Vimal Patel

Object Detection Model:

Object detection is a Task of detecting instances of objects of a certain class within an image

Introduction:

The earlier approach was known as sliding window method where the image was converted into grid then as the name suggests the algorithm checks every grid by sliding one by one through the image checking for the class to be predicted, then the output matrix is fed to the algorithm, this process is slow as every grid need to be predicted which makes it time consuming.

Next, an improvement was made the region of proposal method was introduced where in the probability of finding the output class is much greater than the other and this image snippets are fed to the algorithm to make predictions

To make the object detection model more and more fast further improvements were made where the whole image is fed to the network and output is predicted.

Now let us look more closely into it

The main focus will be YOLO and faster RCNN.

The methods can be divided into two main types:

1.One-stage

- YOLO
- SSD
- Retina Net
- YOLOv3

•YOLOv4

•YOLOR

2.Two-stage

•RCNN

•SPPNET

•Fast RCNN

•Faster RCNN

•Mask R-CNN

•Pyramid Networks

•G-RCNN

Now firstly we shall know the difference between one and two stage One-Stage:

In general, deep learning-based object detectors extract features from input image or video frame this is done by object detectors by solving two subsequent tasks

#1: firstly, is to find an arbitrary number of objects even zero

#2: And then Classify every single object and estimate its size with bounding boxes.

We can either separate these two tasks and call it two-stage or combine both into one for higher performance at a cost of accuracy

Now, in one stage detectors predict bounding boxes over an image without region proposal step (you will know what is region proposal step in RCNN explanation below) the main advantage of one step over two step algorithms it consumes less time, also can be used in real time processing.

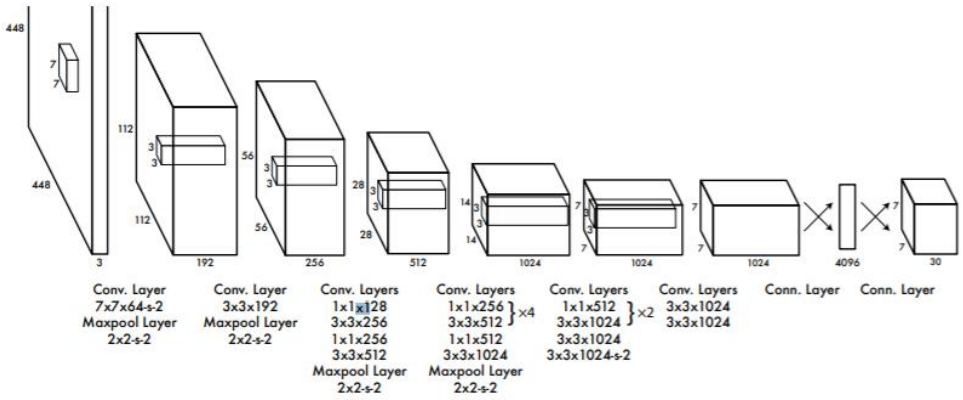
Two-Stage:

In two-stage architecture involves (1) object region proposal with deep networks, then (2) object classification based on features extracted from the region with bounding box regression.

One thing about two stage is they are slower compared to one stage but have good accuracy

Yolo (You only look once):

YOLO was proposed by Joseph Redmond *et al.* in 2015. 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.



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. Sometimes the objects fall outside the grid which is not a problem unless it falls under one grid that grid is responsible for prediction of that bounding box. Each confidence score shows how accurate it is that the bounding that predicts contains an object and how precise it predicts the bounding box coordinates w.r.t. ground truth prediction.

At the time of prediction, we multiply the class probabilities with the individual box confidence predictions

Each bounding box consists of 5 predictions: *(x, y, w, h)* and confidence score.

If each grid has one bounding box, and only one class to be predicted then the augment y would be y:

pc
x
y
h
w
c1

If there are two bounding boxes and two class to be predicted then y would be y:

(This is for yolo v2 where 5 classes can be predicted)

pc
x
y
h
w
c1
c2

The *(x, y)* coordinates represent the center of the box relative to the bounds of the grid cell.

The *h, w* coordinates represent height, width of bounding box relative to *(x, y)*.

The confidence score represents the presence of an object in the bounding box.

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

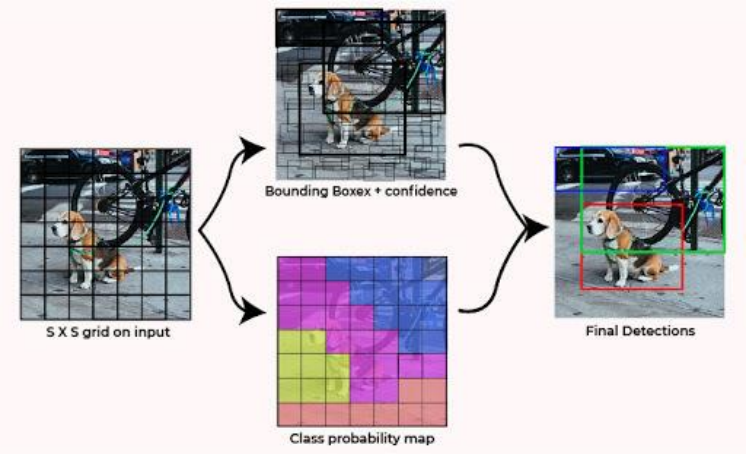
Each grid also predict PC conditional class probability

This probability was conditional based on the presence of an object in grid cell.

After this comes the part where we measure the performance

IoU: measures the overlap between two bounding boxes.

What if same object is being detected multiple times this can be solved by **Non-max suppression** where box with value lower than threshold value of IoU box(i.e,IoU>=0.5) is not used.



YOLO is very fast at the test time because it uses only a single CNN architecture to predict results.

YOLOv2

YOLOv2 was proposed to fix YOLO's main issues—the detection of small objects in groups and the localization accuracy.

YOLOv2 increases the mean Average Precision of the network by introducing batch normalization.

A much more impactful addition to the YOLO algorithm, as proposed by YOLOv2, was the addition of anchor boxes.

YOLO predicts a single object per grid cell.

it creates issues when a single cell has more than one object, as YOLO can only assign a single class to the cell.

YOLOv2 gets rid of this limitation by allowing the prediction of multiple bounding boxes from a single cell. This is achieved by making the network predict 5 bounding boxes for each cell.

YOLO9000

YOLO9000 similar network architecture as YOLOv2,

YOLO9000 was proposed as an algorithm to detect more classes as the name suggests it detects 9000 classes

While YOLO9000 provides a lower mean Average Precision(mAP) as compared to YOLOv2, it is capable of detecting more than 9000 classes, making it a powerful algorithm.

YOLOv3

Although YOLOv2 is a superfast network, various alternatives that offer better accuracies—like Single Shot Detectors—have also entered the market. Although much slower in performance, they outstrip YOLOv2 and YOLO9000 in terms of accuracy.

YOLOv3 was proposed to improve YOLO with modern CNN that make use of residual networks and skip connections

YOLOv3 uses a much more complex DarkNet-53 as the model backbone

YOLOv3 predicts 3 bounding boxes per cell (compared to 5 in YOLOv2) but it makes three predictions at different scales, totaling up to 9 anchor boxes.

YOLOv4

It proposes the addition of Weighted Residual Connections, Cross Mini Batch Normalization, Cross Stage Partial Connections, Self-Adversarial Training, and Mish Activation as methodological changes amongst modern methods of regularization and data augmentation.

A YOLOv4 Tiny version that provides faster object detection and a higher FPS while making a compromise in the prediction accuracy.

YOLOv5

YOLOv5 Backbone employs CSPDarknet as the backbone for feature extraction from images consisting of cross-stage partial networks.

YOLOv5 Neck uses PANet to generate a feature pyramids network to perform aggregation on the features and pass it to Head for prediction.

YOLOv5 Head Layers that generate predictions from the anchor boxes for object detection.

It uses below choices for training –

Activation and Optimization: YOLOv5 uses leaky ReLU and sigmoid activation, and SGD and ADAM as optimizer options.

Loss Function: It uses Binary cross-entropy with logits loss.

R-CNN (Region based Convolution Neural Network (CNN))

Convolution Neural Network (CNN) with a fully connected layer is not able to deal with the frequency of occurrence and multi objects.

Ross Giesick et al.in 2013 proposed an architecture called R-CNN (Region-based CNN) to deal with this challenge of object detection.

R-CNN architecture uses the Region Proposal that generates approximately 2000 region proposals. These 2000 region proposals are then provided to CNN architecture that computes CNN features.

These features are then passed in an SVM model to classify the object present in the region proposal. An extra step is used to perform a bounding box regressor to localize the objects present in the image more precisely.

R-CNN Algorithm flow:

1.CNN for feature extraction

2.Linear SVM classifier for identifying objects

3.Regression model for tightening the bounding boxes.

All these processes combine to make RCNN very slow. It takes around 40-50 seconds to make predictions for each new image, which essentially makes the model cumbersome and practically impossible to build when faced with a gigantic dataset.

Fast RCNN

To overcome this fast RCNN is introduced

the CNN just once per image and then finding a way to share that computation across the 2,000 regions.

In Fast RCNN, we feed the input image to the CNN, which in turn generates the convolutional feature maps. Using these maps, the regions of proposals are extracted.

We then use a RoI pooling layer to reshape all the proposed regions into a fixed size, so that it can be fed into a fully connected network.

But even Fast RCNN has certain problem areas. It also uses selective search as a proposal method to find the Regions of Interest, which is a slow and time-consuming process. It takes around 2 seconds per image to detect objects.

Faster RCNN

Faster RCNN is the modified version of Fast RCNN. The major difference between them is that Fast RCNN uses selective search for generating Regions of Interest, while Faster RCNN uses “Region Proposal Network”, aka RPN.

With help of RPN we find the area where the object can possibly be found we will that area as foreground class whereas the remaining part is named as background class the foreground moves further in the algorithm the RPN takes image feature maps as an input and generates a set of object proposals, each with an object ness score as output.

The below steps are typically followed in a Faster RCNN approach:

1.an image as input and pass it to the ConvNet/RPN which returns the feature map for that image.

2.A RoI pooling layer is applied on these proposals to bring down all the proposals to the same size.

3.Finally, the proposals are passed to a fully connected layer which has a softmax layer and a linear regression layer at its top, to classify and output the bounding boxes for objects.

Anchor boxes are nothing but some reference boxes of different sizes (since not all objects are of same sizes) placed at different positions in the image. k anchor boxes are generated for each pixel in our feature map (output of CNN). Thus, the total

number of anchor boxes is h*w*k (h*w is the output size of the feature map). k here is a hyperparameter.

RPN predicts two things:

- The probability that an anchor is an object (it does not consider which class the object belongs to)
- And the bounding box regressor for adjusting the anchors to better fit the object

We now have bounding boxes of different shapes and sizes which are passed on to the RoI pooling layer. Now it might be possible that after the RPN step, there are proposals with no classes assigned to them. We can take each proposal and crop it so that each proposal contains an object.

RoI pooling layer does. It extracts fixed sized feature maps for each anchor:

Then these feature maps are passed to a fully connected layer which has a softmax and a linear regression layer. It finally classifies the object and predicts the bounding boxes for the identified objects.

Cons:

- The algorithm requires many passes through a single image to extract all the objects
- As there are different systems working one after the other, the performance of the systems further ahead depends on how the previous systems performed

References

R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms | by Rohith Gandhi | Towards Data Science

YOLO : You Only Look Once - Real Time Object Detection - GeeksforGeeks