A simple explanation to the YOLO Algorithm

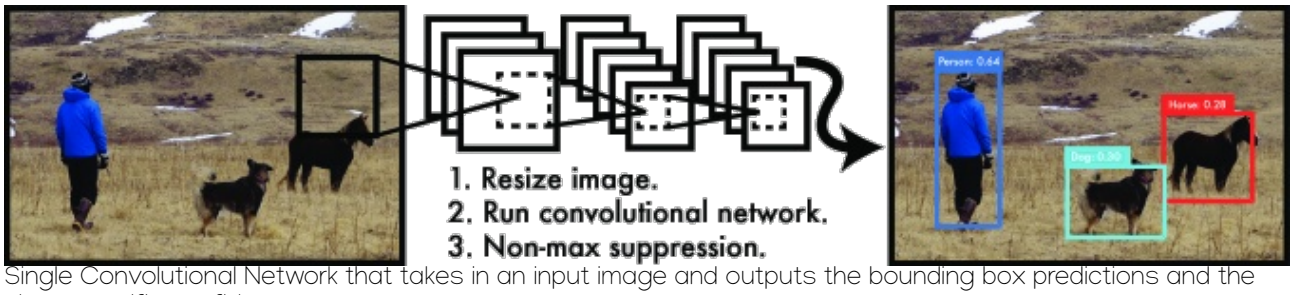
- Shaunak Halbe, 111803053 , Division 1 T-2 Batch

1. Introduction To YOLO V1:

YOLO - stands for You Only Look Once which is a real time yet accurate object detection algorithm proposed by Redmon et. al. It is named so because the YOLO system looks at an image only once and makes final detection and recognition predictions based on it.

Firstly, YOLO is extremely fast at processing an image with the base version running at 45fps and a lighter version at an astounding 150 fps! which is more than the real time expectation.

YOLO re-frames the detection problem as a regression problem as against most of the previously proposed systems which use classifiers to perform detection. YOLO has a simple single flow pipeline which makes it computationally efficient and allows it to achieve the above mentioned real time speed.



Single Convolutional Network that takes in an input image and outputs the bounding box predictions and the class specific confidence scores

Secondly, the YOLO systems focuses less on the pixel values, but learns the shapes, sizes and aspect ratios of objects pretty well this allows it to accurately perform detection on artwork data-sets which are different from natural images at the pixel level. Thus YOLO is said to learn general representations of objects unlike RCNN type models.

Finally, as mentioned above, the entire image is fed at once to the YOLO system which enables it to encode contextual information about the object classes thus making lesser background errors as compared to Fast RCNN which feeds patches/parts of the image to it's system and not the entire image.

2. Detection :

2.1) The YOLO system divides the input image into an S * S grid. If the center of an object lies in that grid cell then that grid cell is responsible to detect that object.

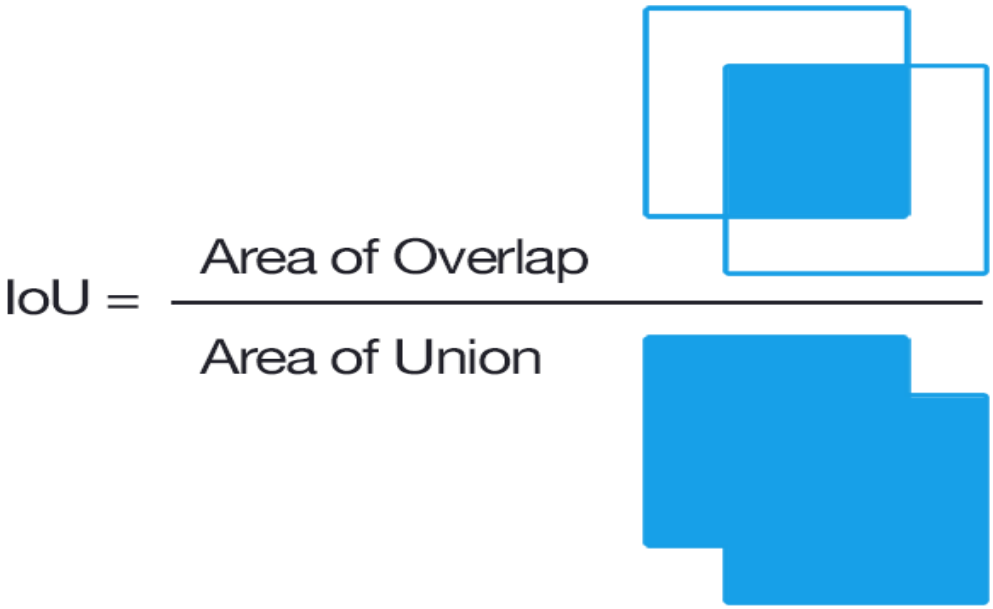
2.2) Each cell predicts B bounding boxes and confidence scores associated with it. The boxes are characterized by:

Height, width relative to the whole image i.e. lying between 0 to 1. Coordinates x and y relative to the bounds of the cell lying between 0 and 1. Confidence Score is given as Pr(Object)* IOU_{pre}.

Pr(Object) is the probability that an object is present in the cell

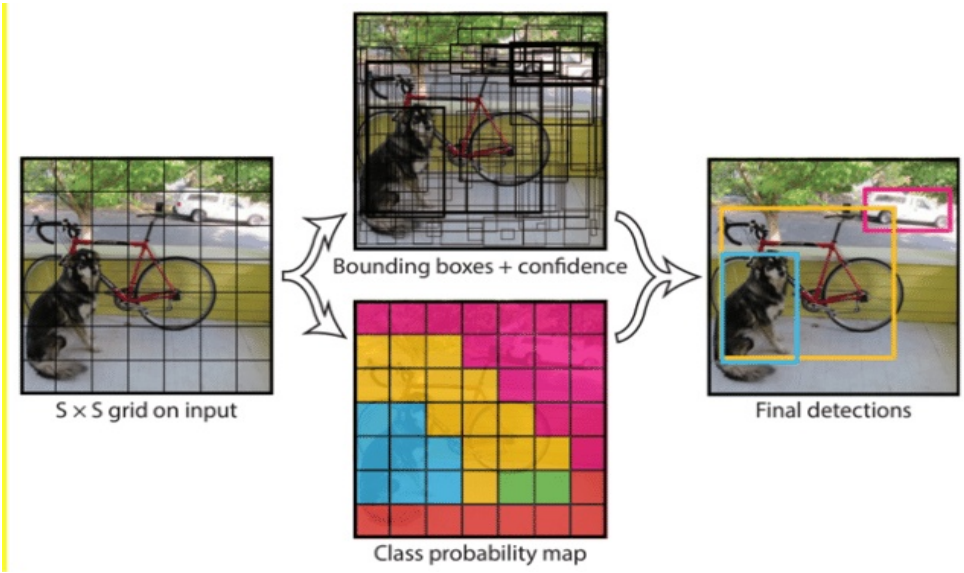
Intersection over Union (IOU) :

It is an evaluation metric used to check the accuracy of the predicted bounding box w.r.t the actual ground truth.



2.3) Each cell also predicts only set of conditional class probabilities which at test time are multiplied with the confidence scores to generate class-wise confidence predictions.

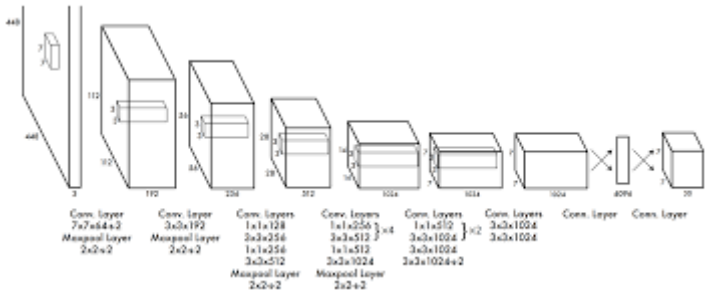
Confidence : $Pr(Object)* IOU_{Pre}*Pr(Class_i / Object) = Pr(Class_i)*IOU_{Pre}$



The output tensor has a size: **S*S*(B*5 + C)** where B is the number of bounding boxes, 5 predictions per box: x,y,h,w,Confidence Score and C class probabilities per cell.

3. Architecture:

The detection network has 24 convolutional layers followed by 2 fully connected layers.



For Pascal VOC data-set, the authors of the paper have chosen S to be 7 and B to be 2 . So, the output size of this network is 7*7*30

4. Training:

The first 20 Conv layers are pre-trained on the Imagenet 1000 class data-set for recognition. The input size used is 224*224 for images.

Now, to perform detection, the input size is increased to 448*448 as detection requires finer visual information.

The last layer predicts the box co-ordinates and class probabilities. Except for the last layer a leaky Relu activation function is used.

5. Loss Function:

A sum- squared error loss function is used as here, detection is treated as a regression task.

A few problems in using a simple squared loss along with their proposed remedies are:

1. Localization and classification errors will have equal weights which is not ideal. So to counter this localization error is multiplied by a λ constant = 5 ,so that the algorithm will try to minimize the squared error due to localization, as if it doesn't then the squared error of localization will get high, and it will get multiplied by 5 which will result in an even higher error that shoots ups the total loss.

2. Confidence Scores of cells having no objects will drop to 0, which will make the model unstable and cause gradient overpowering. To counter this the no -object term in the below given loss function is multiplied by a low valued λ constant such as 0.5 ,thus not penalizing it adequately as compared to the other terms and allowing the predicted confidence scores to slightly deviate from the 0 value.

3. Sum squared error equally weights the errors in small and large boxes. Small error in large boxes is benign but comparable error in small boxes is harmful. To solve this issue, the square roots of the height and width are used in this loss function to reflect the relative deviations.

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

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

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

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

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

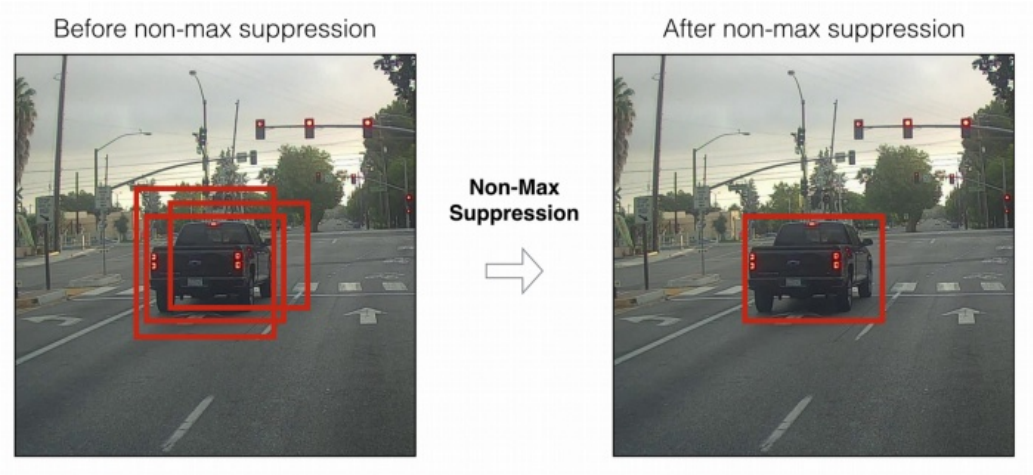
YOLO assigns one predictor responsible for predicting an object based on IOU

The notation used is :

1. $\mathbb{1}_{ij}^{obj}$ denotes if object appears in cell i and bounding box predictor j is responsible for that prediction. Which means it is 1 for such i,j pair and 0 elsewhere. It is a simple squared error loss over all bounding boxes.
2. $\mathbb{1}_{ij}^{noobj}$ denotes if object does not appear in cell i, yet the bounding box predictor j tries to predict the object in the cell no. i.
3. $\mathbb{1}_{i}^{obj}$ denotes if object is present in cell i.

6. Inference:

As a post processing method Non Max suppression is used, which is used to suppress multiple bounding boxes predicting the same object except the one which has the highest IOU score.



Limitations of YOLO:

1. YOLO divides the image into discrete cells and limits the number of object and bounding box predictions in a particular cell, which imposes spatial constraints making it difficult for the algorithm to predict closely spaced objects like birds in a flock etc.

2. YOLO does not use any pre-processing method for generating candidate regions for objects, instead it learns to predict boxes in the same training pipeline. This limits the algorithm to generalize to new settings.
3. Due to many down-sampling and aggregation layers in the network, it learns coarse image features as compared to RCNN,SSD models.
4. It struggles to correctly localize bounding boxes.

Even with these limitations, YOLO is a very popular algorithm used in video object detection, and several other practical applications due to its real time performance and computational efficiency.

References:

1. Joseph Redmon , Santosh Divvala, Ross Girshick , Ali Farhadi .You Only Look Once: Unified, Real-Time Object Detection. In CVPR

2. R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014

Image References:

1. Joseph Redmon , Santosh Divvala, Ross Girshick , Ali Farhadi .You Only Look Once: Unified, Real-Time Object Detection. In CVPR.

2. Google Images