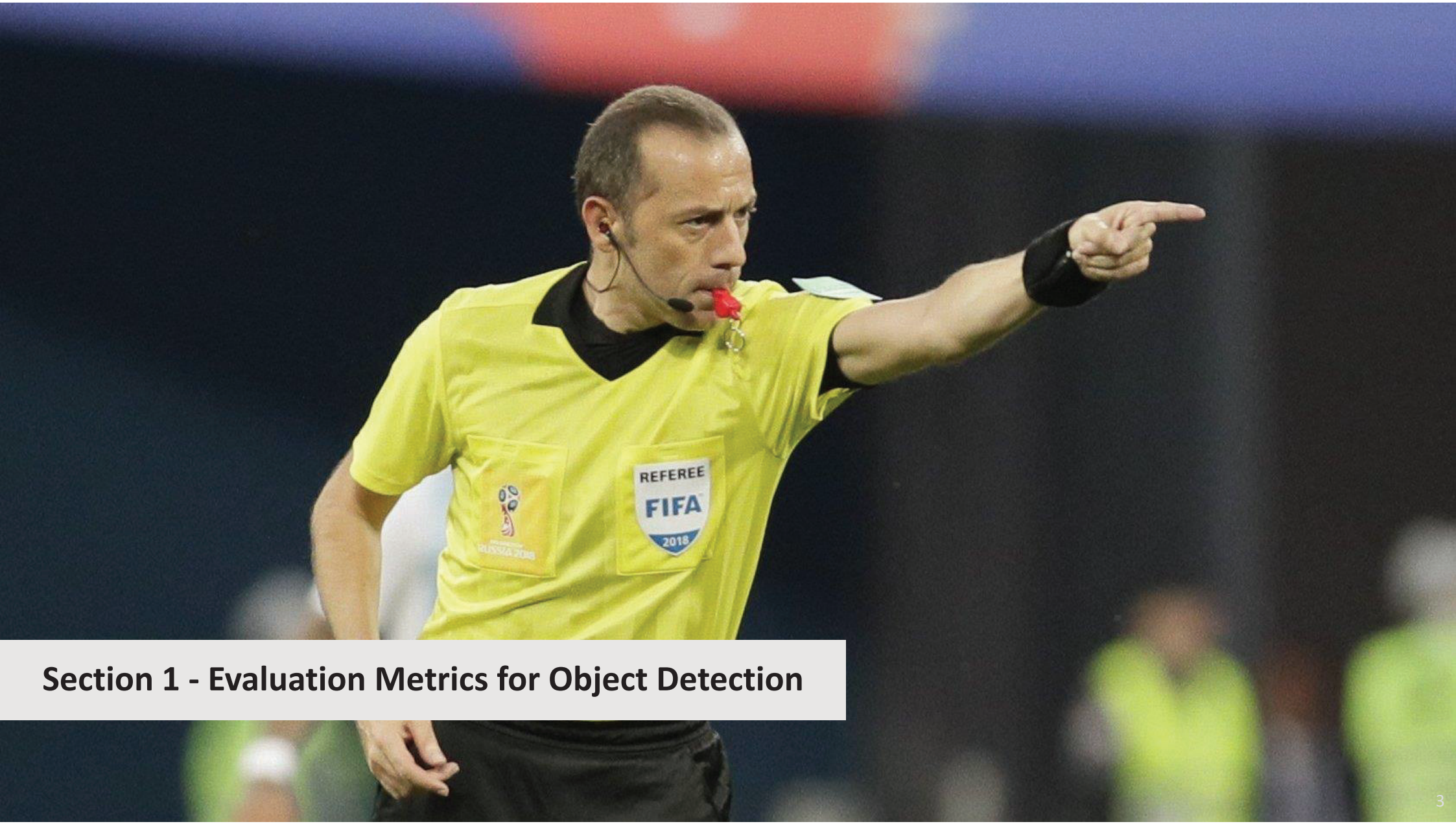
Review

- R-CNN
  - Region proposal from the source image
  - CNN identification
- Fast R-CNN
  - Region proposal from the eventual feature map layer
  - CNN identification
- Faster R-CNN
  - Region Proposal Network (RPN) to generate region proposal (also from feature map layer)
  - CNN identification

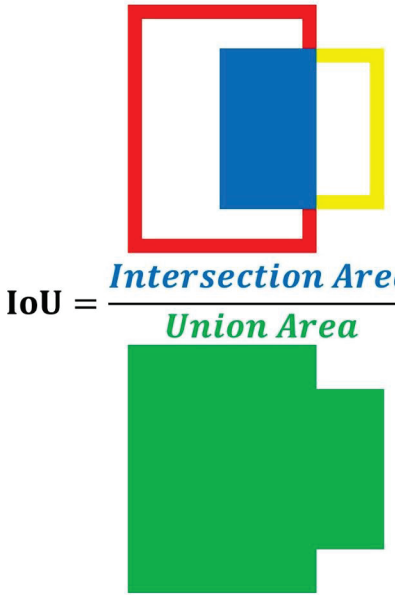


Section 1 - Evaluation Metrics for Object Detection

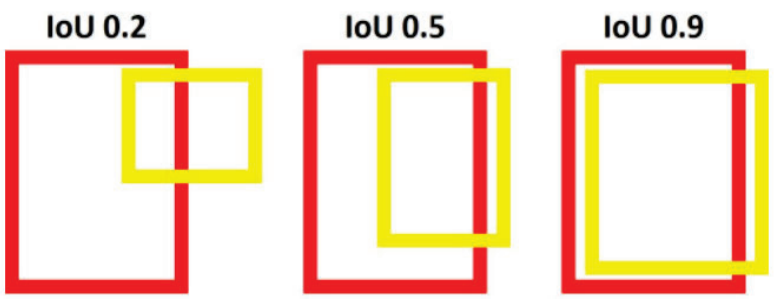
Evaluation Metrics

- Like AUC, F1, Accuracy, for classification problems, we also need evaluation metrics for object detection, to measure the goodness of fit.
  1. Intersection over Union(**IOU**)
  2. Precision and Recall
  3. Average Precision(**AP**)
  4. Mean Average Precision(**mAP**)
- Try to be clear on the difference between loss function and evaluation metrics

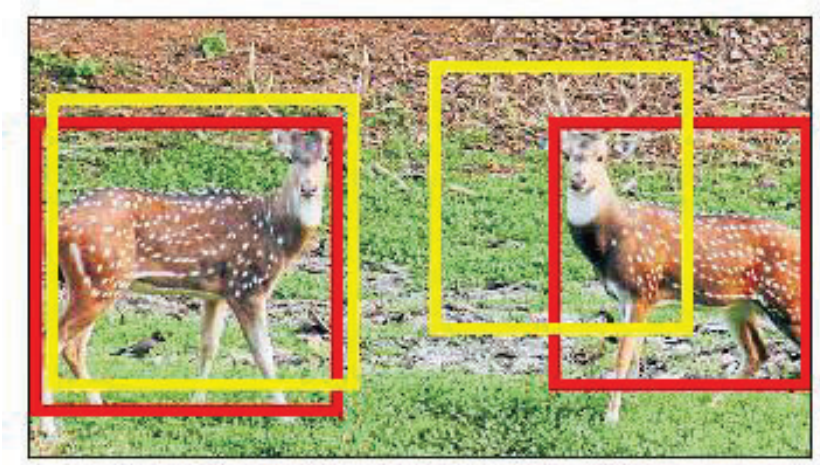
1. Intersection over Union (IoU)



$$\text{IoU} = \frac{\textit{Intersection Area}}{\textit{Union Area}}$$



1. Intersection over Union (IoU)

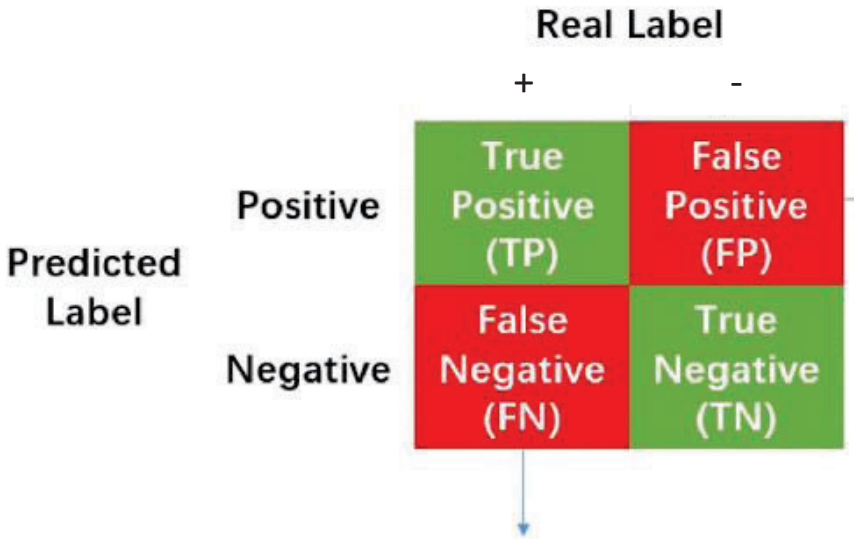


E.g., after test, we got left IoU = 0.8, right IoU = 0.2

Average IoU = 0.5

In practice, we seldom directly use average IoU for evaluation. Instead, we often use IoU as thresholds/conditions to calculate mAP.

2. Precision and Recall (Classification)



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

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

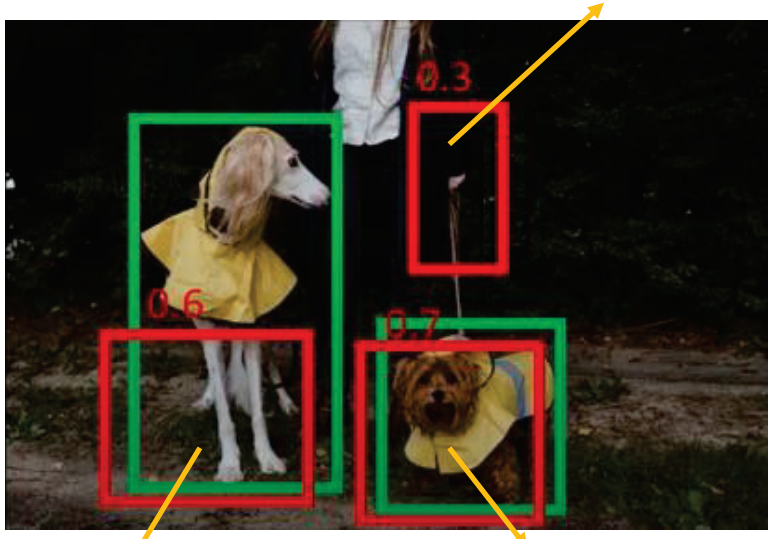
$$\text{Accuracy} = \frac{\sum \text{TP} + \text{TN}}{\sum \text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

True: Correct
False: Incorrect
Positive: positive prediction
Negative: negative prediction

$$\text{Recall} = \frac{\sum \text{TP}}{\text{Real "+" Samples}}$$

2. Precision and Recall (object detection)

		Real Label +	Real Label -
Predicted Label	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)



Green box: ground truth
Red box: model predictions (numbers are probability/confidence)

- 3 red boxes mean the model generates 3 positive predictions (negative = no object = no red box).

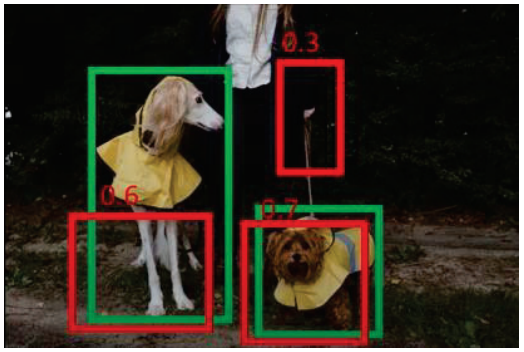
- So, **IoU threshold** is used as a threshold to judge True/False predictions
  - If IoU is large enough (> IoU threhold), we think the prediction is "close" to real "+", so it is correct/True.
  - If no, then it is "too far away", thus an incorrect prediction/False.

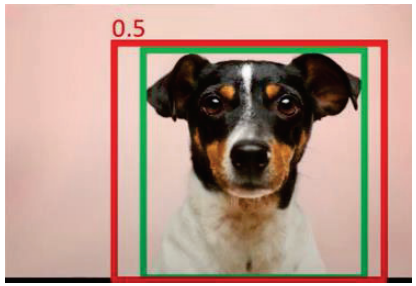
If we set IoU threshold as 0.5.
Then the 3 predictions would be
1. IoU=0, incorrect, False Positive (FP)
2. IoU=0.35, incorrect, FP
3. IoU=0.77, correct, True Positive(TP)

If we set IoU threshold as 0.3, then:
1. IoU=0, incorrect, FP
2. IoU=0.35, correct, TP
3. IoU=0.77, correct, TP

Note that all 3 red boxes are positive prediction (P)
For negative predictions, we won't output them.

2. Precision and Recall (object detection)

① 

② 

③ 

Green: ground truth
Red: model predictions (numbers are probability of being a dog), 7 red = 7 positive (P)
Assume IoU threshold = 0.5

Image	Predictions	Prob	IoU	TP/FP
①	1	0.3	0	FP
①	2	0.6	0.35	FP
①	3	0.7	0.77	TP

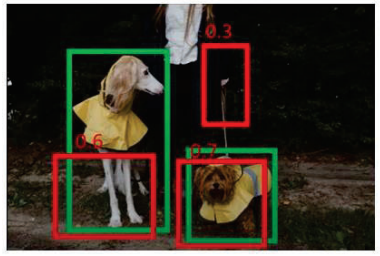
Image	Predictions	Prob	IoU	TP/FP
②	1	0.5	0.8	TP

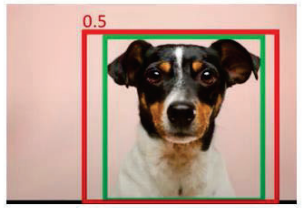
True: Correct prediction
False: Incorrect

Image	Predictions	Prob	IoU	TP/FP
③	1	0.2	0	FP
③	2	0.8	0	FP
③	3	0.9	0.7	TP

*Note: don't confuse probability threshold and IoU threshold

2. Precision and Recall (object detection)

① 

② 

③ 

Assume we set IoU=0.5

Image	Predictions	Prob	IoU	TP/FP
①	1	0.3	0	FP
①	2	0.6	0.35	FP
①	3	0.7	0.77	TP
②	1	0.5	0.8	TP
③	1	0.2	0	FP
③	2	0.8	0	FP
③	3	0.9	0.7	TP

Order by prob →

Image	Predictions	Prob	IoU	TP/FP
③	3	0.9	0.7	TP
③	2	0.8	0	FP
①	3	0.7	0.77	TP
①	2	0.6	0.35	FP
②	1	0.5	0.8	TP
①	1	0.3	0	FP
③	1	0.2	0	FP

→

precision	recall
1/1	1/4
1/2	1/4
2/3	2/4
2/4	2/4
3/5	3/4
3/6	3/4
3/7	3/4

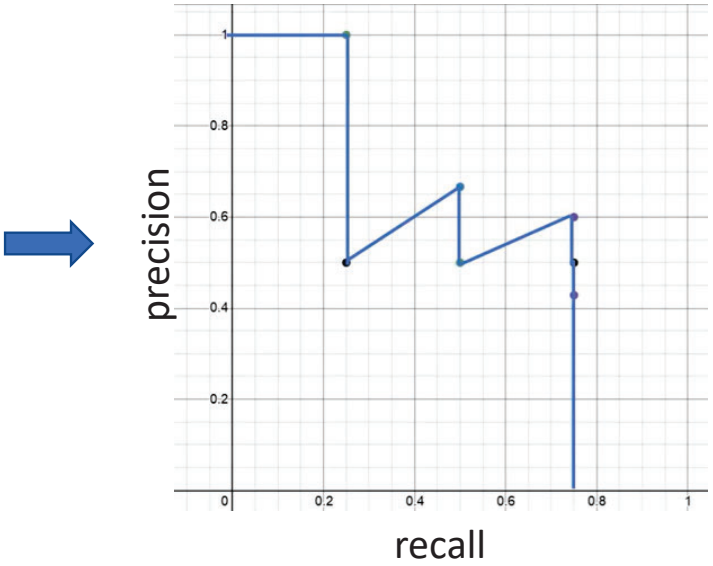
$$\text{Precision} = \frac{\sum \text{TP}}{\sum \text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\sum \text{TP}}{\text{Real "+" Samples}}$$

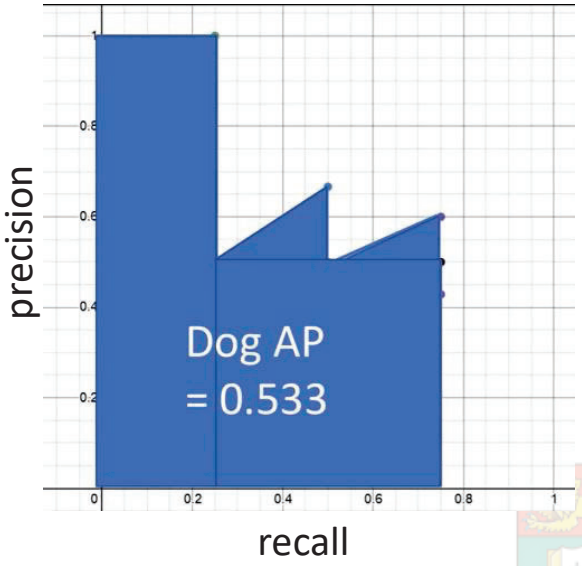
3. Average Precision

Assume we set IoU=0.5

precision	recall
1/1	1/4
1/2	1/4
2/3	2/4
2/4	2/4
3/5	3/4
3/6	3/4
3/7	3/4

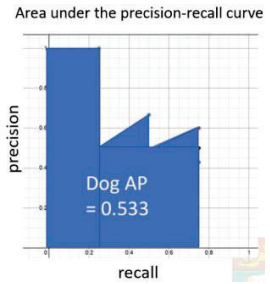


Area under the precision-recall curve



In this case, we say, the AP for **the dog class** in the test/validation is 0.533 when **IoU is 0.5**.

4. Mean Average Precision



- If AP(dog)=0.533, AP(cat)=0.733
- If the problem only has these two class, then mAP=0.633
- Consider IoU = 0.5 in this scenario, so it could be written as
  - mAP@0.5=0.633

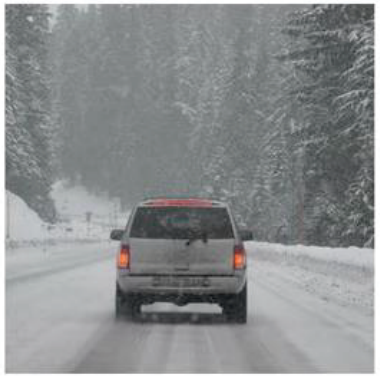
- We often see “mAP@0.5:0.05:0.95 = XXX”, it means the overall average AP when IoU = 0.5, 0.55, 0.6, 0.65 … … 0.85, 0.9.

Q

Section 2 - YOLO

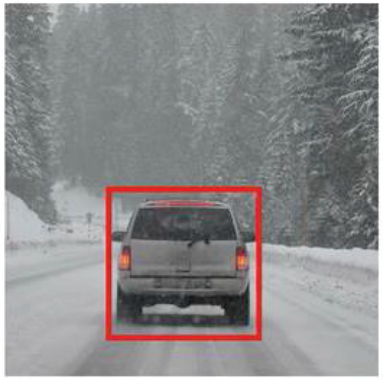
Review Localization and Detection

Image classification



1 “car”

Classification with localization



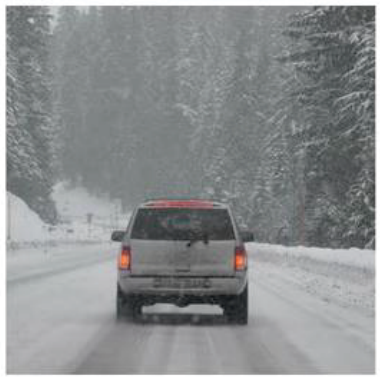
1 “car”

Detection

multiple “car”

Review Localization and Detection

Image classification



1 “car”

X= [image pixels]

network

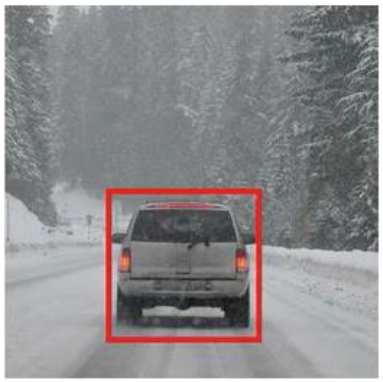
$$Y = \begin{bmatrix} C1 \\ C2 \\ C3 \\ C4 \end{bmatrix}$$

*C represents classes, e.g.:

1 - pedestrian
2 - car
3 - motorcycle
4 - background

Review Localization and Detection

Classification with localization



1 “car”

X= [image pixels]

network

$$Y = \begin{bmatrix} P \\ x \\ y \\ w \\ h \\ C1 \\ C2 \\ C3 \\ C4 \end{bmatrix}$$

*C represents classes, e.g.:

1 - pedestrian
2 - car
3 - motorcycle
4 - background

Review Localization and Detection

Detection

multiple “car”

X= [image pixels]

CV or Network

Region of Interest 1

network

$Y1 = \begin{bmatrix} P \\ x \\ y \\ w \\ h \\ C1 \\ C2 \\ C3 \\ C4 \end{bmatrix}$

...

...

...

Region of Interest n

network

$Yn = \begin{bmatrix} P \\ x \\ y \\ w \\ h \\ C1 \\ C2 \\ C3 \\ C4 \end{bmatrix}$

RCNN use CV based methods
Fast RCNN use CV based methods on feature maps
Faster RCNN use Region Proposal Network

In many real-world applications, we want faster

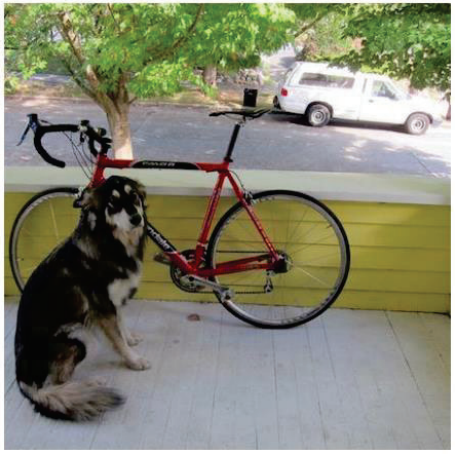
- Faster R-CNN is still slow especially in real time applications.
- Also the complex structure in Faster R-CNN with multiple outputs that are each a potential source of error
- in object detection, to assist the process, we added region proposal, which we know is the major time consumer.
- Do we have to use region proposal?

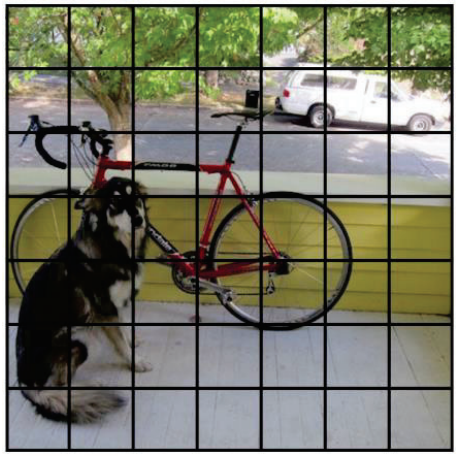
You only look once (YOLO)

- Another type of methods that did not rely on the region proposal.
- Instead of localize the object perfectly, it emphasize more on speed and recognition.
- It is a real time (in inference) object detection algorithm (very fast)

You only look once (YOLO)

- RCNN type of methods will slide through the image using different window size (time consuming)
- YOLO only scan once using one predefined size of window (no overlap, stride = w or h of the cell).





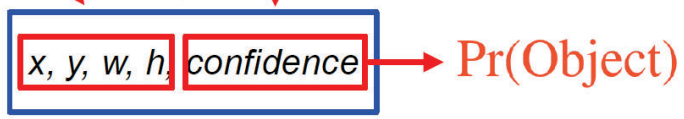
S x S

Training (labeling)

- All BBox, All classes

1) Image → **S x S** grids

2) grid cell
→ **B**: BBoxes and Confidence score



→ **C**: class probabilities w.r.t #classes

Just like typical classification and localization, but for all cells

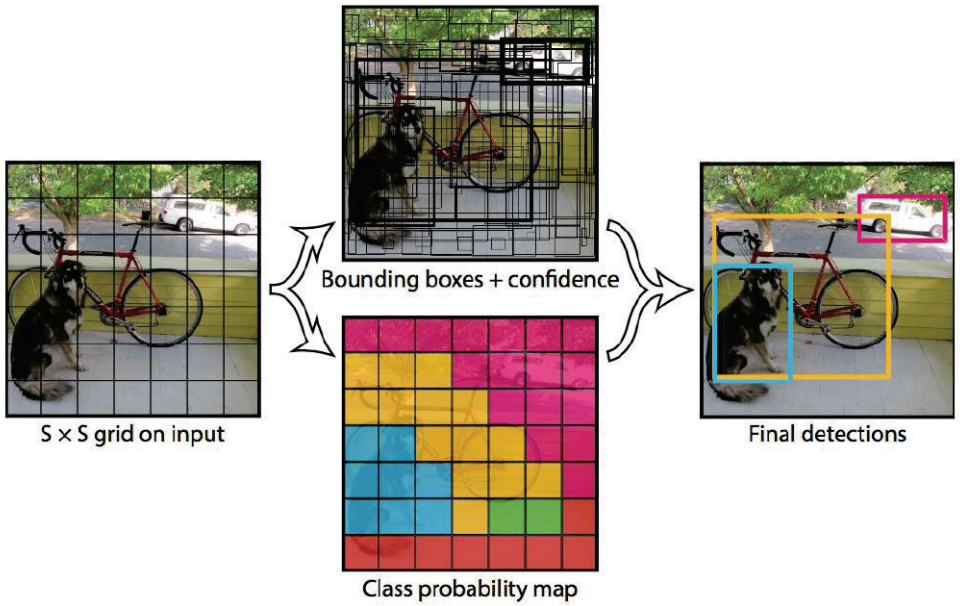


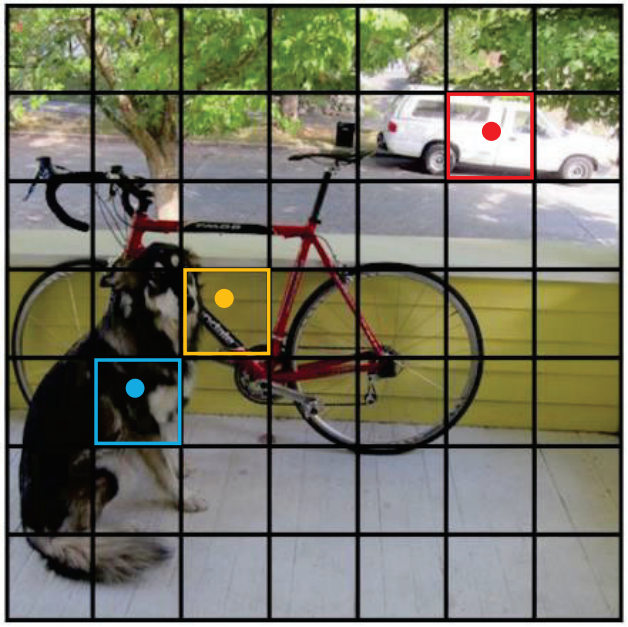
Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

Training (labeling)

You can in fact understand it as we break down one image into SxS (7x7) small training images.

The cells where the centers of the object falls at, are the True cases, and others are the False cases:

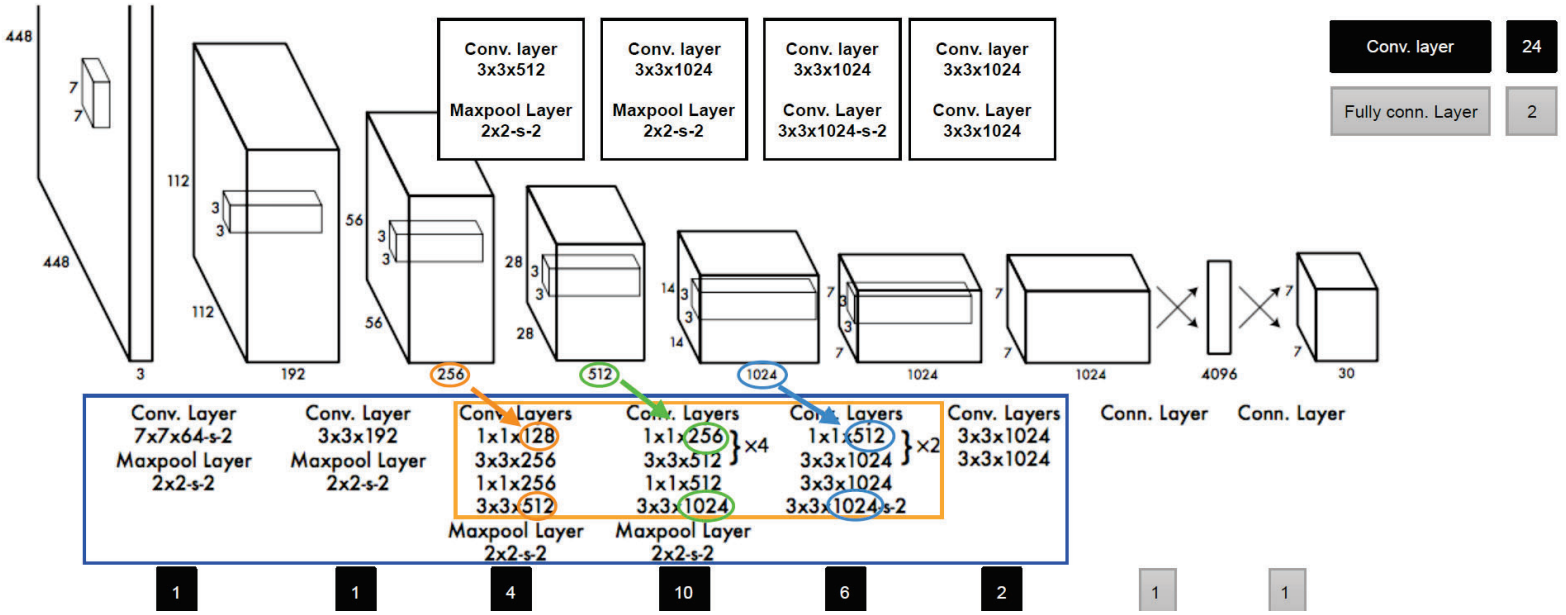
$$Y_b=\begin{bmatrix} P=1 \\ x \\ y \\ w \\ h \\ C1=1 \\ C2 \\ C3 \\ \dots \end{bmatrix} \quad Y_y=\begin{bmatrix} P=1 \\ x \\ y \\ w \\ h \\ C1 \\ C2=1 \\ C3 \\ \dots \end{bmatrix} \quad Y_r=\begin{bmatrix} P=1 \\ x \\ y \\ w \\ h \\ C1 \\ C2 \\ C3=1 \\ \dots \end{bmatrix} \quad Y_{others}=\begin{bmatrix} P=0 \\ x \\ y \\ w \\ h \\ C1 \\ C2 \\ C3 \\ \dots \end{bmatrix}$$



YOLO is somehow guessing your whole object (size and location) by looking at parts of you

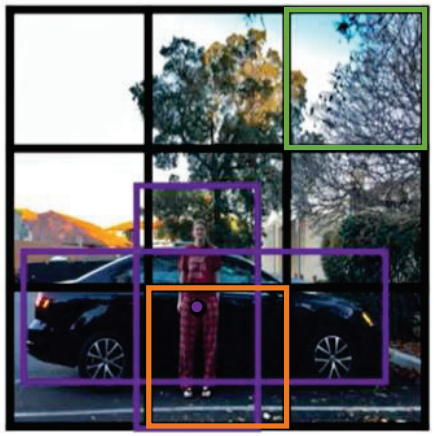
Training (network)

- The network is a bit complicated. It has the network in network concept. You don't need to understand it at this stage, but view it as an modified GoogLeNet. It evolved into more complicated but faster structure in later versions.



Quick Question

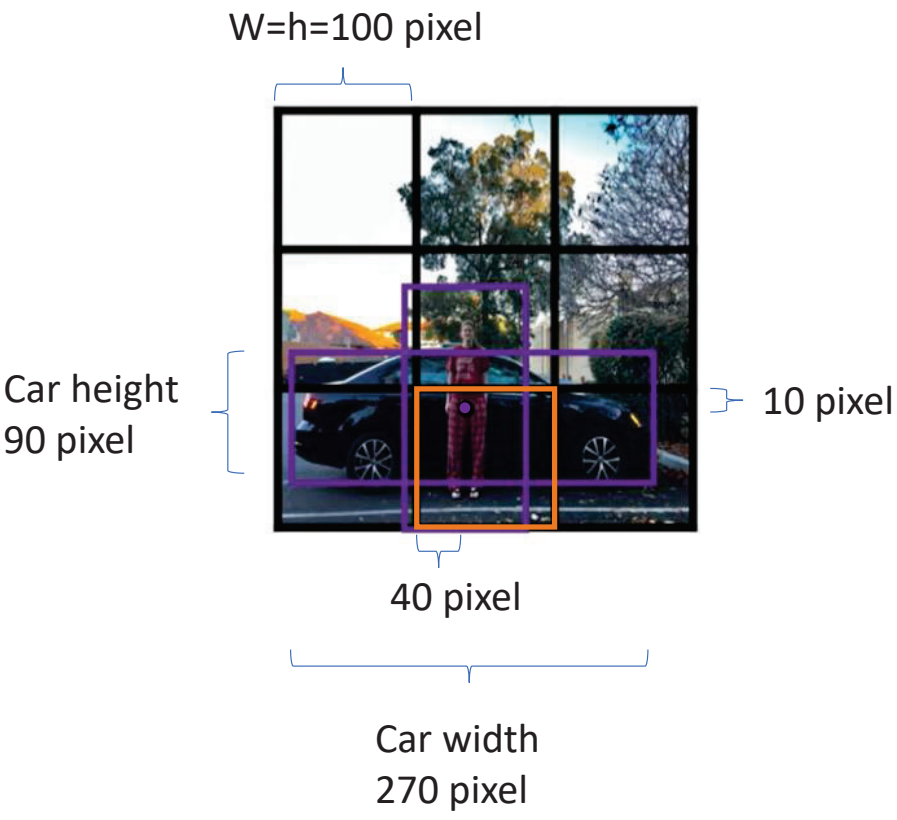
Q: Assume the purple boxes are the ground truth in the following image, what are the Q values in your training data?



$$\mathrm{Y}_{\mathrm{Green}}=\begin{bmatrix} P=\boldsymbol{Q} \\ x \\ y \\ w \\ h \\ C1 \\ C2 \\ C3 \\ \dots \end{bmatrix}$$

$$\mathrm{Y}_{Orange}=\begin{bmatrix} P=\boldsymbol{Q} \\ x \\ y \\ w \\ h \\ C1 \\ C2 \\ C3 \\ \dots \end{bmatrix}$$

Quick Question



Q: if x, y represent the percentage location of the center within the cell, and width and height present the percentage the object w.r.t. the whole image. What are the values of the following 4 Q? (assume upper left corner is 0,0)

$$Y_{car} = \begin{bmatrix} P \\ x = \boldsymbol{Q} \\ y = \boldsymbol{Q} \\ w = \boldsymbol{Q} \\ h = \boldsymbol{Q} \\ C1 \\ C2 \\ C3 \\ \dots \end{bmatrix}$$

Training (loss function)

loss function:

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \qquad (3)$$

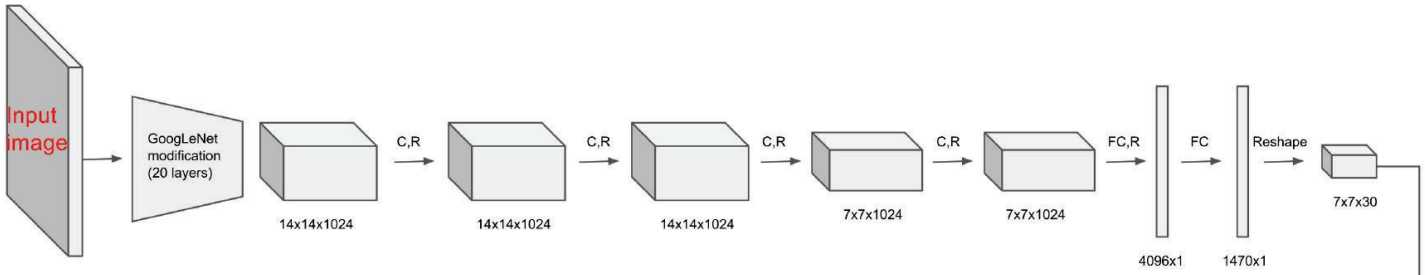
$\mathbb{1}_{ij}^{obj}$ **The *j*th bbox predictor** in ***cell i*** is "responsible" for that prediction

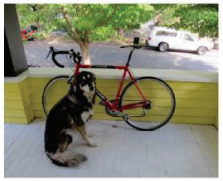
$\mathbb{1}_{ij}^{noobj}$

$\mathbb{1}_{i}^{obj}$ If object appears in ***cell i***

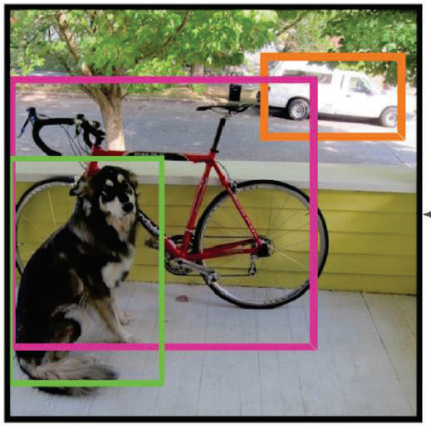
Note that the loss function only penalizes classification error if an object is present in that grid cell (hence the conditional class probability discussed earlier). It also only penalizes bounding box coordinate error if that predictor is "responsible" for the ground truth box (i.e. has the highest IOU of any predictor in that grid cell).

Test/Validation (inference)





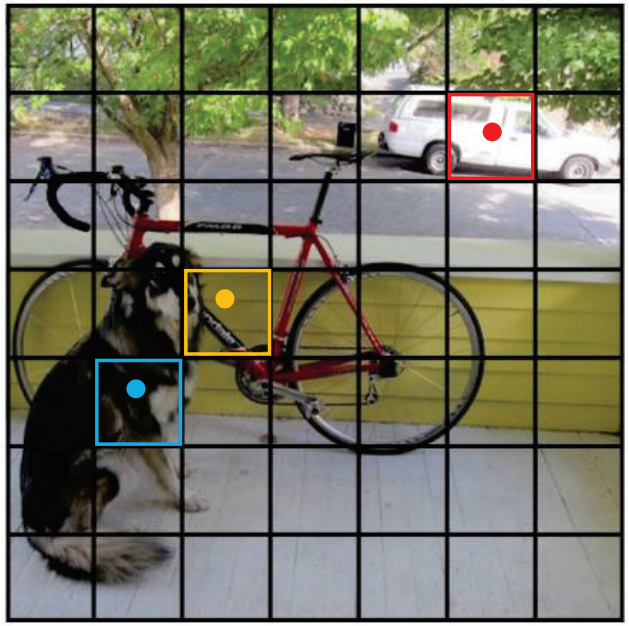
Let's assume this image becomes a test image now



Detection Procedure

Test/Validation (may generate many b-box)

Training: everybody peaceful



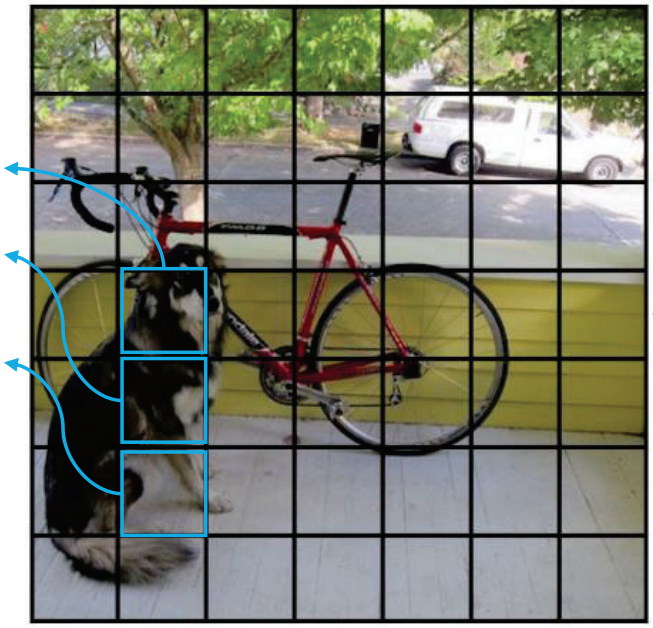
I'm the center

I'm the center

I'm the center

…

Test: cells are fighting



$$Y = \begin{bmatrix} P \\ x \\ y \\ w \\ h \\ C1 \\ C2 \\ C3 \\ \dots \end{bmatrix}$$

Test/Validation (Non-maximal Suppression)

- To overcome the generation of multiple bounding boxes for one object.
- The idea is very easy. Just select the one with the highest p value.
- Steps (the actual steps can be a bit more complicated):
  1. Look at the output vector of each grid cell. Recall that each grid cell will have an output vector with a P value and bounding box coordinates.
  2. Remove all bounding boxes that have a P value less than or equal to some threshold, say 0.5. Therefore, we will only keep a bounding box, if there is more than a 50% chance of an object being inside of it.
  3. Select the bounding box with the highest P value.

Test/Validation (overlap and multi object issue)

- Non-maximal Suppression is fine help us filtered the outputs
- The problem of the current process is that we only considered one object for one cell.
- What if we have overlapped objects in one cell, or multiple objects in one cell??? This is quite common in practice.

- One way is mitigating the issue is to make cell small, but too small will make objects hard to predict, because for big/normal objects, we only look at a smaller portion.
- Another way to help the issue is called the anchor box technique.

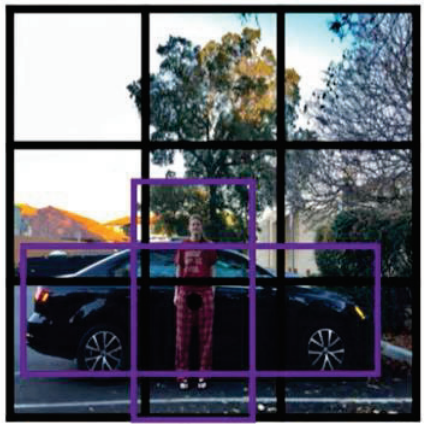
Anchor Box (affect both train and test)

- We use an additional method called anchor boxes. You can view the process as update the original cell into two different shaped cells.
- You can also understand it as adding more b-box in Y, but the b-boxes have different preferred shapes.

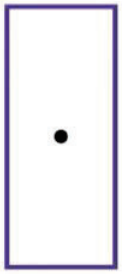
$$Y = \begin{bmatrix} P \\ x \\ y \\ w \\ h \\ C1 \\ C2 \\ C3 \\ \dots \end{bmatrix} \Rightarrow Y = \begin{bmatrix} P1 \\ x1 \\ y1 \\ w1 \\ h1 \\ C11 \\ C12 \\ C13 \\ \dots \\ P2 \\ x2 \\ y2 \\ w2 \\ h2 \\ C21 \\ C22 \\ C23 \\ \dots \end{bmatrix}$$

Simple
Rough
But it efficiently worked

Anchor Box



Anchor box 1:



Anchor box 2:

To facilitate the process, we will usually predefine a ratio for w and h for different boxes. It's like teamwork, one is responsible for tall and thin objects, while the other for short and wide objects.

Previously:

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

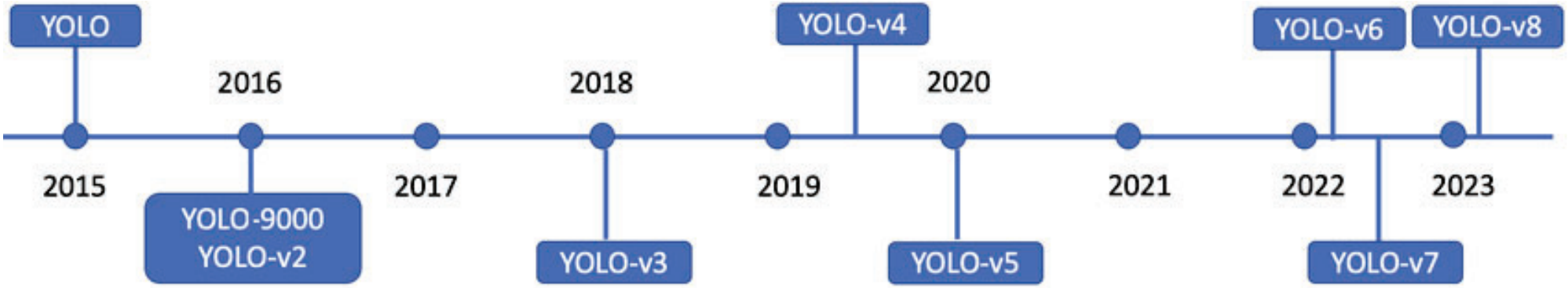
With two anchor boxes:

Each object in training image is assigned to grid cell that contains object's midpoint and anchor box for the grid cell with highest IoU.

Anchor Box

- You can define as many anchor boxes as you want. This is a method to "mitigate" and "get closer"
- The truth is even human eyes are difficult to identify multiple objects if they overlapped. So don't give too much hope on the performance.
- Imagine you have 10 or more objects in one cell, then 5 anchor boxes will not be sufficient. However, if you set more anchor boxes, it will make it slower and more expensive to train, so overall is a balance issue.
- In any case, the concept is very important, as many modern algorithms are using it to improve the efficiency. (e.g., Faster/Mask RCNN)

YOLO Timeline



Q

Demonstration

- Detect objects using YOLO
- Custom your YOLO training

Elective Exercise

- Prepare training datasets for YOLO like networks.

- https://vivek-yadav.medium.com/part-1-generating-anchor-boxes-for-yolo-like-network-for-vehicle-detection-using-kitti-dataset-b2fe033e5807

Demonstration

- Faster R-CNN
- Mask R-CNN
- YOLO