YOLO Algorithm Based Real-Time Object Detection

Priya Kumari[1], Sonali Mitra[2], Suparna Biswas[3], Sunipa Roy[4], Sayan Roy Chaudhuri[5], Antara Ghosal[6], Palasri Dhar[7], Anurima Majumder[8]

[1,2,3,4,5,6,7,8]*Department of Electronics and Communication Engineering, Guru Nanak Institute of Technology, 157/F, Nilgunj Road, Panihati, Sodepur, Kolkata-700114, West Bengal, India*

***Abstract* - The main objective of Real time object detection is to find the location of an object in a given picture accurately and mark the object with the appropriate category. In this paper we have used real time object detection You Look Only Once (YOLO) algorithm to train our machine learning model. YOLO is a clever neural network for doing object detection in real time and with the help of COCO Dataset the algorithm is trained to identify different objects in a particular image. After training this technique detect the object in real time with 90% accuracy.**

***Index Terms* - YOLO, Object Detection, Neural Network, Bounding Boxes, OpenCV.**

I.INTRODUCTION

The goal of object detection is to detect all instances of objects from a known class, such as people, cars or faces in an image. Typically, only a small number of instances of the object are present in the image, but there are a very large number of possible locations and scales at which they can occur and that need to somehow be explored. Each of the detection is reported with some form of pose information. This could be as simple as the location of the object, a location and scale, or the extent of the object defined in terms of a bounding box. In other situations, the pose information is more detailed and contains the parameters of a linear or nonlinear transformation. For example, a face detector may compute the locations of the eyes, nose and mouth, in addition to the bounding box of the face. An example of a bicycle detection that specifies the locations of certain parts is shown in Figure 1. The pose could also be defined by a three-dimensional transformation specifying the location of the object relative to the camera.

Object detection systems construct a model for an object class from a set of training examples. In the case of a fixed rigid object only one example may be needed, but more generally multiple training examples are necessary to capture certain aspects of class variability.

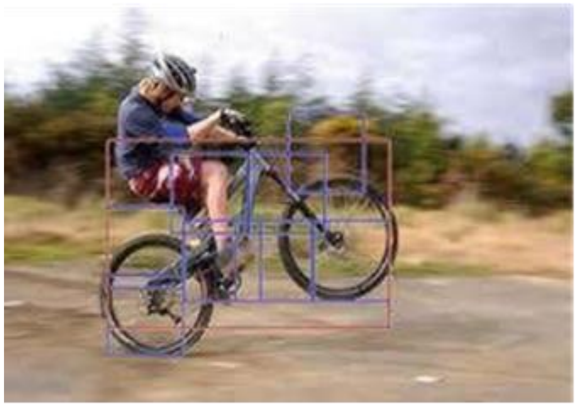

Fig. 1 – Bicycle detection that specifies the locations of certain parts

II. PROPOSED METHODOLOGY

The first method is:

A. YOLO (You Look Only Once)

You can take a classifier like VGG Net or Inception and turn it into an object detector by sliding a small window across the image. At each step you run the classifier to get a prediction of what sort of object is inside the current window.

Using a sliding window gives several hundred or thousand predictions for that image, but you only keep the ones the classifier is the most certain about. This approach works but it's obviously going to be very slow, since you need to run the classifier many times.

A slightly more efficient approach is to first predict which parts of the image contain interesting information — so-called region proposals — and then run the classifier only on these regions. The classifier has to do less work than with the sliding windows but still gets run many times over.

YOLO takes a completely different approach. It is not a traditional classifier that is repurposed to be an object detector. YOLO actually looks at the image just once (hence its name: You Only Look Once) but in a clever way. YOLO divides up the image into a grid of 13 by 13 cells as shown in Figure2.

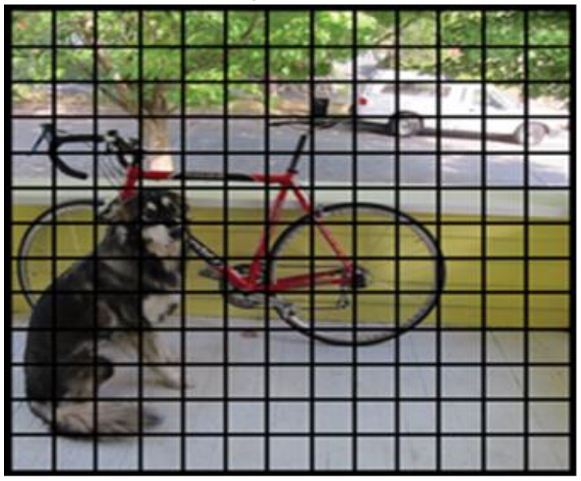

Fig. 2 YOLO divides up the image into a grid of 13 by 13 cells

Each of these cells is responsible for predicting 5 bounding boxes. A bounding box describes the rectangle that encloses an object. YOLO also outputs a confidence score that tells us how certain it is that the predicted bounding box actually encloses some object.
This score does not say anything about what kind of object is in the box, just if the shape of the box is any good. The predicted bounding boxes may look something like the following (the higher the confidence score, the fatter the box is drawn in Fig. 3):

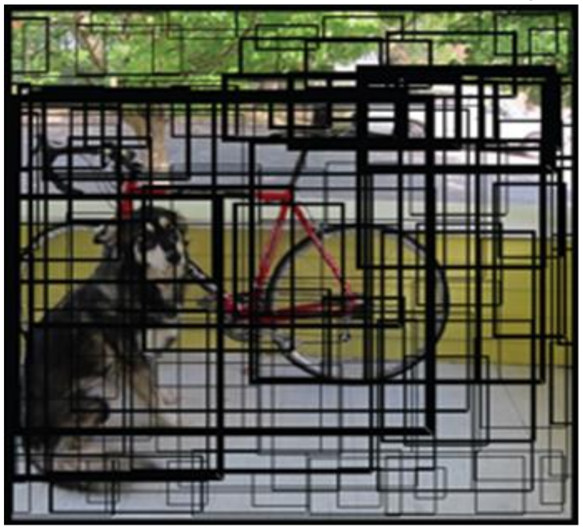

Fig. 3 the higher the confidence score, the fatter the box is drawn

For each bounding box, the cell also predicts a class. This works just like a classifier: it gives a probability distribution over all the possible classes. The version of YOLO we are using is trained on the PASCAL VOC dataset, which can detect 20 different classes such bicycle, boat, car, dog, person, and so on. The confidence score for the bounding box and the class prediction are combined into one final score that tells us the probability that this bounding box contains a specific type of object. For example, the big fat yellow box (Fig.4) on the left is 85% sure it contains the object "dog":

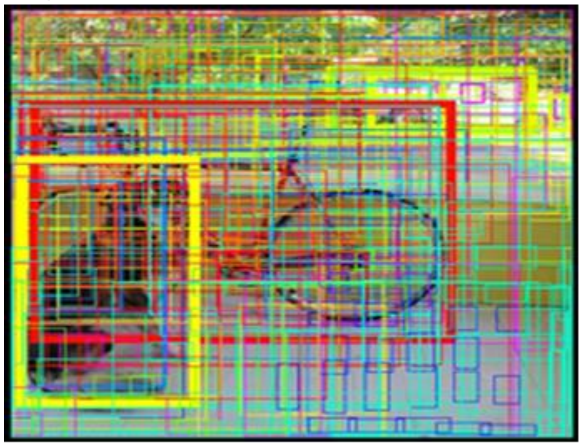

Fig. 4 For example, the big fat yellow box on the left is 85% sure it contains the object "dog"

Since there are 13×13 = 169 grid cells and each cell predicts 5 bounding boxes, we end up with 845 bounding boxes in total. It turns out that most of these boxes will have very low confidence scores, so we only keep the boxes whose final score is 30% or more (you can change this threshold depending on how accurate you want the detector to be).The final prediction is then:

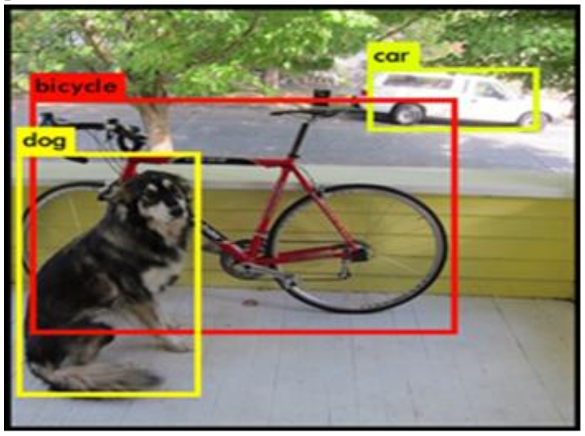

Fig.5. Final detected objects

From the 845 total bounding boxes we only kept these three because they gave the best results. And that's why YOLO is so powerful and fast.
The second method is:

B. COCO DATASET

One of the most important tasks in computer vision is to label the data. There are several tools available where you can load the images, label the objects using per-instance segmentation. This aids in precise object localization using bounding boxes or masking using polygons. This information is stored in annotation files. Annotation file/files can be in COCO. COCO is large scale images with Common Objects in Context (COCO) for object detection, segmentation, and captioning data set. COCO has 1.5 million object instances for 80 object categories.

III. RESULTS

In this program we used the YOLO algorithm to train our machine learning model. YOLO is a machine learning model from Google which was design to work best with dark net framework in 2016 but later it was made compatible to work with OpenCV which we used in this project. To understand the workings of this project we first need to understand how yolo algorithm actual work. So, with the help of the webcam of our laptop it takes image every second and passes those images to the YOLO algorithm. After the image, the algorithm is trained to identify different objects in that particular image with the help of Coco dataset. Coco datasets contain names and data of every object which the algorithm uses to train and learn from. The program takes image every second and throws that image to the algorithm and after getting trained the algorithm slowly learns to detect any object in real time and give out the result with 90% accuracy. You can also enhance the accuracy and differentiate wide variety of object depending on the size of your dataset and the processing power of your laptop.

IV. CONCLUSION AND FUTURE SCOPE

Object detection is a key ability for most computer and robot vision system. Although great progress has been observed in the last years, and some existing techniques are now part of many consumer electronics (e.g., face detection for auto-focus in smart phones) or have been integrated in assistant driving technologies, we are still far from achieving human-level performance, in particular in terms of open-world learning. It should be noted that object detection has not been used much in many areas where it could be of great help. As mobile robots, and in general autonomous machines, are starting to be more widely deployed (e.g., quadcopters, drones and soon service robots), the need of object detection systems is gaining more importance. Finally, we need to consider that we need object detection systems for nano-robots or for robots that will explore areas that have not been seen by humans, such as depth parts of the sea or other planets, and the detection systems will have to learn to new object classes as they are encountered. In such cases, a real-time open-world learning ability will be critical.

ACKNOWLEDGMENT

Authors would like to thank Guru Nanak Institute of Technology for the support in providing the necessary information along with development tools and kits in carrying out this paper. We would like to thank our Professors Dr. Suparna Biswas, Mrs. and Mr. Sayan Roy Chaudhuri, who gave us the opportunity to do this wonderful project on “REAL TIME OBJECT DETECTION”, and who also helped us in completing this project. By doing this project we came to know about many new things, and we are really thankful to them.

REFERENCE

[1] en.wikipedia.org
[2] www.geeksforgeeks.com
[3] https://pjreddie.com/darknet/yolo/
[4] http://web.eecs.umich.edu/~cscott/past_courses/eecs545f09/bib.html
[5] http://object-detection-algorithm-9eae9d6191b7