YOLO V1

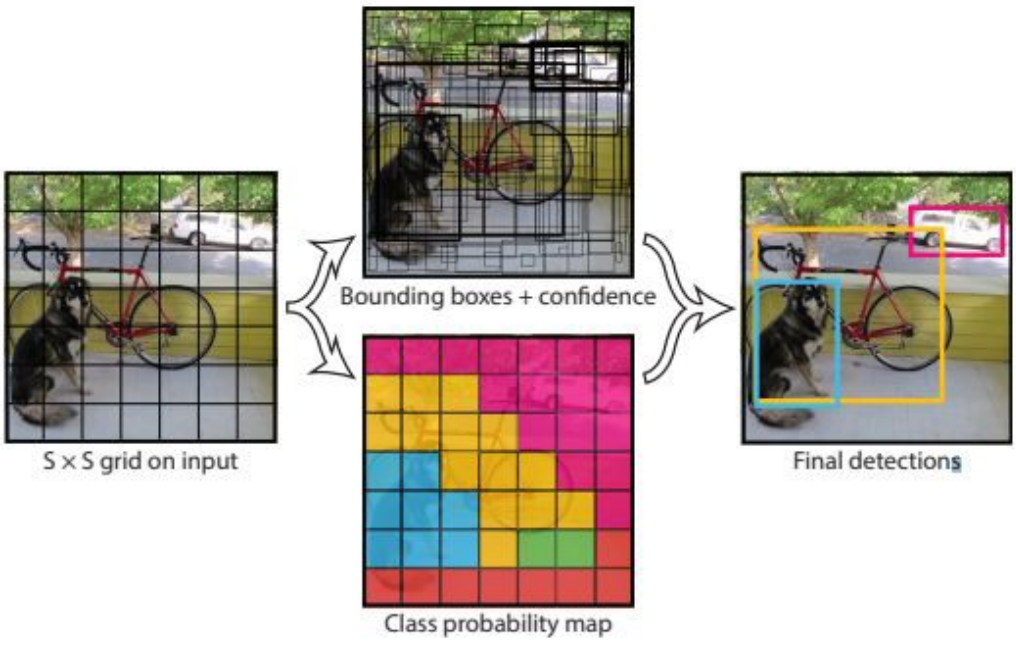
You Look Only Once (YOLO) is a state-of-the art model for object detection task today.

The first version of YOLO architecture came in 2015, where it was presented as the real-time object detection system. Though the accuracy achieved with YOLO is poorer than the already existing architectures but there was a huge leap in the latency of the operation. YOLO is so fast that it can run in real time on a video, as seen in Redmon's (author of YOLO paper) video.

How YOLO work?

As we know that YOLO is a single-stage detector therefore it unifies all the components of object detection into the single neural network.

YOLO divides the whole $S \times S$ grids and each grid will be predicting B boundary boxes. If the centre of an object falls into a grid cell then that cell will be responsible for detecting that particualr object.



- In the left image, YOLO divides the input image into $S \times S$ grid cells.
- In middle top image, each grid cell predicts B bounding boxes and an "objectness score" indicating whether the box contains an object or not.
- In middle bottom image, each grid cell also predicts the class probability if the grid cell contain an object.
- In the right image, it just combined the results of all the intermediate steps to out bounding boxes around the objects.

Design of YOLO

- Each grid cell in YOLO predicts B bounding boxes and a confidence score.
- For each bounding box, YOLO predicts five parameters — $x, y, w, h \; and \; confidence \; score$.
- (x, y) are the centre of the bounding box with respect to the grid cell. The values of x, y are normalised between 0 and 1.
- (w, h) are the width and height of the bounding box relative to the width and height of the whole image.
- An important point to note is that all the coordinates (x, y, h, w) are normalised between 0 and 1, i.e,

$$x, y, h, w \in [0, 1]$$

- $Confidence \; score$ indicates whether the bounding box have an object and how accurate the bounding box is.

$$Confidence \; score = P(object) \times IoU(Between \; predicted \; and \; ground \; truth)$$

- Each grid cell also predicts C (number of classes of objects) class probabilities.
- The class probabilities are conditioned on the presence of object

$$Class \; Probability = P(class|object)$$

- YOLO only predicts one set of C class probabilities per grid cell even though the grid cell has B bounding boxes.

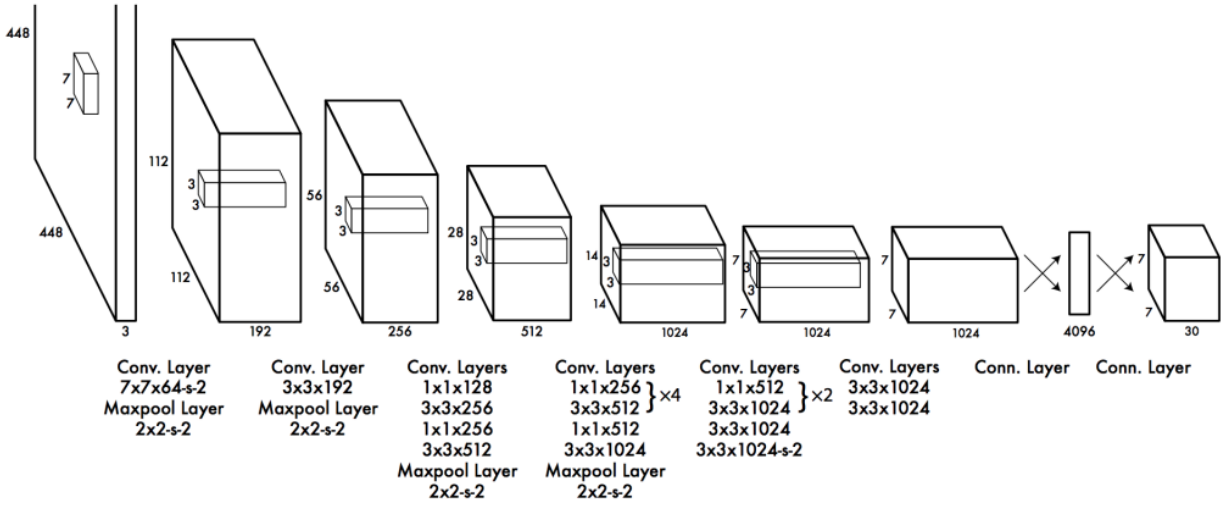
Output tensor shape

- So YOLO predicts $C + (B \times 5)$ parameters for each grid cell
- Total prediction tensor for an image then is

$$S \times S \times (C + B \times 5)$$

- Finally YOLO applies Non Maximum Suppression (**NMS**) and thresholding to report final predictions.

Architecture



- The network has 24 convolutional layers followed by 2 fully connected layers.
- Alternating 1×1 convolutional layers reduce the feature space from preceding layers. [Learn about 1×1 conv layers here]
- It uses the Leaky Relu activation function for all the layers except the last layer, in the last layer it uses linear activation function.
- For optimization, it uses the **Sum of Squared Error**.

Loss Function

The total loss consists of:

1. ***Classification Loss***: It is the sum of squared error between the class probabilities of predicted and actual class of each cell.
2. ***Localisation Loss***: It is the sum of squared error between the coordinates, height, and width of the predicted and actual boxes.
3. ***Confidence Loss:*** It is the sum of squared error of the object confidence score for bounding boxes.

The total loss would be:

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

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

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

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

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

Let's break each component and study them:

Localisation Loss:

▼ Bounding Box centre coordinates loss

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

Calculates the sum-squared loss between the predicted bounding box coordinates and the actual bounding box coordinates.

The loss function only penalizes, if that predictor is responsible for the ground truth box. (Remember, if the centre of the object is inside the cell then that cell is responsible for calculating bounding box for that object, then we calculate IoU for each bounding box for that cell. The bbox with highest IoU is responsible for prediction).

▼ Bounding Box width and height loss

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

Calculate the sum-squared loss in prediction of bounding box width and height.

If the magnitude of error is same for a small bounding box versus a large bounding box, they produce same loss. but the same magnitude of error is more wrong for a small bounding box than a large bounding box, hence, the square root of the values is used to calculate loss.

The loss function penalizes only if that predictor is responsible for the ground truth box

Confidence Loss:

▼ Object confidence loss

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

Calculates the error in prediction of object confidence score for bounding boxes that have an object.

The loss function penalizes only if that predictor is responsible for the ground truth box

▼ No Object confidence loss

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

Calculates the error in prediction of object confidence score for bounding boxes that do not have an object.

The loss function penalizes only if that predictor is responsible for the ground truth box

Classification Loss:

▼ Class Probability loss

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

Calculates the error in the prediction of class probabilities for grid cells that have an object.

The loss function only penalizes class probabilities error, if an object is present in that grid cell. (Remember we conditioned our class probabilities on the presence of object!)

Predictions

During predictions, YOLO predicts multiple boundary boxes for each object. So to remove all the irrelevant boundary boxes and keep the best one, YOLO applies Non-Max Suppression.

Non-Max Suppression

In NMS first we find the bounding box with the highest objectness score, and get rid of all the other bounding boxes that overlap a lot with it. (e.g, with an IoU greater than 60%).

Non-Max Suppression ensures that only the best predicted box for each object will be kept.

Limitations of YOLO v1

1. Fails when we have more than one object inside the grid cell.
2. Not good for small object detections.
3. Has difficulties in detecting objects having unusual aspect ratios.
4. Makes more localization error compared to Fast R-CNN