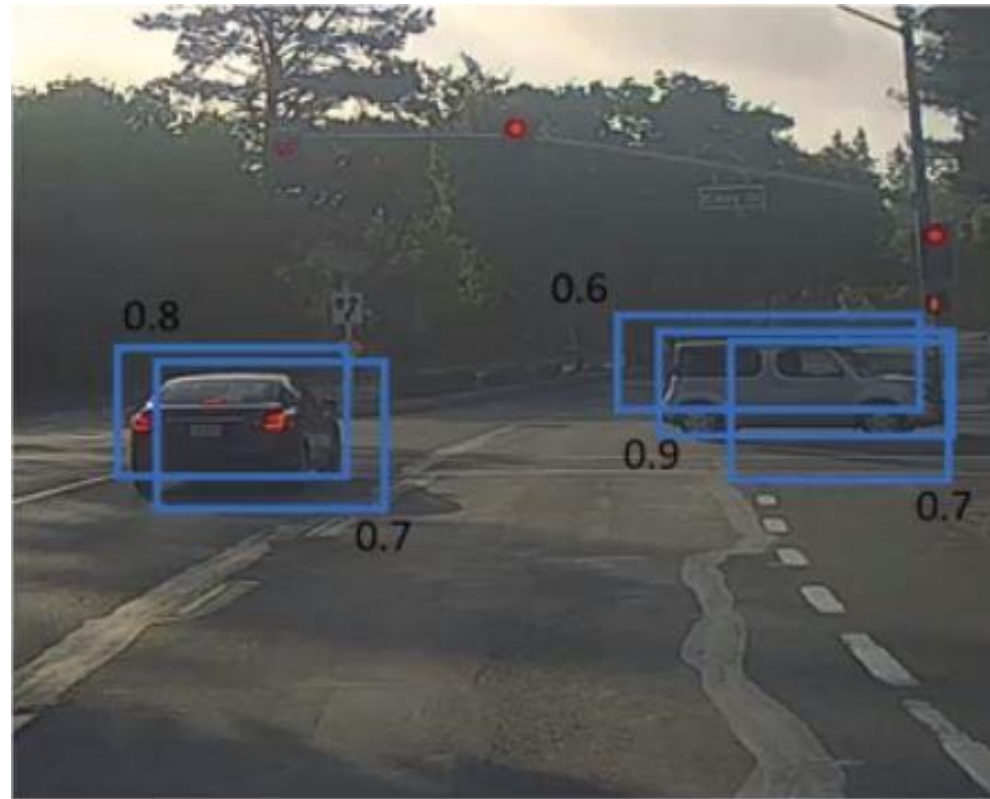
# Multiple Bbox single object: NMS

- Non max suppression is a technique used mainly in object detection that aims at selecting the best bounding box out of a set of overlapping boxes
- Sort the bboxes in descending order of class confidence/probability score
- remove the boxes which have a confidence score < threshold confidence score (e.g. remove boxes with score < 0.5)
- Loop over the remaining boxes
  - Start with the box which has the highest confidence
  - Discard other boxes if they have an IoU > predefined IoU thresh with the highest confidence box
  - Repeat the steps until all the boxes are either taken as output prediction or discarded



# Multiple Bbox single object: NMS

---



# Disadvantages of RCNN

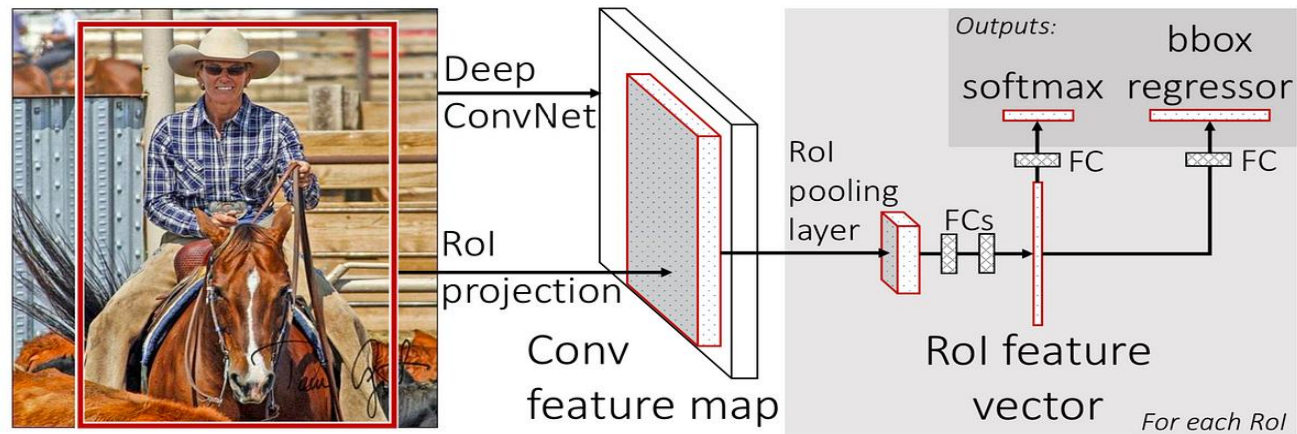
---

1. Object Detection is very slow
  - For each image selective search proposes 2000 Rols to be examined by the entire pipeline. These 2000 Rols are then passed on to the CNN 2000 times for a single image i.e. 2000 forward propagations for a single image. This process makes it computationally very expensive. During inference object detection in a single image takes 50 seconds (47.5 seconds for CNN + time taken for selective search)
2. Multiple Training stages (3 stages)
  - Finetuning of CNN feature extraction
  - Training of object classes using SVMs
  - Training of Bbox regressors for bounding box refinement
3. Two stage pipeline
  - First stage is for Region Proposals
  - Second stage is for Object Detection on Region Proposals
4. Training is expensive in terms of both time and space
  - 2000 Rols per image need to be finetuned
  - 2000 Rols per image need to be stored on the disk

# Fast RCNN (Inspired by SPPNet: Spatial Pyramid Pooling Network)

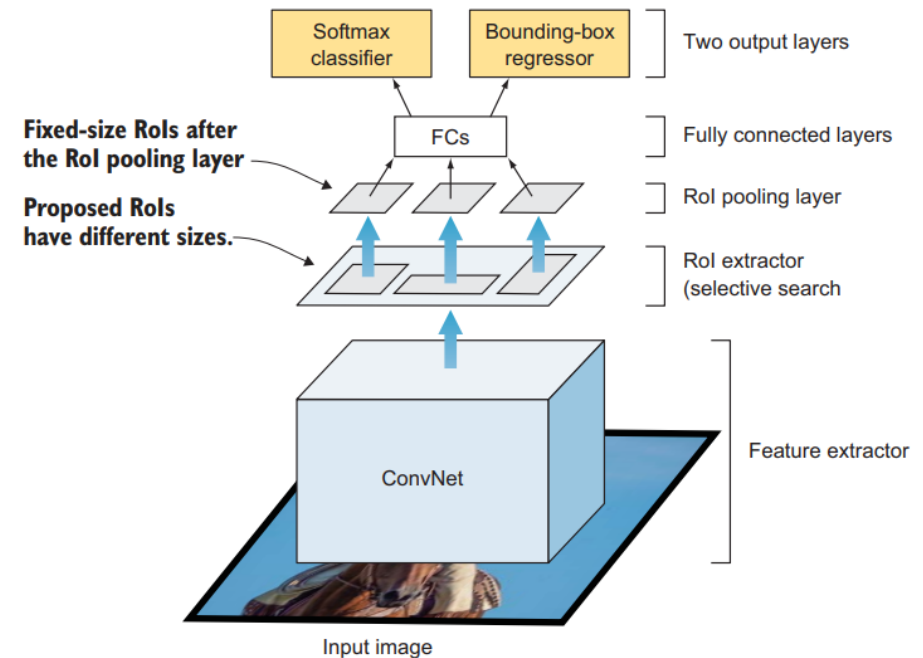
---

- Since feeding 2000 Rols to the CNNs 2000 times for a single image was a bad idea the author of the paper on RCNN came up with a new architecture
- Bounding box coordinate format:  $(x_c, y_c, w, h)$ 
  - $x_c$  – x center,  $y_c$  – y center,  $w$  – width,  $h$  - height



# Fast RCNN

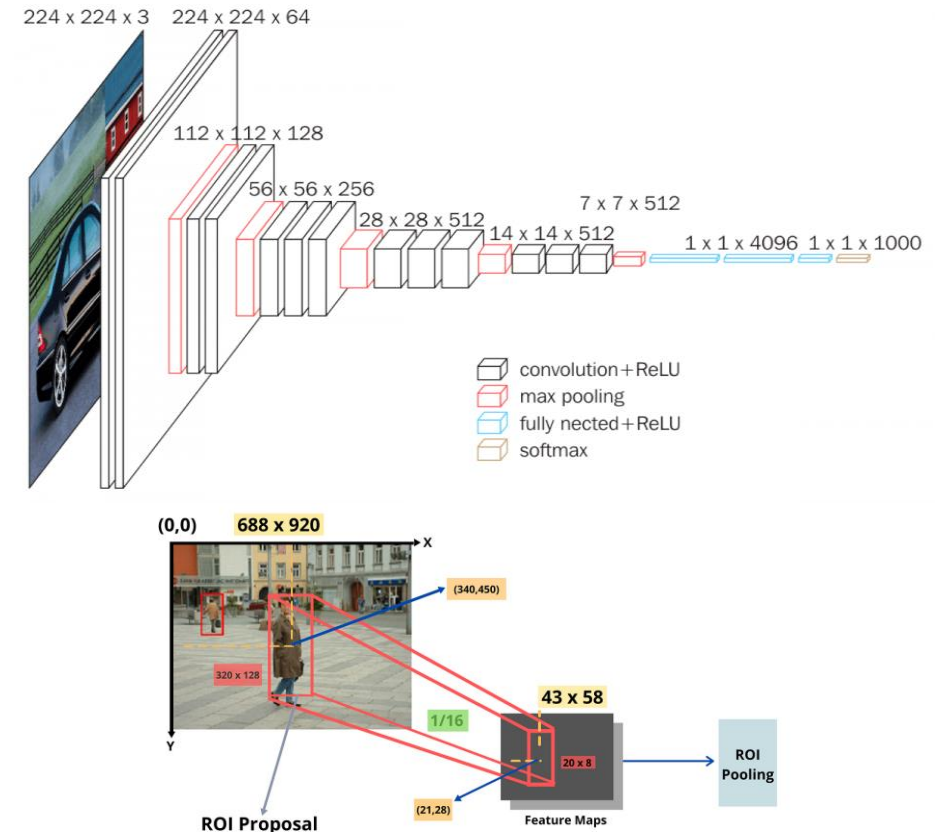
1. Pass the input image to the feature extractor just once
2. Pass the input image to selective search algorithm to get Rols/Region Proposals
3. Feature maps are produced by the pooling layer of the pretrained network
4. Rols from step 2 are projected onto the feature maps
5. Rol pooling is performed on the projected Rols
6. Rol pooling output is passed to the fully connected layers
7. Fully connected layer is connected to the multi-head output
  1. Classifier
  2. Bbox Regressor





# RoI Projection

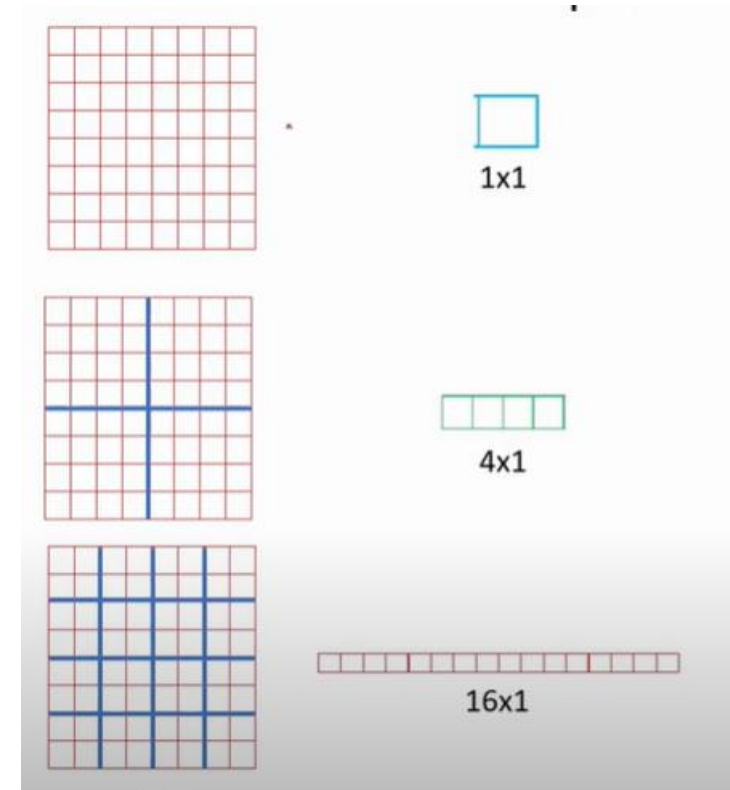
- The final max pooling layer produces a feature map of size  $7 \times 7 \times 512$
- An image of size  $224 \times 224$  gets downsampled to  $7 \times 7$ , therefore we can calculate its **subsampling ratio** by  $224/7 = 32$
- Any object that is present in the image of size  $224 \times 224$  can be projected to the feature map of size  $7 \times 7$  if we divide the height and width of that object by **32**.



# ROI Pooling

---

- The projected Rols on the feature maps are then pooled and passed on to the fully connected layers
- But since the projected Rols will vary in size we need to do something about it
- We perform a special type of pooling known as ROI pooling.
- ROI pooling takes the max value from each grid
- We use a 7x7 grid to perform ROI pooling on a feature map which will produce a vector of size 49x1
  - 1x1 grid ROI pooling will produce 1x1 vector
  - 2x2 grid will produce a 4x1 vector
  - 4x4 grid will produce a 16x1 vector



# Classification and Bbox Regression Module

---

1. Classification: Softmax classifier is used for classification prediction
2. Bbox Regression: Offsets between the selective search region proposals and ground truth bounding boxes are predicted for each class (this method is known as Relative Bounding Box Regression)
  1.  $tx = x\_center\_gt - x\_center\_ss$
  2.  $ty = y\_center\_gt - y\_center\_ss$
  3.  $tw = w\_gt - w\_ss$
  4.  $th = h\_gt - h\_ss$

---

# Loss function

---

## Fast R-CNN

- Multi-task loss

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

Where

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{x, y, w, h\}} \text{smooth}_{L_1}(t_i^u - v_i).$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

# Advantages and Disadvantages of Fast RCNN

---

## ADVANTAGES

1. Requires only one propagation for each image unlike 2000 in RCNN
2. Three stage training from RCNN reduced to one stage training
3. Faster in inference time than RCNN: 2 seconds (0.32 seconds for CNN + time required for selective search)

## DISADVANTAGES

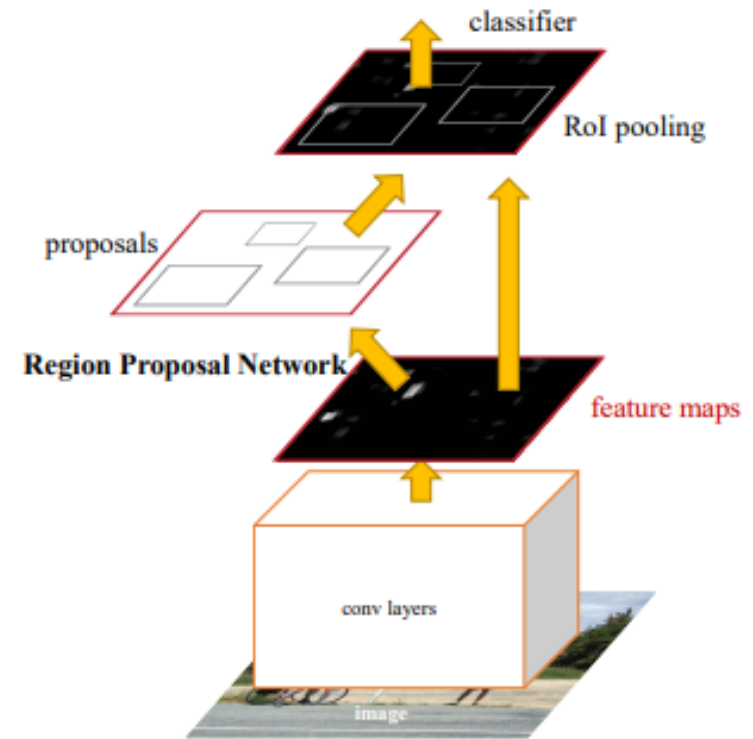
1. It is still a two stage pipeline
  1. Selective Search Region Proposal
  2. CNN
2. Selective search still remains to be a bottleneck as it is very slow to generate region proposals. 2 seconds of inference time is too much for real time object detection in videos.



# Faster RCNN

---

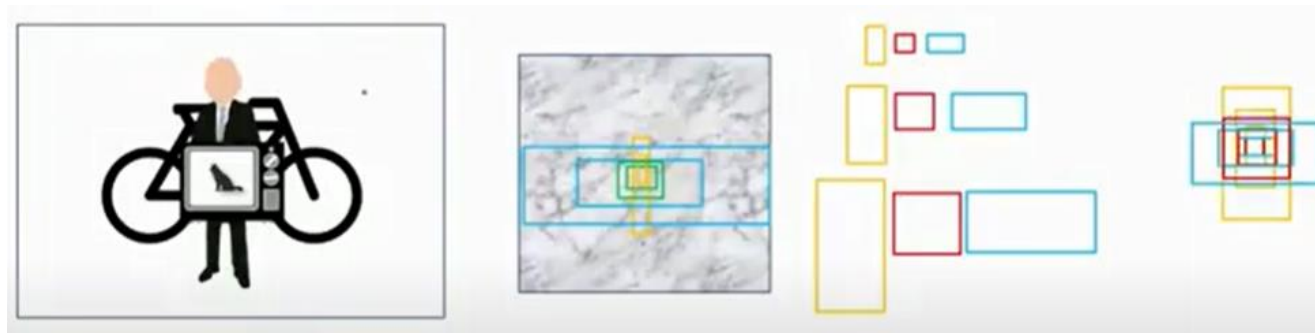
- We now know that the major bottleneck with region based CNNs till now is the traditional and slow region proposal method (selective search)
- We need to replace the selective search method with some other region proposal method based on the following criteria
  - < 2000 region proposals
  - As fast as SS or better
  - As accurate as SS or better
  - Should be able to propose overlapping ROIs with different aspect ratios and scales



# Faster RCNN: Region Proposal Aspect Ratios

---

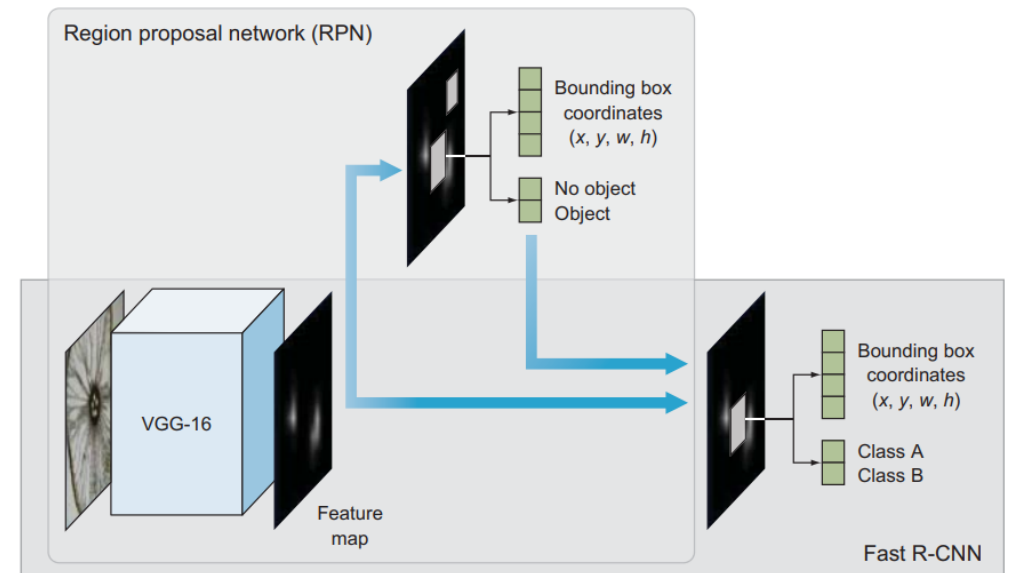
- To detect most kinds of objects you typically use one of the 3 aspect ratios given below
  - Vertical rectangle
  - Square
  - Horizontal rectangle
- Most of the object will be obstructing the one behind and hence you need overlapping ROIs
- Objects maybe of different size therefore you need ROIs of different scale



# Faster RCNN: Components

To address the constraint of selective search for region proposals we have the following addition in Faster RCNN

1. Region Proposal Network (RPN): Selective Search is replaced by a ConvNet that proposes RoIs from the last feature maps of the feature extractor that are considered for investigating objects of interest. RPN has two outputs
  1. Objectness Score: whether there is an object present (1) or absent(0)
  2. Box location
2. Fast RCNN: same components
  1. Base network for feature extraction
  2. RoI pooling layer
  3. Output layer that contains two fully connected layers: softmax classifier and bounding box regression



# Region Proposal Network (RPN)

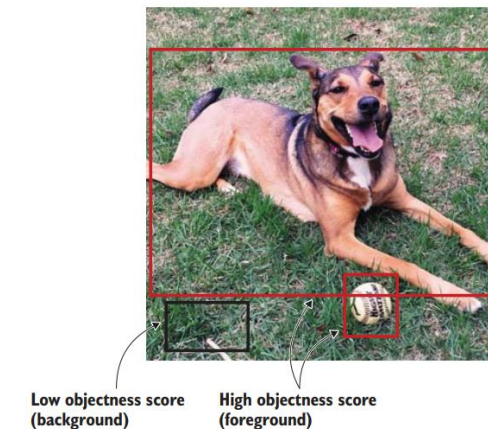
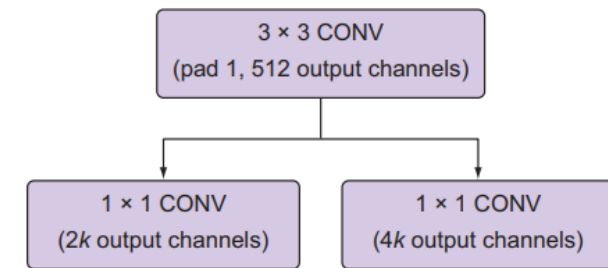
---

RPN identifies RoIs based on the last feature map of the pretrained CNN. Faster RCNN uses RPN to perform the region proposals instead of using selective search like its earlier versions

The architecture of RPN is composed of two layers

1. A 3x3 conv layer with 512 filters
2. Two parallel 1x1 conv layers
  1. Classification layer: Predicts whether there is an object present in the proposed region (or just random background)
  2. Bounding box regression layer: bounding box of proposed RoIs

**Note:** Remember that RPN is not doing object detection but it is only finding objects of interest that the later components can both localize and classify



# How are the region proposed bounding boxes predicted ?

---

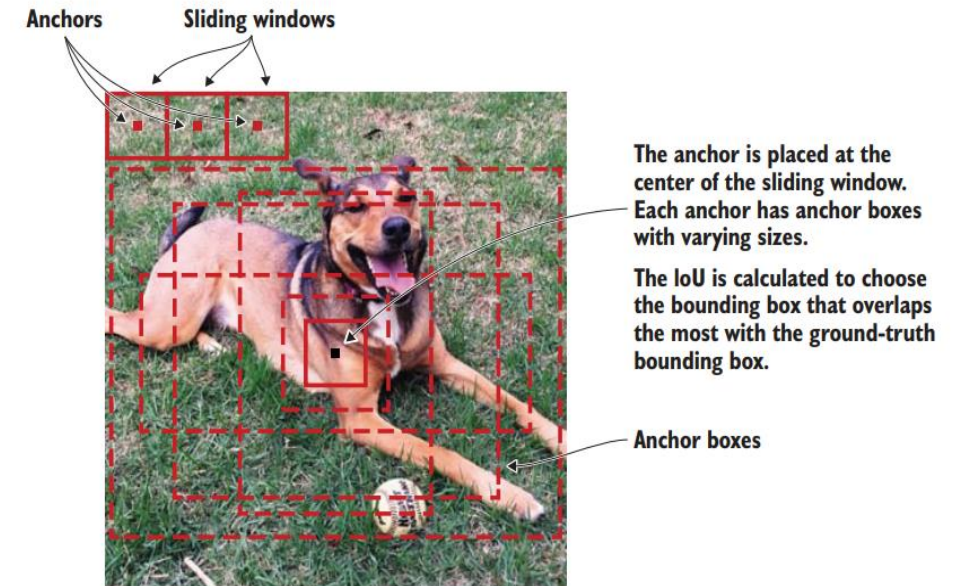
Bounding box can be defined as  $(x,y,w,h)$  where  $x$  and  $y$  are the centers of the bounding box and  $w$  and  $h$  are the widths and heights respectively

We perform relative bounding box regression i.e. we create reference boxes called as anchor boxes in the image and make the regression layer predict the offsets from these boxes called deltas  $(\Delta x, \Delta y, \Delta w, \Delta h)$  to adjust the anchor boxes to better fit the object to get final proposals

# Anchor Boxes

---

- RPN generates  $k$  Rols for each location in the feature map using a sliding window approach. These regions are represented using anchor boxes.
- The anchors are centered in the middle of their corresponding sliding window and are of different aspect ratios and scales to cover a wide variety of different sized objects
- They are fixed bounding boxes that are placed throughout the image to be used for reference when first predicting the region proposals for objects of interest
- In the Faster RCNN paper, anchor boxes with 3 different aspect ratios and 3 different scales were proposed



# Training the RPN: Need to ensure there are less than 2000 RoIS

---

RPN is used to classify whether an object is present or not and propose the bounding box for an object

**How does one prepare the data for whether there is an object present or not inside the anchor box ?**

For each anchor the overlap value is computed which indicates how much these anchor boxes overlap with the ground truth boxes (or how useful these anchor boxes are going to be to help us find the region proposals)

Only the following anchor inputs are taken forward for the objectness classification

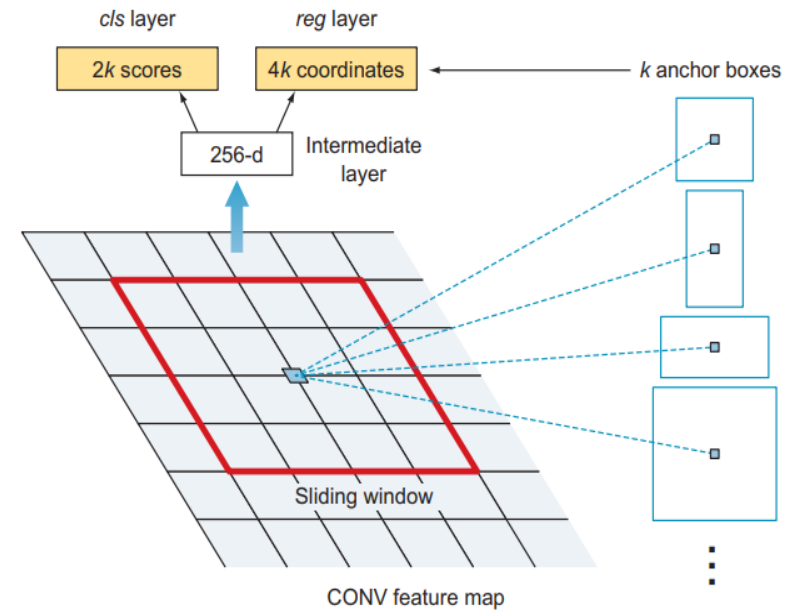
1. Anchor boxes having high IoU with ground truth boxes  $\text{IoU} > 0.7$  (positive anchor boxes)
2. Anchor boxes having low IoU with ground truth boxes  $\text{IoU} < 0.3$  (negative anchor boxes)
3. Rest of the anchor boxes are ignored  $0.3 < \text{IoU} < 0.7$



# Training the RPN Contd.

During the training process the positive and negative anchors are passed as input to two fully connected layers corresponding to the classification of anchors as containing an object or no object, and to the regression of location parameters (four coordinates)

Corresponding to the  $k$  number of anchors from a location, the RPN network outputs  $2k$  scores and  $4k$  coordinates. Thus, for example, if the number of anchors per sliding window ( $k$ ) is 9, then the RPN outputs 18 objectness scores and 36 location coordinates



# Training the Fast RCNN component of Faster RCNN

---

The final fully connected layers of the network take two inputs:

1. Feature maps from the feature extractor
2. Rols coming from the RPN

The Rols are projected on to the feature maps in the same way like Fast RCNN

Softmax Classifier and Bounding Box regressor are used to predict the object class and bounding box offsets respectively for the detected object

# RPN: Loss Function

---

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

$p_i^* = 1$  if anchor box contains ground truth object  
 $= 0$  otherwise

$p_i$  = predicted probability of anchor box containing an object

$N_{cls}$  = batch-size

$N_{reg}$  = batch-size  $\times k$

$k$  = anchor boxes

# Faster RCNN Training in a Nutshell

---

1. Finetune RPN using a pretrained network
2. Finetune Fast RCNN using a pretrained network on the proposed images from step 1
3. Finetune RPN using the pretrained network used in step 2
4. Now finetune the Fast RCNN using the pretrained network used in step 3 on the images proposed in step 3

# Review of RCNN Family

---

## 1. mAP on Pascal VOC 2007

1. RCNN: 66%
2. Fast RCNN: 66.9%
3. Faster RCNN: 66.9%

## 2. Limitations

1. RCNN: 3 stage training, high computation time and memory requirements
2. Fast RCNN: Selective search is slow and hence computation time is still high
3. Faster RCNN: It is still a 2 stage network

## 3. Test Time Per Image

1. RCNN: 50 seconds
2. Fast RCNN: 2 seconds
3. Faster RCNN: 0.2 seconds

## 4. Speedup from RCNN

1. Fast RCNN: 25x
2. Faster RCNN: 250x

# mAP: mean Average Precision

---

Mean Average Precision (mAP) is a performance metric popularly used for object detection models. It is the most popular metric that is used by benchmark challenges for object detection such as PASCAL VOC, COCO, ImageNET challenge, Google Open Image Challenge, etc.

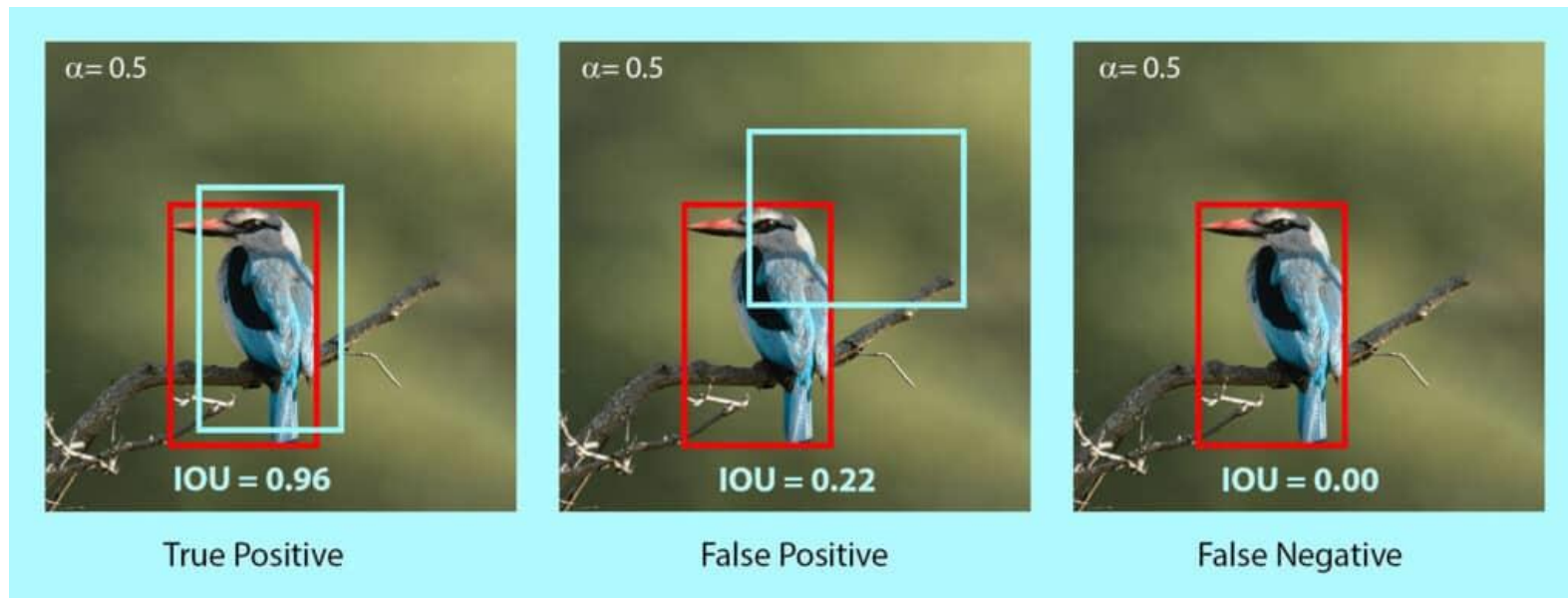
Is it just average of precisions ? Its not that simple.



# mAP: mean Average Precision

---

Remember IoU ?





# mAP: mean Average Precision

---

Remember Precision ?

**Precision** measures the proportion of predicted positives that are actually correct. If you are wondering how to calculate precision, it is simply the **True Positives** out of **total detections**. Mathematically, it's defined as follows.

$$P = TP / (TP + FP)$$

$$= TP / \text{Total True Predicted}$$

The value ranges from 0 to 1.

# mAP: mean Average Precision

---

Remember Recall ?

**Recall** measures the proportion of actual positives that were predicted correctly. It is the True Positives out of all **Ground Truths**. Mathematically, it is defined as follows.

$$R = TP / (TP + FN)$$

$$= TP / \text{Total Ground Truths}$$

Similar to Precision, the value of **Recall** also ranges from 0 to 1.

# mAP: mean Average Precision

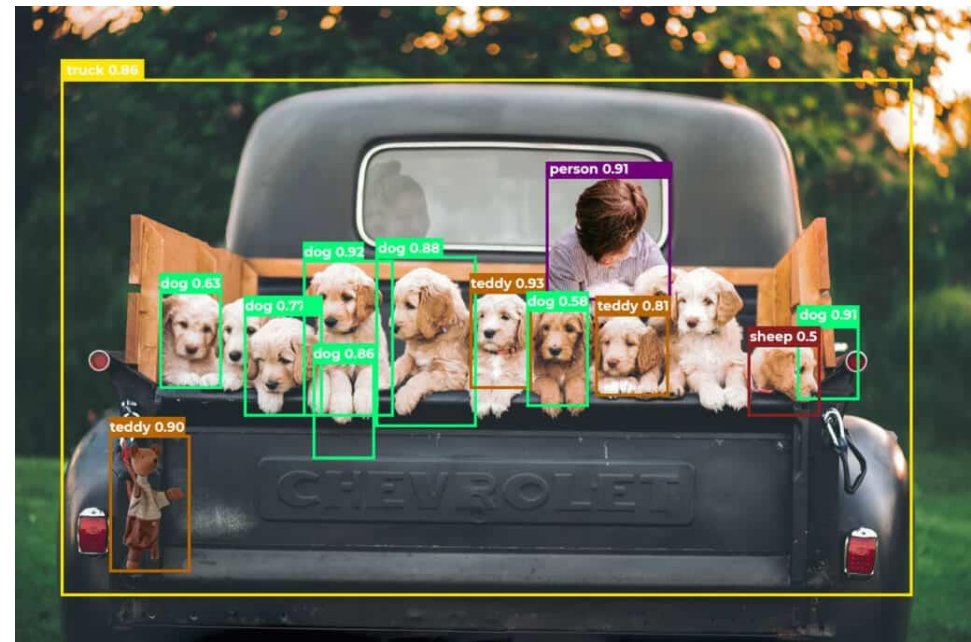
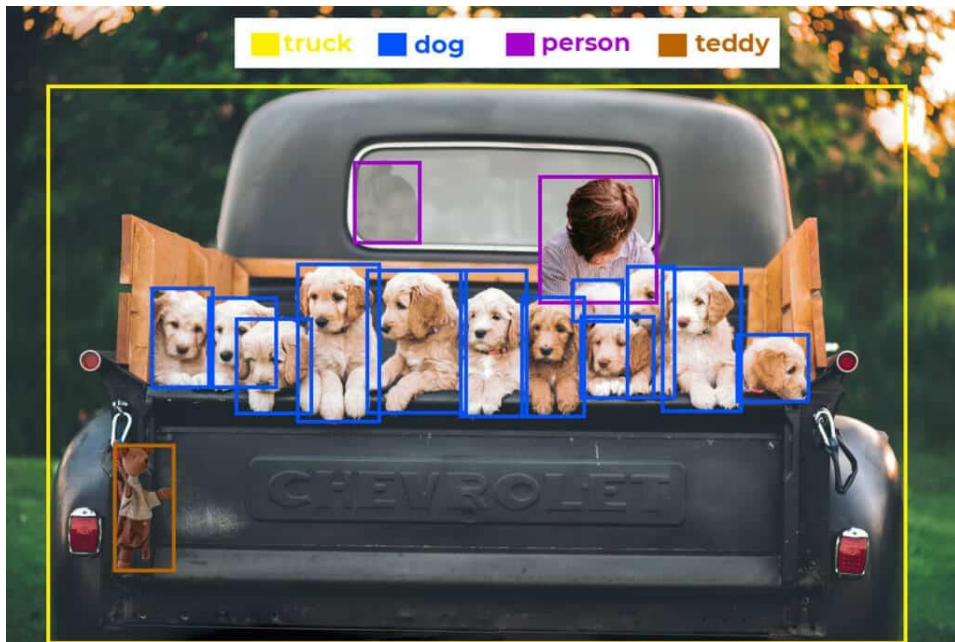
---



# mAP: mean Average Precision

---

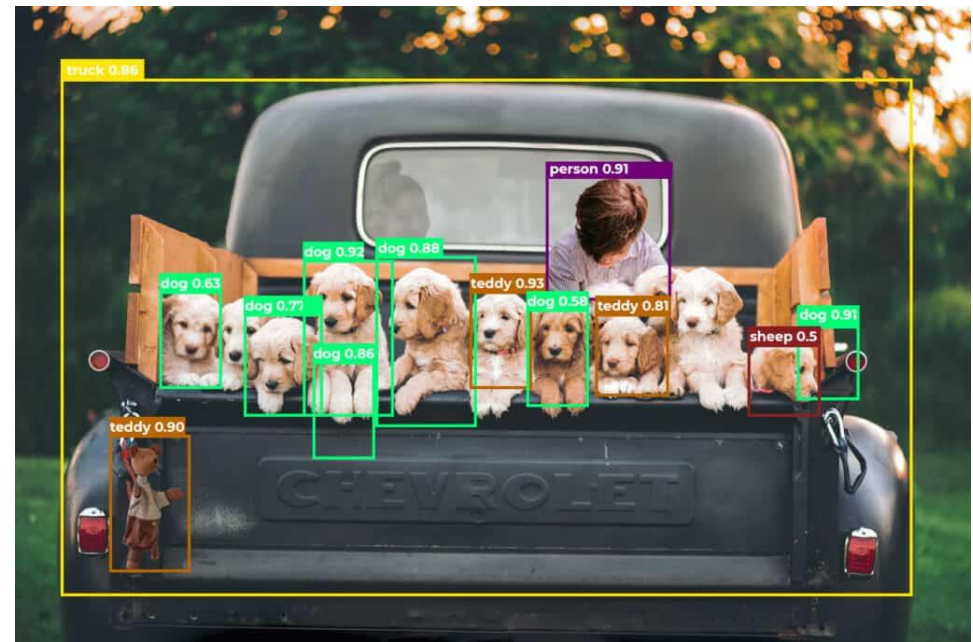
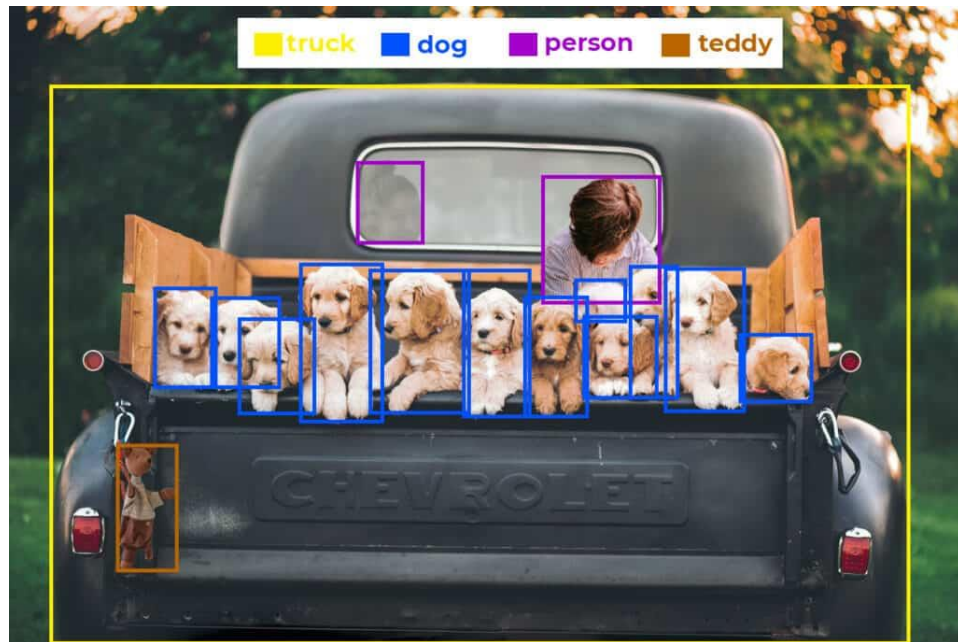
Average Precision (AP) is actually not the average of Precision (P). The term AP has evolved with time. For simplicity, we can say that it is the area under the precision-recall curve.



# mAP: mean Average Precision








---

There are 12 dogs, 2 persons, 1 teddy and 1 truck. The predictions are 7 dogs, 1 person, 3 teddies and 1 truck.



# AP: for class Dog

---

Detections							
Conf.	0.63	0.77	0.92	0.86	0.88	0.58	0.91
Matches GT by IoU?	TP	TP	TP	FP	TP	TP	FP

# mAP: mean Average Precision

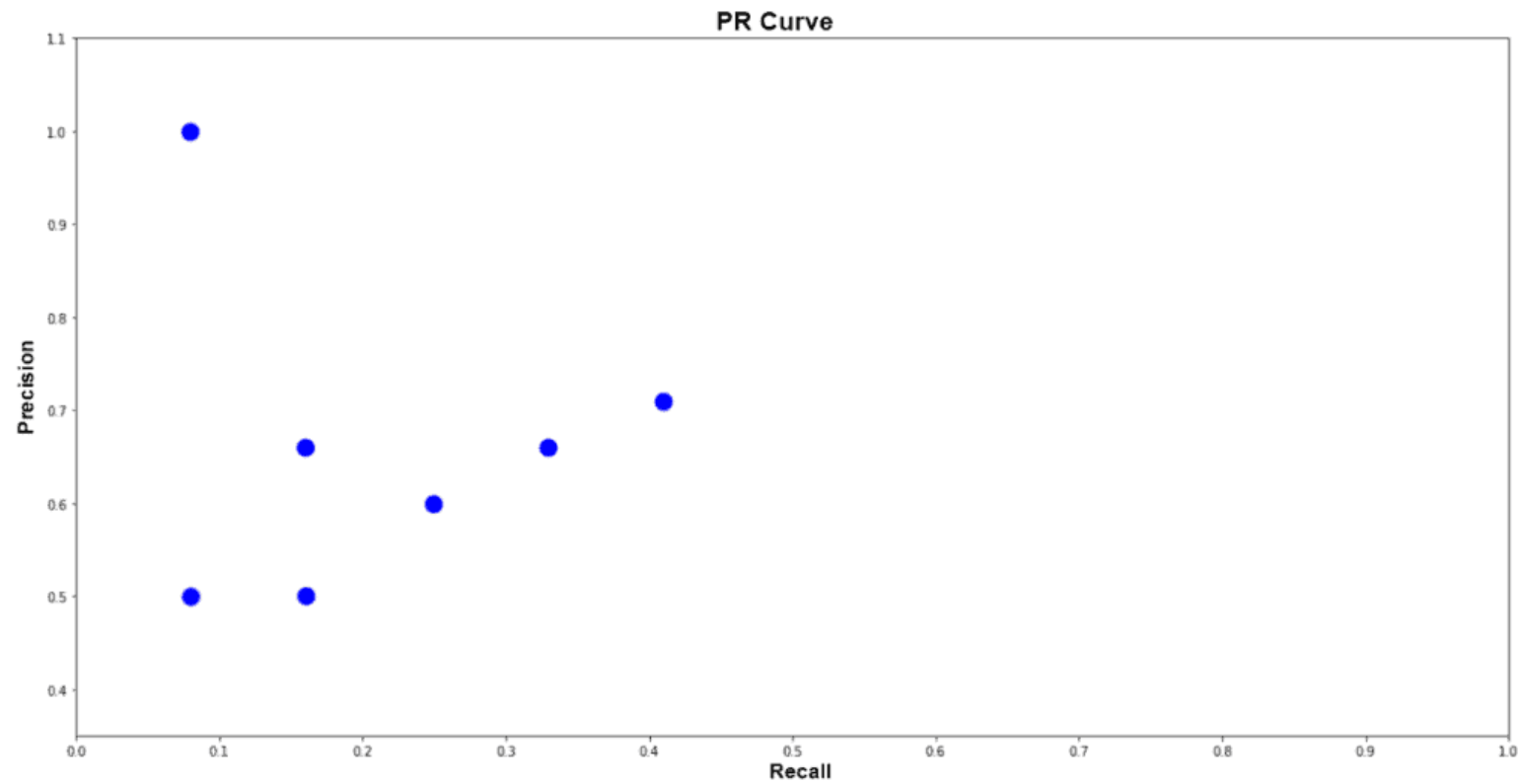
---

\*\* space for lethal math \*\*



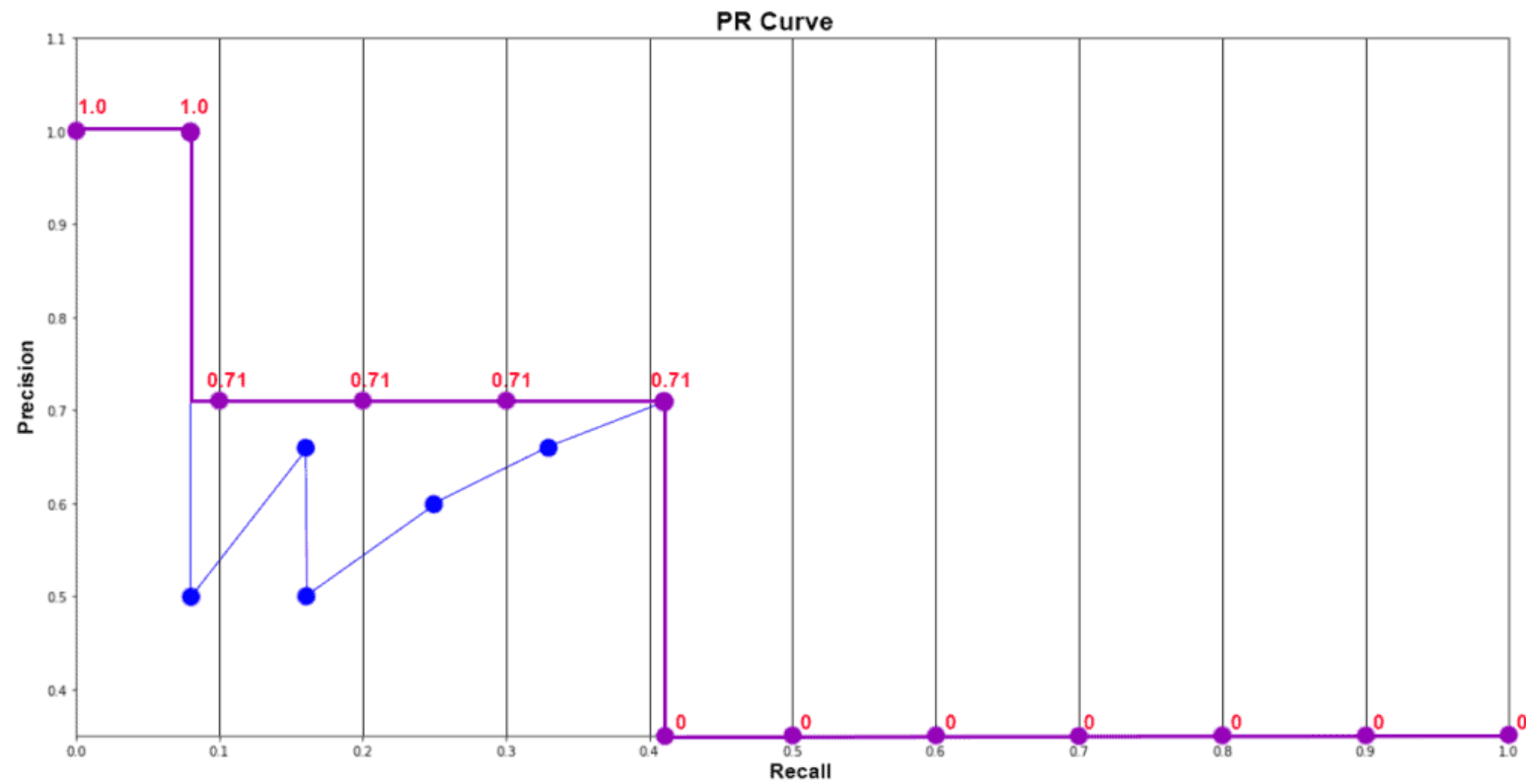
# mAP: PR curve

---



# mAP: PR curve

---



# mAP: mean Average Precision

---

If we calculated just AP for dog ..... does that mean mAP is nothing but the mean of average precision calculated for each class ? Yes

As the name suggests, **Mean Average Precision** or **mAP** is the average of **AP** over all detected classes.

$\text{mAP} = 1/n * \text{sum}(\text{AP})$ , where n is the number of classes.

In the example above, we have 5 classes. Therefore, the calculated mAP is;

$\text{mAP} = 1/5 * (0.349 + 0.545 + 0 + 1 + 0.5)$  (example APs)

= 0.4788

= 47.88 %

# What is $\text{mAP}@0.50$ or $\text{mAP}@0.50:0.95$ ?

---

We decided the TPs and FPs wrt the IoU values threshold at 0.5. i.e. if  $\text{IoU} > 0.5$  then it's a TP else it's a FP. Therefor the mAP calculation done for IoU thresh = 0.5 is known as  $\text{mAP}@0.50$

Similarly MS COCO defines Average Precision (AP) as  $\text{mAP}@[0.5:.05:.95]$  i.e. an average of mAPs at 0.50, 0.55, 0.60.....0.95

*Thank you*

For any queries drop an email at: [quadeershaikh15.8@gmail.com](mailto:quadeershaikh15.8@gmail.com)