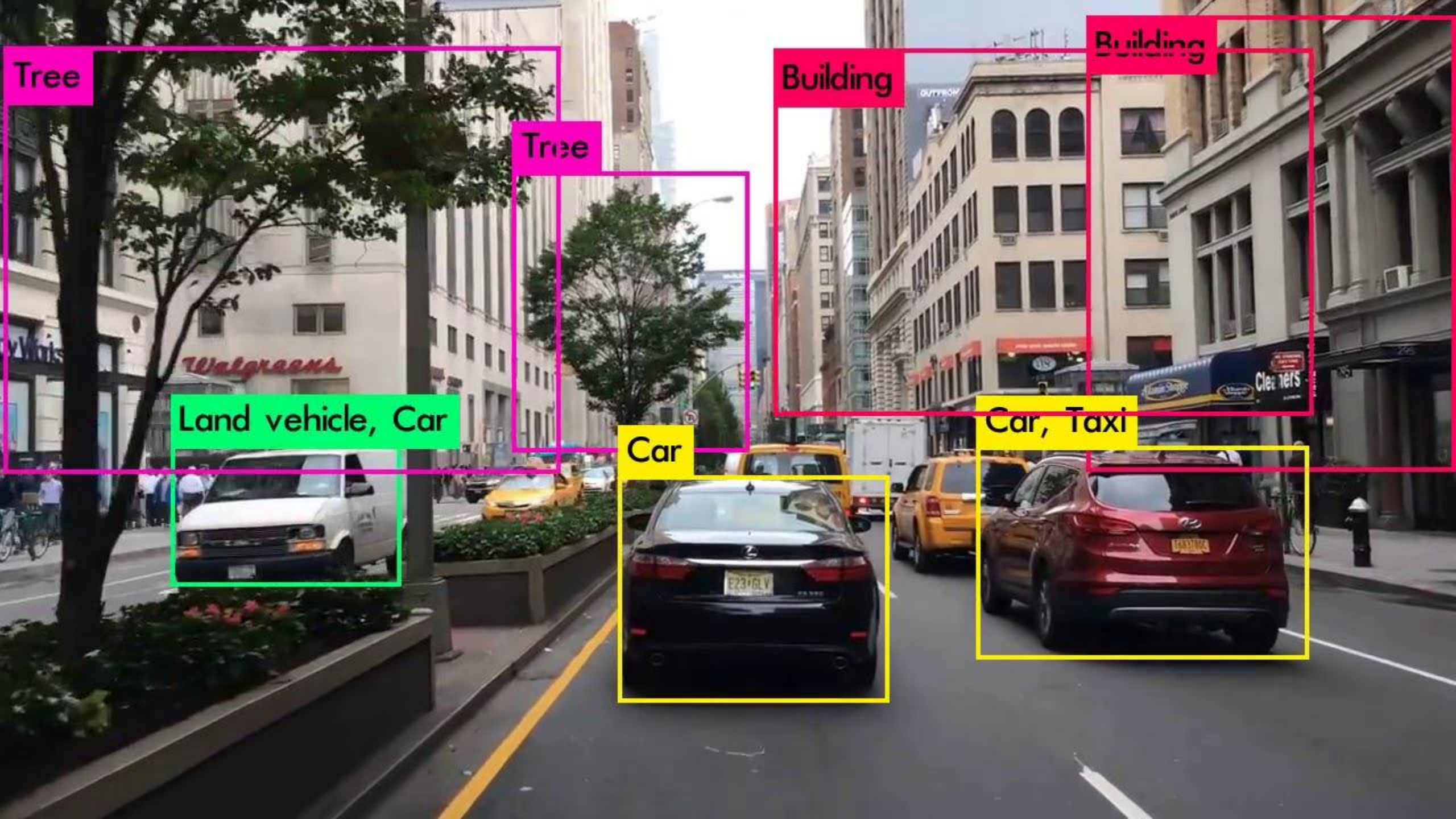
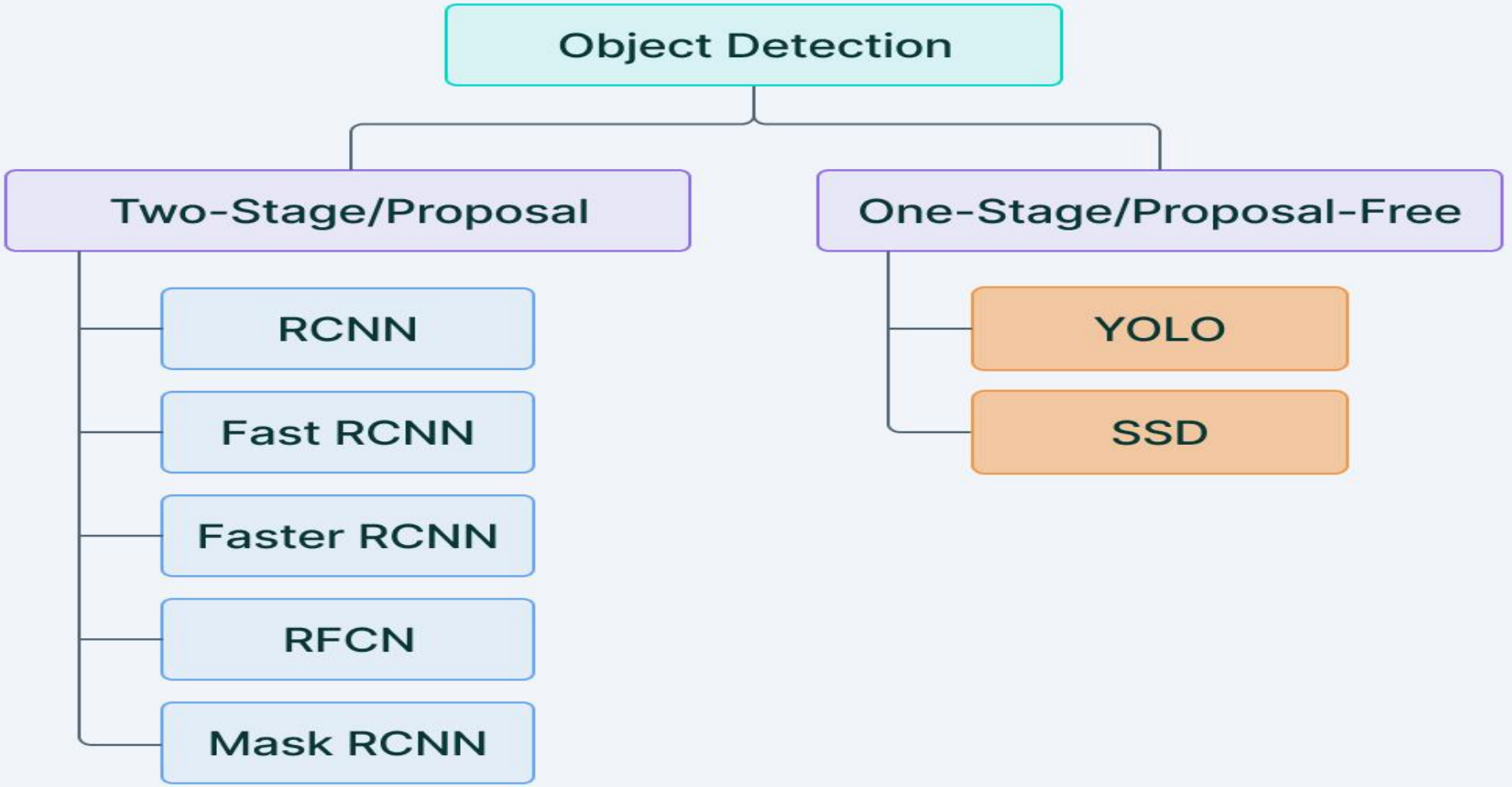

YOLO
Object
Detection
truck
traffic light
car
person
car
car
bicycle
car
car
car
car
car
bicycle


Tree
Tree
Building
Building
Land vehicle, Car
Car
Car, Taxi
Walgreens

One and two stage detectors



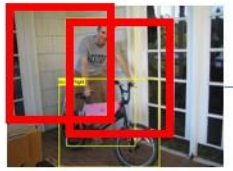
Before YOLO

Car/non-car Classifier

Training set:

Crop + Resize with Sliding Window + Image Pyramid

Sliding Window – Location
Image Pyramid - Scale



AlexNet/VGG

Conv and Pool Layers
As Feature Extractors

Feature Maps

Get Class scores
Using Softmax

Get Bounding boxes
Using L2 loss
(x1, y1, x2, y2)

Training set:

x	y
	1
	1
	1
	0
	0

Image is Scanned by Multiple Windows with Different Sizes.
Sub Images are extracted and annotated with:
(W,H) of Sliding window and (x,y) of sliding window position

System is trained to classify each sub image
(No training for W,H,x,y)

The more windows sizes, the more accurate W,H of object
The more smooth sliding, the more accurate Position of object

Examples of Objects Detection Data sets

VOC 2007 / 2012:

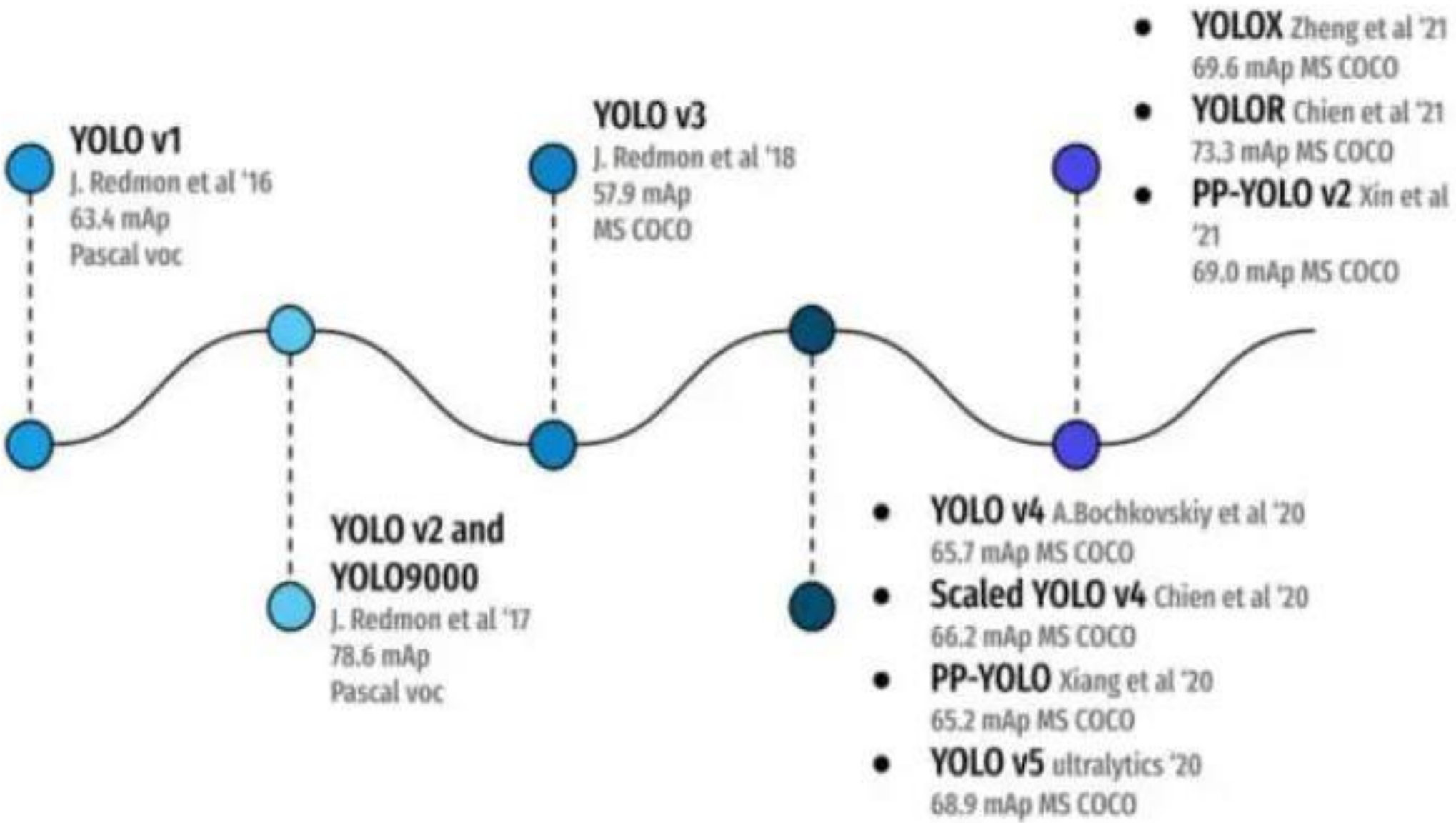
- 20 classes
- e.g. person, cat, dog, car, chair, bottle

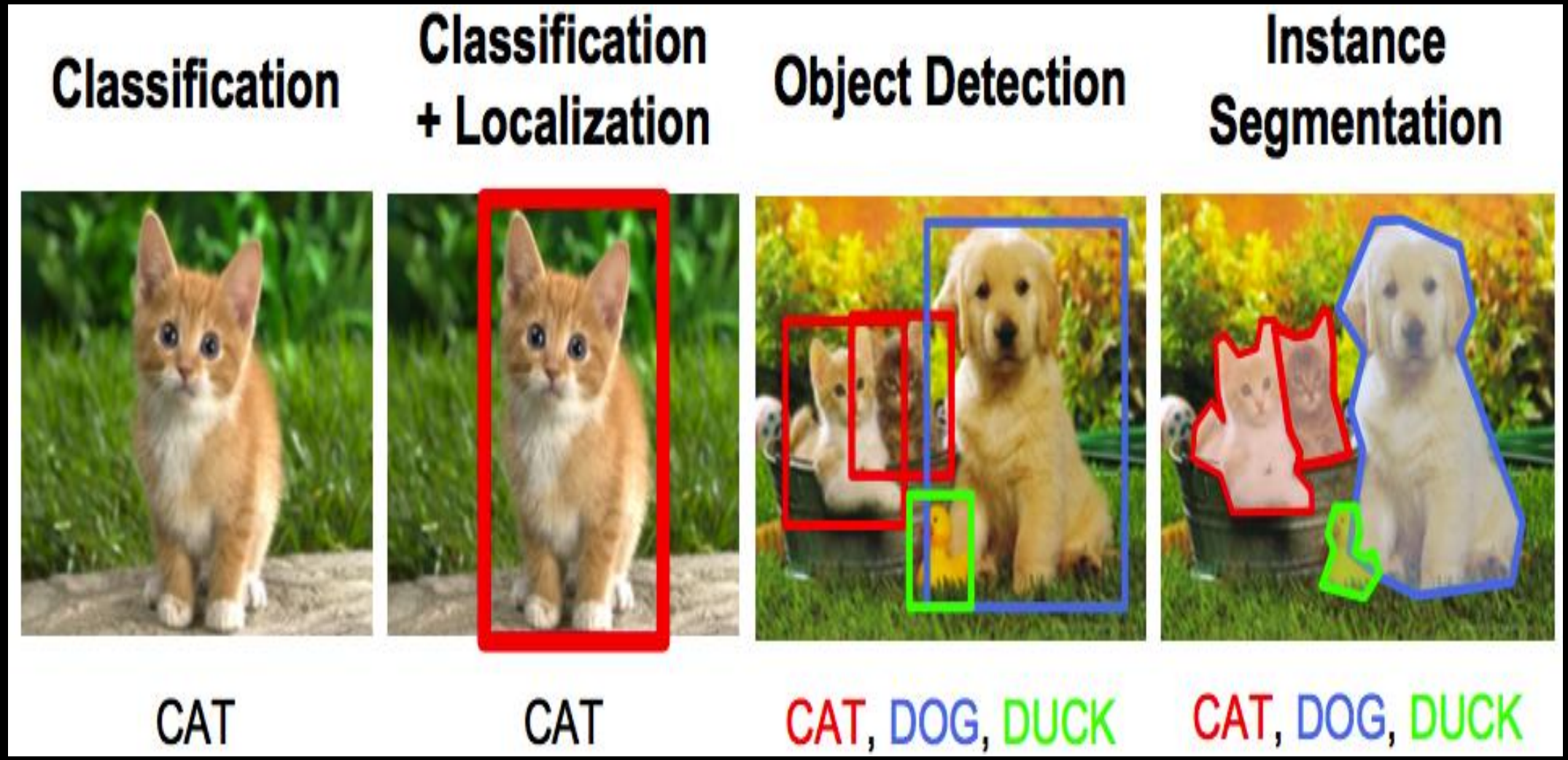
MS COCO:

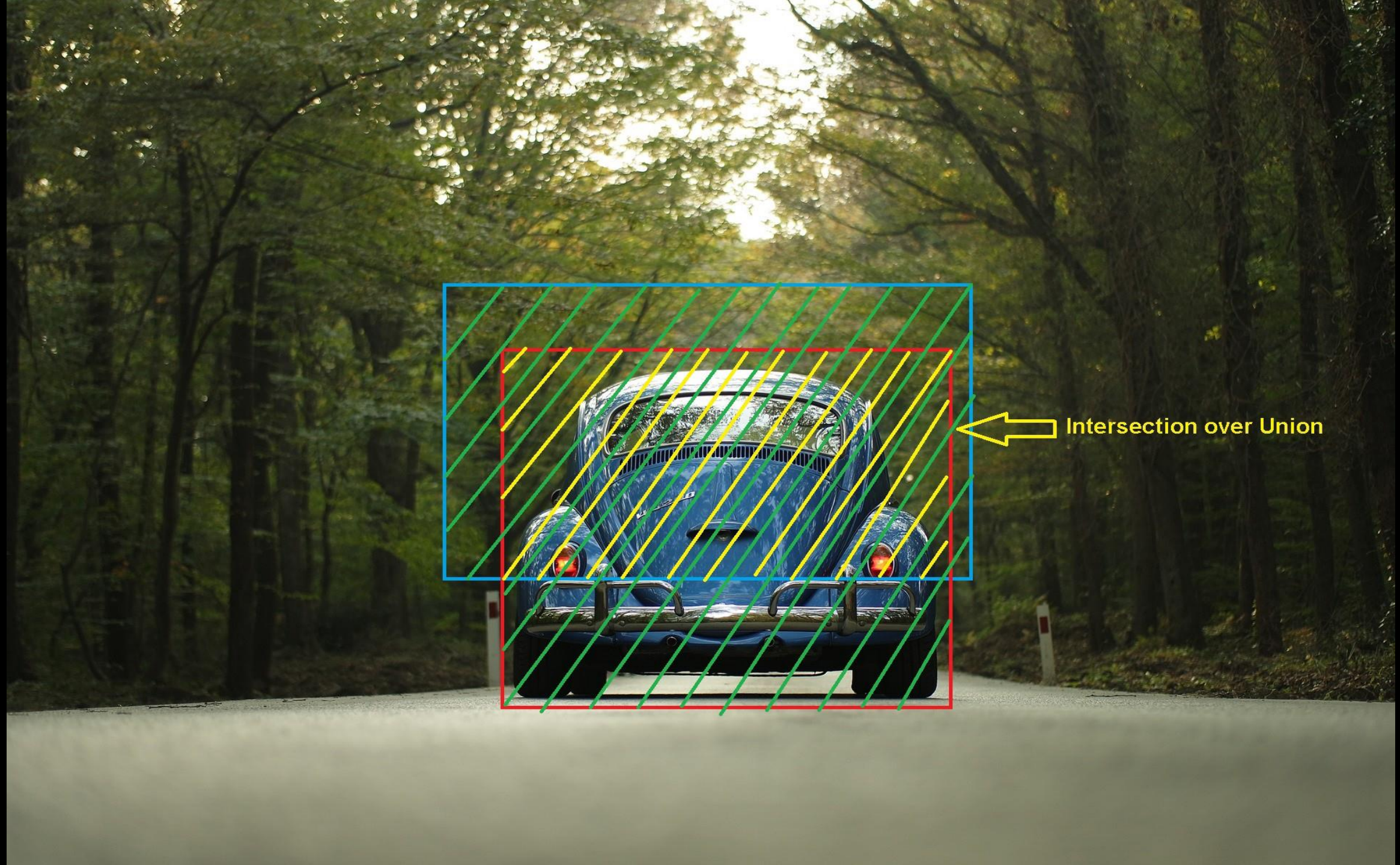
- 80 classes
- e.g. book, apple, teddy bear, scissors

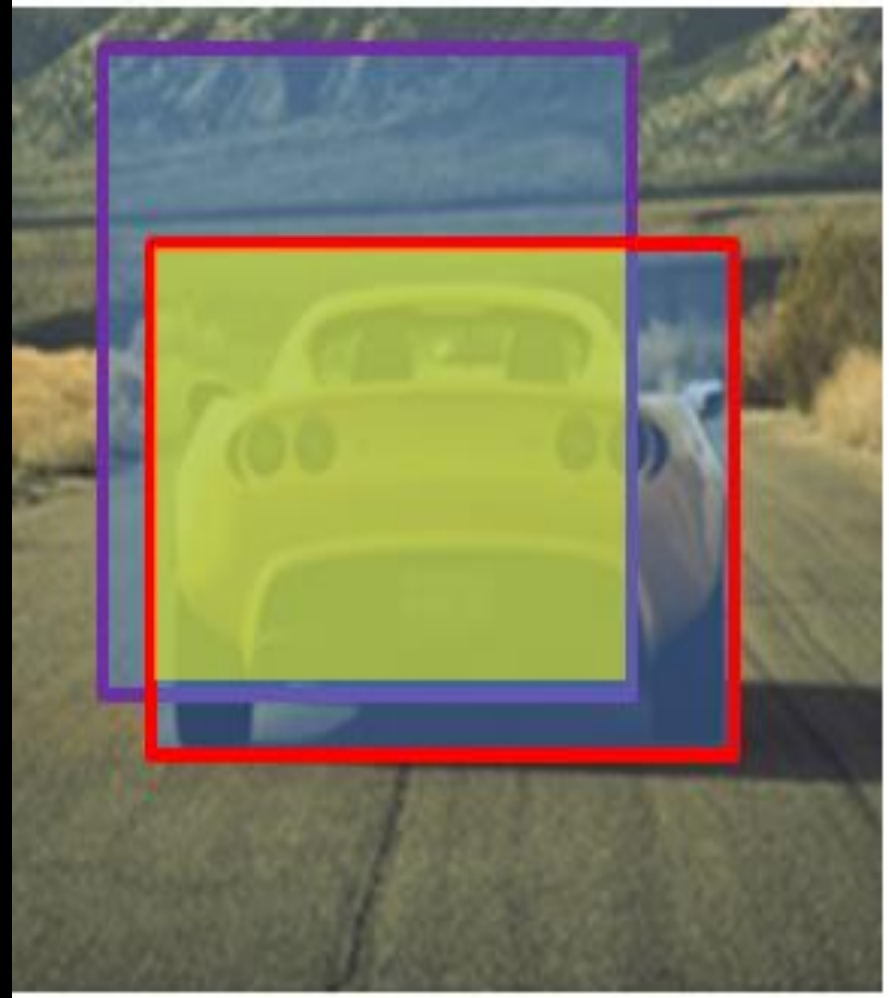
ImageNet1000:

- 1000 classes
- e.g. German shepherd, golden retriever, European fire salamander


YOLO v1
J. Redmon et al '16
63.4 mAp
Pascal voc
YOLO v2 and YOLO9000
J. Redmon et al '17
78.6 mAp
Pascal voc
YOLO v3
J. Redmon et al '18
57.9 mAp
MS COCO
YOLO v4 A.Bochkovskiy et al '20
65.7 mAp MS COCO
Scaled YOLO v4 Chien et al '20
66.2 mAp MS COCO
PP-YOLO Xiang et al '20
65.2 mAp MS COCO
YOLO v5 ultralytics '20
68.9 mAp MS COCO
YOLOX Zheng et al '21
69.6 mAp MS COCO
YOLOR Chien et al '21
73.3 mAp MS COCO
PP-YOLO v2 Xin et al '21
69.0 mAp MS COCO


Classification
Classification + Localization
Object Detection
Instance Segmentation
CAT
CAT
CAT, DOG, DUCK
CAT, DOG, DUCK

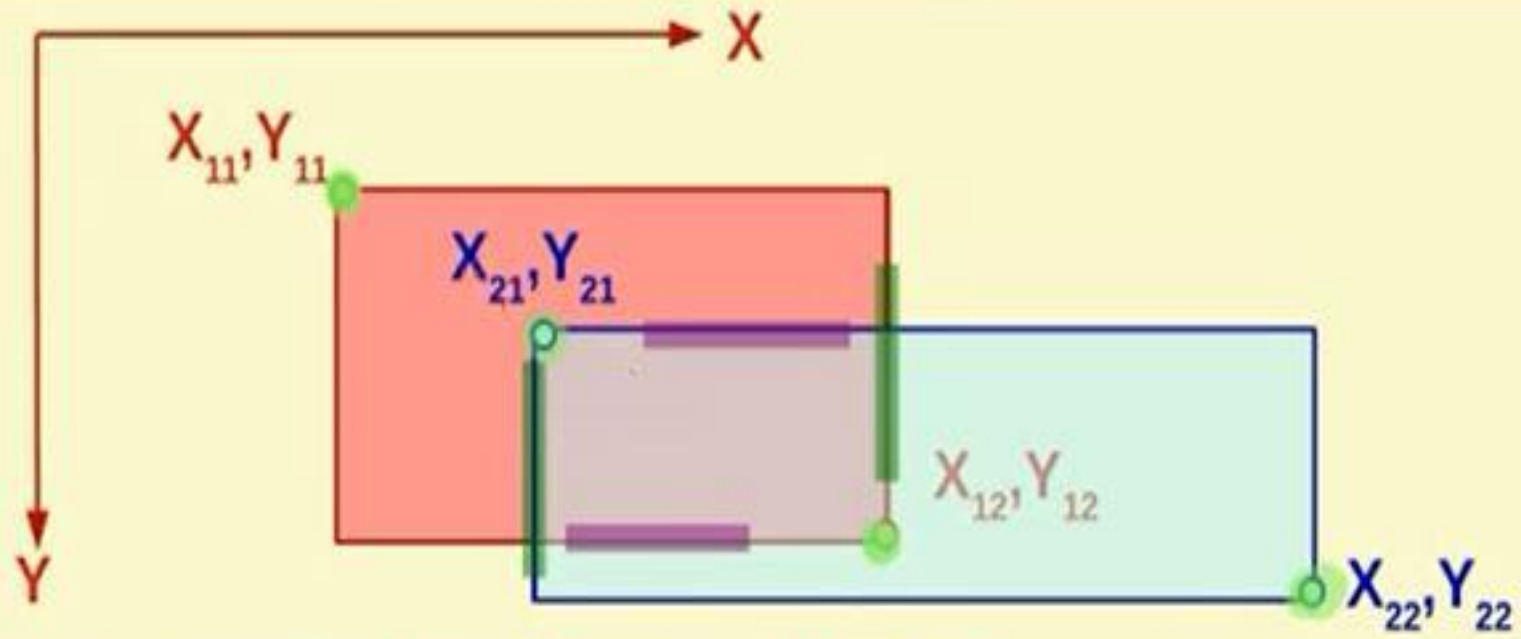

Intersection over Union



Intersection over union (IoU)

$$= \frac{\text{size of } \square}{\text{size of } \square}$$

"Correct" if IoU $\geq 0{,}5$



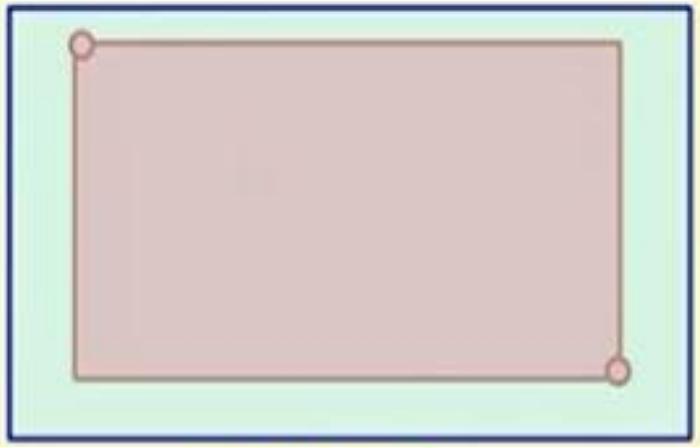
Intersection Width = Min (X_{12}, X_{22}) - Max(X_{11}, X_{21})

Intersection Height = Min (Y_{12}, Y_{22}) - Max(Y_{11}, Y_{21})

Intersection AREA = Intersection Width * Intersection Width

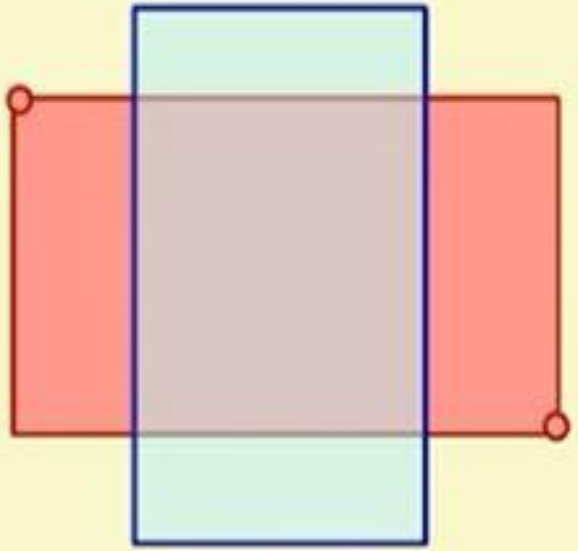
Union AREA = $(X_{12} - X_{11}) * (Y_{12} - Y_{11}) + (X_{22} - X_{21}) * (Y_{22} - Y_{21})$ - Intersection AREA

IOU = [Intersection AREA] / [Union AREA]

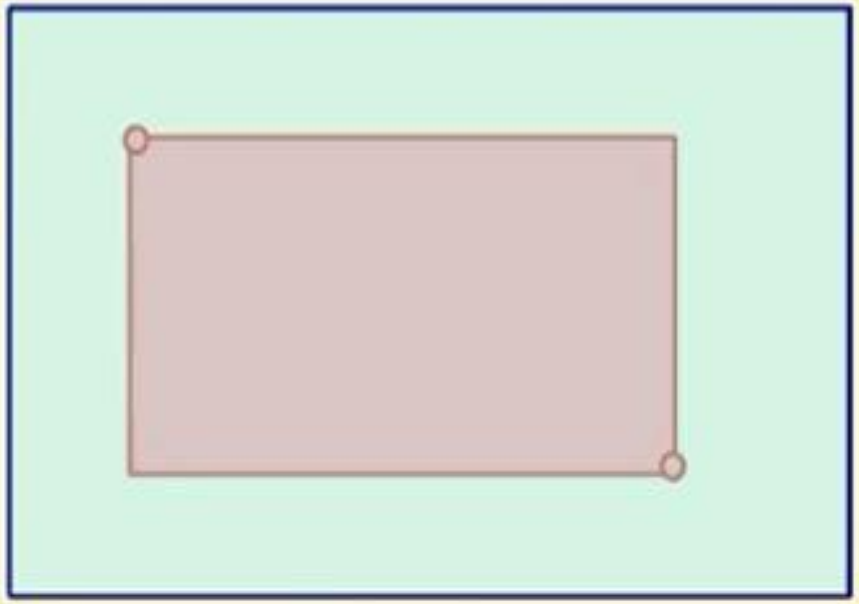


IOU < 1

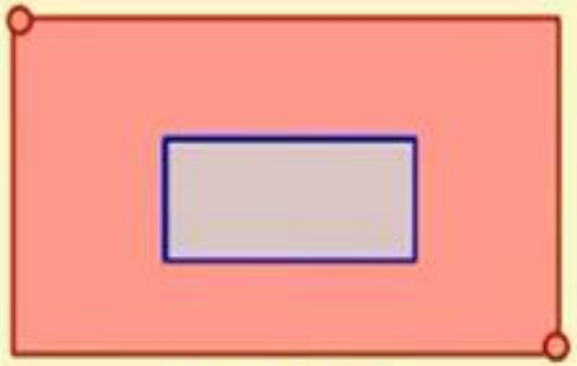
IOU = 1



IOU < 1

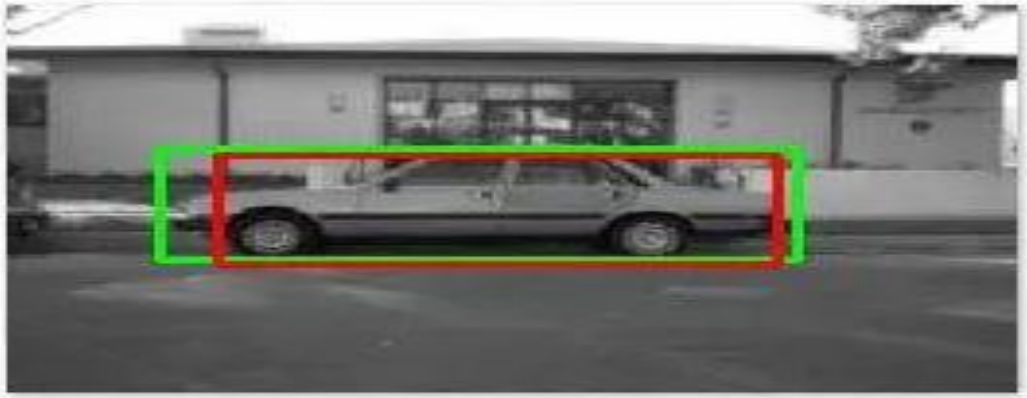


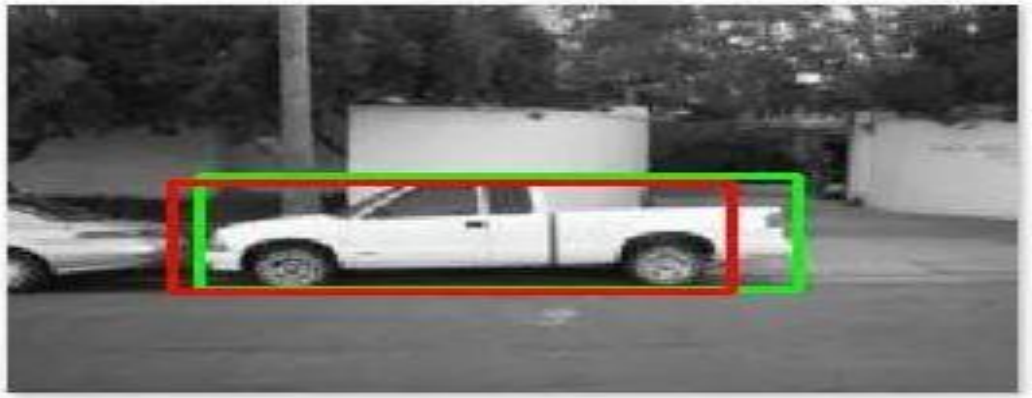
IOU << 1

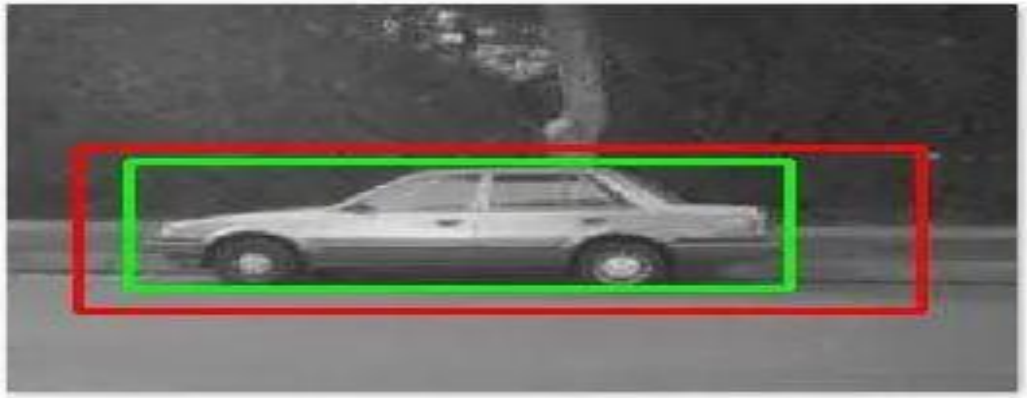


IOU << 1

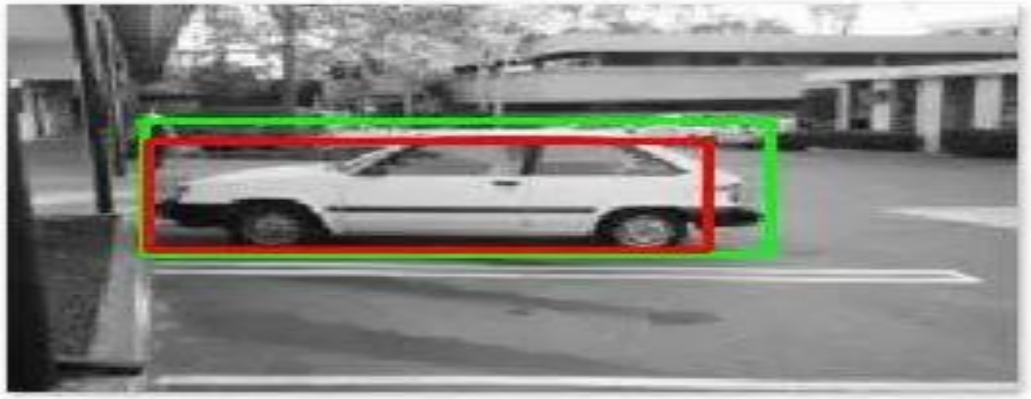

IoU: 0.4034
IoU: 0.7330
IoU: 0.9264
Poor
Good
Excellent







ground-truth bounding boxes (**green**)
predicted bounding boxes (**red**)



[1] Used as Quality measure of Predicted Bounding Boxes

[2] Used in Loss Calculation (YOLO Version 2,3)

[3] Used in Assignment of Anchor Box to Object Bounding Box

[4] Used in Selection of BEST Predicted Box (in Non-Max Suppression Algorithm)

[5] Used as Similarity Measure in K-Means Clustering Algorithm (in order to select Best “K” Anchors sizes and Aspect Ratios)

How yolo algorithm work

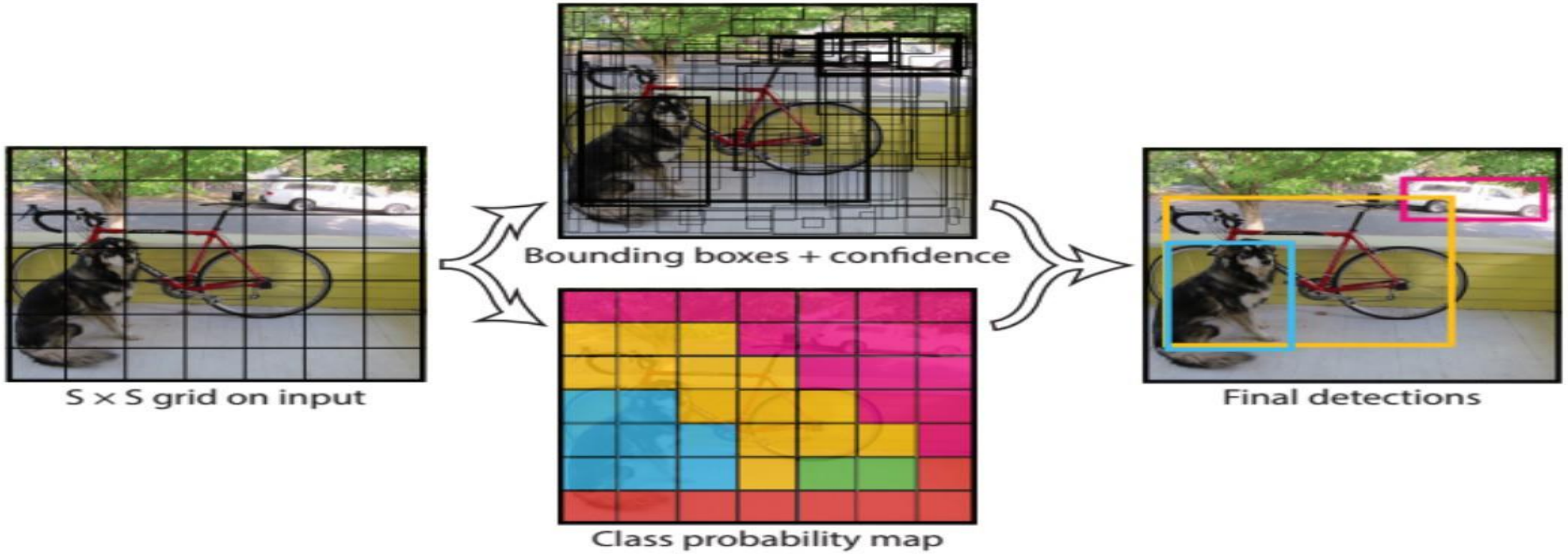
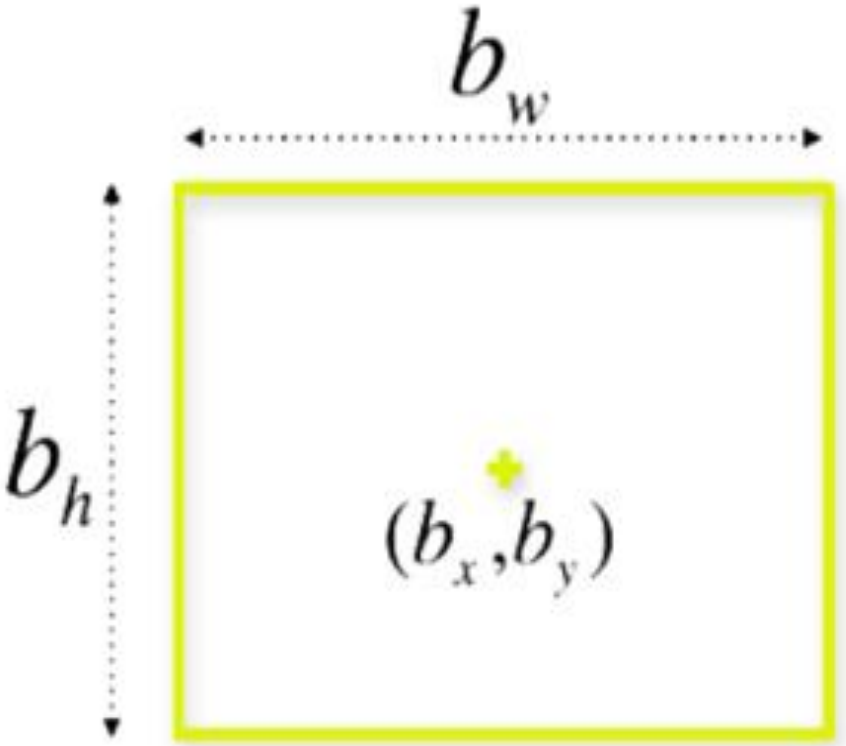


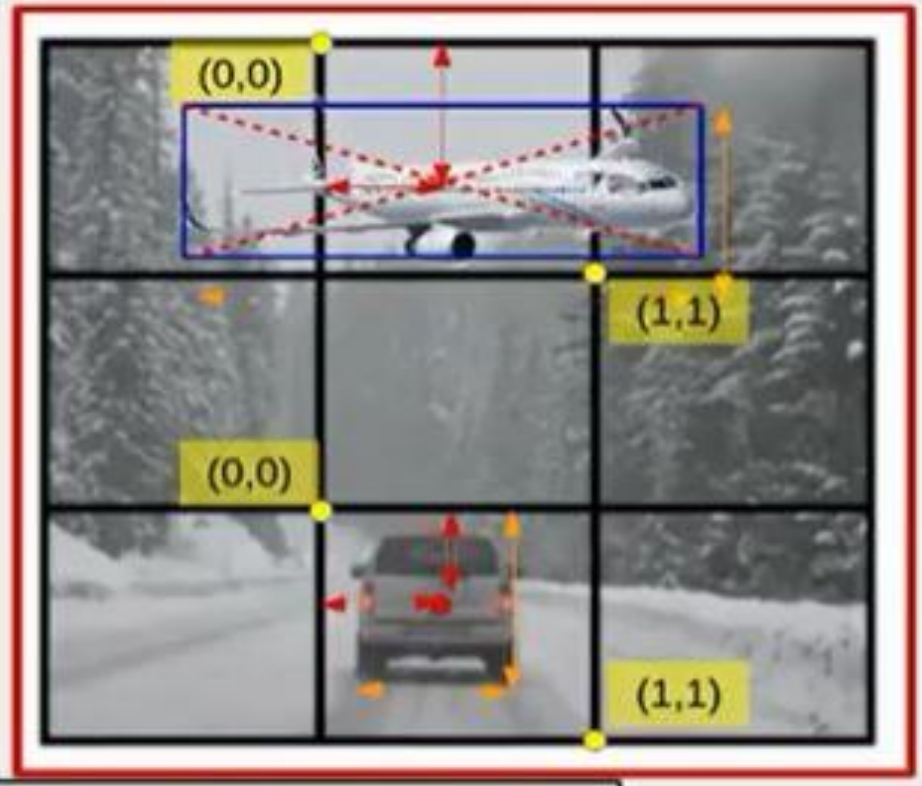
Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

Labeling data for training process

$$y = (p_c, b_x, b_y, b_h, b_w, c)$$



Encoding of Object Coordinate (Cell Based Encoding)



	X,	Y,	W,	H
Plane:	0.4,	0.7,	1.9,	0.6
Car:	0.5,	0.4,	0.5,	0.7

Object Center

Image is divides into Grid Cells.

Each Grid box is associated with

ONE Object Center *(One Anchor)*

For Each Gird Cell,

(0,0) is the top left corner

(1,1) is the bottom right corner

X,Y are relative to top left corner **of Cell**

X,Y values ranges from (0 to 1)

Width and height may exceed “1”

Encoding of Object Coordinate (Image Based Encoding)

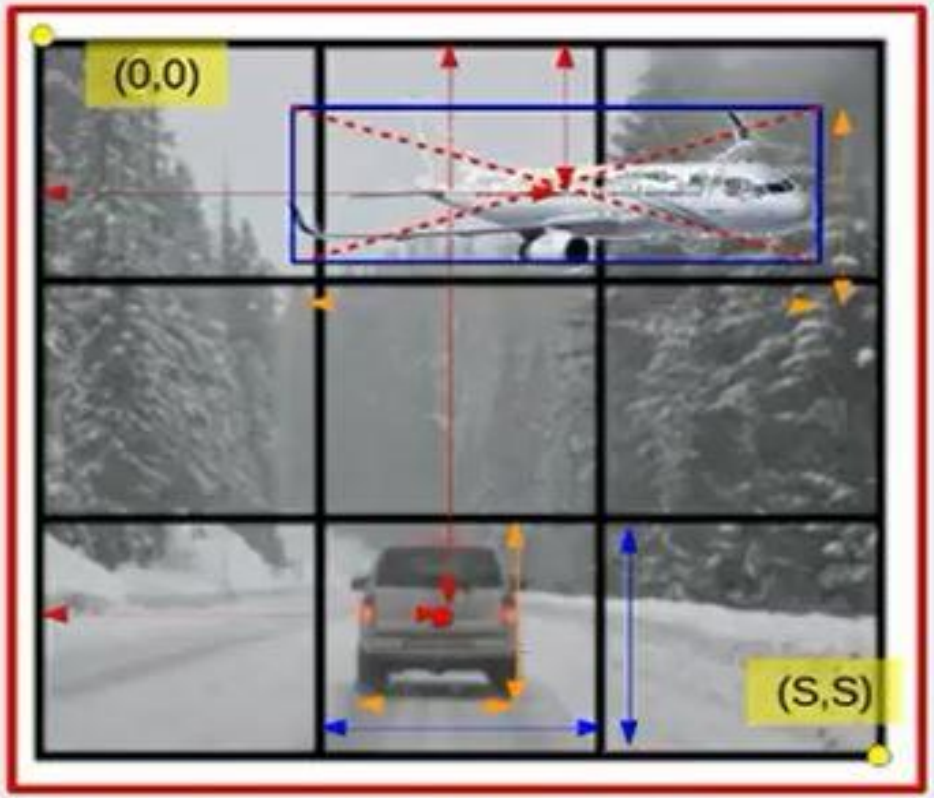


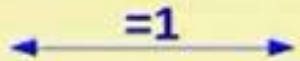
*Image is divided into S*S Cells (e.g. 3x3 Cells)*

	X,	Y,	W,	H
Plane:	1.8,	0.7,	1.9,	0.6
Car:	1.5,	2.4,	0.5,	0.7

Image is divides into Grid cells.

Each Grid box is associated with ONE Object Center *(One Anchor)*

Using Cells Unit



(0,0) is the top left corner of the IMAGE

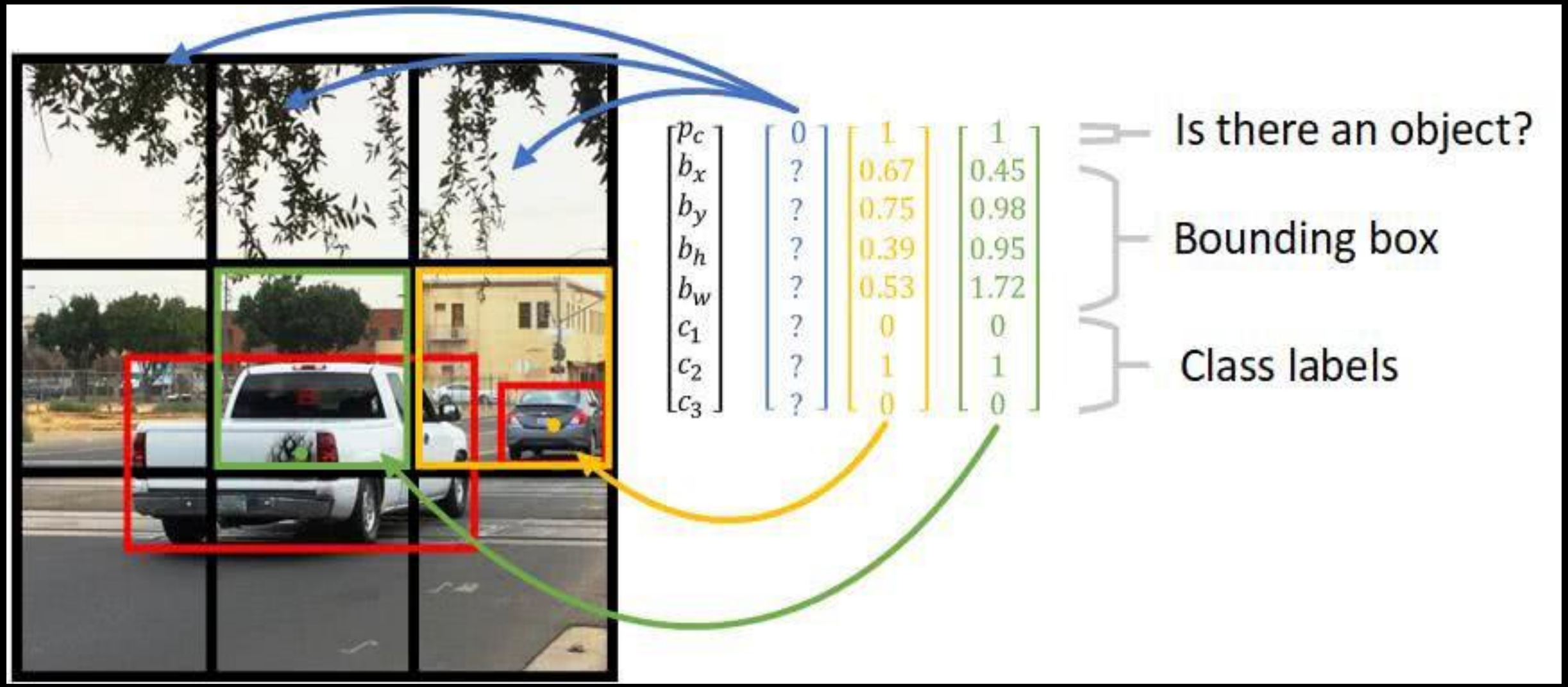
(S,S) is the bottom right corner of the Image

X,Y are relative to top left corner **of Image**

X,Y values ranges from (0.x to S)

Width and height ranges from (0.x to S)

Encode labels



One Anchor Box

One Grid Cell Contains One Object at Most (No Overlapped Objects, small Grid Boxes)

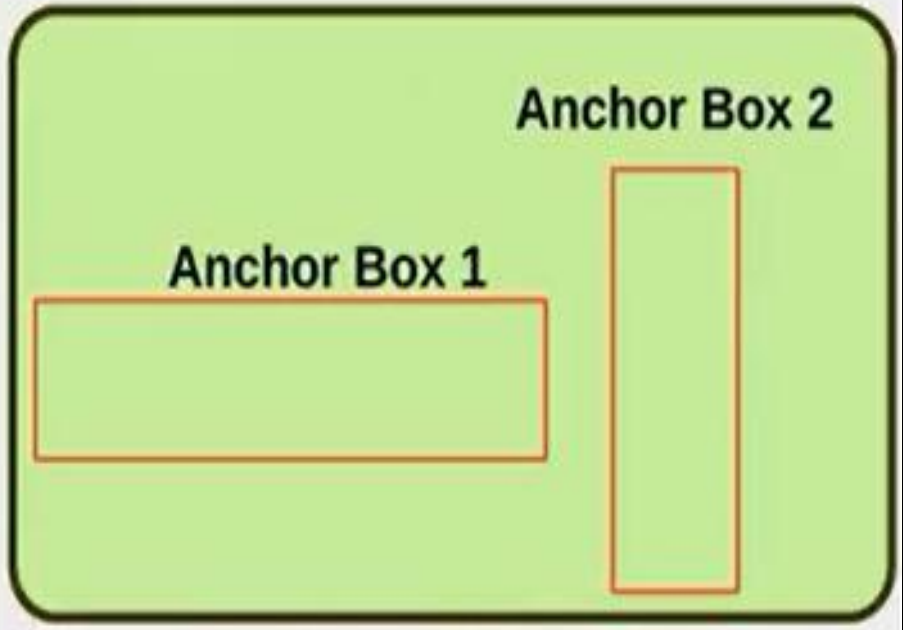
Max Number of Detected Objects = Cell Conut

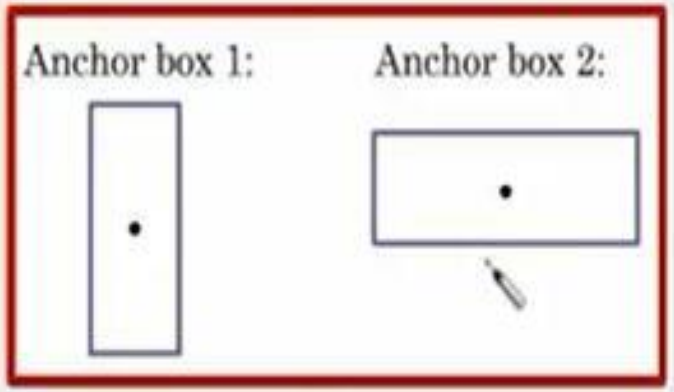
Each Object in training image is assigned to grid cell that contains that object's midpoint

What if there are multiple objects in a single grid?

That can so often be the case in reality in Many Cases
Such as:
- Overlapped Objects in Same grid
- Large grid size and small objects

That leads to the concept of anchor boxes

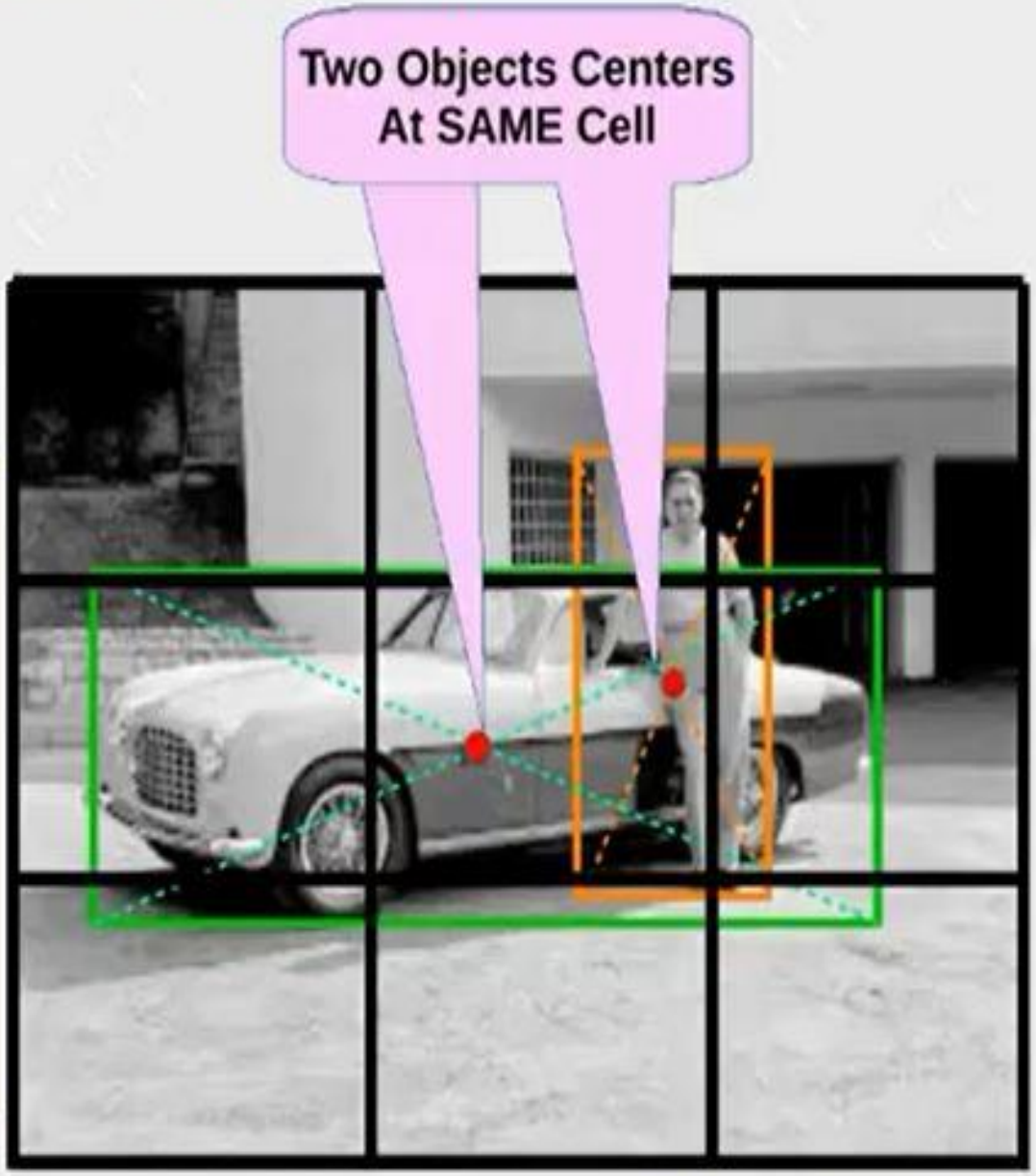




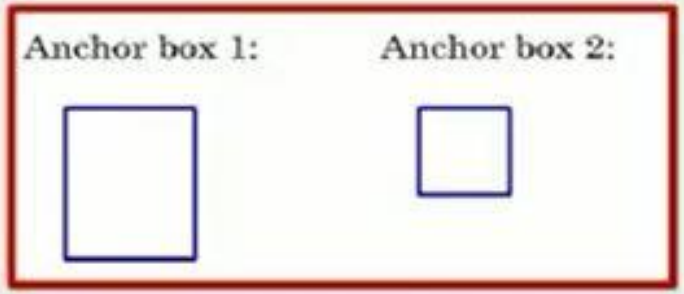
The objects are assigned to the anchor boxes based on:

The similarity of the bounding boxes and the anchor box shape.

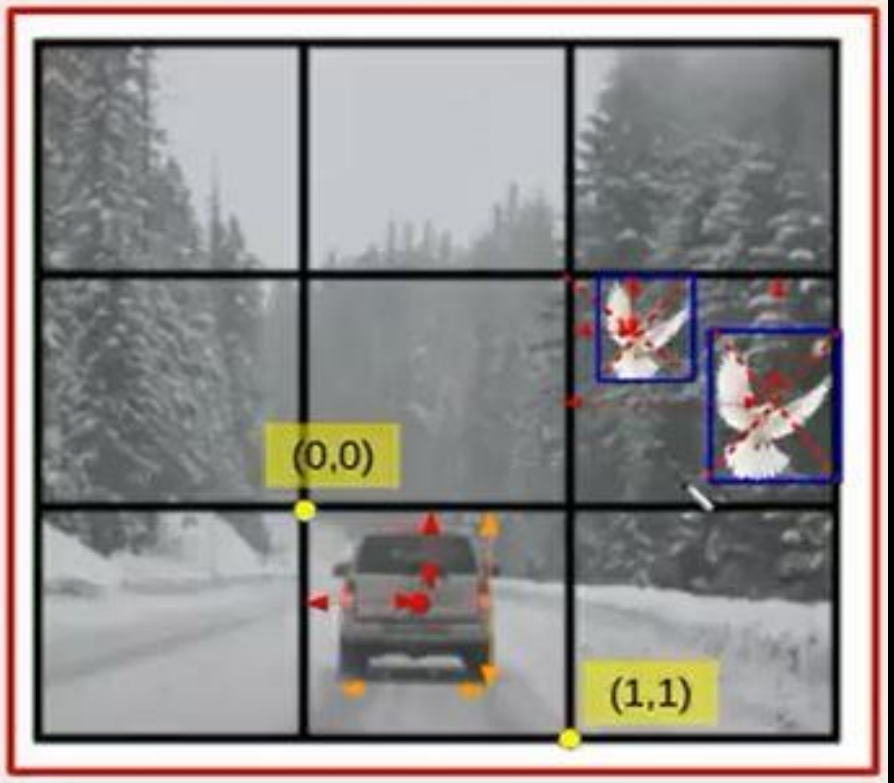
Since the shape of **anchor box 1 is similar to the bounding box for the person**, the latter will be assigned to anchor box 1 and the **car will be assigned to anchor box 2**.

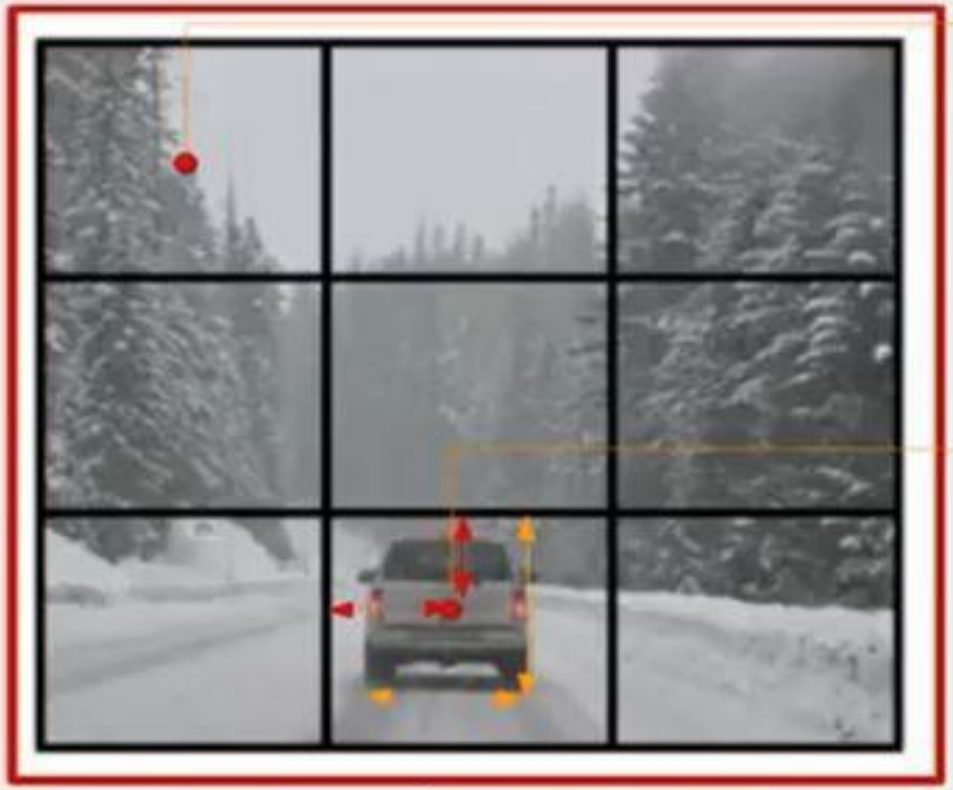


Multiple Anchor Boxes (for Small Objects)



- Select Anchor Box Similar to Each Object (size and Aspect Ratio)
- Each Anchor Box will be associated with (Object X,Y,W,H, Object Class Type, In Addition to Confidence Probability that there Exist an object)





1 - pedestrian
2 - car
3 - motorcycle

y is $3 \times 3 \times 2 \times 8$

Num. of Boxes In the Grid

Num. of Anchor Boxes

P_c , b_x, b_y, b_w, b_h, + Num. of Classes

1

2

p_c b_x b_y b_h b_w c_1 c_2 c_3

p_c b_x b_y b_h b_w c_1 c_2 c_3

0 ? ? ? ? ? ? ? 0 ? ? ? ? ? ? ?

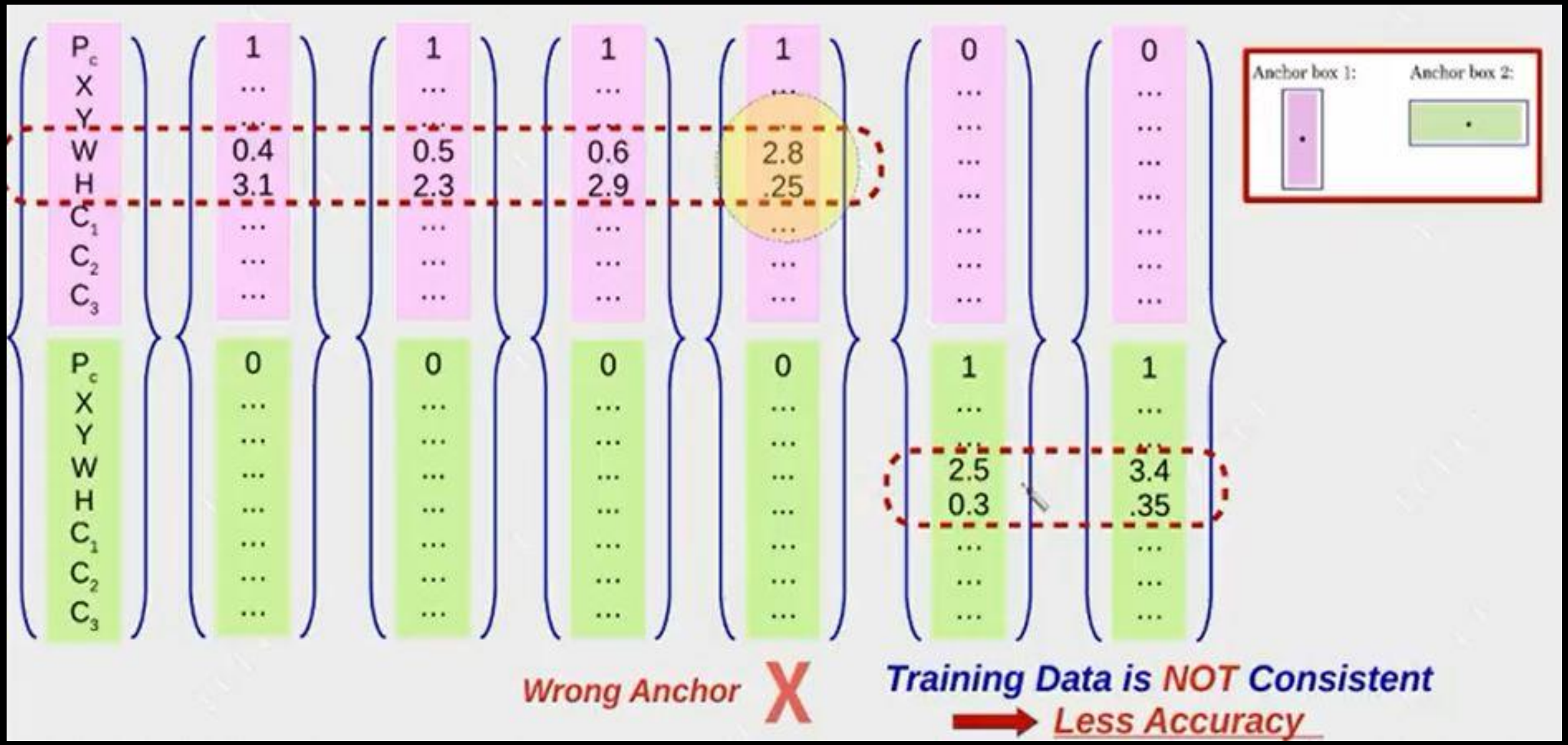
0 ? ? ? ? ? ? ? 1 b_x b_y b_h b_w 0 1 0

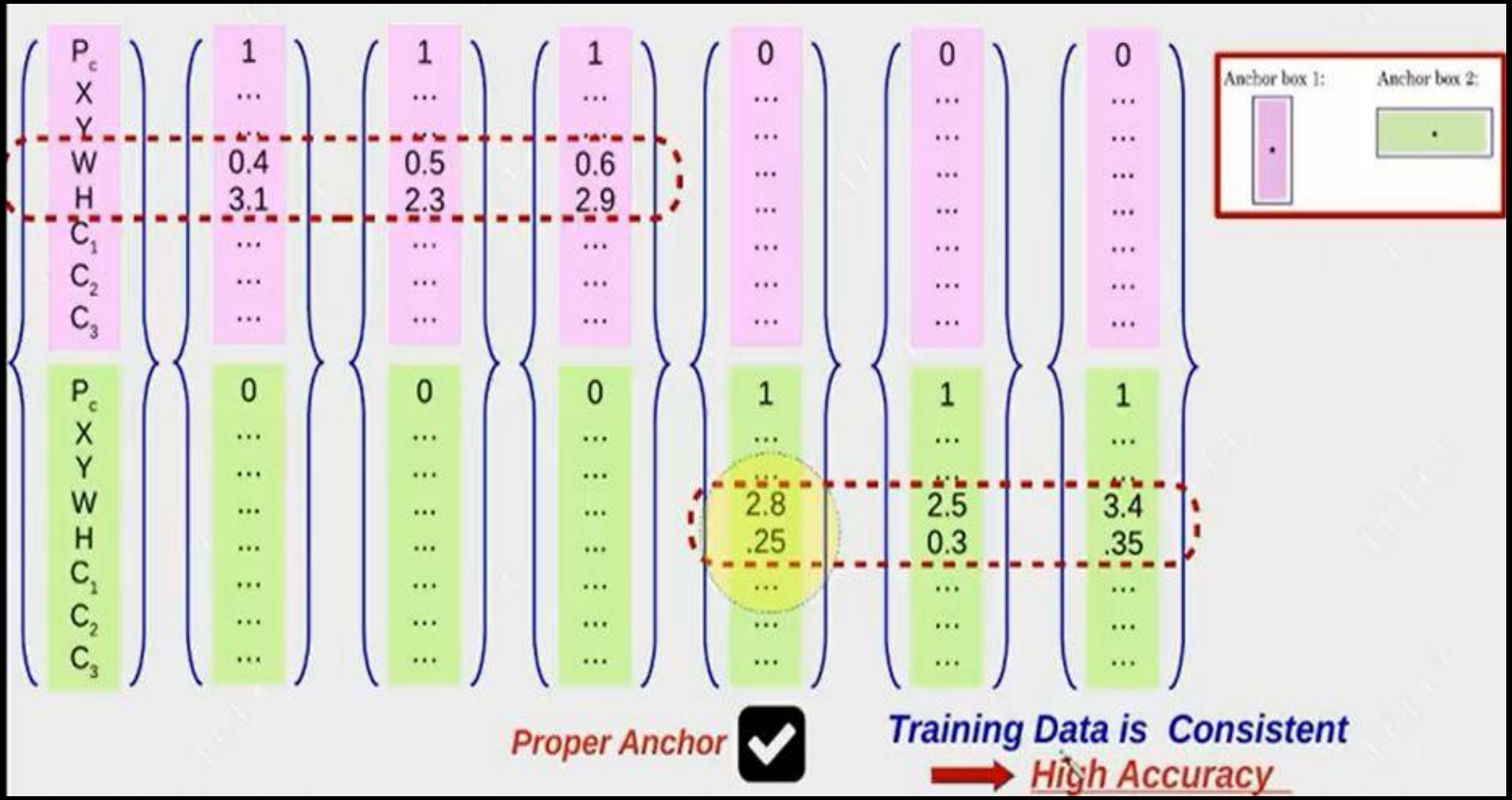
Two Anchor Boxes

2

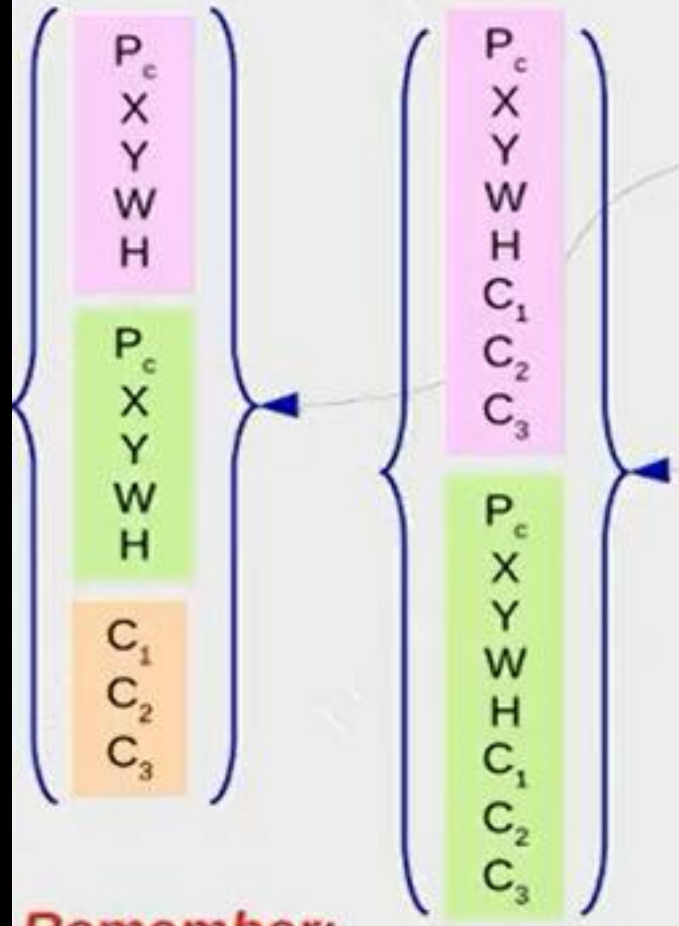
1

What if Object is Assigned to Wrong Anchor Box?




Pc
X
Y
W
H
C1
C2
C3
1
0.4
3.1
1
0.5
2.3
1
0.6
2.9
0
0
0
0
0
0
0
2.8
.25
2.5
0.3
3.4
.35
Anchor box 1:
Anchor box 2:
Proper Anchor
Training Data is Consistent
High Accuracy

Multiple Anchors, ONE Object / Cell



Remember:
ONE Vector / Cell

For SxS Cells, M Anchors , N Classes

One Object per Cell

*Total Size = [S X S] * [M * 5 + N]*

Multiple Objects per Cell

Total Size = [S X S] * M * [5 + N]

Example:
S=13 , M=6 Anchors, N=100 Class

One Object per Cell
Total Size = 13 * 13 * [6 * 5 + 100] = **21970**

Multiple Objects per Cell
Total Size = 13 * 13 * 6 * [5 + 100] = **106470**

1 : 4.84

How to Assign Anchor Box to Object

For Each Object Bounding Box

Calculate IOU_1

Calculate IOU_2

Calculate IOU_3

Calculate IOU_4

1

2

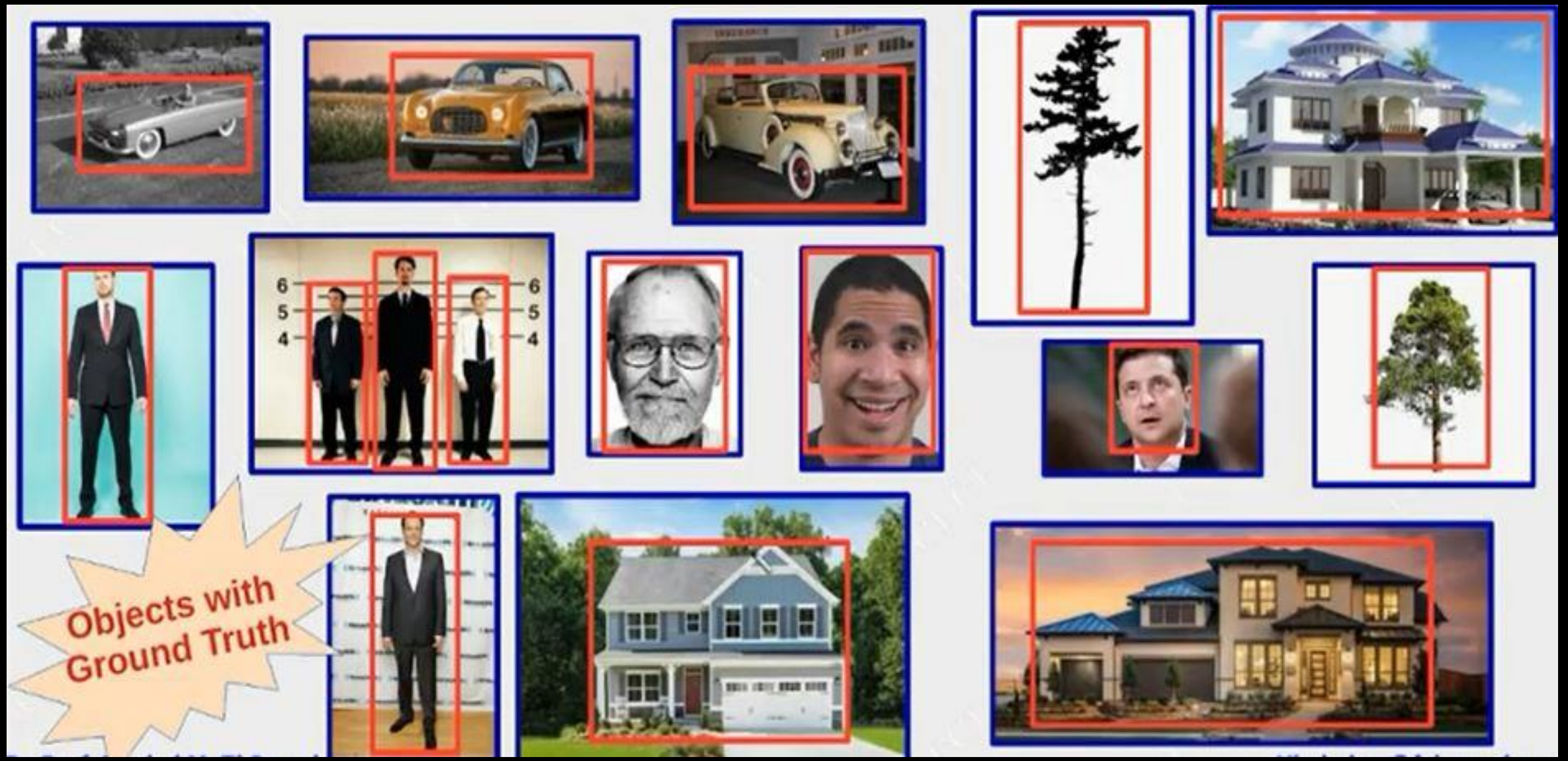
3

4

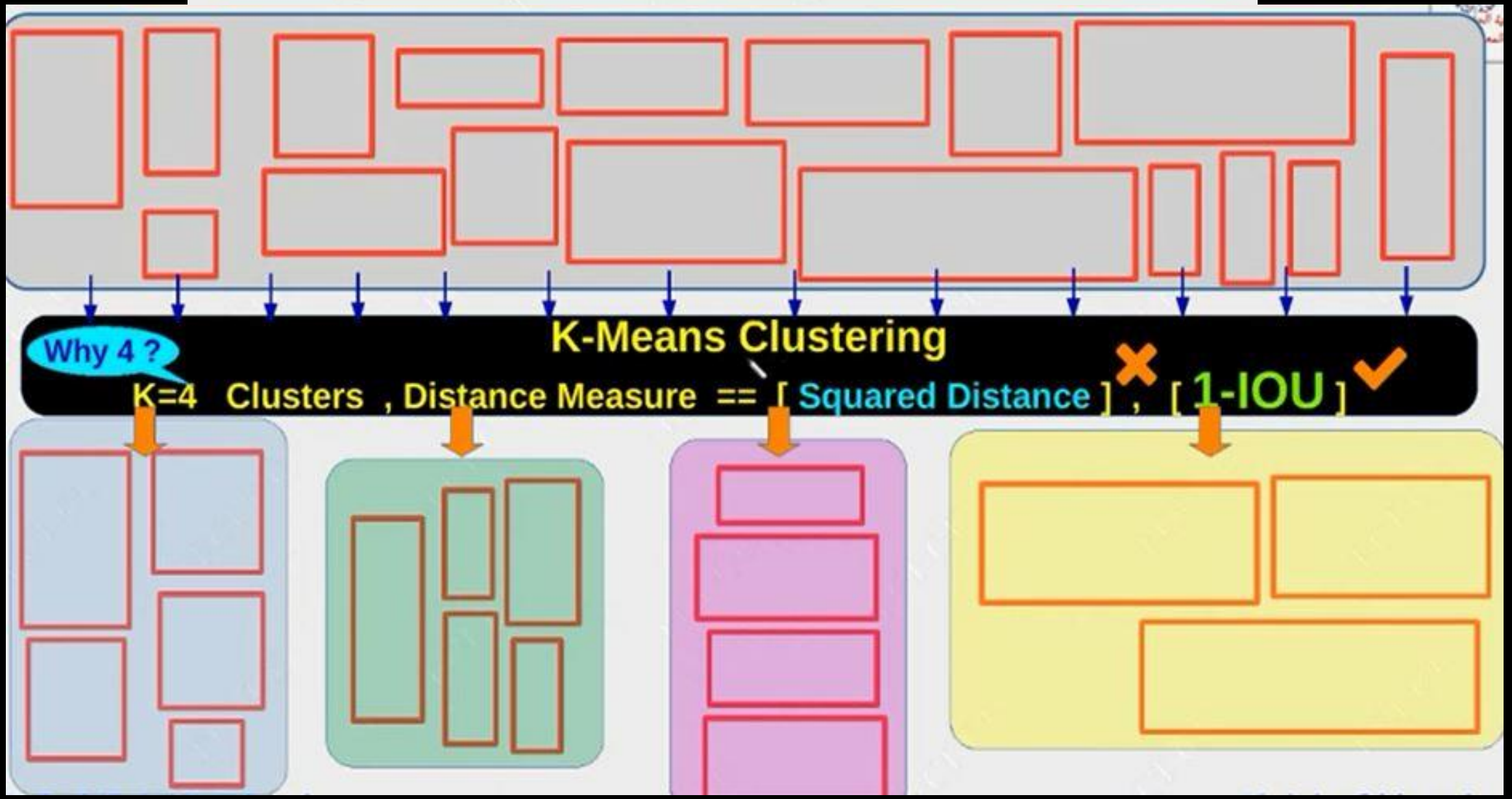
Anchor Boxaes

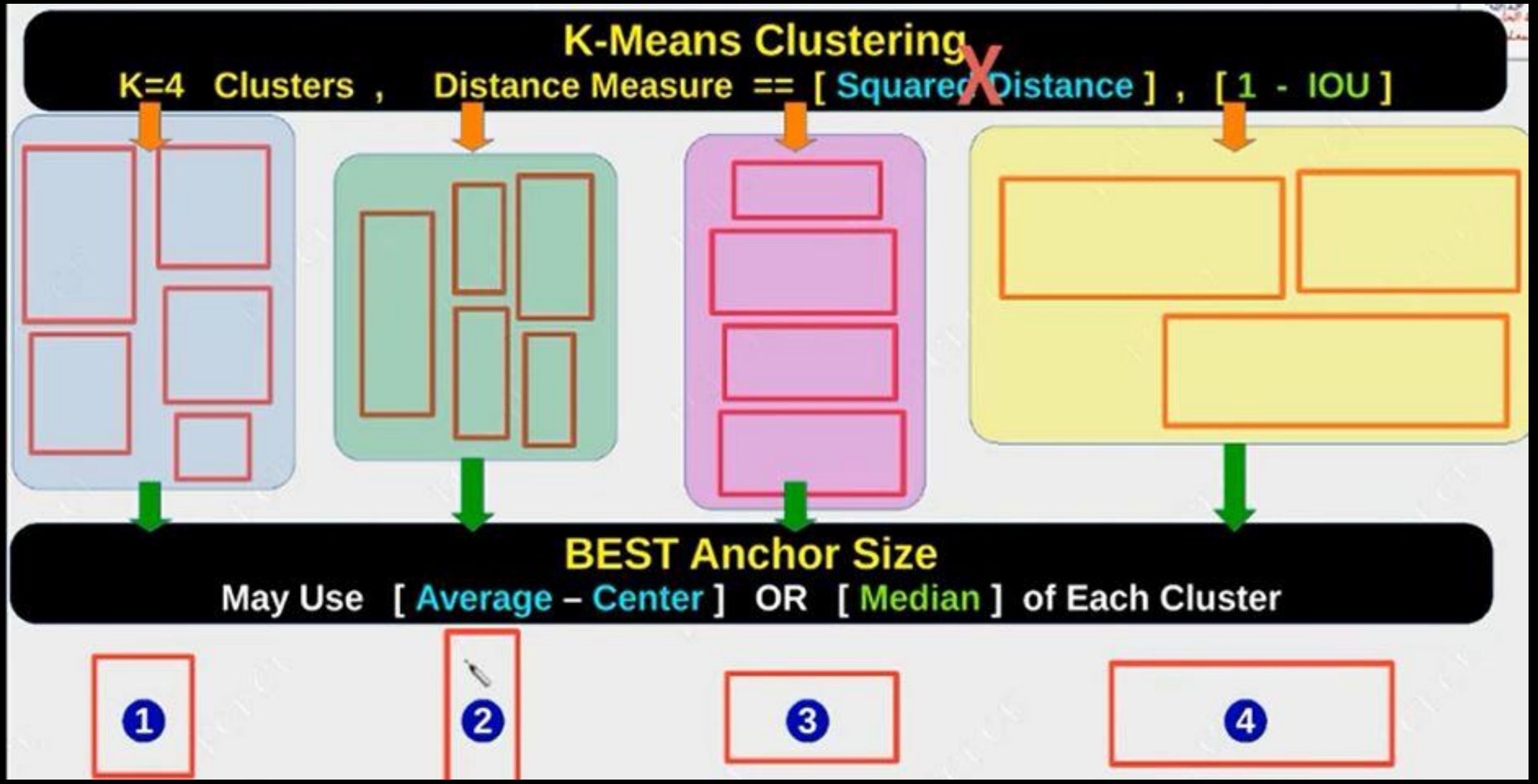
Select Anchor Box with HIGHEST IOU

How to Define Sizes of Anchor Boxes



How to Define Sizes of Anchor Boxes




K-Means Clustering
K=4 Clusters , Distance Measure == [Squared Distance] , [1 - IOU]
BEST Anchor Size
May Use [Average – Center] OR [Median] of Each Cluster
1
2
3
4

How to Define Number of Anchor Boxes

K-Means Clustering
RUN with K= 3, 4, 5, 6, 7, 8, 9, ... Clusters]

Assign Ground Truth Bounding Box of Each Object to one of the "K" Anchor boxes (based on Max IOU)

Calculate IOU between Ground Truth Bounding Box of Each Object and assigned Anchor Box, Calculate Mean IOU

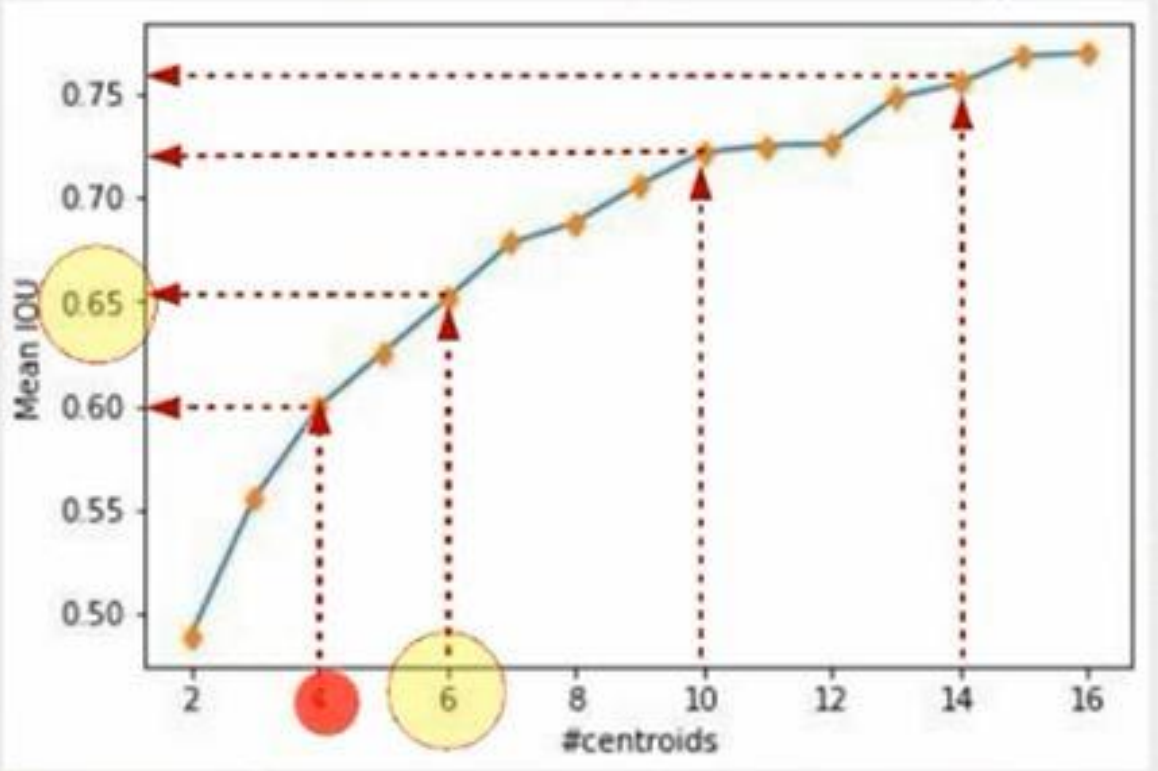
Repeat for K=2, 3, 4, ... , Obtain Mean IOU for each K

Select Minimum "K" Value that achieves "Acceptable IOU" [e.g. >0.6]

How to Define Number of Anchor Boxes

K-Means Clustering

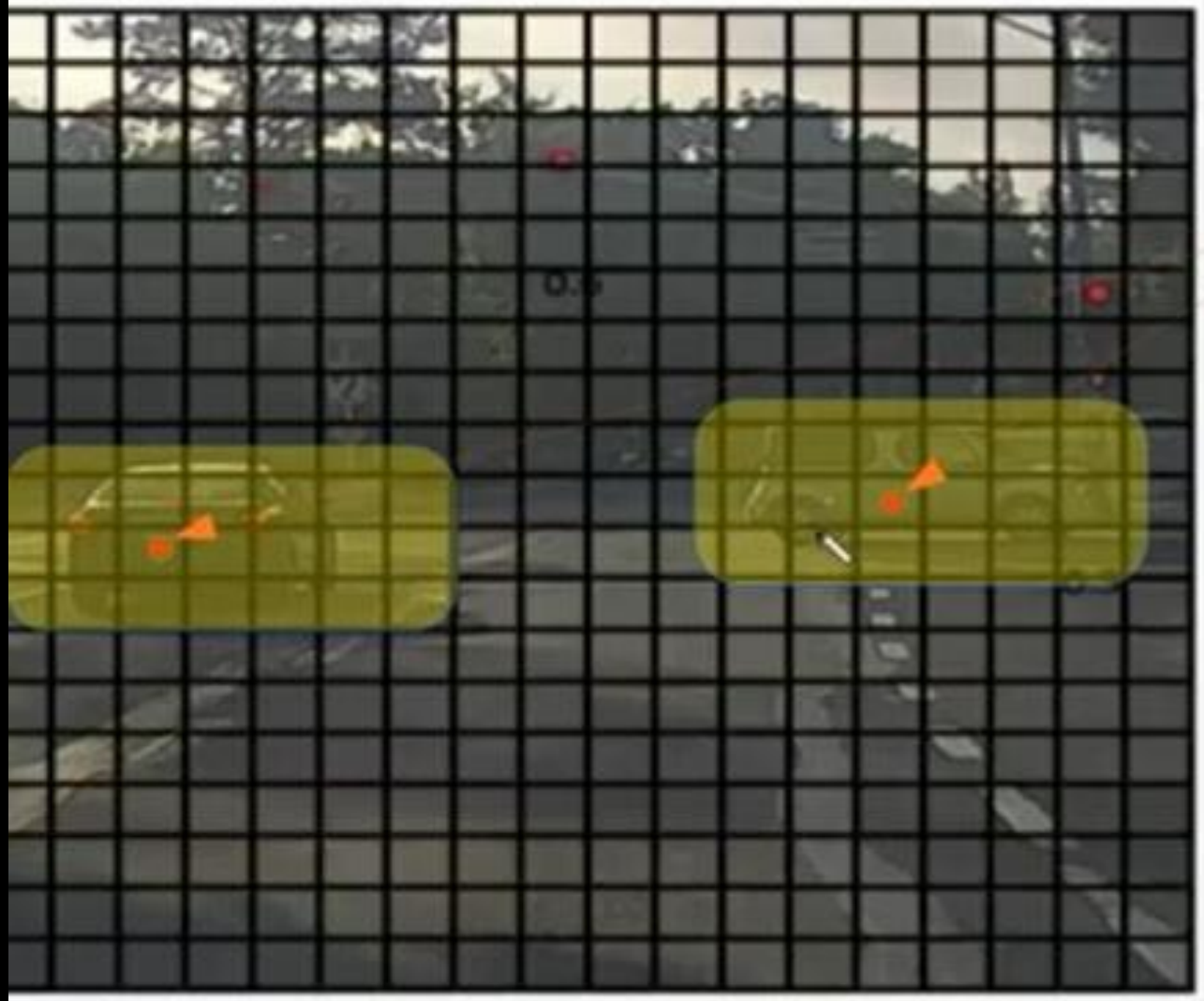
RUN with K= 3, 4, 5, 6, 7, 8, 9, ... Clusters]



Select Minimum “K” Value that achieves “Acceptable IOU” [e.g. >0.6]

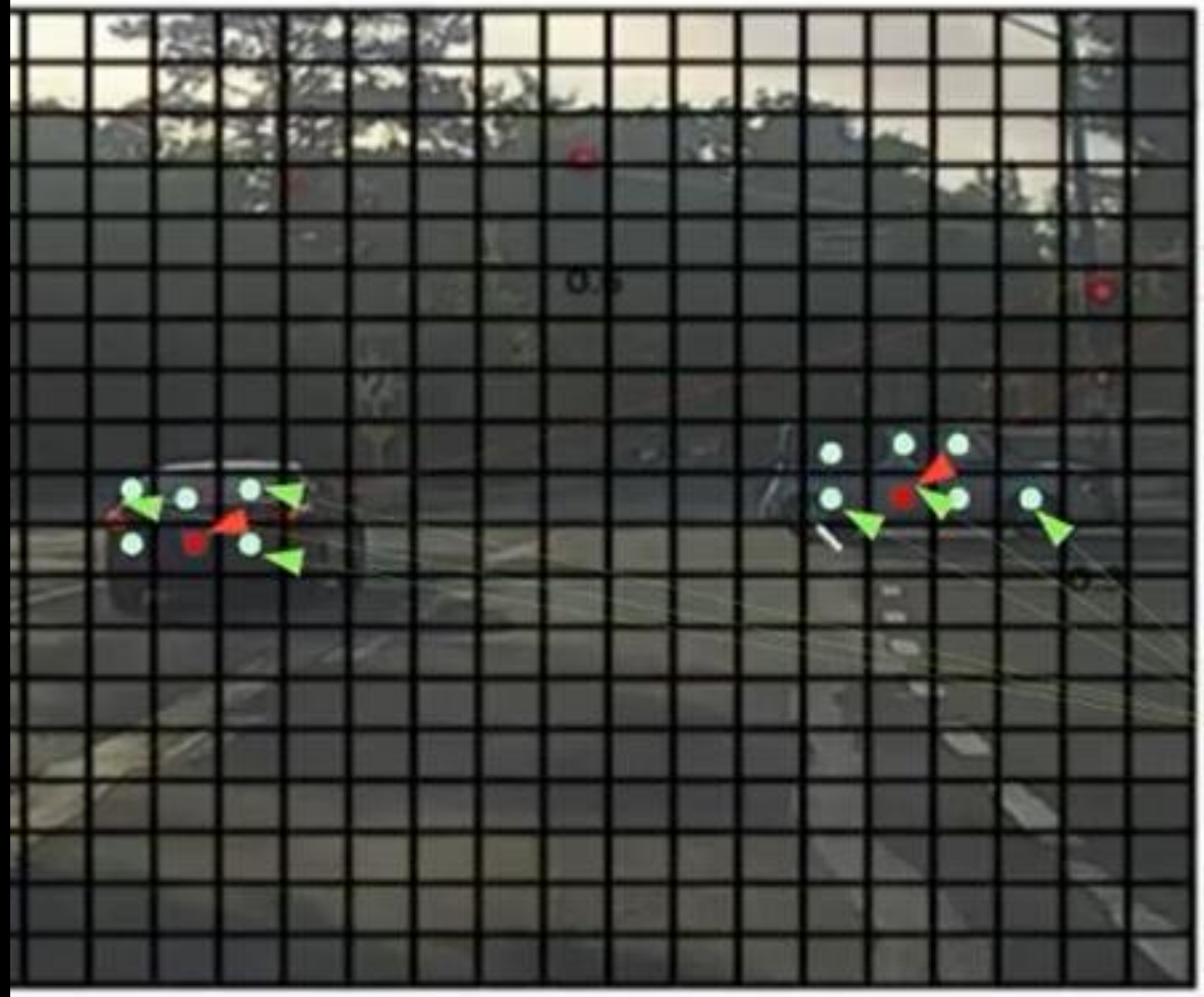
Non Maximum Supression

Non-Max Supression



Ground Truth
Objects centers and corresponding Bounding Boxes

Non-Max Supression



Ground Truth
Objects centers and corresponding Bounding Boxes
(Two Boxes ONLY)

Detection Output
Objects centers and corresponding Bounding Boxes
(Multiple Boxes for Same Object)

Non-Max Suppression Algorithm

1- Discard All Bounding Boxes with P_c <0.6
[Low probability to be an Object]

2- Select One Box with Highest P_c,
Calculate IOU between it and ALL other Boxes

3- Discard Boxes with IOU >0.5 (Highly Overlap)

4- Repeat 2,3, excluding the previously accepted predicted Box, till no Boxes to discard

Non-Max Suppression

Detection Output

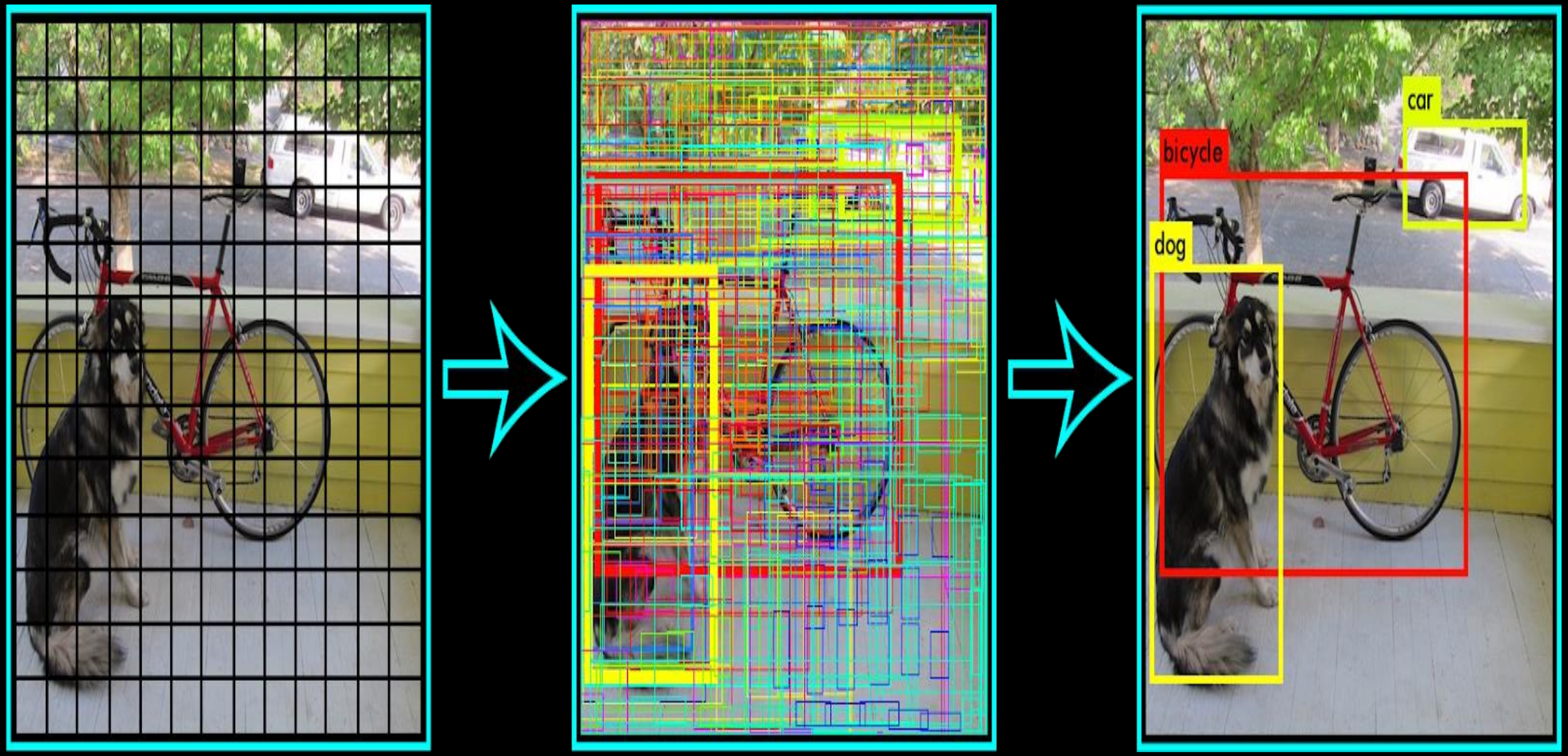
Objects centers and corresponding to

Bounding Boxes

(Multiple Boxes for Same Object)

Remember:

Each Box has its own P_c


car
bicycle
dog

In the next videos

- **Loss function**
- **mAP**
- **FPS**
- **Difference between YOLO versions**

YOLO
Loss Function

Difference between y and $\hat{y}$

$$Y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$$Y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

1. Pedestrian
2. Car
3. Motorcycle

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \quad y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$y = 3 \times 3 \times 16$

$y = 3 \times 3 \times 2 \times 8$

anchors # parameters

Original Loss Function of YOLO

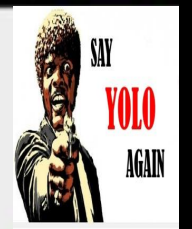
$$
\begin{aligned}
\text{LOSS} = \; & \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\
& + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\
& + \lambda_{obj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (C_i - \hat{C}_i)^2 \\
& + \lambda_{cls} \sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p(c)})^2
\end{aligned}
$$

coordinate loss

confidence loss

class loss

Enhanced Loss Function of YOLO



$$\text{loss}_{i,j} = \text{loss}_{i,j}^{xywh} + \text{loss}_{i,j}^{p} + \text{loss}_{i,j}^{c}$$

Coord Loss + Class Prob Loss + Object Confidence Loss

$$\text{loss}_{i,j}^{xywh} = \frac{\lambda_{\text{coord}}}{N_{L^{obj}}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{i,j}^{\text{obj}} \left[(x_{i,j} - \hat{x}_{i,j})^2 + (y_{i,j} - \hat{y}_{i,j})^2 + \left(\sqrt{w}_{i,j} - \sqrt{\hat{w}}_{i,j}\right)^2 + \left(\sqrt{h}_{i,j} - \sqrt{\hat{h}}_{i,j}\right)^2 \right]$$

$$\text{loss}_{i,j}^{p} = -\frac{\lambda_{\text{class}}}{N_{L^{obj}}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{i,j}^{\text{obj}} \sum_{c \in \text{class}} p_{i,j}^{c} \log(\hat{p}_{i,j}^{c})$$

$$\text{loss}_{i,j}^{c} = \frac{\lambda_{\text{obj}}}{N^{conf}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{i,j}^{\text{obj}} \left(IOU_{\text{prediuction}_{i,j}}^{\text{ground truth}_{i,j}} - \hat{C}_{i,j} \right)^2 + \frac{\lambda_{\text{noobj}}}{N^{conf}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{i,j}^{\text{noobj}} \left(0 - \hat{C}_{i,j} \right)$$

Where:

- **B:** Number of Anchor Boxes
- **S^2:** Number of Cells /Image = (13 x 13)
- **i:** Current Cell Number
- **j:** Current Anchor Number
- **x,y,w,h:** for Bonding Box (Ground Truth)
- **x^,y^,w^,h^:** of Predicted Box

- **$C_{i,j}$** : = Confidence (there exist an object cell # i, Anchor # j = 1 or 0 [Ground Truth]
- **$C_{i,j}$^** := Predicted Confidence (there exist an object @ cell # i, Anchor # j , values between {0 -->1}
- **$L_{i,j}^{obj}$** : = 1 if $C_{i,j}$ = 1 , 0 ~~ Apply Eq. If there exist an object

Enhanced Loss Function of YOLO

Coordinates Loss

*Instead of predicting the **width** and **height** directly, Yolo **predicts the square roots** to account for deviations in small numbers being more significant than the same deviation in big numbers.*
The square root mapping is used to expand smaller numbers,
e.g. any number in [0,0.25] gets mapped to [0,0.5].

Weight of Coordinates Loss w.r.t. Other Losses

Centers Coordinates for Ground Truth and Predicted Bonding Boxes

Width and Height for Ground Truth and Predicted Bonding Boxes

$$\text{loss}_{i,j}^{xywh} = \frac{\lambda_{\text{coord}}}{N_{I^{\text{obj}}}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{i,j}^{\text{obj}} \left[\left(x_{i,j} - \hat{x}_{i,j}\right)^2 + \left(y_{i,j} - \hat{y}_{i,j}\right)^2 + \left(\sqrt{w}_{i,j} - \sqrt{\hat{w}}_{i,j}\right)^2 + \left(\sqrt{h}_{i,j} - \sqrt{\hat{h}}_{i,j}\right)^2 \right]$$

Normalization Term

Object Exists

Squared Distance for Centers Coordinates

Squared Distance for Widths and Heights

Enhanced Loss Function of YOLO

$$\text{loss}^{p}_{i,j} = -\frac{\lambda_{\text{class}}}{N_{L^{obj}}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L^{\text{obj}}_{i,j} \sum_{c \in \text{class}} p^{c}_{i,j} \log(\hat{p}^{c}_{i,j})$$

Class Prob.

$$\text{loss}^{c}_{i,j} = \frac{\lambda_{\text{obj}}}{N^{conf}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L^{\text{obj}}_{i,j} \left(IOU^{\text{ground truth}_{i,j}}_{\text{preduiction}_{i,j}} - \hat{C}_{i,j} \right)^2 + \frac{\lambda_{\text{noobj}}}{N^{conf}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L^{\text{noobj}}_{i,j} \left(0 - \hat{C}_{i,j} \right)$$

Object Confidence

The scales of the $C_{i,j}$ and $p^{c}_{i,j}$ are different from $\hat{C}_{i,j}$ and $\widehat{p^{c}}_{i,j}$, as the their ground truths take 0/1 values

	Ground Truth	Predicted
$C_{i,j}$	1 (object exists) or 0 (No Object)	between 0 and 1
$p^{c}_{i,j}$	1 (C^{th} class Object exists) or 0 (C^{th} class Object NOT exists)	between 0 and 1

Prefer NOT to use Squared Error Distance, Use Another metric

Enhanced Loss Function of YOLO
Class Probability Loss

For Objects Classes Prediction, Use Cross Entropy Loss

$$CrossEntropyLoss = -(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i))$$

If y_i =1 this term will take effect

If y_i =0 this term will take effect

Class Prob. Loss Weight

$$\text{loss}_{i,j}^{p} = -\frac{\lambda_{\text{class}}}{N_{L^{obj}}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{i,j}^{\text{obj}} \sum_{c \in \text{class}} p_{i,j}^{c} \log(\hat{p}_{i,j}^{c})$$

Normalization Term

If there is **NO Object** in this Anchor Box **Don't Consider** this LOSS

Since C belongs to Class $p_{i,j}^{c}$ =1

Use $-(y_i log(\hat{y}_i)$ ONLY

Enhanced Loss Function of YOLO
Object Confidence Loss

Learn the model to Predict Confidence Value based on IOU between Ground Truth and Predicted Boxes (in case of Object Exists) And Predict "0" if NO Object Exists

Different Confidence Loss Weights For Object Exists and No Object

Actually, there is an Object in this {Cell, Anchor} Pairs

Actually, there is No Object in this {Cell, Anchor} Pairs

$$\text{loss}_{i,j}^{c} = \frac{\lambda_{\text{obj}}}{N^{conf}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{i,j}^{\text{obj}} \left(IOU_{\substack{\text{Ground Truth}_{ij} \\ \text{Prediction}_{ij}}} - \hat{C}_{i,j} \right)^2 + \frac{\lambda_{\text{noobj}}}{N^{conf}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{i,j}^{\text{noobj}} \left(0 - \hat{C}_{i,j} \right)$$

Object Exist, Confidence calculated based on IOU

No Object, Confidence =0

Normalization Term

Enhanced Loss Function of YOLO

Weights, Normalization , Functions Activation's

$$N_{L^{obj}} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{i,j}^{\text{obj}}$$

$$N^{conf} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{i,j}^{\text{obj}} + L_{i,j}^{\text{noobj}}(1 - L_{i,j}^{\text{obj}})$$

$$L_{i,j}^{\text{obj}} = \begin{cases} 1 \text{ if } C_{i,j} = 1 \\ 0 \text{ else} \end{cases}$$

$$L_{i,j}^{\text{noobj}} = \begin{cases} 1 \text{ if } \max_{i',j'} IOU_{\text{preduiction}_{i,j}}^{\text{ground truth}_{i',j'}} < 0.6 \text{ and } C_{i,j} = 0 \\ 0 \text{ else} \end{cases}$$

Suggested Values
(Hyper Parameters)

$\lambda_{\text{coord}} = 5$

$\lambda_{\text{obj}} = 1$

$\lambda_{\text{noobj}} = 0.5$

$\lambda_{\text{class}} = 1$

In the next video we will talk about mAP

YOLO

Basic Terminologies

Part Four Mean Average Precision (mAP)

TP, FP, TN, FN


True Positive
Prediction: Hot Dog
Prediction: Hot Dog
False Positive
False Negative
Prediction: Not Hot Dog
Prediction: Not Hot Dog
True Negative

TP, FP, TN,FN

True Positives

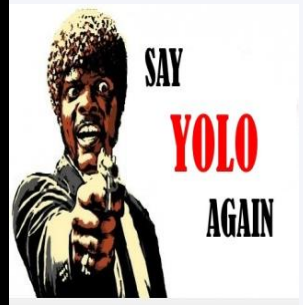
These are **predicted bounding boxes** which have an **IOU >0.5** with the **target bounding box** for a given class.

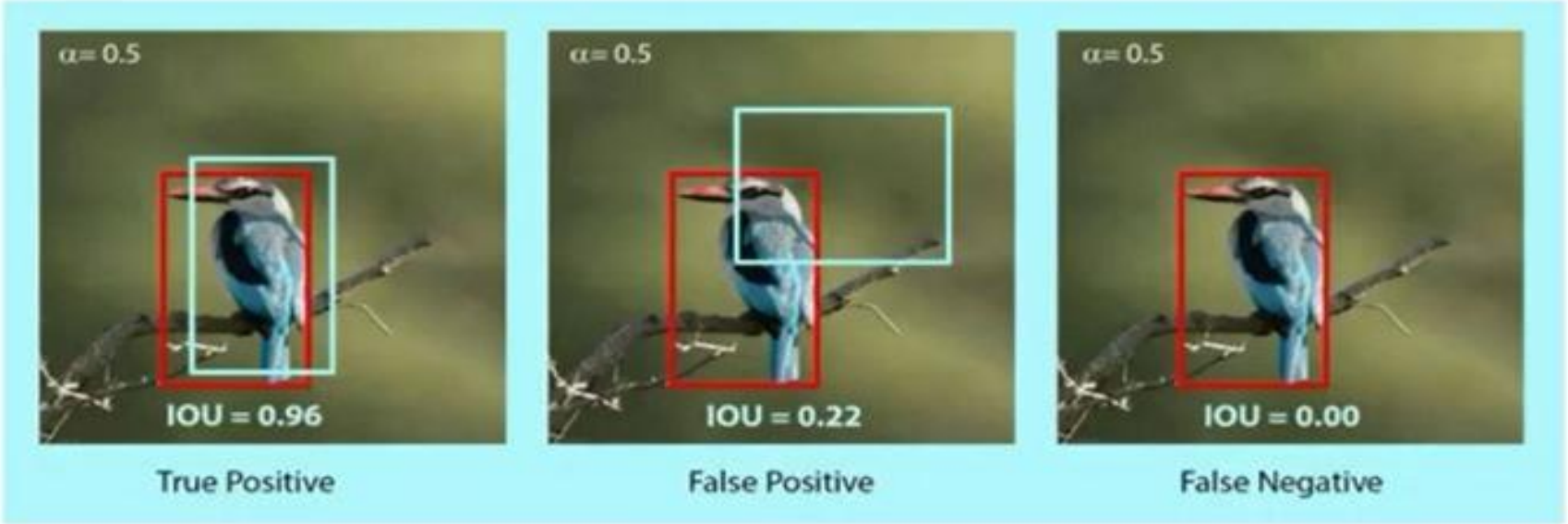
False Negatives

There are **no predicted bounding boxes** for a **target bounding box**.

False Positves

These are **predicted bounding boxes** which have an **IOU < 0.5** with the **target bounding box** for a given class.





Note that **True Negative** is not applicable to object detection and segmentation.

It is **correctly detecting the background of an object as background**.

In other words, it means we didn't output a bounding box that wasn't there.

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**.

$$Precision = \frac{TP}{\text{total positive results}}$$

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

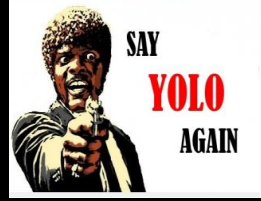
Recall

Recall measures the proportion of **actual positives** that were **predicted correctly**.

It is the **True Positives** out of all **Ground Truths**.

Recall = TP / Total Ground Truth boxes

$$Recall = \frac{TP}{TP + FN}$$



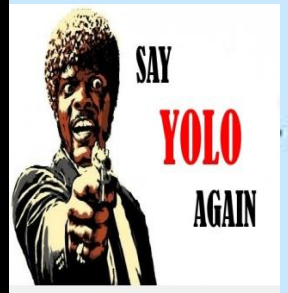
Average Precision

AP (Average precision) is a popular metric in measuring the accuracy of object detectors like YOLO, Faster R-CNN, SSD, etc

Average precision computes the **average precision value** for **recall value** over **0** to **1**.

Average Precision (**AP**) is not the **average of Precision** (**P**).

For simplicity, we can say that it is the **area under the precision-recall curve**.



Average Precision Use Case

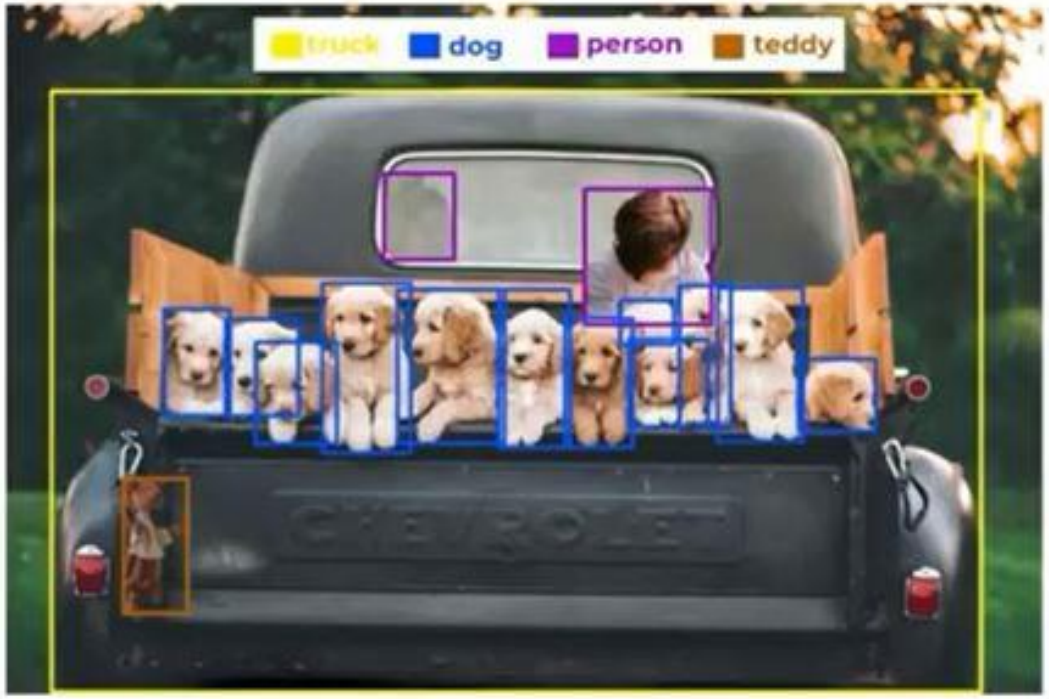
Let us consider the following image that has various classes.

The **YOLOv5 nano** model is predicting bounding boxes on the objects.

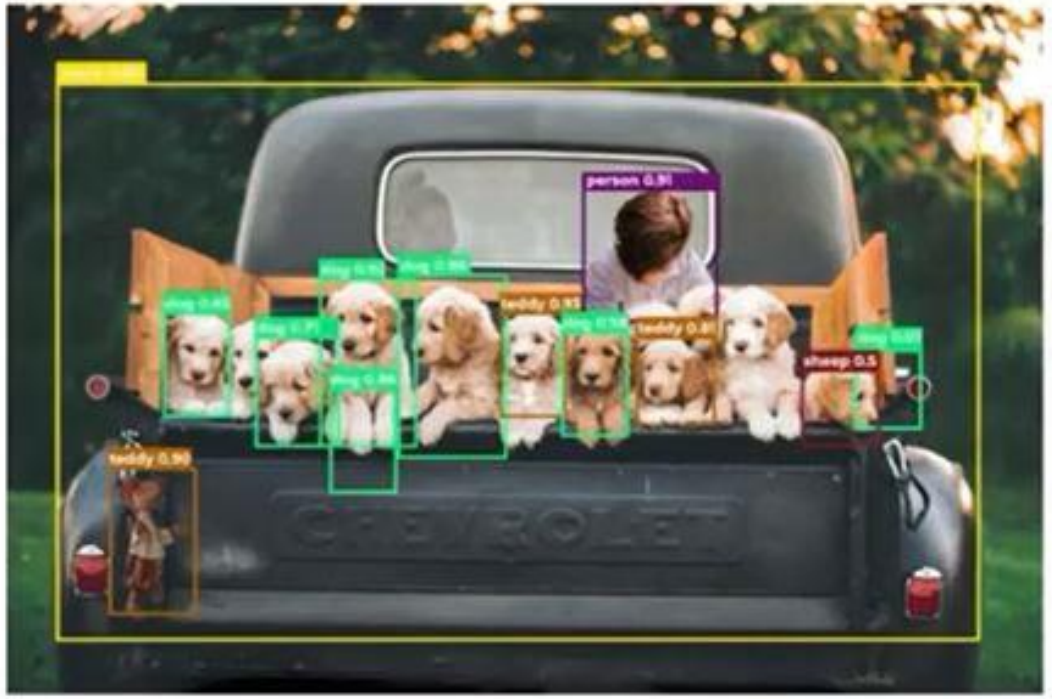
IoU threshold is set at **0.5**.

The **ground truths** are already known.

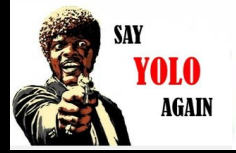
Average Precision Use Case



Ground truths



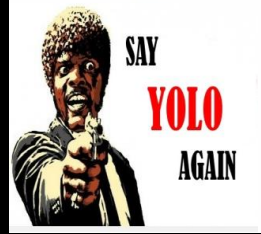
Predictions by YOLOv5 Nano



We can see that the images have the following objects (Ground Truths).

- 2 person
- 12 dog
- 1 teddy
- 1 truck

The YOLOv5 nano model has predicted the following objects.

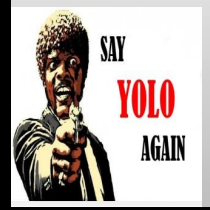


Average Precision Use Case

Average Precision (**AP**) is calculated **class-wise**.

Now, let's get started with calculating **AP** of **dog class**.

The steps to be followed are as follows.

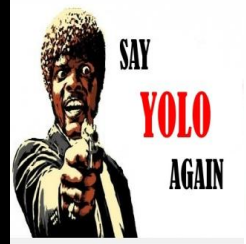


Step 1 - Record every Dog detection along with the Confidence score

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

TP - If IoU of **Predicted box** matches **GT** is **> IoU threshold**

FP - If IoU of **Predicted box** matches **GT** is **< IoU threshold**



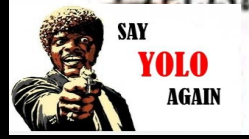
Step 2 - Calculate Precision and Recall

Follow the steps below to tabulate the data.

1. Sort the table in descending order of confidence.
2. Tabulate cumulative TP and FP (Keep on adding the current value with the previous row).
3. Calculate row-wise Precision and Recall.

Preds.	Conf.	Matches	Cumulative TP	Cumulative FP	Precision	Recall
	0.92	TP	1	0	1/(1+0) = 1	1/16 = 0.08
	0.91	FP	1	1	1/(1+1) = 0.5	1/16 = 0.08
	0.88	TP	2	1	2/(2+3) = 0.66	2/16 = 0.16
	0.86	FP	2	2	0.5	0.16
	0.77	TP	3	2	0.6	0.25
	0.63	TP	4	2	0.66	0.33
	0.58	TP	5	2	0.71	0.41

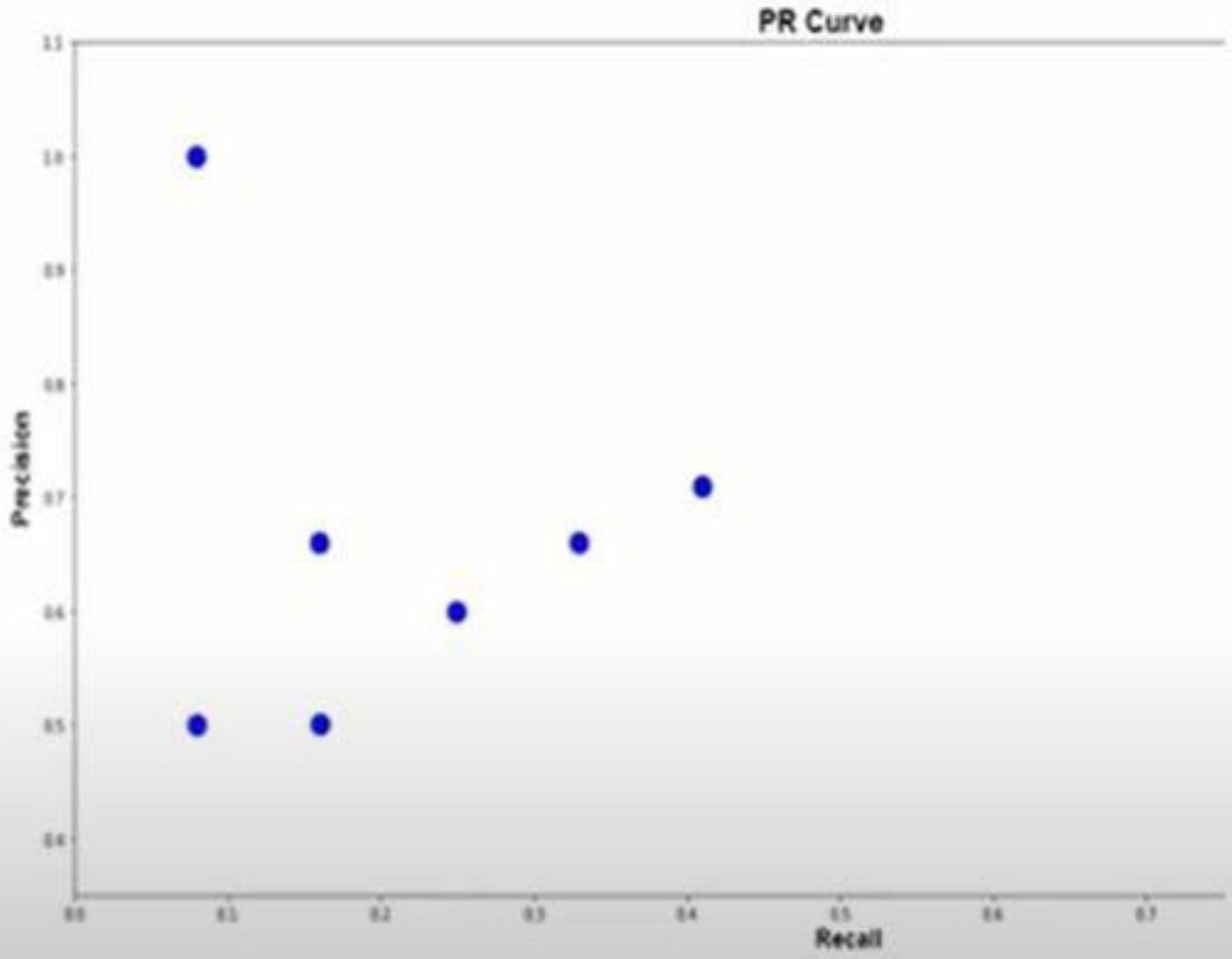
$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$



Step 3 - Plot Precision-Recall graph

Note : If the table contains **multiple precision values** for the **same recall values**, you can consider the **highest value** and **discard the rest.**

Not doing so will **NOT** affect **the final result**. This is for **simplifying the plot**. In our case, we are **plotting everything**.



Step 4 - Calculate AP using 11 Point Interpolation Method

The 11 point interpolation method was introduced in the **2007 PASCAL VOC challenge**.

Precision values are recorded across 11 equally spaced Recall values.

A prediction is positive if **IoU ≥ 0.5**.

$$AP = \frac{1}{11} \sum_{r \in \{0.0, \ldots, 1.0\}} AP_r$$

$$= \frac{1}{11} \sum_{r \in \{0.0, \ldots, 1.0\}} p_{interp}(r)$$

where

$$p_{interp}(r) = \max_{\tilde{r} \geq r} p(\tilde{r})$$

Step 4 - Calculate AP using 11 Point Interpolation Method

- First, we divide the recall value from **0** to **1.0** into **11 points** — **0, 0.1, 0.2, ..., 0.9 and 1.0**.

- Next, we compute the average of maximum precision value for these 11 recall values.

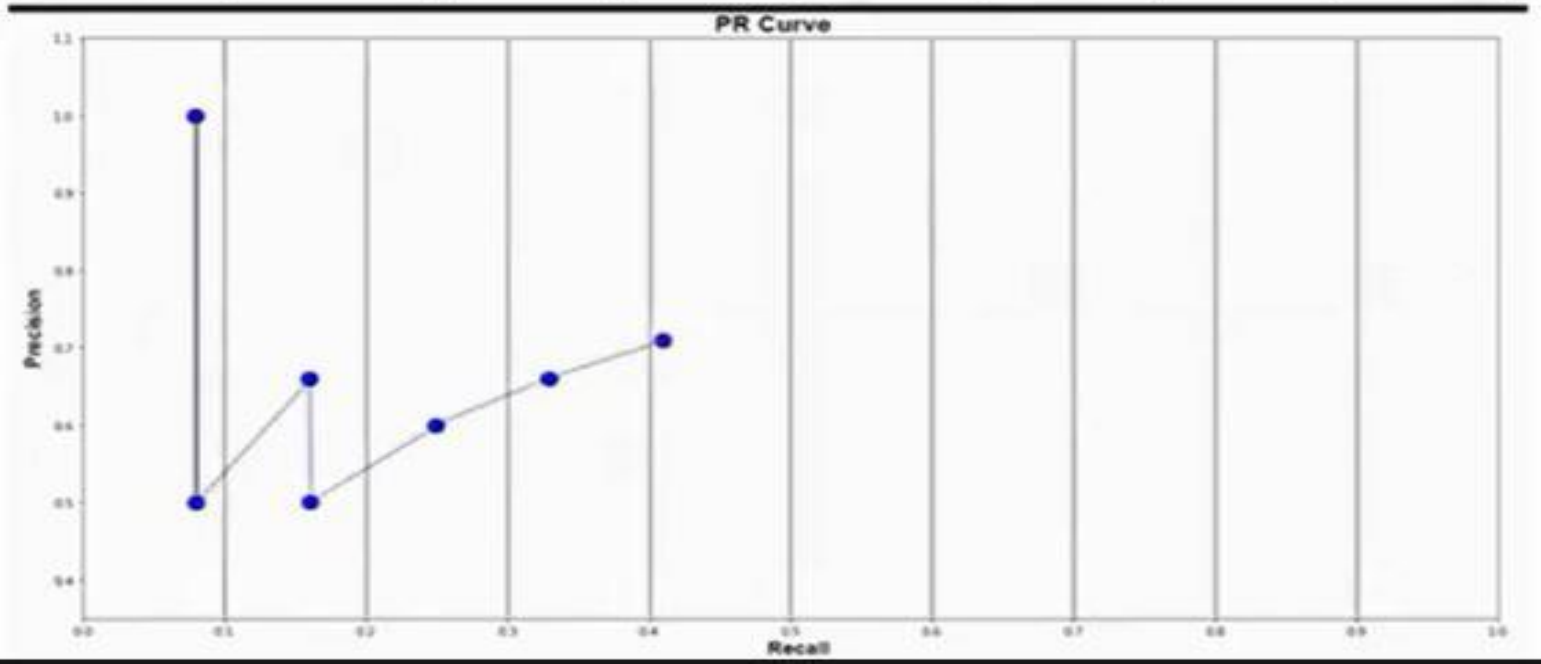
$$\textbf{AP} = 1/11 * \textbf{Sum}(\textbf{11 point interpolated precision})$$

- The **interpolated Precision** is the **maximum Precision** corresponding to the **Recall value greater than the current Recall value**.

- In simple terms, it is the **maximum precision value to the right**.

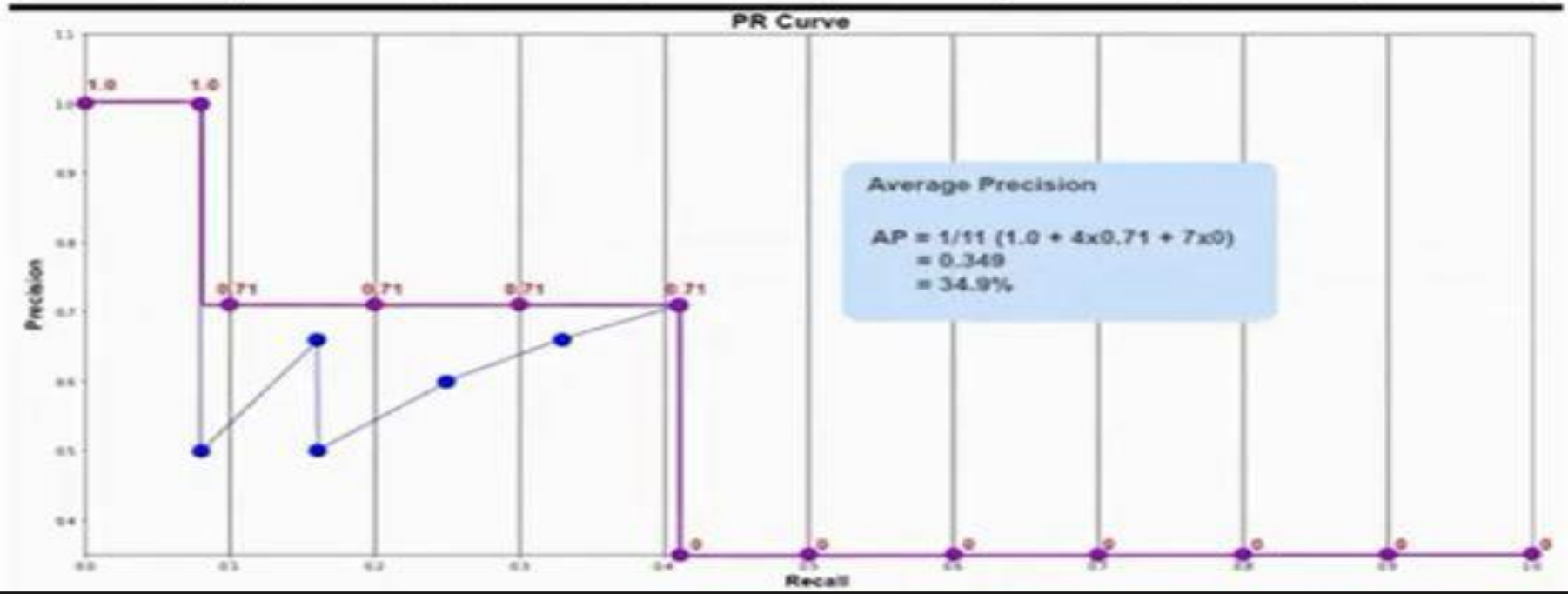
Step 5 - Plot Final Interpolated graph and calculate Average Precision for the Dog Class

Detections							
Precision	1	0.5	0.66	0.5	0.6	0.66	0.71
Recall	0.08	0.08	0.16	0.16	0.25	0.33	0.41

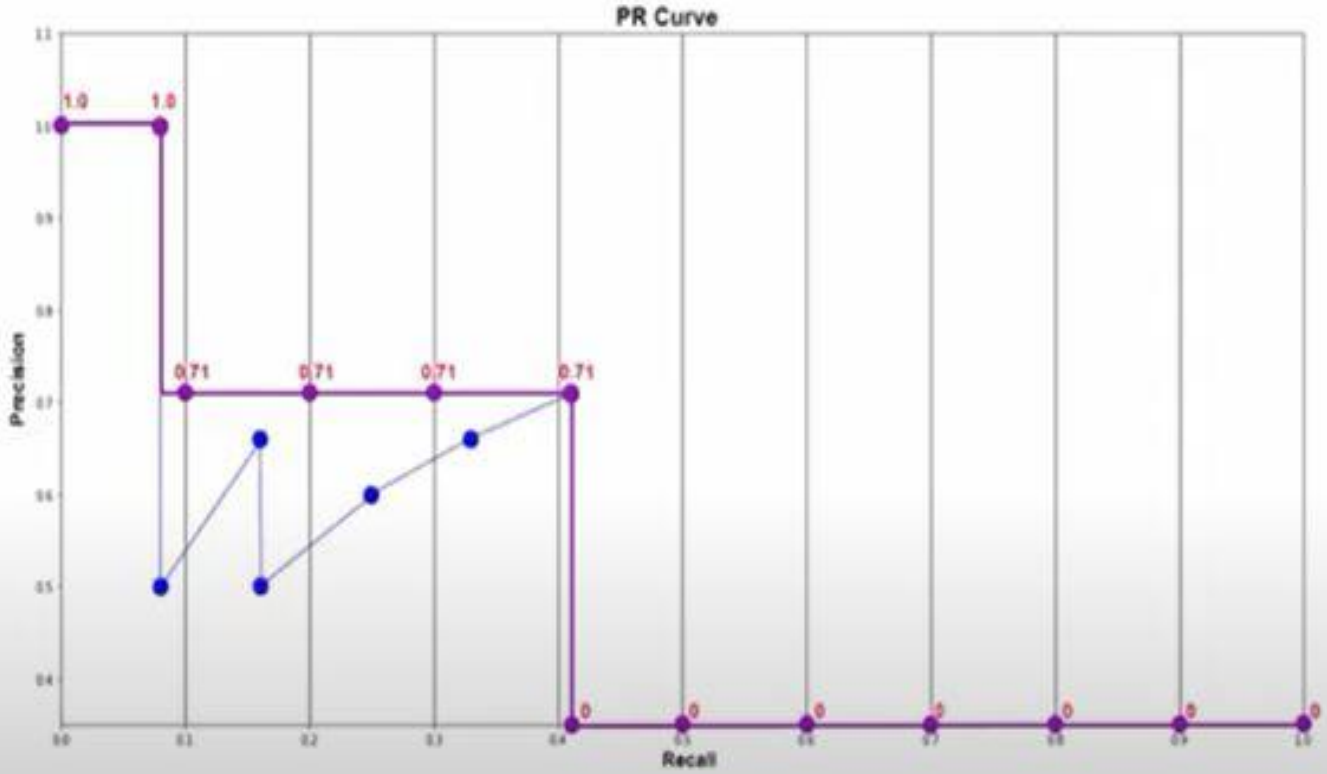


Step 5 - Plot Final Interpolated graph and calculate Average Precision for the Dog Class

Detections							
Precision	1	0.5	0.66	0.5	0.6	0.66	0.71
Recall	0.08	0.08	0.16	0.16	0.25	0.33	0.41



Step 5 - Plot Final Interpolated graph and calculate Average Precision for the Dog Class



APdog = 1/11 * (Sum of 11 interpolated Precision values)

= 1/11 * (1 + 4*0.71 + 6*0)

= 0.349

= 34.9%

Why Precision is interpolated only for 11 Recall points?

Quoting from the paper, "**The intention in interpolating the precision/recall curve in this way is to reduce the impact of the "wiggles" in the precision/recall curve, caused by small variations in the ranking of examples.**"

In reality, the evaluation dataset is **huge**. When we plot the graph for all predictions, the **difference between adjacent points** will be **very very less**. Hence, to compare two models, 11 point interpolation is sufficient.

Similarly, we calculate **Average Precision**(**AP**) for **person, teddy, sheep, and truck**. Make sure to calculate **AP** for **at least two classes** below for clear understanding. Once done, we will proceed with calculating **Mean Average Precision** (**mAP**).

CLASS	dog	person	sheep	truck	teddy
AP	0.349	0.545	0.00	1.00	0.50

Mean Average Precision

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

mAP = 1/n * sum(AP), where n is the number of classes.

CLASS	dog	person	sheep	truck	teddy
AP	0.349	0.545	0.00	1.00	0.50

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

$$mAP = 1/5 * (0.349 + 0.545 + 0 + 1 + 0.5)$$

$$= 0.4788$$

$$= 47.88\ \%$$

To arrive at the mAP, while evaluating a model; Average Precision (AP) is calculated for each class separately.

Why Average Precision (AP) is it not calculated for all the classes at once?