Yolo Algorithm:

The first thing, YOLO means You Only Look Once.
Fastest Algorithm for detection and classification of objects in real world scenarios.

But first my understanding of the field of object detection. When it comes to object detection I can divide it into 3 parts :- Image Classification, Object Classification with Localisation and Multiple object detection.

In Object Classification with Localisation for a single object :-
We have a basic image and we need to say whether it is a car or not, we can easily say we will use CNN and then at the end a vector of features will be produced. We then put it through a Softmax function for classification.

But now for Localization, we need to output a vector of values which will “Bound” the image of the car. The values we will need are, the centre coordinates of the object, and the required width and length of the box that will bound the object. Along with that we can also add a variable whether it is a car or anything else.

Based on the project given to me, We have to use the YOLO algorithm.

How I chalked out the problem:
Let’s say we have an image of dimensions 100 X 100 X 3 (Basic RGB image). This will be the input.

The expected output will be a vector = [p, bx, by, bh, bw, c].
Where:
p: probability/confidence that the object is in the block/cell
bx, by : the centre point of the object in the image
bw, bh : expected height and width of the bounding box
c : the probability of the c being the car category(or any required category)

The convolutional layers of the YOLO Algorithm, encodes the image into (n, n, 2, 85). The (n,n) means the no. of cells the picture is divided into. 2 is used as the number of anchors, the car and the traffic, and 85 because, in the cocomo dataset there are already 80 categories and 5 is added because the vector [p, bx, by, bh, bw] is included.

Now each cell will have the output of [p, bx, by, bh, bw, c], where c is the vector of probability of 80 classes.

Each cell is given a “score”.
Score is basically, the probability of having the car and the cell is let’s say cell x. So for cell x we have the vector = [Px, bx, by, bh, bw, c].
Px : The probability of the car might exist in the cell 1
c : The probability of the category being car

Based on the anchor boxes and the scores dedicated to the category, we can visualise the portion of the actual image the model is predicted.

Now using the central coordinates and the height and width of the bounded box, we can find the bounding boxes.

But this will also result in many bounded boxes, to reduce the number of boxes we need to use the concept of Intersection over Union(IoU).
After that also, we would still have some boxes that might overlap, that's why we need to set thresholds on the values, based on which we should discard the boxes. Let the threshold value for first IoU implementation is 0.6, any box having value below that will be removed and then we would implement the concept of Non max suppression, where the box with the higher px will be selected and the boxes whose IoU with these are greater than 0.5 will discarded, as these boxes will be same as the one we selected.

This is the concept I have understood, until now.

The coding part is still a mystery to me on how to start working, and what sets of images to use for training. Also about the libraries, I have a rough idea but I haven’t done any project that required such extensive usage of the code or concept.

I know about yad2k file and we can use that for making our boxes and thresholding and also rebuild the images maybe along with the bounded boxes.

Other than this, I took the liberty to go deeper and find other algorithms that might be used in object detection, rather I was looking for the answer why YOLO is used so much and is said to be the fastest, but why? The approach very brute force as each of the grids are needed to be analysed.

So I have found the various concepts of : RCNN and Fast RCNN that can also be used as Object detectors.

For RCNN the output of the CNN was a 4,096 element vector that describes the contents of the image that is fed to a linear SVM for classification, specifically one SVM is trained for each known class.

Fast RCNN:

A Fast R-CNN network takes as input an entire image and a set of object proposals. The network first processes the whole image with several convolutional (conv) and max pooling layers to produce a conv feature map. Then, for each object proposal a region of interest (RoI) pooling layer extracts a fixed-length feature vector from the feature map. Each feature vector is fed into a sequence of fully connected (fc) layers that finally branch into two sibling output layers: one that produces softmax probability estimates over K object classes plus a catch-all “background” class and another layer that outputs four real-valued numbers for each of the K object classes. Each set of 4 values encodes refined bounding-box positions for one of the K classes.

Research Papers referred :

1. You Only Look Once: Unified, Real-Time Object Detection

   Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi

2. YOLO9000: Better, Faster, Stronger

   Joseph Redmon, Ali Farhadi