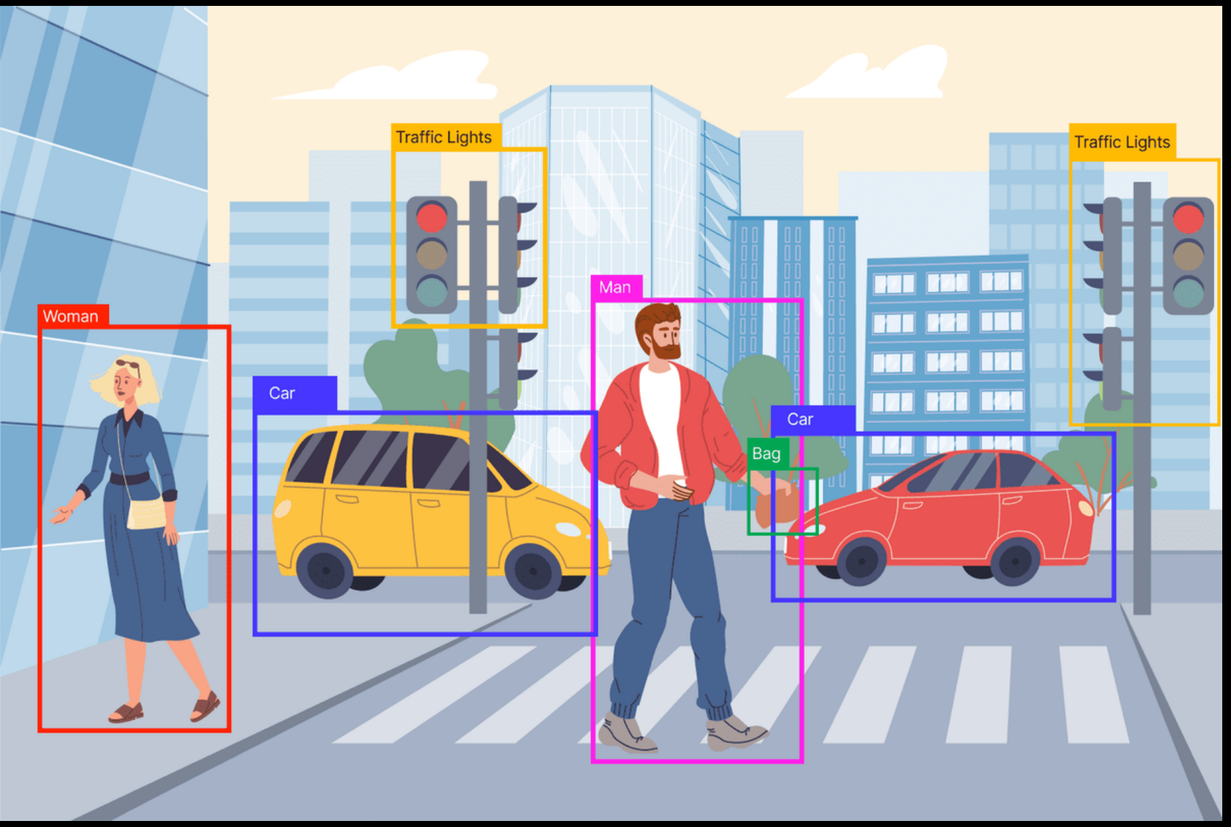
Robotics Corner

YOLOv9

Eng. Ahmed Mady, MSc

YOLO



YOLO (You Only Look Once) is a real-time object detection system. It's a convolutional neural network (CNN) designed for fast and efficient object detection in images and videos.

YOLO processes the entire image in a single pass and directly predicts bounding boxes and class probabilities for these boxes. YOLO has several versions, with each version improving upon speed and accuracy. It's widely used in applications such as autonomous vehicles and robotics for real-time object detection, tracking, and segmentation tasks.

Before YOLO?

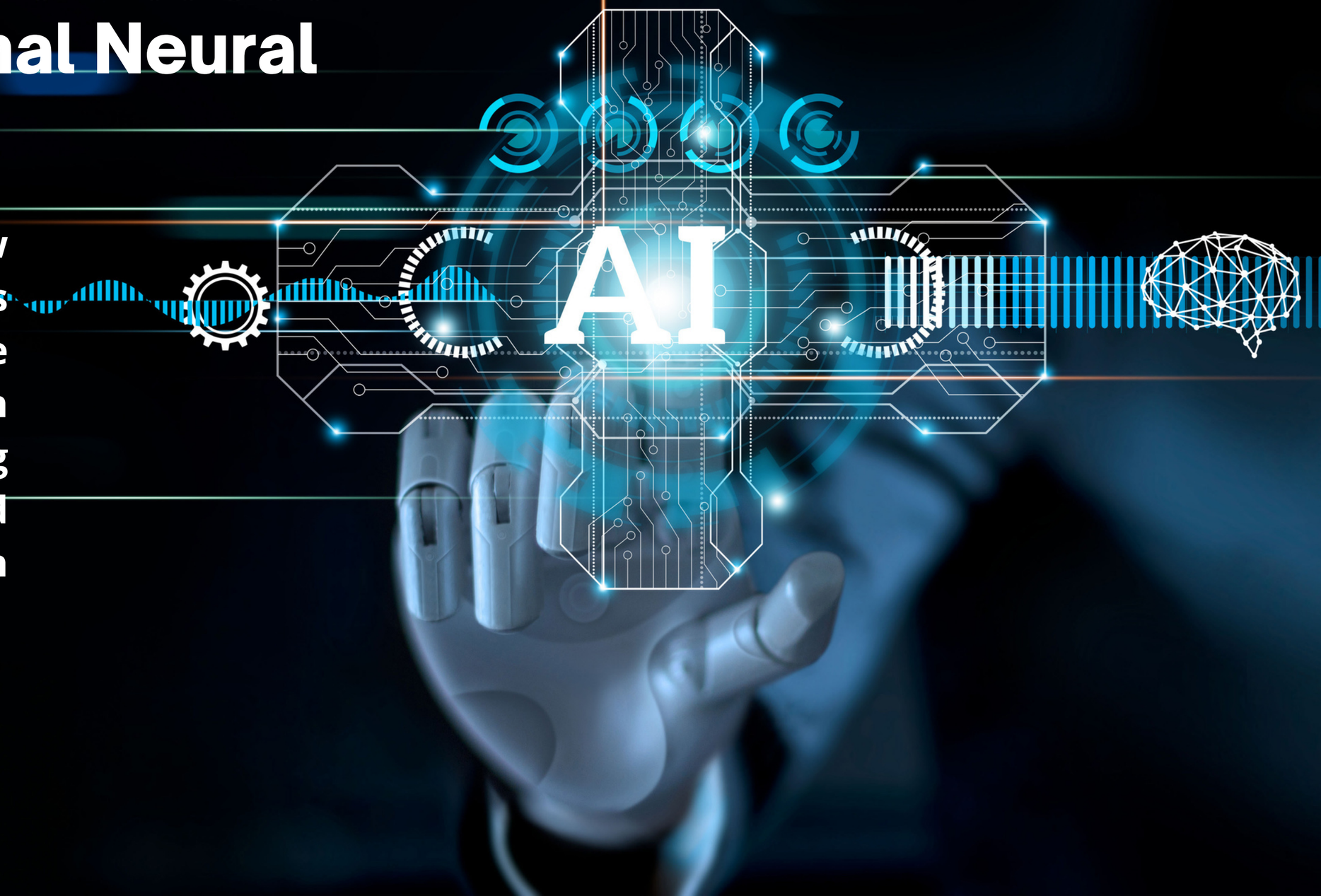
Previously, people were using Sliding Window Object Detection, then more
faster versions were invented which include RCNN, Fast RCNN, Faster RCNN.

Sliding Window

- **Limitation: Sliding window methods involve scanning the image with multiple windows at different scales to detect objects. This approach is computationally expensive and inefficient because it requires running a classifier at each window position and scale.**

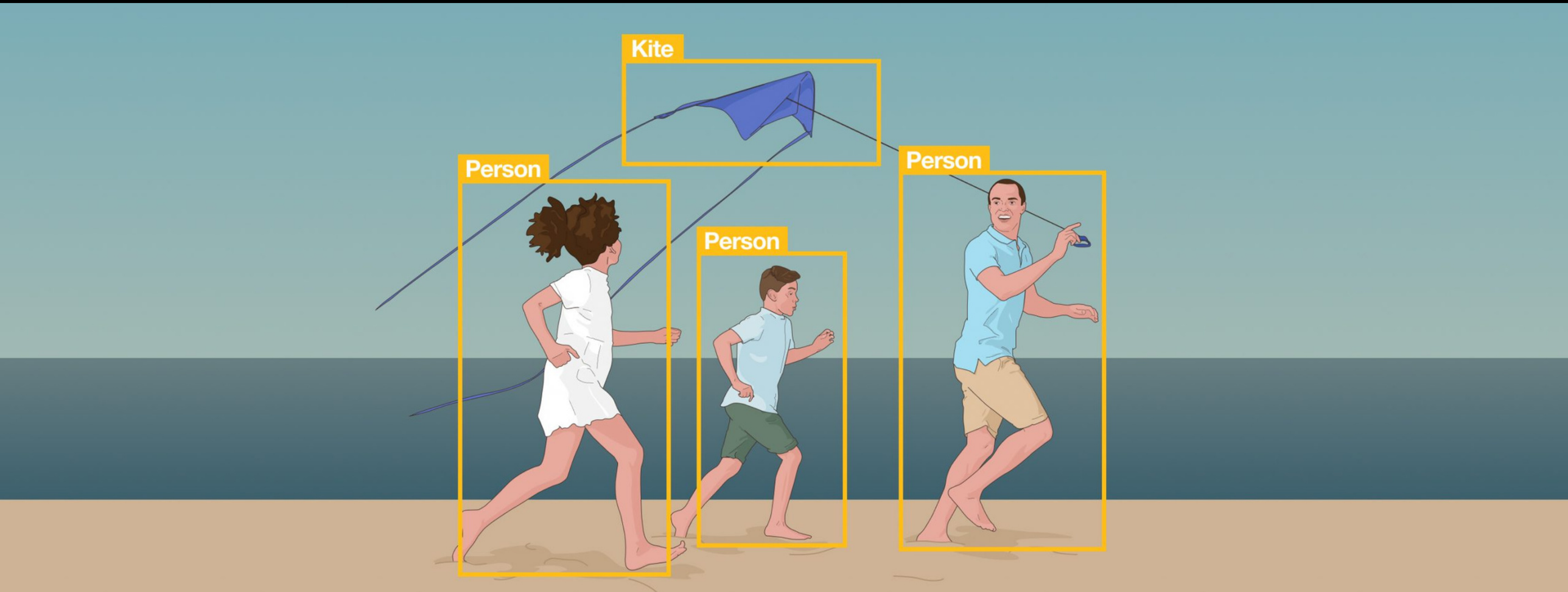
R-CNN (Region-based Convolutional Neural Network)

Limitation: Slow because it involves generating a large number of region proposals using selective search and then classifying each region independently.



Why YOLO is better

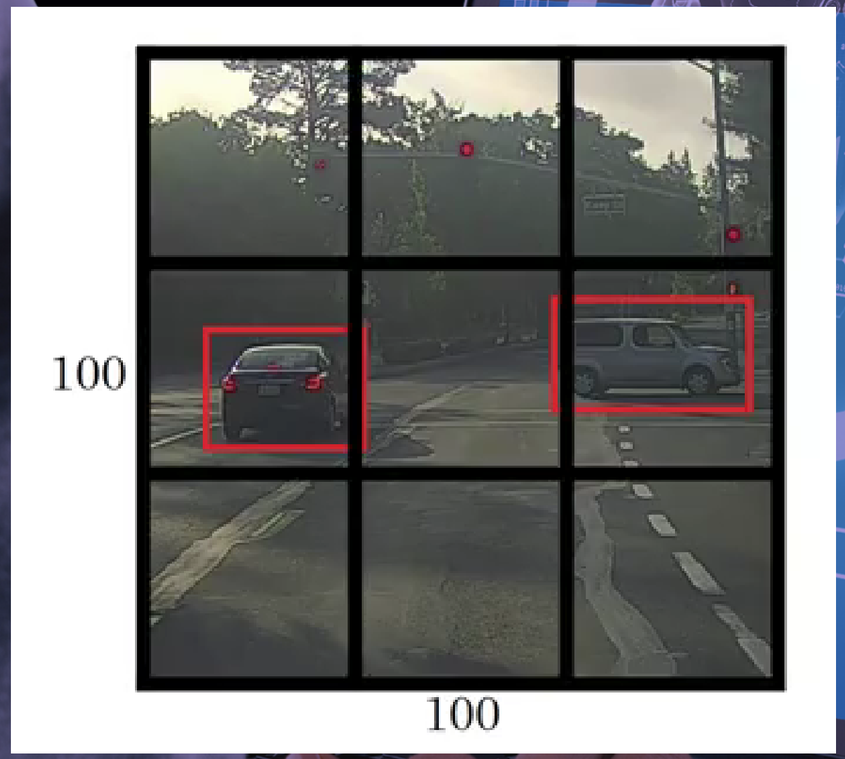
Speed, Efficiency, Real-time Performance



How it Works?

YOLO first takes an input image

The framework then divides the input image into grids (say a 3 X 3 grid)



- Image classification and localization are applied on each grid. YOLO then predicts the bounding boxes and their corresponding class probabilities for objects (if any are found)

We need to pass the labeled data to the model to train it. Suppose we have divided the image into a grid of size 3 X 3, and there is a total of 3 classes that we want the objects to be classified into. The classes are Pedestrian, Car, and Motorcycle, respectively. So, for each grid cell, the label y will be an eight-dimensional vector:

y =	pc
	bx
	by
	bh
	bw
	c1
	c2
	c3

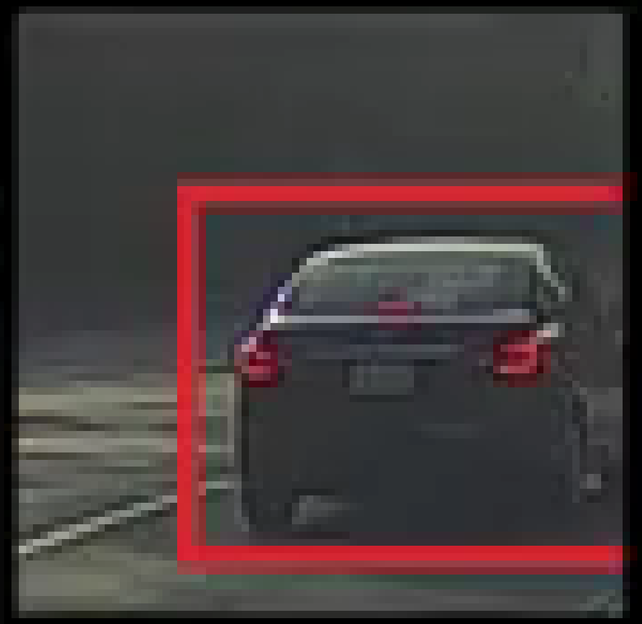
Here,

- pc defines whether an object is present in the grid or not (it is the probability)
- bx, by, bh, bw specify the bounding box if there is an object
- c1, c2, c3 represent the classes. So, if the object is a car, c2 will be 1 and c1 & c3 will be 0, and so on

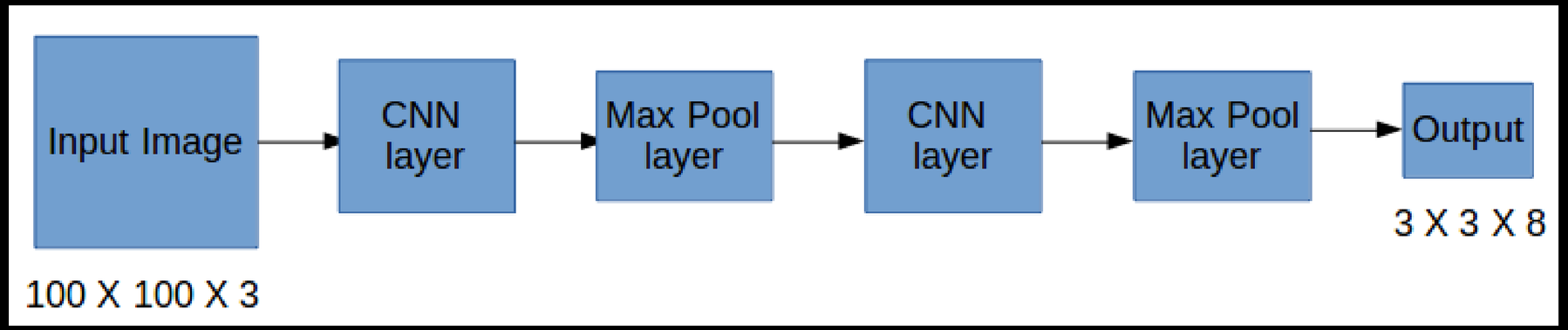
Since there is no object in this grid, pc will be zero and the y label for this grid will be:

y =	0
	?
	?
	?
	?
	?
	?
	?
	?

y =	1
	bx
	by
	bh
	bw
	0
	1
	0

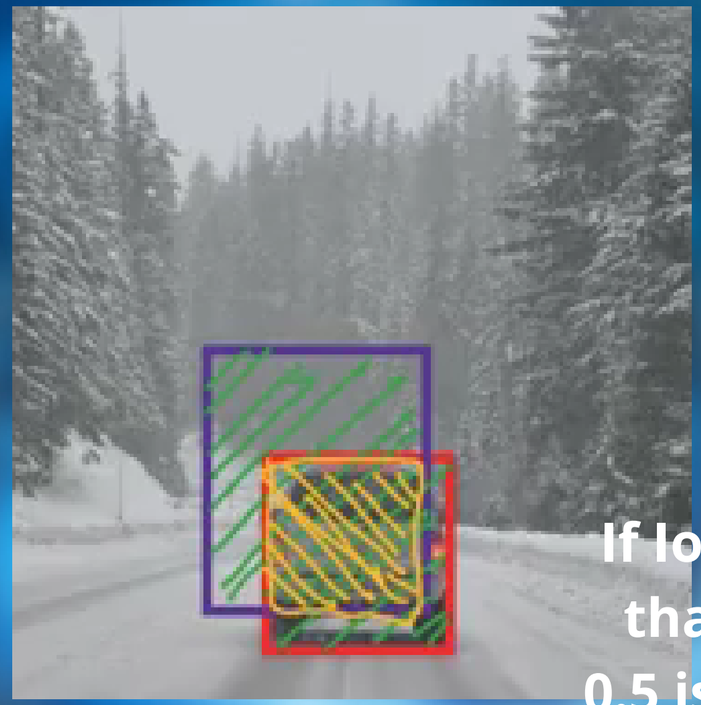


We will run both forward and backward propagation to train our model. During testing, we pass an image to the model and run forward propagation until we get an output y. To keep things simple, I have explained this using a 3 X 3 grid here, but generally, in real-world scenarios, we take larger grids (perhaps 19 X 19).



Intersection over Union

how can we decide whether the predicted bounding box is giving us a good outcome (or a bad one)? This is where Intersection over Union comes into the picture. It calculates the intersection over the union of the actual bounding box and the predicted bounding box.



Here, the red box is the actual bounding box and the blue box is the predicted one
IoU = Area of the intersection / Area of the union, i.e.

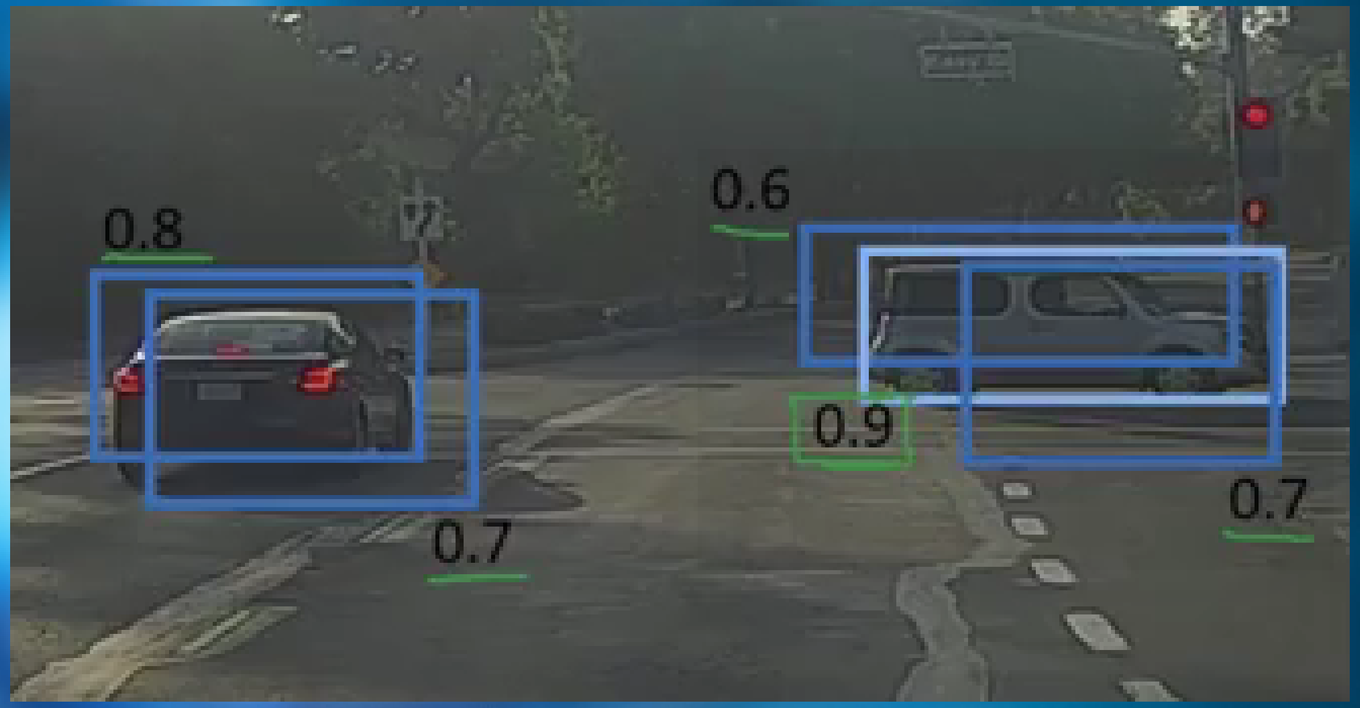
If IoU is greater than 0.5, we can say that the prediction is good enough. 0.5 is an arbitrary threshold we have taken here, but it can be changed according to your specific problem. Intuitively, the more you increase the threshold, the better the predictions.

Non-Max Suppression

One of the most common problems with object detection algorithms is that rather than detecting an object just once, they might detect it multiple times

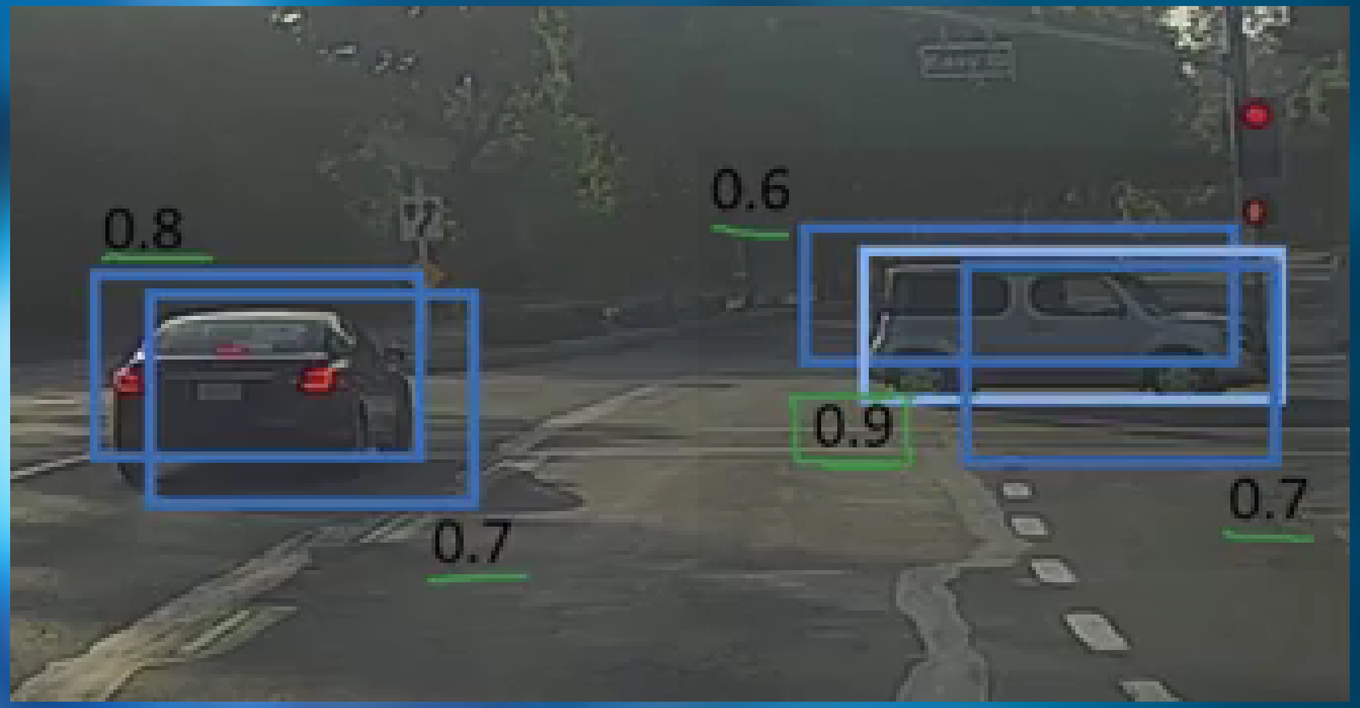
Non-Max Suppression

It first looks at the probabilities associated with each detection and takes the largest one. In the above image, 0.9 is the highest probability, so the box with 0.9 probability will be selected first

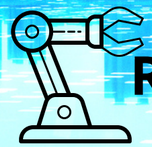


Non-Max Suppression

It first looks at the probabilities associated with each detection and takes the largest one. In the above image, 0.9 is the highest probability, so the box with 0.9 probability will be selected first



Thank You!



Robotics Corner