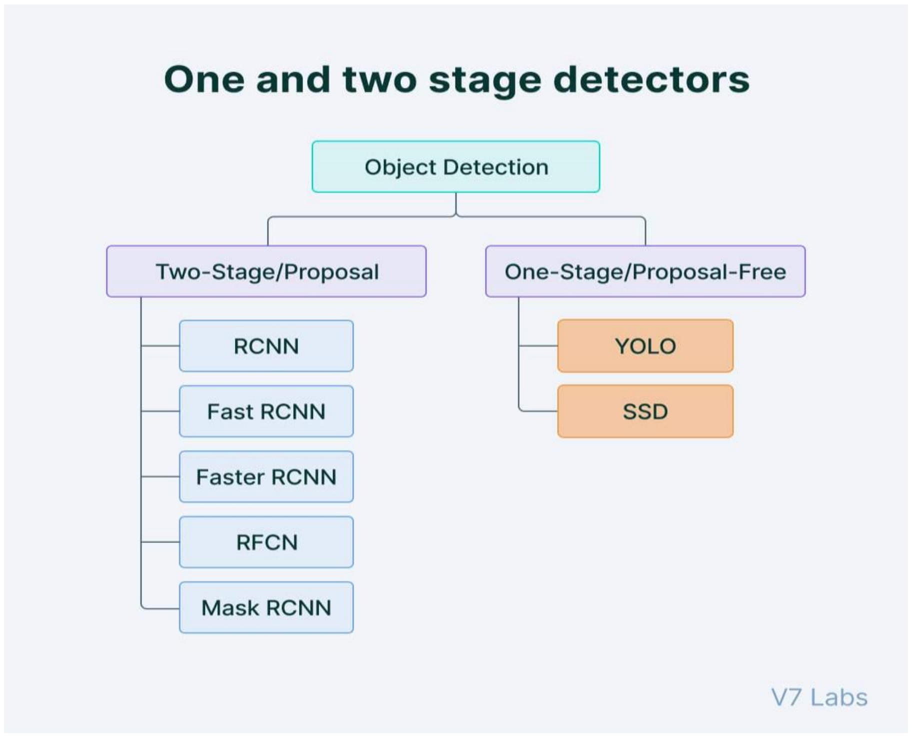
YOLO: You Only Look Once – Real Time Object Detection

What is object detection?

Object detection 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. When humans look at images or video, we can recognize and locate objects of interest within a matter of moments. The goal of object detection is to replicate this intelligence using a computer. It is an important part of many applications, such as surveillance, self-driving cars, or robotics. Object detection algorithms can be divided into two main categories: single-shot detectors and two-stage detectors.



Single-shot object detection uses a single pass of the input image to make predictions about the presence and location of objects in the image. It processes an entire image in a single pass, making them computationally efficient.

However, single-shot object detection is generally less accurate than other methods, and it's less effective in detecting small objects. Such algorithms can be used to detect objects in real time in resource-constrained environments.

YOLO is a single-shot detector that uses a fully convolutional neural network (CNN) to process an image.

Two-shot object detection uses two passes of the input image to make predictions about the presence and location of objects. The first pass is used to generate a set of proposals or potential object locations, and the second pass is used to refine these proposals and make final predictions. This approach is more accurate than single-shot object detection but is also more computationally expensive.

What is YOLO?

The original YOLO (You Only Look Once) was written by Joseph Redmon in a custom framework called Darknet. Darknet is a very flexible research framework written in low level languages and has produced a series of the best real time object detectors in computer vision: YOLO, YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOV6, YOLOV7, YOLOv8, and YOLO-NAS. It was proposed to deal with the problems faced by the object recognition models at that time, Fast R-CNN is one of the state-of-the-art models at that time but it has its own challenges such as this network cannot be used in real-time, because it takes 2-3 seconds to predicts an image and therefore cannot be used in real-time. Whereas, in YOLO we have to look only once in the network i.e., only one forward pass is required through the network to make the final predictions.

YOLO or **You Only Look Once**, is a popular real-time object detection algorithm. YOLO combines what was once a multi-step process, using a single neural network to perform both classification and prediction of bounding boxes for detected objects. As such, it is heavily optimized for detection performance and can run much faster than running two separate neural networks to detect and classify objects separately. It does this by repurposing traditional image classifiers to be used for the regression task of identifying bounding boxes for objects.

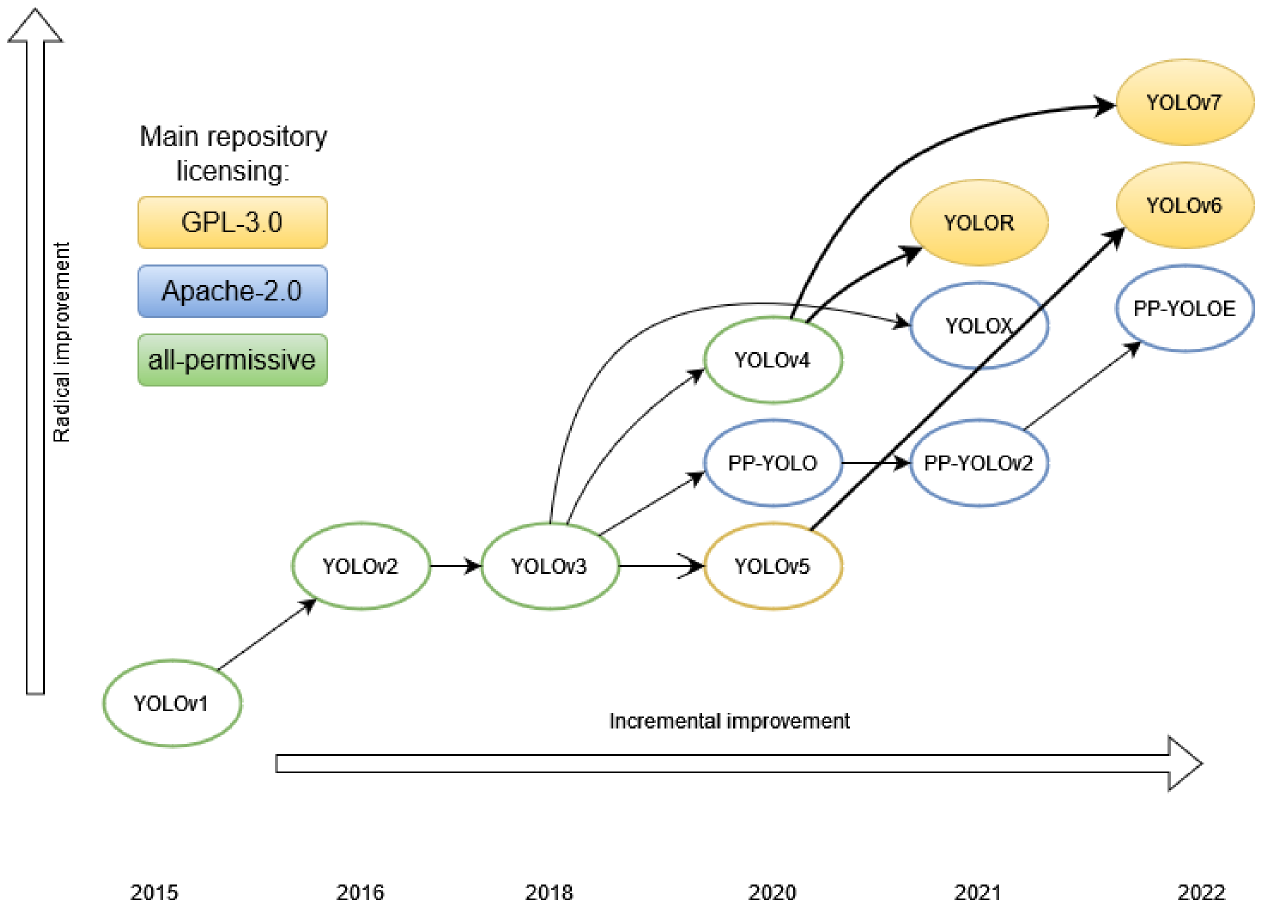


Fig: The evolution of the YOLO neural networks family from v1 to v7.

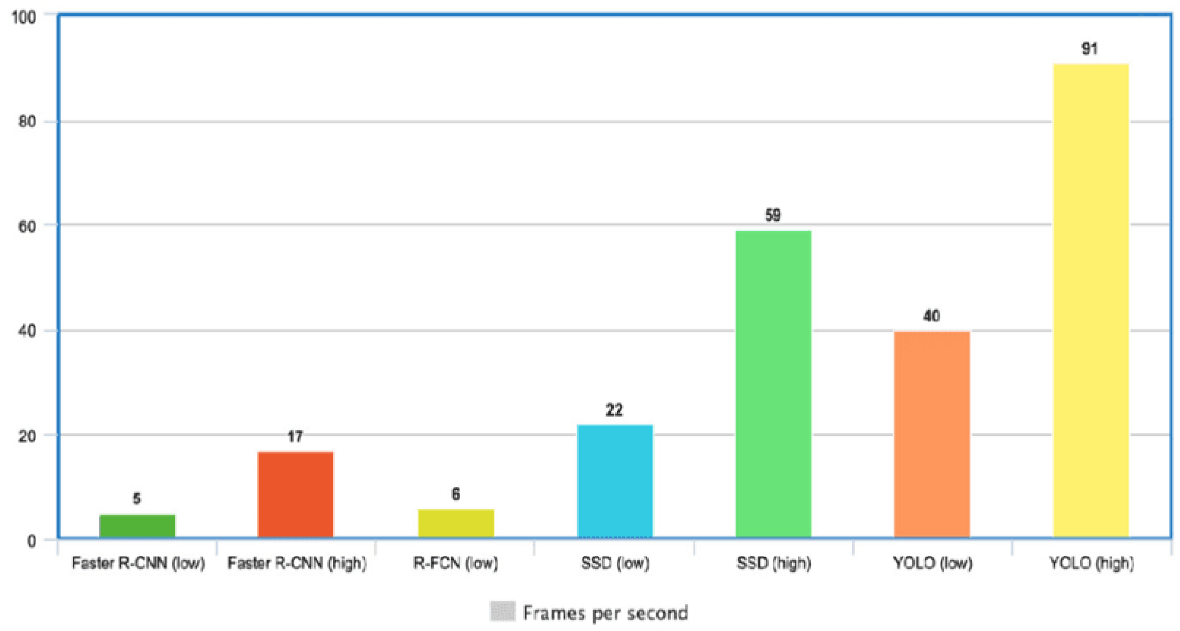
What Makes YOLO Popular for Object Detection?

Some of the reasons why YOLO is leading the competition include its:

- **Speed**
- **Detection accuracy**
- **Good generalization**
- **Open-source**

Speed

YOLO is extremely fast because it does not deal with complex pipelines. It can process images at 45 Frames Per Second (FPS). In addition, YOLO reaches more than twice the Mean Average Precision (MAP) compared to other real-time systems, which makes it a great candidate for real-time processing.



YOLO Speed compared to other state-of-the-art object detectors

High detection accuracy

YOLO is far beyond other state-of-the-art models in accuracy with very few background errors.

Better generalization

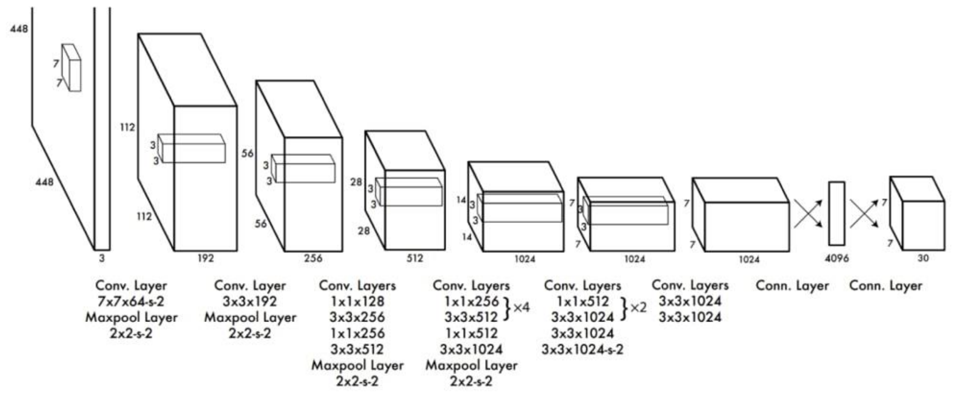
This is especially true for the new versions of YOLO. With those advancements, YOLO pushed a little further by providing a better generalization for new domains, which makes it great for applications relying on fast and robust object detection.

Open source

Making YOLO open-source led the community to constantly improve the model. This is one of the reasons why YOLO has made so many improvements in such a limited time.

How does YOLO work? YOLO Architecture

The YOLO algorithm takes an image as input and then uses a simple deep convolutional neural network to detect objects in the image. The architecture of the CNN model that forms the backbone of YOLO is shown below.



The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

This architecture takes an image as input and resizes it to 448*448 by keeping the aspect ratio same and performing padding. This image is then passed in the CNN network. This model has 24 convolution layers, 4 max-pooling layers followed by 2 fully connected layers. For the reduction of the number of layers (Channels), we use 1*1 convolution that is followed by 3*3 convolution. Notice that the last layer of YOLOv1 predicts a cuboidal output. This is done by generating (1, 1470) from final fully connected layer and reshaping it to size (7, 7, 30).

Detection: This architecture divides the image into a grid of S*S size. If the center of the bounding box of the object is in that grid, then this grid is responsible for detecting that object. Each grid predicts bounding boxes with their confidence score. Each confidence score shows how accurate it is that the bounding box predicted contains an object and how precise it predicts the bounding box coordinates wrt. ground truth prediction.

YOLO Image (divided into S*S grid)

The cell in which the center of an object, for instance, the center of the dog, resides, is the cell responsible for detecting that object. Each cell will predict B bounding boxes and a confidence score for each box. The default for this architecture is for the model to predict two bounding boxes. The classification score will be from `0.0` to `1.0`, with`0.0` being the lowest confidence level and `1.0` being the highest; if no object exists in that cell, the confidence scores should be `0.0`, and if the model is completely certain of its prediction, the score should be `1.0`. These confidence levels capture the model's certainty that there exists an object in that cell and that the bounding box is accurate. Each of these bounding boxes is made up of 5 numbers: **the x position, the y position, the width, the height, and the confidence.** The coordinates ` (x, y) ` represent the location of the center of the predicted bounding box, and the width and height are fractions relative to the entire image size. The confidence represents the IOU between the predicted bounding box and the actual bounding box, referred to as the ground truth box.

The IoU stands for **Intersection Over Union**. IoU is a widely adopted metric for gauging the precision of localization and quantifying localization errors within object detection models. **The IoU calculation involves several steps. Initially, it determines the overlapping region between the predicted bounding box and the ground truth bounding box for the same object.** To calculate the IoU between the predicted and the ground truth bounding boxes, we first take the intersecting area between the two corresponding bounding boxes for the same object. Following this, we calculate the total area covered by the two bounding boxes— also known as the "Union" and the area of overlap between them called the "Intersection."

The intersection divided by the Union gives us the ratio of the overlap to the total area, providing a good estimate of how close the prediction bounding box is to the original bounding box.

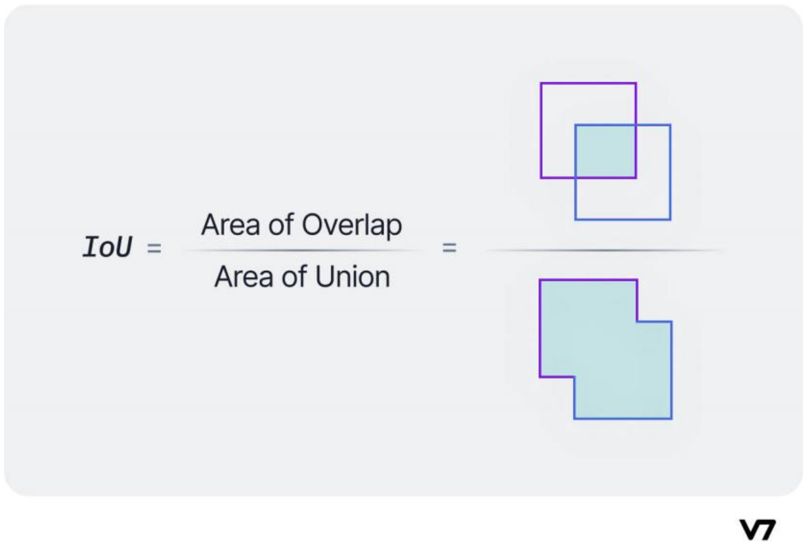
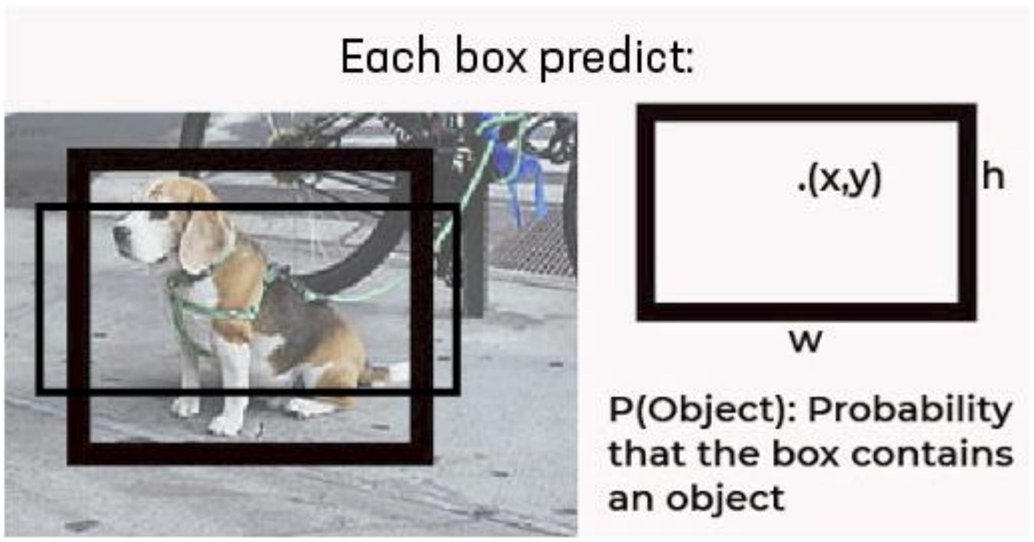
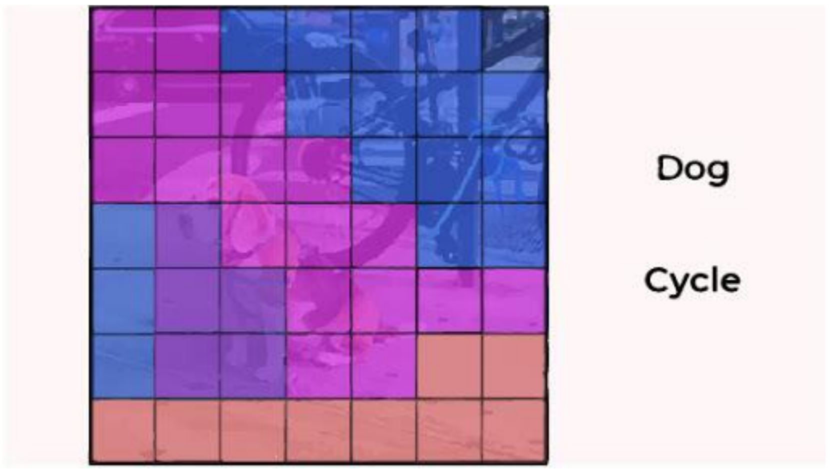


Fig: IoU Calculation compute the IoU by dividing the intersection area (I) by the union area (U)



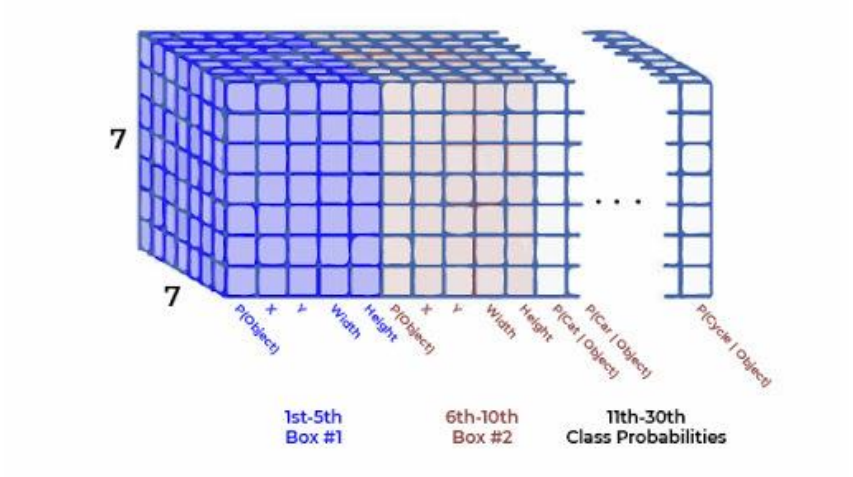
YOLO single Grid Bounding box-Box

This results in combination of bounding boxes from each grid like this.



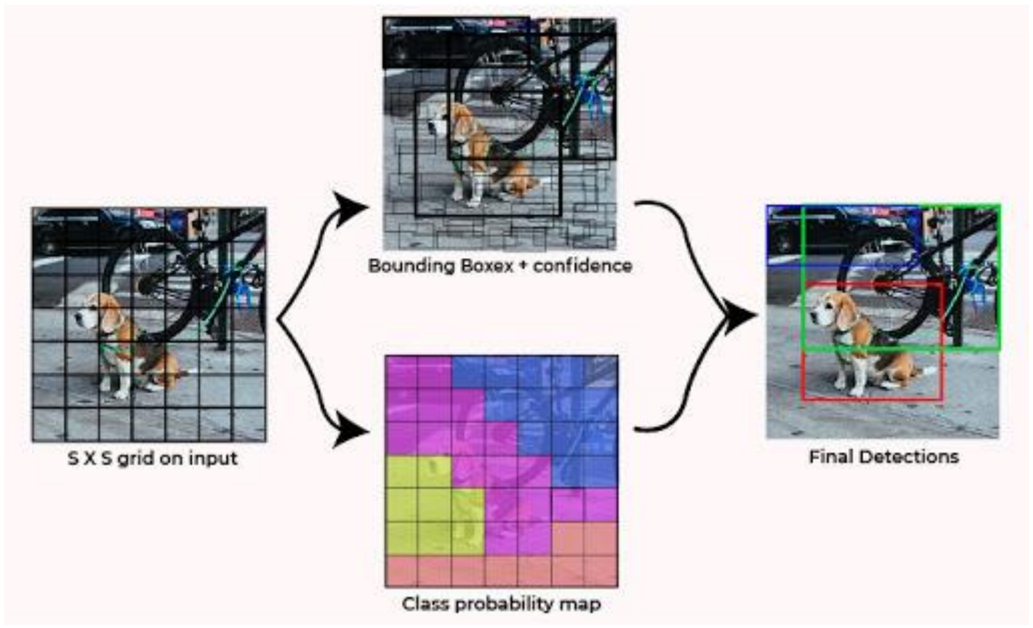
YOLO conditional probability map

This probability was conditional based on the presence of an object in grid cell. Regardless the number of boxes each grid cell predicts only one set of class probabilities. These predictions are encoded in the 3D tensor of size S * S * (5*B +C). Now, we multiply the conditional class probabilities and the individual box confidence predictions,



YOLO output feature map

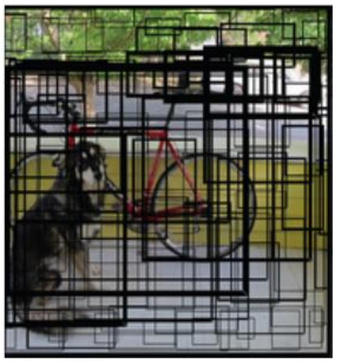
$$P_r\left(Class_i|Object\right) * P_r\left(Object\right) * IOU_{pred}^{truth} = P_r\left(Class_i\right) * IOU_{pred}^{truth}$$

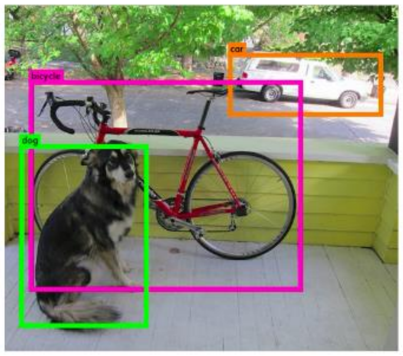


YOLO test results

which gives us class-specific confidence scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object. Then after we apply non-maximal suppression for suppressing the non max outputs (when a number of boxes are predicted for the same object). And at last, our final predictions are generated.

What is non-max suppression?


Multiple Bounding Boxes



Final Bounding Boxes

The objects in the image can be of different sizes and shapes, and to capture each of these perfectly, the object detection algorithms create multiple bounding boxes. (Left image). Ideally, for each object in the image, we must have a single bounding box. Something like the image on the right.

To select the best bounding box, from the multiple predicted bounding boxes, these object detection algorithms use non-max suppression. This technique is used to "suppress" the less likely bounding boxes and keep only the best one.

The goal of the NMS is quite simple: We must "remove" the less likely bounding boxes and keep only the best.

1. Does YOLO actually divide image into S x S grid before processing?

Yes, YOLO (You Only Look Once), specifically YOLOv1 and its subsequent versions, divides an input image into an S x S grid before processing. YOLO divides an input image into an S × S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks the predicted box is.

YOLO predicts multiple bounding boxes per grid cell. At training time, we only want one bounding box predictor to be responsible for each object. YOLO assigns one predictor to be "responsible" for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at forecasting certain sizes, aspect ratios, or classes of objects, improving the overall recall score.

Here's how it works in brief:

- **Grid Division:** YOLO divides the input image into a grid of cells. The number of cells in this grid is determined by the "S" parameter. Each cell is responsible for predicting bounding boxes and class probabilities for objects within its spatial region.

- **Bounding Box Predictions:** In each grid cell, YOLO predicts multiple bounding boxes (usually, four) along with confidence scores. These bounding boxes represent the potential locations of objects within the cell.

- **Class Predictions:** For each bounding box, YOLO also predicts class probabilities to determine the type of object present within that bounding box.

- **Output:** The final output of YOLO is a tensor that combines information from all the grid cells, bounding boxes, and class probabilities. This tensor represents the detections made by the model across the entire image.

This grid-based approach makes YOLO very efficient for real-time object detection because it processes the entire image in one forward pass through the neural network, rather than using sliding windows or region-based proposals like some other object detection methods. It significantly reduces computation time and allows YOLO to deliver fast and accurate object detection results.

2. How is overlapping between bounding boxes measured?

Overlapping between bounding boxes is typically measured using a metric called "Intersection over Union" (IoU), also known as the Jaccard index. IoU is a widely adopted metric for gauging the precision of localization and quantifying localization errors within object detection models. The IoU calculation involves several steps. Initially, it determines the overlapping region between the predicted bounding box and the ground truth bounding box for the same object. This shared area is referred to as the 'Intersection.' Concurrently, it computes the combined area of the two bounding boxes, commonly termed the 'Union.' By dividing the Intersection by the Union, we obtain a ratio that signifies the extent of overlap in relation to the overall area. This ratio provides a valuable approximation of how closely the predicted bounding box aligns with the original bounding box.

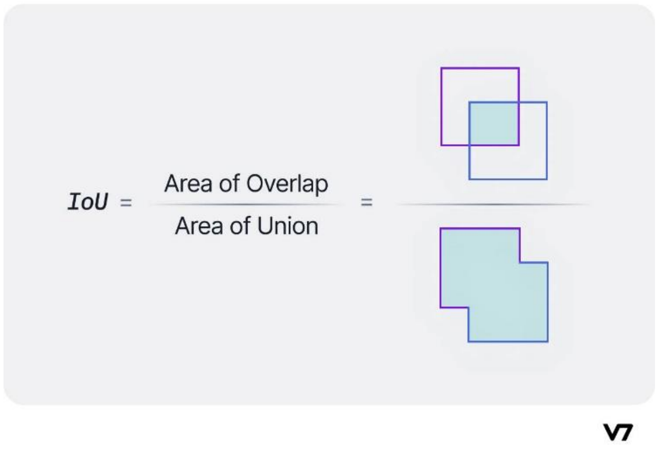
Here's how overlapping is measured in YOLO:

- **Bounding Box Prediction:** YOLO predicts multiple bounding boxes for objects in each grid cell. Each bounding box is characterized by its (x, y) coordinates, width (w), height (h), and confidence score. These bounding boxes represent potential object locations.

- **Intersection Area (I):** For a pair of predicted bounding boxes, calculate the area of intersection (I) between them. This is done by finding the coordinates of the intersection rectangle formed by the overlapping regions of the two bounding boxes.

- **Union Area (U):** Calculate the total area encompassed by both bounding boxes. This involves adding the individual areas of the two bounding boxes and then subtracting the intersection area (I) to avoid double-counting.

- **IoU Calculation:** Compute the IoU by dividing the intersection area (I) by the union area (U):



In YOLO, IoU is used for post-processing steps like non-maximum suppression (NMS). During NMS, predicted bounding boxes with IoU values exceeding a specified threshold (e.g., 0.8) are suppressed to retain only the most confident and non-overlapping detections. This helps ensure that each object is detected only once and reduces redundancy in the final output.

NMS: Non-maximum suppression is a post-processing step in YOLO and many other object detection algorithms. It is used to eliminate redundant or highly overlapping bounding box predictions. During NMS, the following steps are usually performed for each class of objects detected:

- Sort the predicted bounding boxes by their confidence scores (the likelihood that the object is present).
- Start with the highest-confidence bounding box.
- Compare this box to all other remaining boxes.
- If the IoU between the two boxes is above the specified threshold, discard the box with the lower confidence score.
- Repeat this process for all remaining boxes in descending order of confidence.
- The result is a list of non-overlapping bounding boxes, each with its confidence score.

By applying NMS with an IoU threshold, YOLO ensures that only the most confident and non-overlapping detections are retained. This is crucial for reducing redundancy in object detection results and providing a cleaner and more accurate output.

What happens when multiple objects have their centroid in same grid cell?
When multiple objects have their centroids (center points) in the same grid cell in the YOLO (You Only Look Once) algorithm, YOLO is designed to handle such scenarios as follows:

1. **Multiple Bounding Box Predictions:** YOLO predicts multiple bounding boxes for each grid cell, typically four bounding boxes. These bounding boxes are positioned relative to the grid cell, and each bounding box is responsible for potentially detecting an object within that cell.

2. **Confidence Scores:** Along with each bounding box prediction, YOLO assigns a confidence score. This score represents how confident YOLO is that an object exists within that bounding box. It's based on the probability that an object is present and the accuracy of the bounding box.

3. **Class Probabilities:** YOLO also predicts class probabilities for each bounding box, indicating the probability of the detected object belonging to various predefined classes (e.g., "car," "dog," "cycle," etc.).

Non-Maximum Suppression (NMS): After YOLO makes its predictions for all grid cells, it performs a post-processing step called Non-Maximum Suppression (NMS). NMS is applied separately to each class of objects. The goal of the NMS is quite simple: We must “remove” the less likely bounding boxes and keep only the best.

- During NMS, YOLO sorts the bounding box predictions within each class by their confidence scores in descending order.

- It then iterates through these sorted predictions, starting with the one with the highest confidence score.
- For each prediction, it calculates the Intersection over Union (IoU) with all remaining predictions.
- If the IoU between two predictions is above a specified threshold (e.g., 0.5), YOLO retains the prediction with the higher confidence score and discards the one with the lower score.
- This process continues until all predictions have been processed, resulting in a list of non-overlapping, high-confidence bounding boxes for each class.

So, in the scenario where multiple objects have their centroids in the same grid cell, YOLO will predict multiple bounding boxes for that cell, and during the NMS step, it will keep the bounding boxes with the highest confidence scores and eliminate redundant or overlapping predictions. This ensures that only the most confident and non-overlapping detections are retained, even when multiple objects are close to each other within a grid cell.