Advanced Driver Assistant Systems

Varun Goel and Harmandeep Singh Paul

June 24, 2021

Advanced Driver Assistant Systems

Varun Goel
Assistant Professor
Department of Information Technology
Maharaja Agrasen Institute of Technology (IPU), Delhi, India

Harmandeep Singh Paul
Department of Information Technology
Maharaja Agrasen Institute of Technology (IPU), Delhi, India

ABSTRACT

In recent years, traffic is increasing exponentially. The queueing of the vehicles on the road or intersection has risen sharply and the this increase in traffic flow has led to the traditional traffic lights systems to failure resulting in widespread delays and congestion, resulting in need for vehicle and traffic sign detection. In this paper, a real – time vehicle detection and traffic signs detection is proposed to increase the safety on the roads and to reduce the delays. A object detection model that detects in real time and based on the deep convolutional Neural Networks referred as you only look once (YOLO). This paper is focused on analyzing the costs and benefits of using YOLO as an alternative model in detecting vehicles and traffic signs.

Detecting data in a video stream is an object detection problem. An object detection problem can be approached as either a classification problem or a regression problem. In the classification approach, the image is divided into small patches, each of which will be run through a classifier to determine whether there are objects in the patch. The bounding boxes will be assigned to patches with positive classification results. This is all is done using the YOLOv4 approach which is combined with HOG and SVM approach. Linear SVM classifier algorithm is used to classify the car and traffic signs, Sliding Window and Heatmap algorithm are used to make it more robust.

INTRODUCTION

Autonomous vehicles have sparked a lot of interest in academics and industry during the last decade, especially in urban contexts. Due to a variety of difficulties, such as occlusion, color variation, rotation, and skew caused by camera setup or environmental impacts, detecting traffic signs and vehicles are difficult. In many places, traffic congestion is a severe issue, and fixed-cycle controllers are failing to address the long wait times at intersections. We frequently see a police officer in charge of traffic instead of a traffic light. He examines the state of the roads and determines the length of time each direction is permitted to travel. This human feat motivates us to develop a smart traffic light control system that takes into account a variety of factors. This human achievement motivates us to develop a smart traffic signal control system that takes into consideration real-time traffic conditions and intelligently manages the intersection. To put such a system in place, we'll need two primary components: eye to monitor the current road state and a mind to analyze it. We use deep learning as well as YOLO, a new Real-Time Object Detection method to gain from the advancement of computer vision techniques to complete objects of this paper.

In the traffic sign detection dataset, there are a total of 13 classes with various shapes and colors, as well as a "unlabeled" class for any sign that does not fit into one of the 12 initial classes. Triangle signs, red circle signs, blue circle signs, red and blue circle signs, diamond signs, reverse triangle signs, stop signs, forbidden signs, square signs, vertical rectangle signs, horizontal rectangle signs, other traffic signs, and an undefined class for all other signs are among these classes.

This project is for object detection which can be used in self-driving cars, Advanced Driver Assistance System (ADAS). By this system we can avoid the accident in collision with other vehicles, detect traffic signals and other traffic road indicators. We do not have to do everything manually in nowadays, we can do it easily with the help of advanced computers, which will be very fast and easy to use in ADAS.

Rest of the paper is organized as follows: Section II comprises of the methodology used and in Section III the experimental results are analyzed. In

Section IV the paper is concluded and the future scope is outlined.

METHODOLOGY

This section discusses the methodology used for vehicle detection and in detection of traffic signs and lights using the YOLO algorithm. Figure 1 shows the workflow diagram of our methodology.

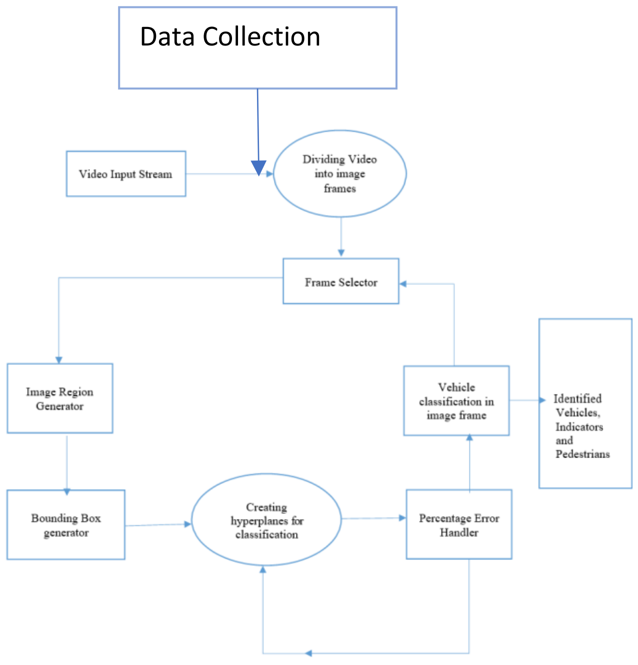
Data Collection
Video Input Stream
Dividing Video into image frames
Frame Selector
Image Region Generator
Vehicle classification in image frame
Identified Vehicles, Indicators and Pedestrians
Bounding Box generator
Creating hyperplanes for classification
Percentage Error Handler

Fig 1. Work-flow diagram for ADAS

A) Data Collection:

For the dataset to be in real-time, we have captured the real-time traffic images and recorded a sample of video for the identification and classification of the vehicles and traffic signs. For the traffic sign recognition, we use the “German Traffic Sign Recognition Benchmark (GTSRB)” as our de-facto dataset.

B) Data-set preparation:

In this paper we use an approach that aims to uncover the already known classes of vehicles and traffic signs on the roads. For this project we have downloaded the dataset from Kaggle, The German Traffic Sign Benchmark is a single-image classification task with many classes that was held at the International Joint Conference on Neural Networks (IJCNN) in 2011. The images and the videos downloaded were of different dimensions and also of different image quality. To accommodate these images into our dataset, we had to normalize the images using a python script into the desired format of the shape (*m*, *608*, *608*, *3*).

C) Dividing video into frames:

Furthermore, the video is then split into frames using a python script and each frame is interpolated as a single image and used to feed into the YOLO model.

Model Details:

Inputs:

- The input is a collection of photos, each of which has a unique shape (m, 608, 608, 3).

Anchor Boxes:

- The anchor boxes are picked by looking over the training data for suitable height/width ratios that represent the various classes.
- The dimension for anchor boxes is the encoding's second-to-last dimension: m, nH, nW, anchors, and classes.
- IMAGE (m, 608, 608, 3) -> DEEP CNN -> ENCODING is the YOLO architecture (m, 19, 19, 5, 85).

Encoding Architecture for YOLO:

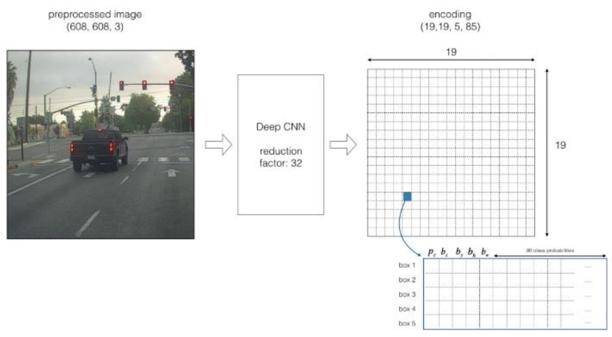

Fig 2.1 Encoding Architecture for YOLO

If an object's center/midpoint falls into a grid cell, that grid cell is in charge of detecting that object. Because we're employing five anchor boxes, each of the 19 × 19 cells has information about five of them. The width and height of an anchor box are the only parameters that define it. We'll flatten the last two dimensions of the shape encoding (19, 19, 5, 85) for simplicity's sake. As a result, the Deep CNN's output is (19, 19, 425).

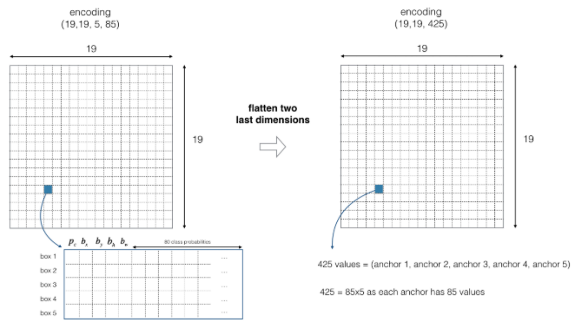
encoding
(19,19, 5, 85)
19
19
flatten two
last dimensions
p_c b_x b_y b_h b_w
80 class probabilities
box 1
box 2
box 3
box 4
box 5
encoding
(19,19, 425)
19
19
425 values = (anchor 1, anchor 2, anchor 3, anchor 4, anchor 5)
425 = 85x5 as each anchor has 85 values

Fig 2.2 Flattening the last two dimensions

Class Score

Now we'll compute the element-wise product for each box (of each cell) and extract a probability that the box has a specific class.

Score ci = p c * ci — the chance that an object exists p c multiplied by the likelihood that the object belongs to a specific class ci — is the class score.

Visualizing Classes

On an image, here's one method to see what YOLO is predicting:

Find the maximum probability score for each of the 19 × 19 grid cells (that is, a maximum across the 80 classes, one maximum for each of the 5 anchor boxes).

Color that grid cell according to what it thinks is the most likely object.

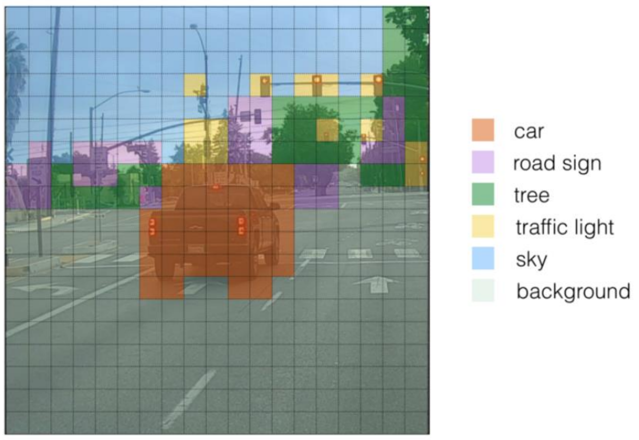

Fig 3 Highlighting the largest predicted probability

Visualizing bounding boxes

Another way to visualize **YOLO**'s output is to plot the bounding boxes that it outputs. Doing that results in a visualization like this:

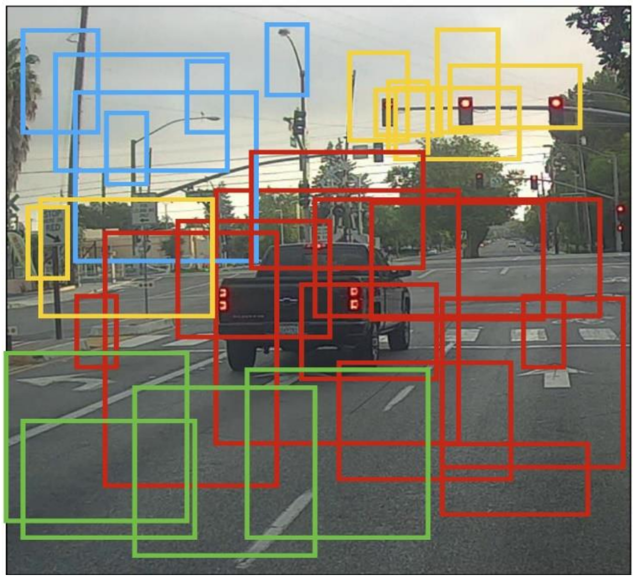

Fig 4. Image before Non-Max Suppression

Non-Max Suppression

We plotted just boxes for which the model gave a high probability in the image above, but there are still too many boxes. We'd like to reduce the number of recognized objects in the algorithm's output.

We'll employ non-max suppression to do this. We'll take the following measures in particular:

- Get rid of low-scoring boxes. The box is not confident in detecting a class, either because of the low likelihood of any item or because of the low likelihood of this particular class.

- When numerous boxes overlap and detect the same object, only one box should be selected.

Experimental results:

This section discusses the experimental results obtained after applying the real-time object detection algorithm based on deep convolutional neural network referred as YOLO on the created dataset as explained in the previous section

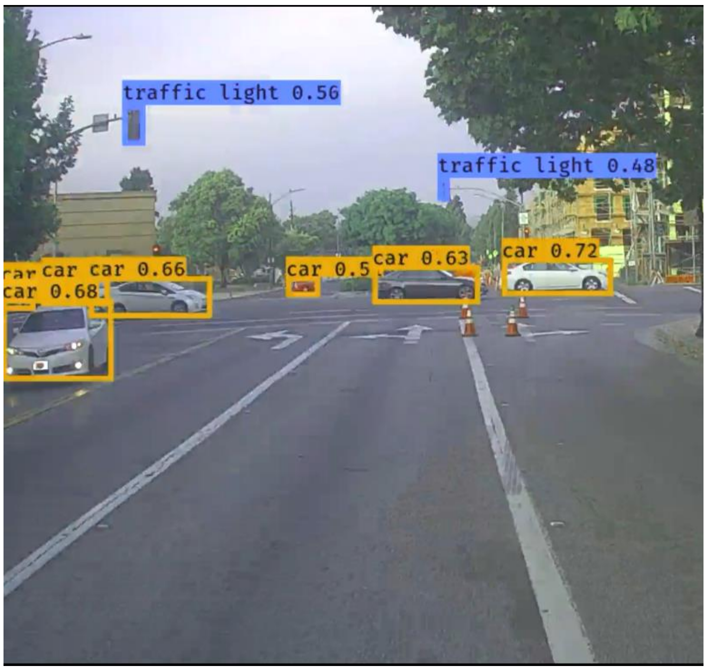

Fig 5.1 Final result

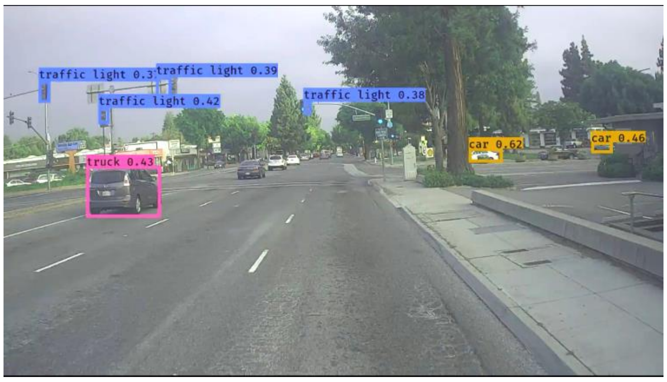

Fig 5.2 Final result

The figure 5.1 and 5.2 gives us the real-time output. The YOLO results in the creation of bounding boxes over the objects that are detected according to the particular dataset images.

The images containing the bounding boxes were then filtered to identify the bounding boxes that do not overlap, for this we use Non-Max Suppression method and further it was filtered according to class scores matching the threshold of 0.6.

The output is a list of bounding boxes together with the classes that have been identified. Each bounding box is represented by six numbers: pc, bx, by, bh, bw, and c. Each bounding box is represented by 85 values when c is expanded into an 80-dimensional vector.

Conclusion and Future Scope:

This paper is for Advanced Driver Assistant System which can be used in autonomous vehicles, and self-driving cars. By this system we can reduce our work. We do not have to do everything manually in nowadays, we can do it easily with the help of pcs, which will be very fast and easy to use. A forward pass of an entire image through the network is more expensive than extracting a feature vector of an image patch and passing it through an SVM. However, this operation needs to be done exactly once for an entire image, as opposed to the roughly 150 times in the SVM+HOG approach. A linear SVM processes video at a measly 3FPS on an i7 CPU. Yolo, a blazingly fast convolutional neural network for object detection on a fast GPU (GTX 1060) the video gets processed at about 65FPS. YOLO is more than 20x faster than the SVM+HOG and at least as accurate.

The problem of implementing object detection and classification on Indian roads is that Indian road signs are not well organized and printed. There is blurring and occlusion which lead to noise in images. One model which works one foreign roads necessarily may not work for Indian roads. This project can be extended to many dimensions such as self-driving vehicles, autonomous drones, robotics, ADAS. For its enhancement it will need more mathematical advancement and better hardware.

References:

[1] Guide to Car Detection using YOLO | by Bryan Tan | Towards Data Science

[2] Yolo Framework | Object Detection Using Yolo (analyticsvidhya.com)

[3] Joseph, R., Santosh, D., Ross, G., Ali, F.: You Only Look Once: Unified, Real-Time Object Detection. arXiv preprint arXiv:1506.02640 (2015)

[4] Joseph,R., Ali.,F.: YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767 (2018)

[5] J. Greenhalgh and M. Mirmehdi, “Real-time detection and recognition of road traffic signs,” IEEE Transactions on Intelligent Transportation Systems, vol. 13, no. 4, pp. 1498– 1506, 2012.

[6] The PASCAL Visual Object Classes (VOC) Challenge Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. and Zisserman, A. International Journal of Computer Vision, 88(2), 303-338, 2010.

[7] https://pjreddie.com/darknet/yolo/

[8] https://www.kaggle.com/andrewmvd/road-sign-detection

[9] https://github.com/thtrieu/darkflow

[10] https://ieeexplore.ieee.org/abstract/document/5010438

[11] https://www.kaggle.com/andrewmvd/road-sign-detection\

[12] https://ieeexplore.ieee.org/abstract/document/5010438