ECE 592 - Topics in Data Science
Project Report
on
“Object Detection for Autonomous Driving”

Submitted by -
1. Aman Waoo (200483401)
2. Girish Wangikar (200475452)
3. Lokesh Kondapaneni (200368051)
4. Manthan Talegaonkar (200481652)

Abstract

The concept of a self-driving car is that of being capable of sensing its environment and moving safely with little or no human interference. In order to perform environment sensing, high performance sensors, cameras and lidars are used. To ensure satisfactory performance of autonomous driving vehicles, the need for higher accuracy and faster inference speed, while implementing any object detection algorithm, is exponentially increasing.
YOLOv3 is an object detection algorithm that uses convolutional neural networks for the application of environment sensing. YOLOv3 has a greater number of layers than its predecessors which makes use of residual networks. It can perform a higher number of operations per second, with good FPS (Frames Per Second) and lesser mAP (mean Average Precision) values.

Introduction

Autonomous driving is not an exception to the current expansion of computer vision and artificial intelligence that has affected almost every industry in the globe. The primary responsibility of autonomous driving is environment sensing, or the ability to swiftly and precisely identify nearby things in order to create a model that will work. Processing speed has substantially increased along with the domains of computational resources' rapid development. However, it is particularly challenging to provide accurate environment sensing in autonomous driving settings because of the diversity of those scenes (such as cars and people in various weather conditions, varied lighting conditions, and with or without occlusion). As a result, the detection task still presents several challenges.
Object detection is a growing field of research in the field of computer vision. The use of object detection algorithms is becoming increasingly important in autonomous vehicles, and

autonomous driving will increasingly require more and more dependable network-based mechanisms, requiring redundant, real-time implementations. A false positive (FP) due to a false localization during autonomous driving can lead to fatal accidents and hinder safe and efficient driving. Therefore, an object detection algorithm for autonomous vehicles should satisfy the following two conditions; First, high accuracy while detection of road objects is needed. Second, a real-time detection speed is very important for incorporating used in driving.

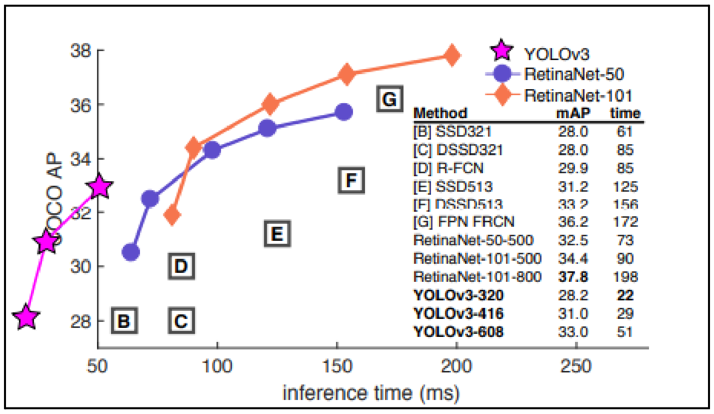
Object detection algorithms based on deep learning can be categorized in two classes; one-stage and two-stage. Two-stage algorithm generates region proposals in the first stage, and goes on to box regression and object classification prediction in these regions in the second stage, e.g., R-CNN based algorithms. Two-stage algorithms usually have a high accuracy but have a relatively slow detection speed. One-stage algorithms, such as YOLO (You Only Look Once), perform classification and regression in just one stage. This Report is based on YOLO

YOLO - You Only Look Once

YOLO is a powerful technique as it achieves high precision whilst being able to manage in real time. The YOLO method has several advantages as compared to other object detection algorithms. In other algorithms like Convolutional Neural Network (CNN), Fast-Convolutional Neural Network, the algorithm will not look at the image completely, but in YOLO, the algorithm looks at the image completely by predicting the bounding boxes using convolutional network and finds class probabilities for these boxes. This report explains the architecture and working of the YOLO algorithm for the purpose of detecting and classifying objects.
YOLO v3 algorithm takes input image or video of size 416 x 416 and gives output of multiple bounding boxes and their classes. In this project we used the Darknet53 architecture codebase which is trained on MSCOCO dataset.

Why YOLOv3?

According to the above figure, which was collected from **[9]**, YOLOv3 has a superior mAP and requires less time than other detection algorithms for the COCO dataset.



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Figure 1. Compares YOLOv3 to various algorithms in terms of mAP and time.

Now, we walk you through the process of implementing YOLOv3 while defining all the terms we'll need.

1. **Bounding Box**

   The code predicts the bounding boxes using predefined boxes called anchor boxes. The output from the code is in terms of four coordinates for the bounding box -Tx, Ty, Tw, Th. The width and height of box is predicted as offsets from centroid of clusters

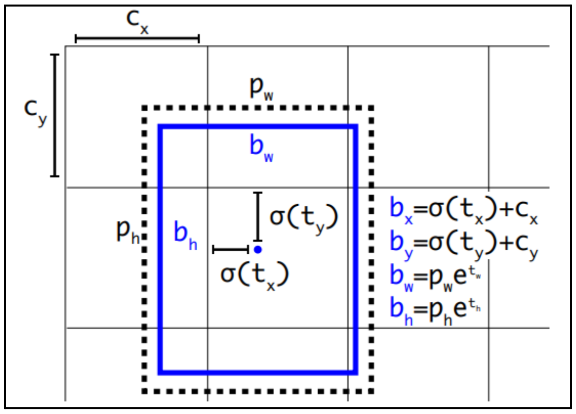


Figure 2. According to **[9]** Bounding boxes with dimension priors and location prediction.

2. **Anchor box**

   Anchor boxes are bounding boxes with varying scales and aspect ratios which are centered on each pixel. Instead of choosing these prior boundary boxes manually, a dimensional clustering algorithm such as K-means clustering is used to train on bounding boxes. Training on these bounding boxes with K-means clustering we find k= 5 works best and we get prior anchor boxes which we used in our code.

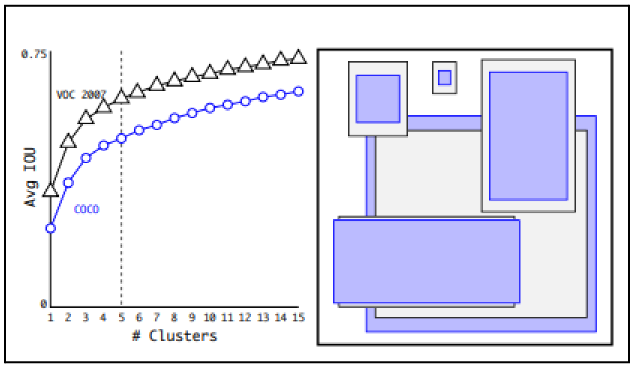


Figure 3: According to **[9]**, we see that k=5 works best on the COCO dataset when using the tradeoff for avg-IOU and number of clusters.

3. **Code Output**

   The code outputs multiple bounding boxes of different scales and ratios, the objectness score, and the multi-label class predictions. We will talk more about how we use bounding boxes of different scales in a later section. The objectness score tells if a particular box has any object recognized in it. The last section of the output, which gives us the class prediction to which the object in the bounding box belongs to. Multi-label approach is used instead of softmax as using softmax assumes that there is only one object class in the box and which is not in the case of object detection in self-driving cars.

4. **Feature Extractor**

The YOLOv3 architecture is built around Darknet-19 and the Residual Network layers. The Residual Network layer is a convolution layer - Batch Normalization Layer and a Leaky ReLU layer in sequence. There are three places from which outputs of different scales are pulled out from, this is to detect smaller objects in the image.

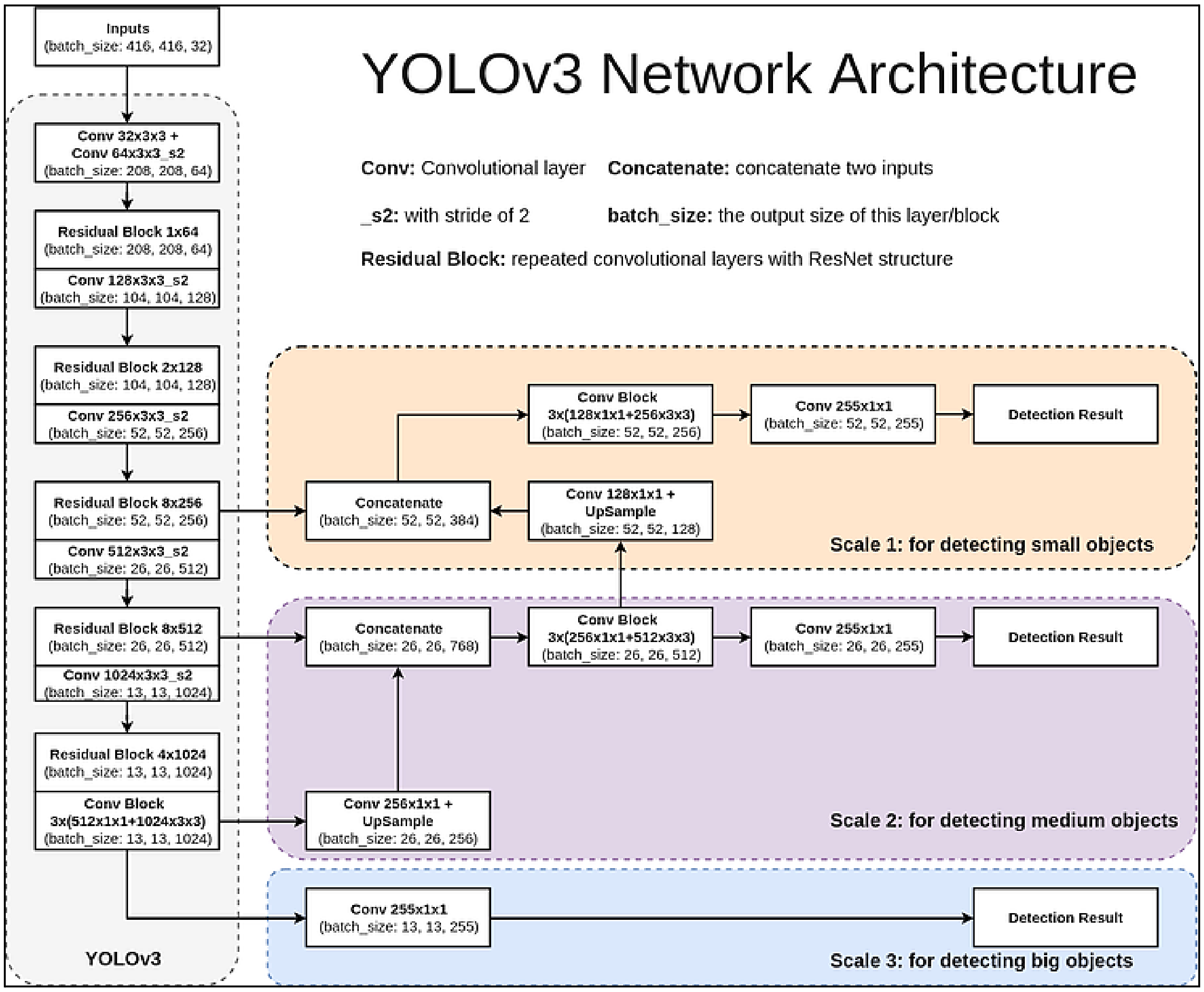

YOLOv3 Network Architecture
Conv: Convolutional layer
Concatenate: concatenate two inputs
_s2: with stride of 2
batch_size: the output size of this layer/block
Residual Block: repeated convolutional layers with ResNet structure
Inputs (batch_size: 416, 416, 32)
Conv 32x3x3 + Conv 64x3x3_s2 (batch_size: 208, 208, 64)
Residual Block 1x64 (batch_size: 208, 208, 64)
Conv 128x3x3_s2 (batch_size: 104, 104, 128)
Residual Block 2x128 (batch_size: 104, 104, 128)
Conv 256x3x3_s2 (batch_size: 52, 52, 256)
Residual Block 8x256 (batch_size: 52, 52, 256)
Conv 512x3x3_s2 (batch_size: 26, 26, 512)
Residual Block 8x512 (batch_size: 26, 26, 512)
Conv 1024x3x3_s2 (batch_size: 13, 13, 1024)
Residual Block 4x1024 (batch_size: 13, 13, 1024)
Conv Block 3x(512x1x1+1024x3x3) (batch_size: 13, 13, 1024)
YOLOv3
Conv Block 3x(128x1x1+256x3x3) (batch_size: 52, 52, 256)
Conv 255x1x1 (batch_size: 52, 52, 255)
Detection Result
Concatenate (batch_size: 52, 52, 384)
Conv 128x1x1 + UpSample (batch_size: 52, 52, 128)
Scale 1: for detecting small objects
Concatenate (batch_size: 26, 26, 768)
Conv Block 3x(256x1x1+512x3x3) (batch_size: 26, 26, 512)
Conv 255x1x1 (batch_size: 26, 26, 255)
Detection Result
Conv 256x1x1 + UpSample (batch_size: 26, 26, 256)
Scale 2: for detecting medium objects
Conv 255x1x1 (batch_size: 13, 13, 255)
Detection Result
Scale 3: for detecting big objects

Figure 4: Each scaled output detects objects of various sizes in the input image. **[10]**

This idea of grabbing outputs from different locations in a network of different scales is taken from the concept of Feature Pyramid Network(FPN). So, scale 1 which detects smaller objects outputs a matrix of size 52x52x255. Scale 2 has an output of size 26x26x255, detects medium sized objects. Similarly, scale 3 is used to detect larger objects and the size of the output is 13x13x255. These three outputs are combined and sent as an output of the network.

5. **Filtering boxes**

   Once we get the outputs from the network, we have to pick each of these arrays and decode the boxes and class predictions of the boxes. A threshold of 0.6 is used to filter out boxes which do not classify an object with good confidence.
   This finally returns us the candidate bounding boxes which have good confidence on the object in it (if any object is detected/classified).

6. **Non-Maximum Suppression (NMS):**

   Now, we might have a lot of candidate boxes and these might be referring to the same object. So, we need to filter these boxes with a confidence threshold and boxes that are overlapping and pick the best bounding box for that image based on IOU metric. This is done by taking the IOU of each of these boxes based on their confidence scores. IOU is used to measure the overlap between two bounding boxes.

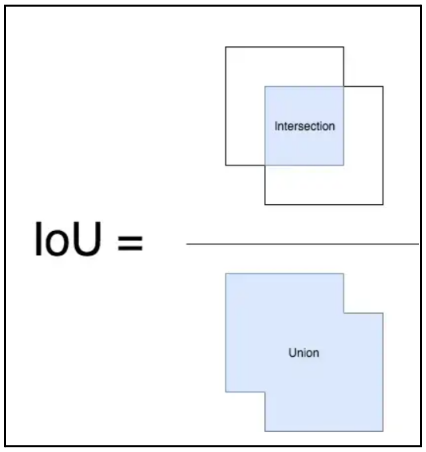
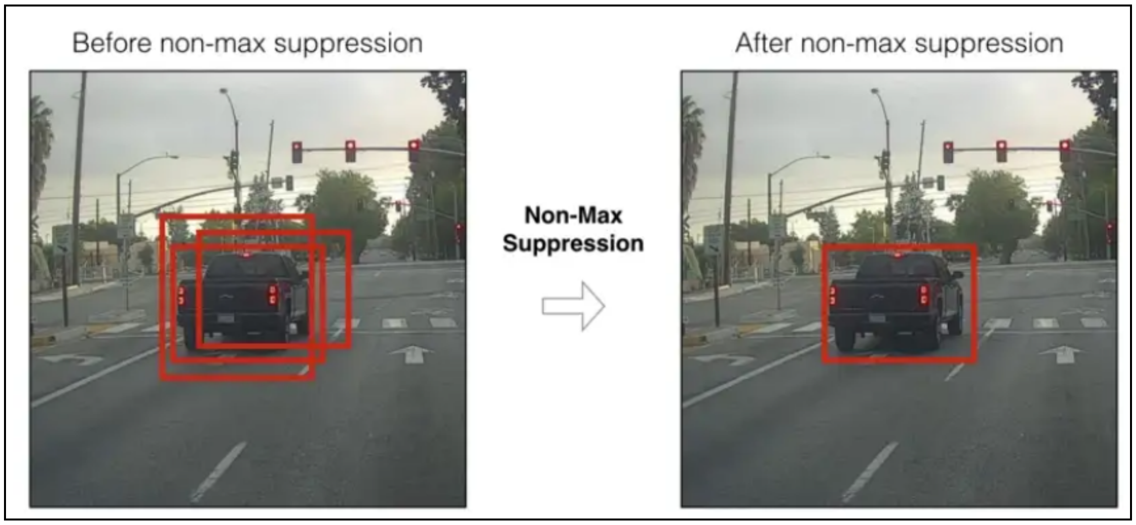


Figure 5: Intersection over Union is explained by **[9].**

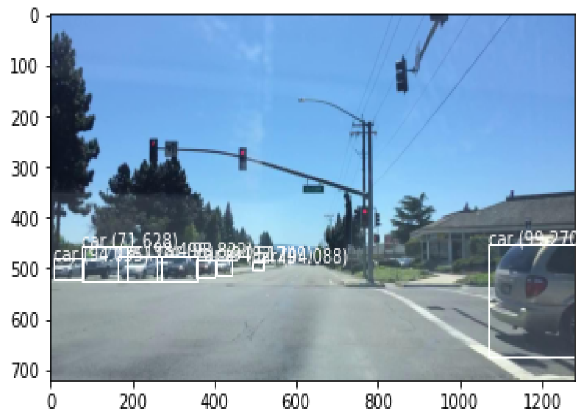
We used an overlap threshold of 0.5 to pick the best bounding boxes using IOU.

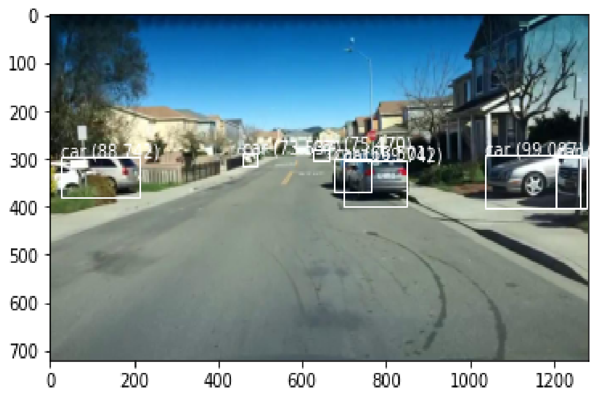
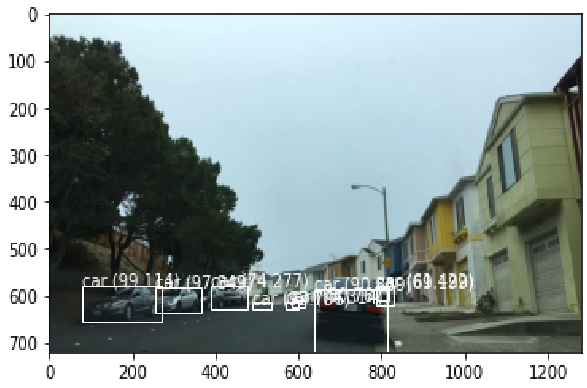
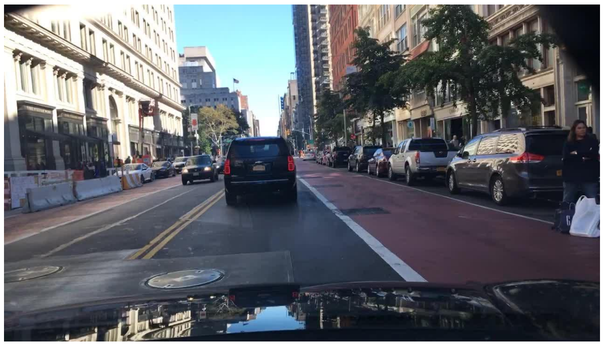
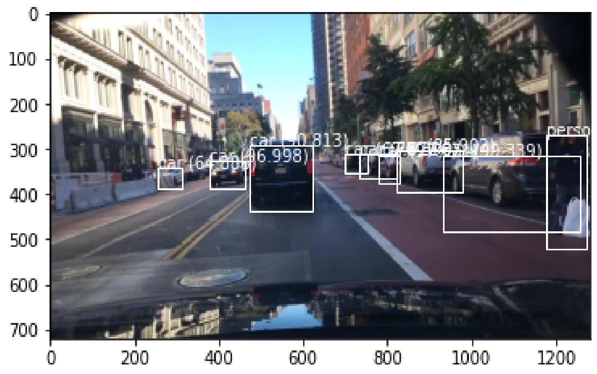
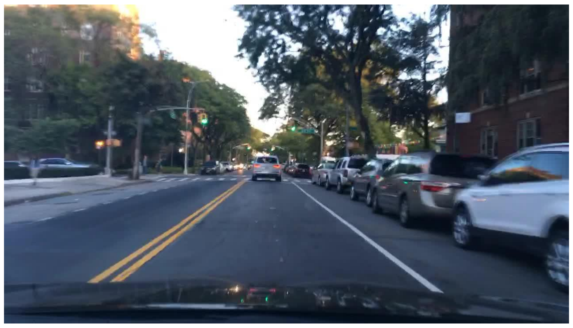
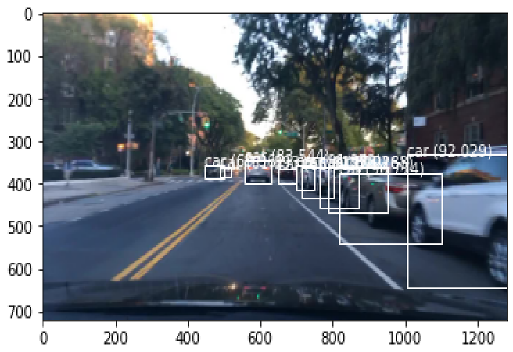


<u>Figure 6</u>: According to the Algorithm used in **[4]** ,the image on the left shows the output before non-max suppression is done. We can see multiple boxes around the same object. The image on the right is the output of NMS which gives us the best bounding box for that image.

Results

As we run the code we get the following as results for different images:

Input Image	Detected output image with bounding boxes and class probabilities
	

	car (88.2
	car (99.11 car (74.277)
	car (70.813) car (86.998) perso
	car (92.029)

References

1. https://link.springer.com/article/10.1007/s11042-022-13644-y
2. https://web.stanford.edu/class/cs231a/prev_projects_2016/object-detection-autonomous.pdf
3. How to Perform Object Detection With YOLOv3 in Keras - MachineLearningMastery.com
4. Image spam filtering using convolutional neural networks | SpringerLink
5. Non-maximum Suppression (NMS). A Technique to remove duplicates and… | by Sambasivarao. K | Towards Data Science
6. https://scholarworks.calstate.edu/downloads/t435gk20h
7. https://www.researchgate.net/publication/360688760_Object_detection_in_self_driving_cars
8. https://www.ijert.org/research/object-detection-and-classification-using-yolov3-IJERTV10IS020078.pdf
9. https://arxiv.org/abs/1804.02767
10. Training Yolo v3 model using custom dataset on Google colab | Tejashwi Kalp Taru