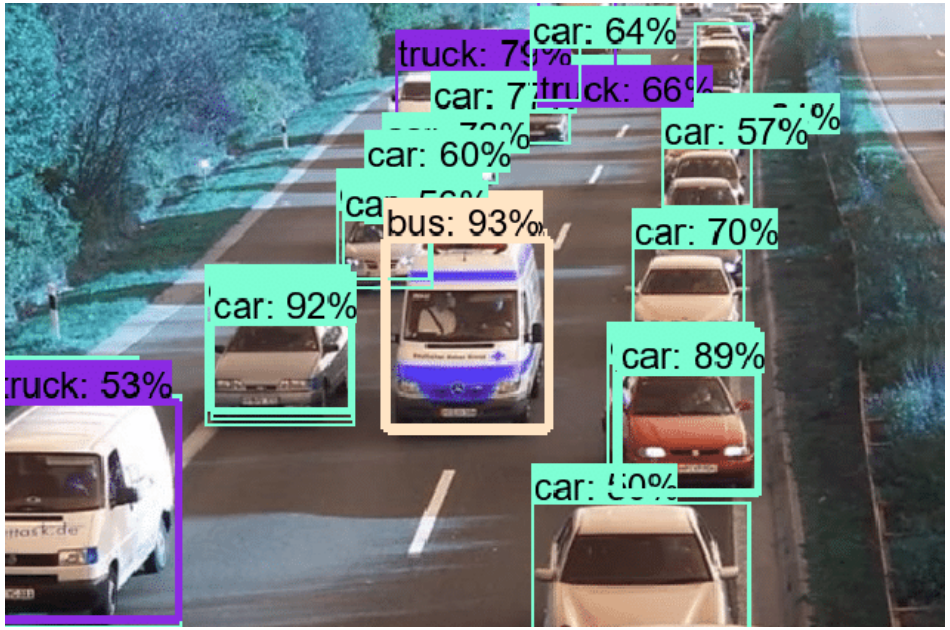
Machine Learning for Robotics: **Object Detection with YOLO**

Prof. Navid Dadkhah Tehrani

- What is object detection:
  Predicting bounding boxes, class probabilities of objects from the input image.



- Yolo [1] is a single stage object detection. This means that it predicts the bounding box and class probability in one step.
- In two-stage detectors, the first step is for generating possible object regions (region proposal) and another for classifying them.

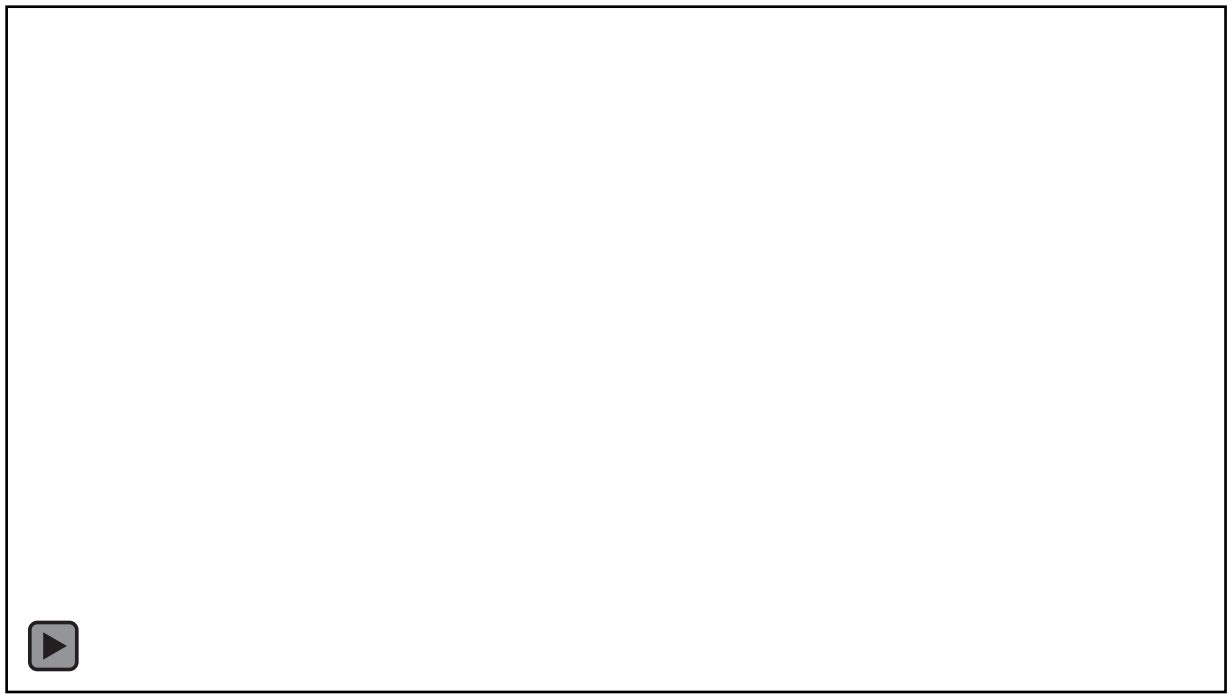
- By handling object detection in one pass,
  YOLO reduces the computational load and is used for real-time applications.

[1] Joseph Redmon, et. al. You Only Look Once: Unified, Real-Time Object Detection, 2015

What is object tracking?

continuously identifying and following objects across a sequence of video frames,
assigning each object a unique identifier so it can be recognized as the same entity from frame to frame.

YOLO itself is not inherently designed for object tracking, as it focuses on object detection.
However, it can be adapted for object tracking when paired with additional algorithms to link objects across frames.



Detection-based tracking

In many applications, object tracking without unique identifiers can still serve the purpose effectively.

YOLO is well-suited for detection-based tracking.

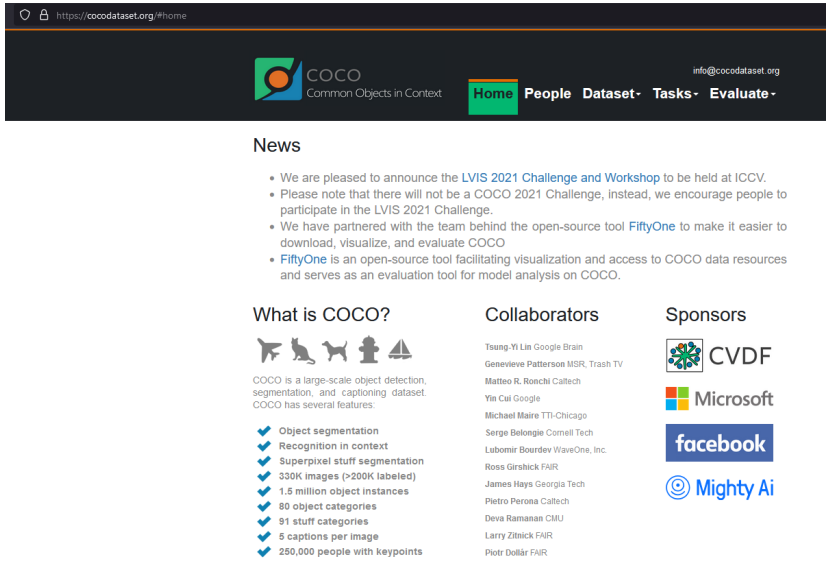
Some application of object detection:

- Density Analysis in traffic.
- Motion Detection and Anomaly Detection
- Heatmap Generation: which parts of a store are most visited.
- Obstacle Avoidance
- Sports Analysis: tracking players positions in football/soccer.
- Agricultural Monitoring: monitor animal activities.

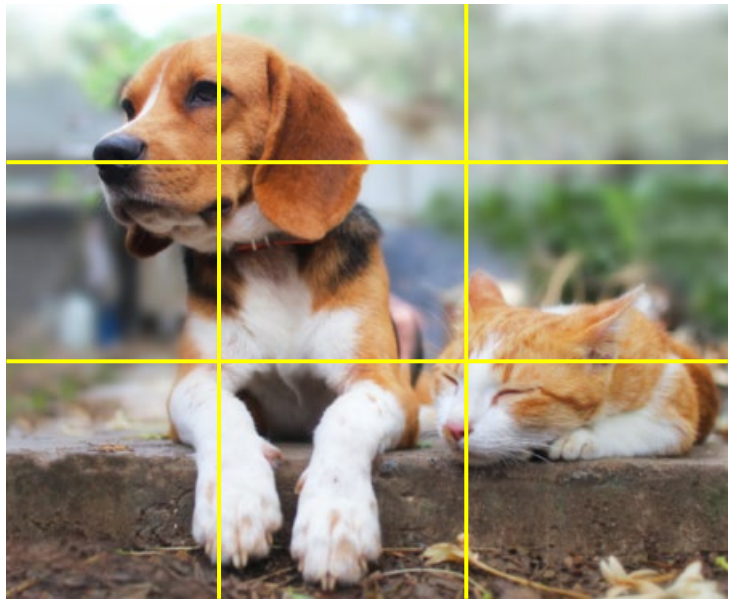
The datasets that are used for object detection, should have ground truth bounding box location as well as class probabilities.

Most papers use COCO dataset for object detection. It has a few hundred thousands images and 80 object categories (e.g., people, animals, vehicles).

https://cocodataset.org

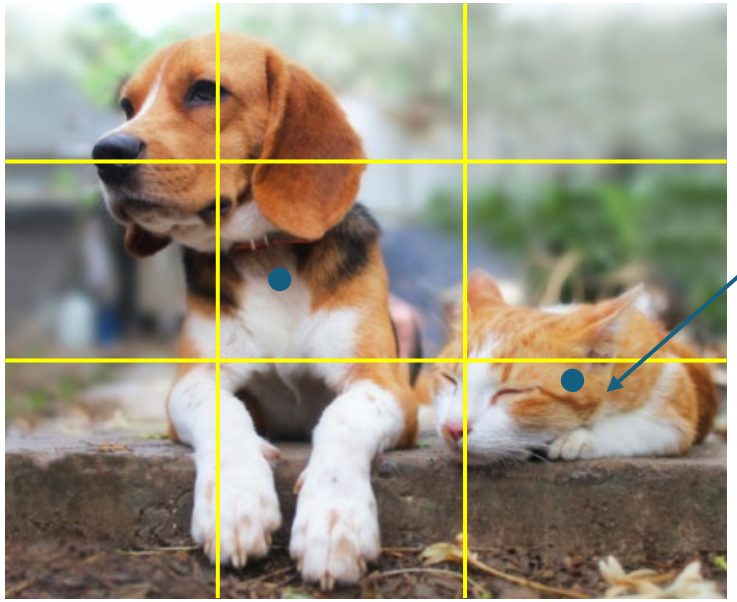


The image is split into $S * S$ grid. In YOLO they used $S = 7$.
Here for demonstration purpose, we use $S = 3$. and we assume to have 20 classes in the dataset.

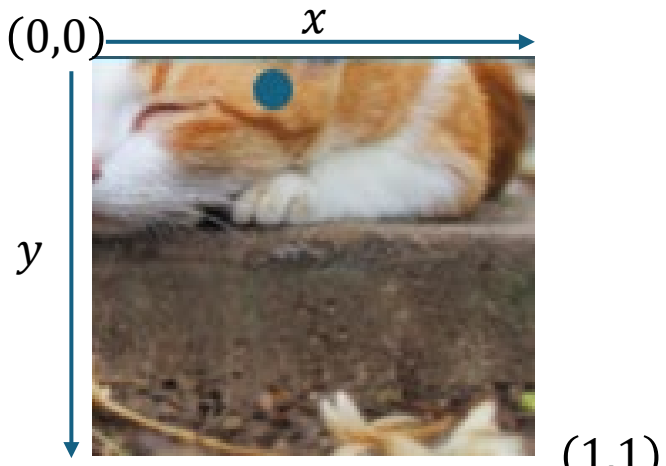


We have 9 cells.
Each cell outputs a class prediction,
and a bounding box.

Q:Each dog is in multiple cells. How do we make sure only one object is actually outputted for each object?
A: we find one cell that's responsible for outputting the object. The cell that is responsible for outputting the object contains the objects midpoint.



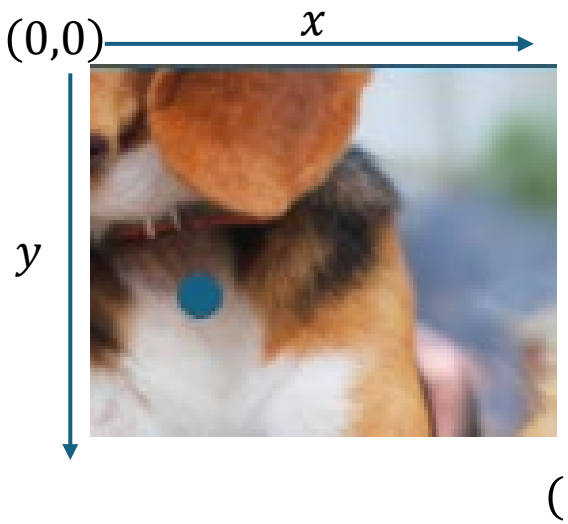
Center of object belongs to this cell



Each bounding box output is (x, y, w, h) relative to the cell.

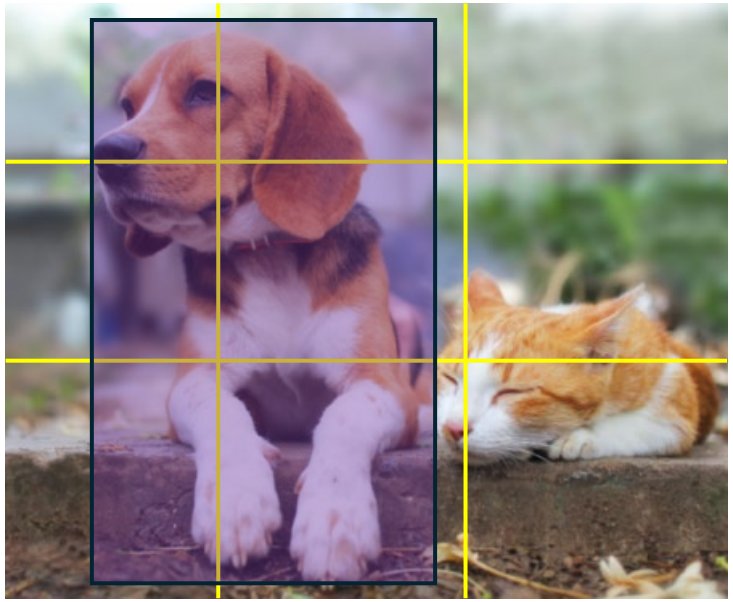
For example, for this cell the bounding box
Is: $(0.25, 0.5, 1.2, 2.8)$

h is 1.2 because the height of the dog is 120 percent of the cell size. And w is 2.8 because its width is 280 percent of the cell.



Therefore:

- the object can be bigger or smaller than a cell.
- The bbox does not have to align with a cell.



Note that in dataset the location of bounding boxes and their dimension is not w.r.t cell and are global. This is because resizing the image should not affect the location of the bounding box.

Once the we read the dataset, additional operations is need to convert the location of the bounding boxes to Yolo format (i.e. with respect to the cell coordinate)

What's the ground truth for each cell?

$$cell\ label = [p_1, p_2, \dots, p_{20}, C, x, y, w, h, 0,0,0,0,0] \rightarrow 1 * (25 + 5\ \text{zeros}) = 1 * 30$$

$p_1, p_2, \dots.: class\ probability\ for\ all\ the\ 20\ classes$ → one hot
$C: confidence\ score\ that\ there's\ an$ object in the cell $\rightarrow \in \{0,1\}$ (if there's no object in the cell, the confidence=0)

What's the prediction for each cell?
It's similar to ground truth, except that it output B bounding boxes, and they used $B = 2$ in the paper
(Tall and wide objects)

$$cell\ prediction = [p_1, p_2, \dots, p_{20}, C_1, x_1, y_1, w_1, h_1, C_2, x_2, y_2, w_2, h_2] \rightarrow 1 * 30$$

$C_1: confidence\ that\ there's\ an\ object\ in\ box\ 1$
$C_2: confidence\ that\ there's\ an\ object\ in\ box\ 2$

Shape of the cell label for the entire image is $(S, S, 25)$ or $(S, S, 30)$ if we add zeros.
Shape of the cell prediction for the entire image is $(S, S, 30)$

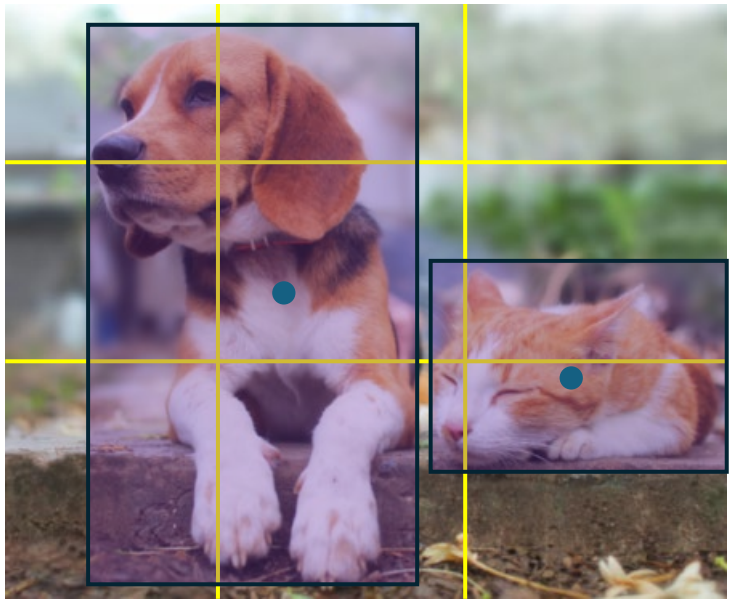
Example of cell labels

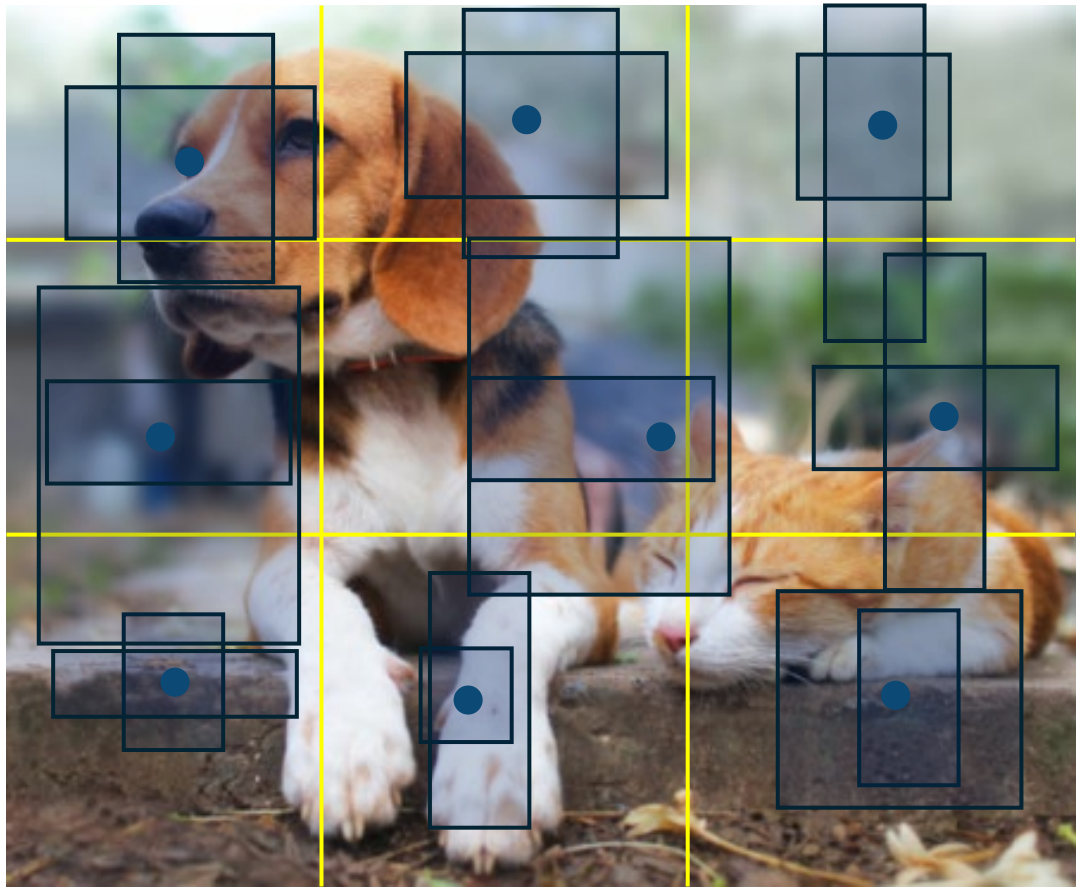
```
[1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
0.0000, 1.0000, 0.4930, 0.3469, 6.7620, 2.7755, 0.0000, 0.0000, 0.0000, 0.0000,
0.0000],


[0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000,
0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
0.0000, 1.0000, 0.5140, 0.0480, 6.9440, 5.9700, 0.0000, 0.0000, 0.0000, 0.0000,
0.0000],

[0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000,
0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
0.0000, 1.0000, 0.4540, 0.1733, 1.0920, 2.0160, 0.0000, 0.0000, 0.0000, 0.0000,
0.0000]
```

Ground truth→





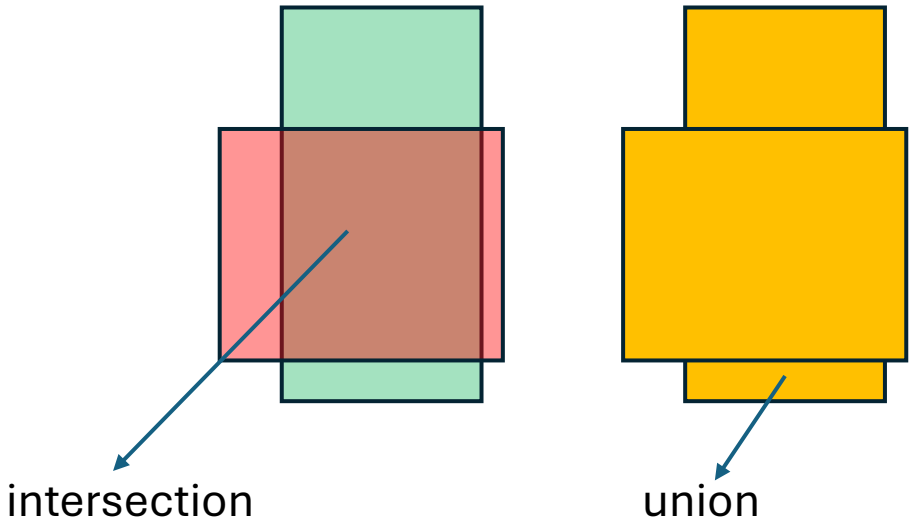
← a prediction in the middle of training

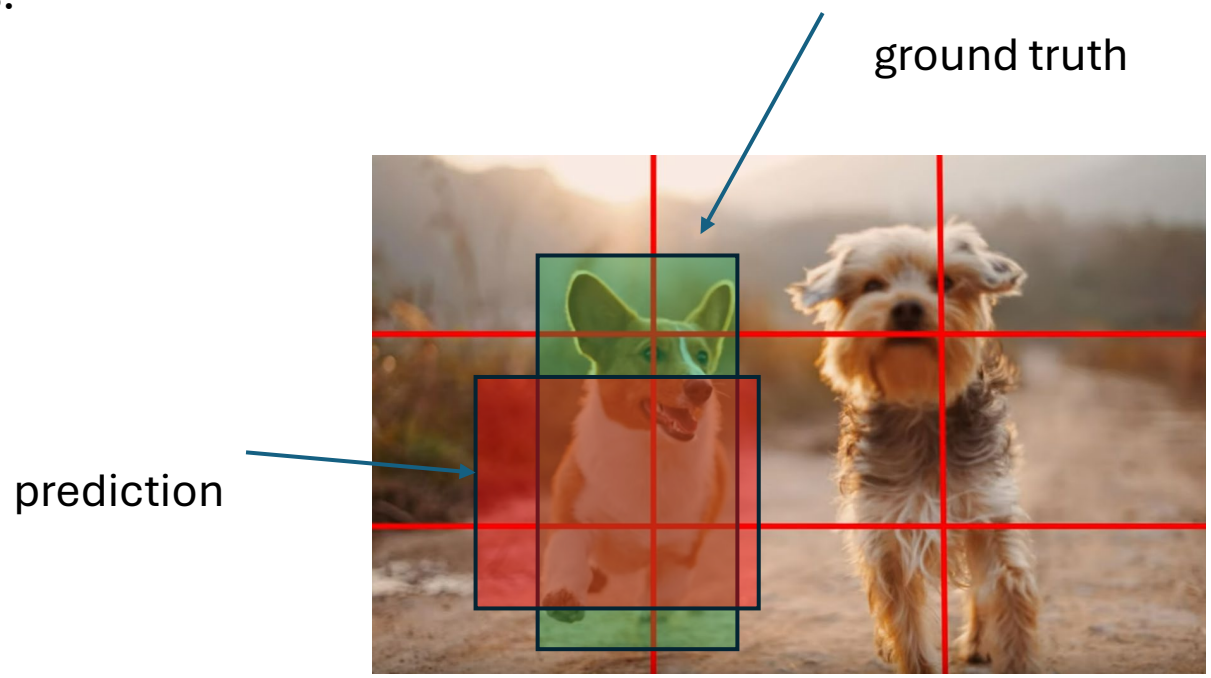
With $S = 3$ and $B = 2$, the model generates 18 bounding boxes.

with S=7 and B=2, the model generates 98 bbox

Intersection Over Union (IOU)

IOU is a way to quantify how good the predicted bbox is.





$$IOU = \frac{area\ of\ the\ intersection}{area\ of\ the\ union} \in [0,1]$$

$$IOU > 0.5 \rightarrow ok$$

$$IOU > 0.7 \rightarrow good$$

box_iou

torchvision.ops.box_iou(*boxes1: Tensor*, *boxes2: Tensor*) → Tensor [SOURCE]

Return intersection-over-union (Jaccard index) between two sets of boxes.

Confidence Score as IoU

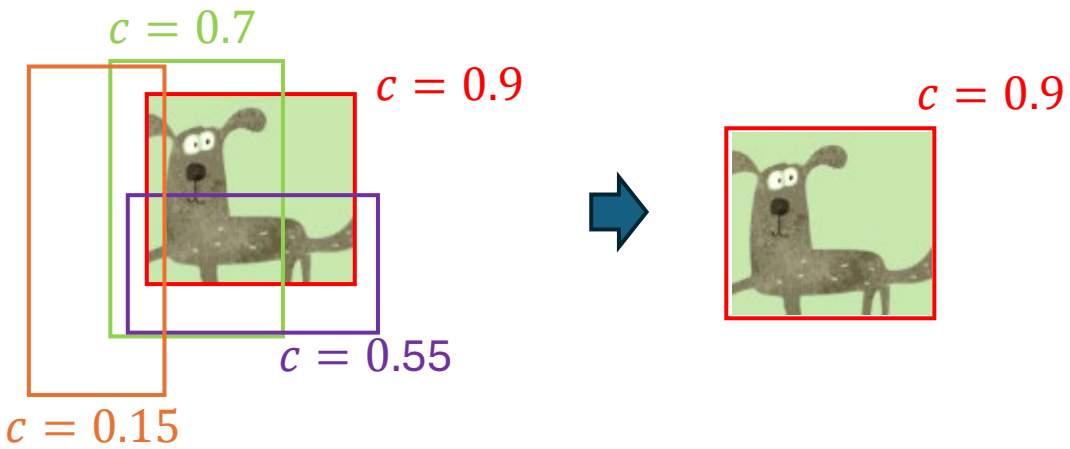
Yolo v1 uses the IoU between the predicted bounding box and the ground-truth box as the confidence score.

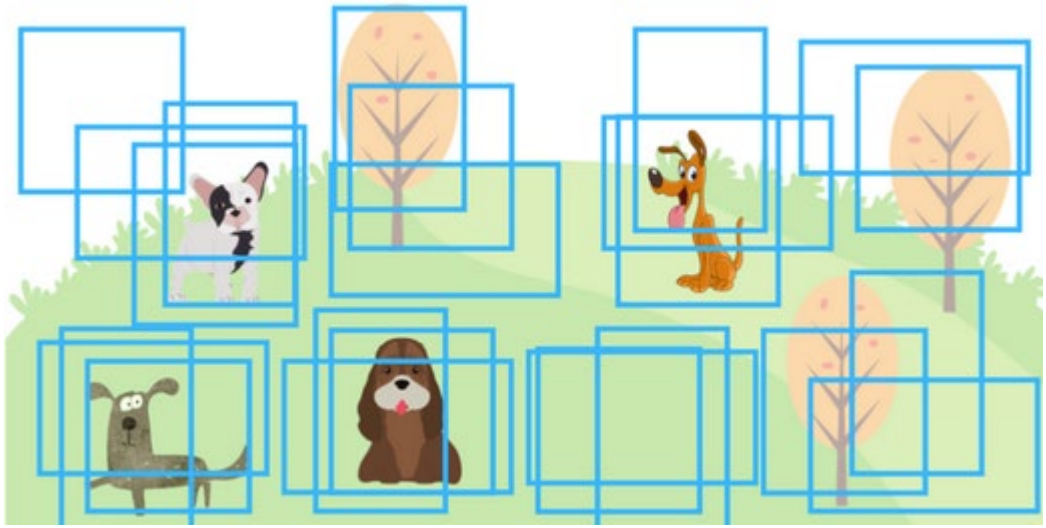
$$\hat{C} = IoU(predicted\ bbox, ground\ truth\ bbox)$$

Non-Max Suppression (NMS)

The goal of NMS is to clean up the predicted bounding boxes.

As seen before, the model predicts many bboxes for a single object (each cell predicts ‘B’ bbox)
But we need to have a single bbox per object.





confidence_threshold = 0.2 (hyperparameter)
iou_threshold = 0.5 (hyperparameter)
prob(orange) < confidence_threshold → get rid of the orange box.
IOU (red,green) = 0.6 > iou_threshold → get rid of the green box.
IOU (red,purple) = 0.55 > iou_threshold → get rid of the purple box.

we perform NMS for each class independently as model outputs scores
for all classes.

Here’s the process:

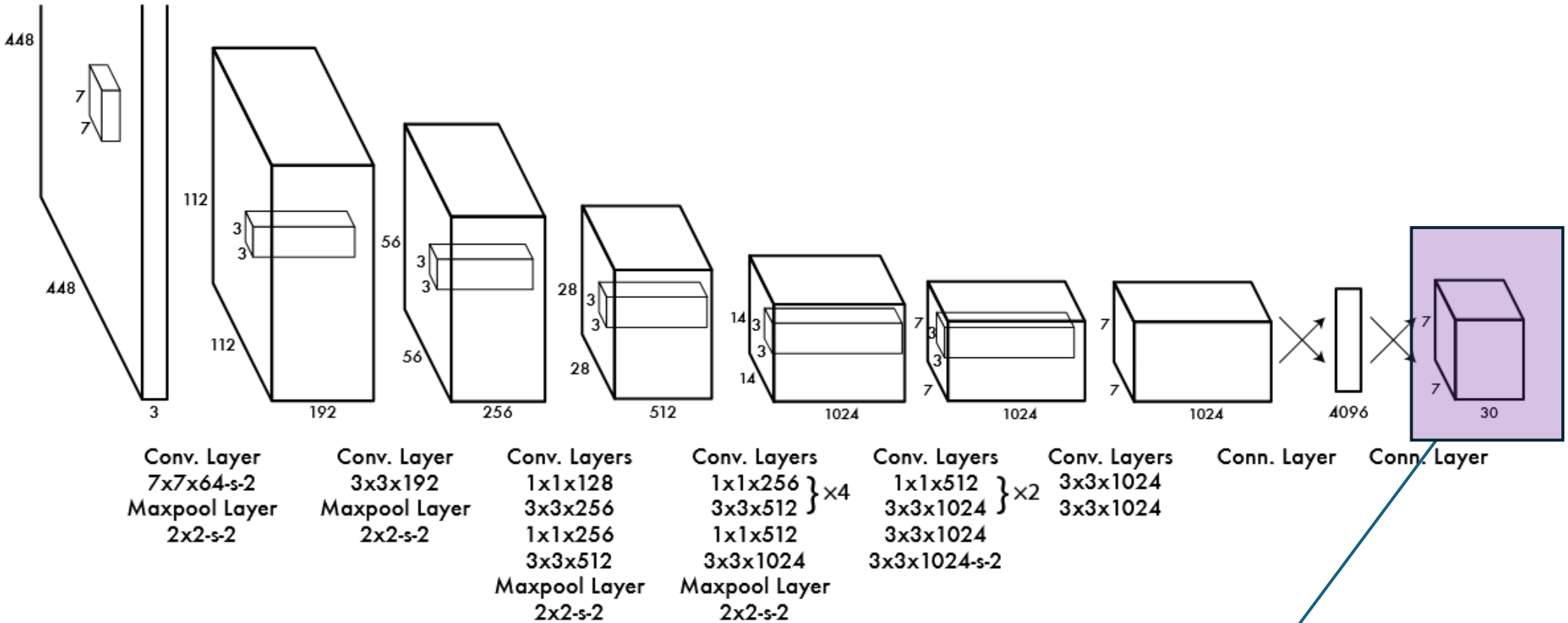
1. start by selecting one class (e.g., "person" or "car") and considers only the bounding boxes predicted to contain objects of that class.

2. then sort these bounding boxes by their confidence score

3. Suppress overlaps by applying NMS (like in the previous slide)

Network Architecture

The yolo model is fully convolutional.



This is the $S * S$ grid ($S = 7$) and each grid has 30 elements associated with it (class prob, bbox, confidence)

Loss function

The yolo v1 lost function is a regression lost.

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

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

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

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

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

$1_i^{obj} = 1$ *, if object appears in cell* i

$1_{ij}^{obj} = 1$, if the jth bounding box predictor in cell i is "re-sponsible" for that prediction

Loss has 3 terms:

1. Coordinate loss: penalizes location and height/width of bbox for the cells that have objects.
2. Confidence loss: penalizes the error in confidence for boxes that contain objects and Penalizes confidence for boxes that do not contain objects, using a smaller scaling factor $\lambda_{noobj} = 0.5$
3. Classification loss: Penalizes the error in the predicted class probabilities for grid cells containing an object.

Q: Why can’t we use IoU directly in the loss?
A: the basic IoU introduced in yolo v1 does not have gradient if the predicted bbox and ground truth bbox do not overlap. In other words the IoU of two non-intersecting bbox is zero regardless if they’re close or far from each other.

There are different versions of IoU that remedy this. Generalized IOU (GIoU), distance IOU (DIoU), Complete IOU (CIoU). These IoUs all have gradient even if the predicted and groundtruth bbox do not overlap.

More recent versions of Yolo incorporated GIoU and CIoU.

[1] Zhaohui Zheng, et. al. "Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression", 2019

Yolo v1 versus more recent Yolo versions:

- Ultralytics Inc. has taken over releasing of the open-source YOLO models.

Their Github page contains the implementation of the latest yolo models.

https://github.com/ultralytics/ultralytics

- After yolo v4 there's no academic paper for yolo, and the only way to understand it is to read the code.
- Fortunately, Ultralytics added many documentation in the github for making datasets, reproducing and running the code, etc.
- They also made it easy for non-software engineers to train and use the model

The recent versions of yolo share many characteristic with yolo v1.
Some key differences:

- Model make use of Utilizes multi-scale feature pyramids techniques(as opposed to VGG style), making it capable of effectively detecting both small and large objects within the same scene.

- The loss function uses IoU-based losses that improve bounding box regression by focusing specifically on the overlap quality between predicted bounding boxes and ground-truth boxes

- Cross entropy loss for classification loss.