YOLO(You Only Look Once)

Introduction:

You Only Look Once (YOLO) is a new and faster approach to object detection. Traditional systems repurpose classifiers to perform detection. Basically, to detect any object, the system takes a classifier for that object and then classifies its presence at various locations in the image. Other systems generate potential bounding boxes in an image using region proposal methods and then run a classifier on these potential boxes. This results in a slightly efficient method. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detection, etc. Due to these complexities, the system becomes slow and hard to optimize because each component has to be trained separately.

Why YOLO?

The base model can process images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO can process images at 155 frames per second while achieving double the mAP of other real-time detectors. It outperforms other detection methods, including DPM (Deformable Parts Models) and R-CNN.

How Does It Work?

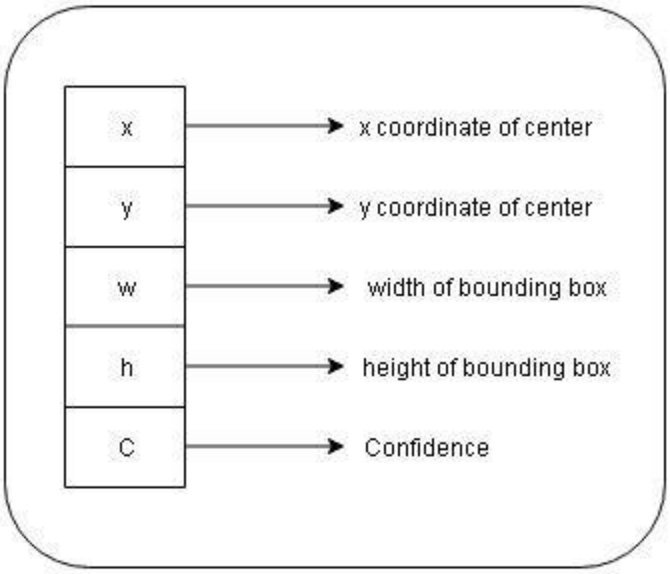
YOLO reframes object detection as a single regression problem instead of a classification problem. This system only looks at the image once to detect what objects are present and where they are, hence the name YOLO.

The system divides the image into an S x S grid. Each of these grid cells predicts B bounding boxes and confidence scores for these boxes. The confidence score indicates how sure the model is that the box contains an object and also how accurate it thinks the box is that predicts. The confidence score can be calculated using the formula:

*C = Pr(object) * IoU*

IoU: Intersection over Union between the predicted box and the ground truth.

If no object exists in a cell, its confidence score should be zero.



Bounding Box Predictions (Source: Author)

Each bounding box consists of five predictions: **x, y, w, h**, and confidence where,

(x,y): Coordinates representing the center of the box. These coordinates are calculated with respect to the bounds of the grid cells.

w: Width of the bounding box.

h: Height of the bounding box.

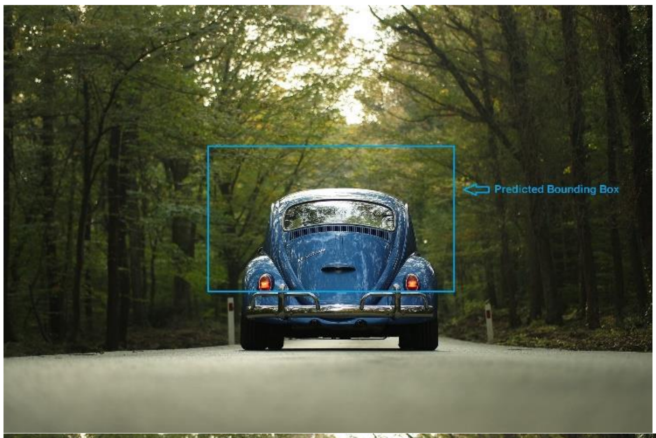
Each grid cell also predicts C conditional class probabilities *Pr(Classi|Object)*. It only predicts one set of class probabilities per grid cell, regardless of the number of boxes B. During testing, these conditional class probabilities are multiplied by individual box confidence predictions which give class-specific confidence scores for each box. These scores show both the probability of that class and how well the box fits the object.

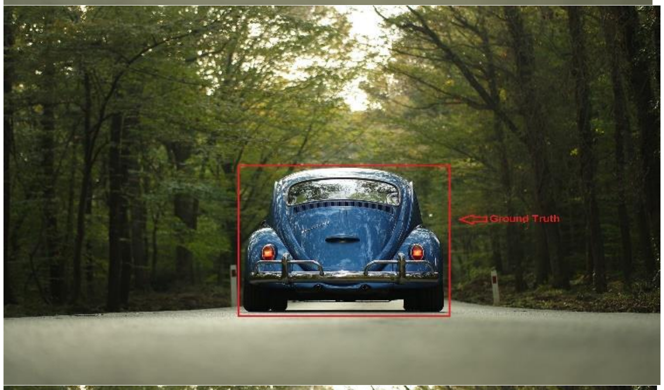
*Pr(Class i|Object)*Pr(Object)*IoU = Pr(Class i)*IoU.*

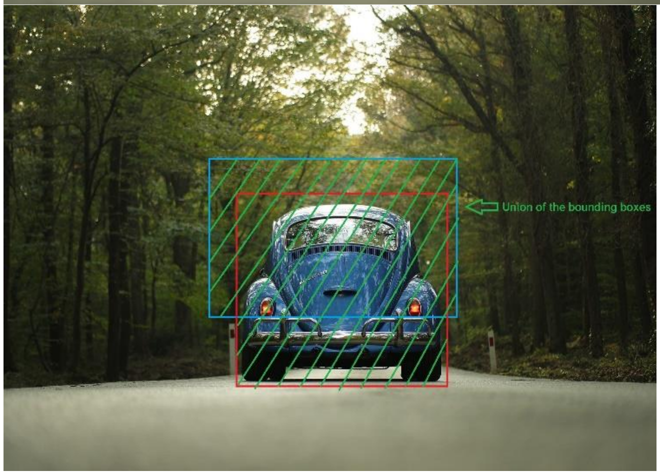
The final predictions are encoded as an S x S x (B*5 + C) tensor.

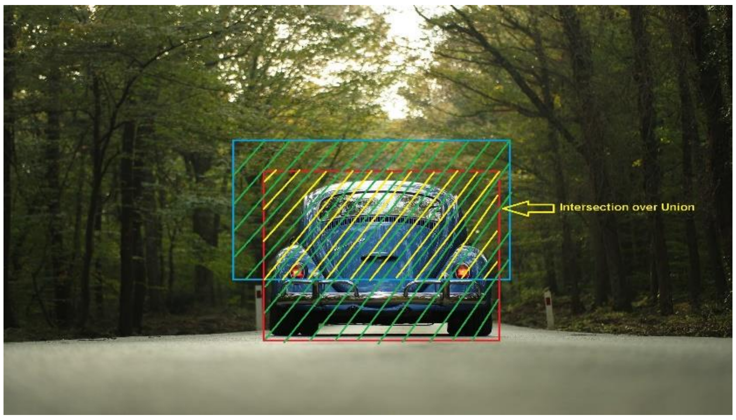
Intersection Over Union (IoU):

IoU is used to evaluate the object detection algorithm. It is the overlap between the ground truth and the predicted bounding box, i.e it calculates how similar the predicted box is with respect to the ground truth.


Predicted Bounding Box


Ground Truth

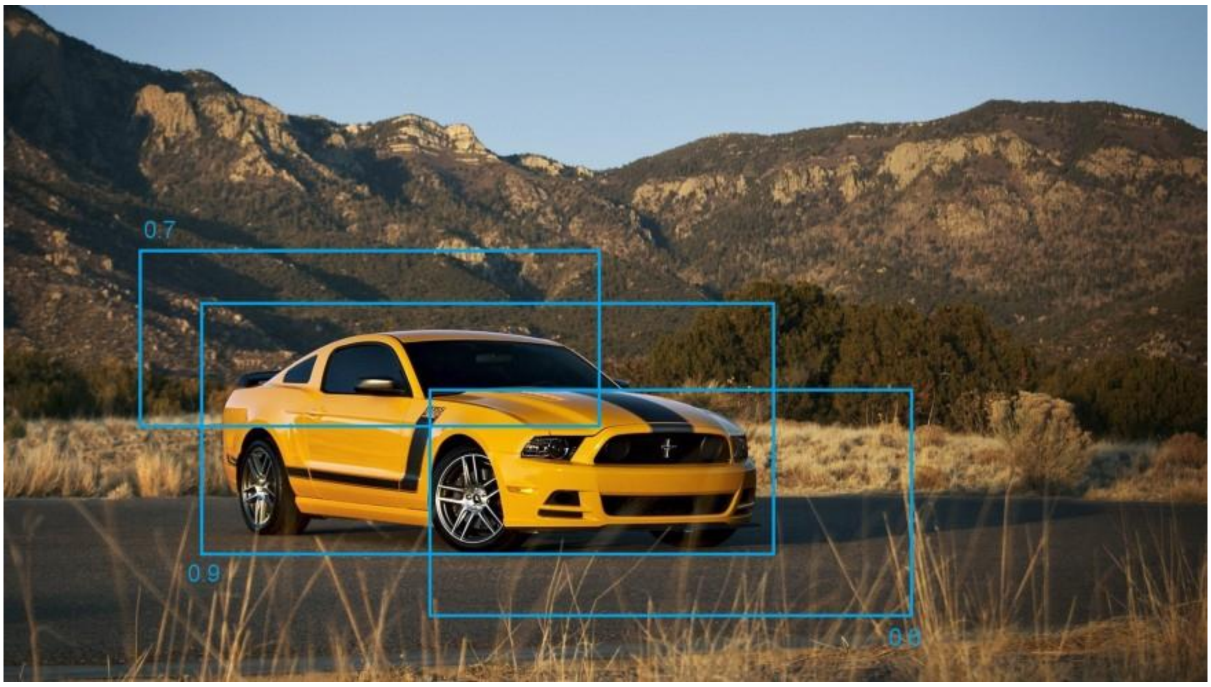

Union of the bounding boxes


Intersection over Union

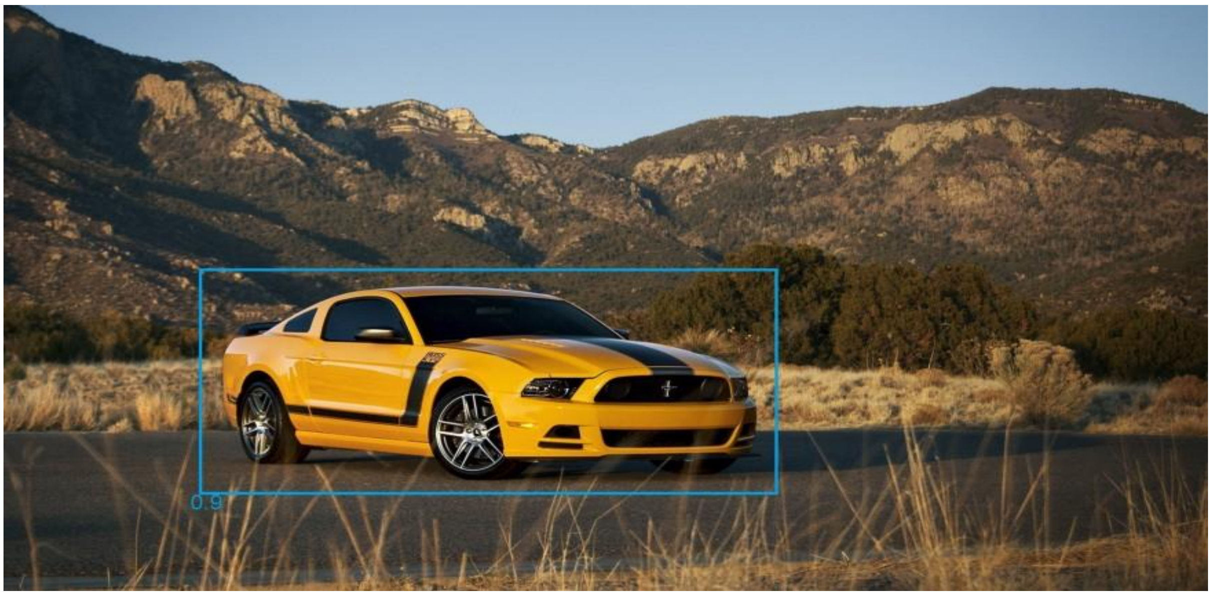
Usually, the threshold for IoU is kept as greater than 0.5. Although many researchers apply a much more stringent threshold like 0.6 or 0.7. If a bounding box has an IoU less than the specified threshold, that bounding box is not taken into consideration.

Non-Max Suppression:

The algorithm may find multiple detections of the same object. Non-max suppression is a technique by which the algorithm detects the object only once. Consider an example where the algorithm detected three bounding boxes for the same object. The boxes with respective probabilities are shown in the image below.

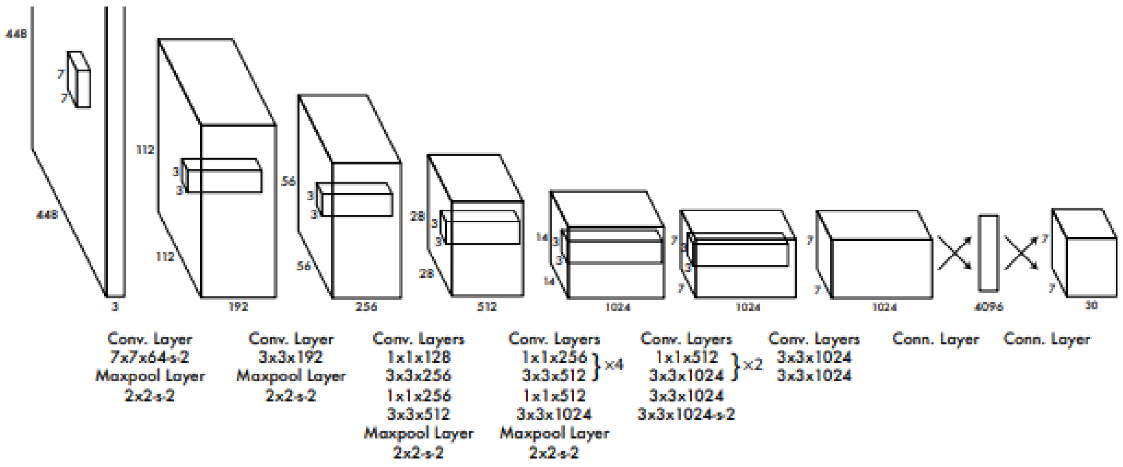


The probabilities of the boxes are 0.7, 0.9, and 0.6 respectively. To remove the duplicates, we are first going to select the box with the highest probability and output that as a prediction. Then eliminate any bounding box with IoU > 0.5 (or any threshold value) with the predicted output. The result will be:



Network Architecture:

The base model has 24 convolutional layers followed by 2 fully connected layers. It uses 1 x 1 reduction layers followed by a 3 x 3 convolutional layer. Fast YOLO uses a neural network with 9 convolutional layers and fewer filters in those layers. The complete network is shown in the figure.



Note:

- The architecture was designed for use in the Pascal VOC dataset, where S = 7, B = 2, and C = 20. This is the reason why final feature maps are 7 x 7, and also the output tensor is of the shape (7 x 7 x (2*5 + 20)). To use this network with a different number of classes or different grid size you might have to tune the layer dimensions.
- The final layer uses a linear activation function. The rest uses a leaky ReLU.

Training:

- Pre train the first 20 convolutional layers on the ImageNet 1000-class competition dataset followed by average — pooling layer and a fully connected layer.
- Since detection requires better visual information, increase the input resolution from 224 x 224 to 448 x 448.
- Train the network for 135 epochs. Throughout the training, use a batch size of 64, a momentum of 0.9, and a decay of 0.0005.
- Learning Rate: For first epochs raise the learning rate from 10–3 to 10–2, else the model diverges due to unstable gradients. Continue training with 10–2 for 75 epochs, then 10–3 for 30 epochs, and then 10–4 for 30 epochs.
- To avoid overfitting, use dropout and data augmentation.

Limitations Of YOLO:

- Spatial constraints on bounding box predictions as each grid cell only predicts two boxes and can have only one class.
- It is difficult to detect small objects that appear in groups.
- It struggles to generalize objects in new or unusual aspect ratios as the model learns to predict bounding boxes from data itself.