Advanced Driver Assistance System (ADAS)

Project Report

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 ADAS

Advanced Driver Assistance Systems or ADAS, is a set of subsystems that assist humans in driving. Given the fact that the majority of accidents occur due to human error, the necessity of these systems is significant. They reduce the number of fatalities that happen due to accidents, by minimizing human error. Adaptive Cruise Control, automatic parking, driver monitoring system, forward collision warning, lane centring, lane change assistance, traffic light and sign detection, are some of the subsystems that help in assisting humans. The greatest advantage of using assistance systems is that they enable communication between different vehicles, vehicle-to-infrastructure systems, and transportation management centres. This enables the exchange of information for better vision, localization, planning, and decision making of the vehicles. Yet these systems have a long way to go to achieve complete autonomy. We believe our project is a small step towards achieving this giant objective.

1.2 Why traffic lights and signs?

Traffic lights and signs around the world are not smart and efficient. Traffic lights turn from red to green and from green to red after a specified amount of time. They don't take into account the density of the traffic, the number of pedestrians on the road, etc., On the other hand, traffic signs are just still. For example, speed limits don't change according to the number of cars on a particular road. Until and unless a combined communication infrastructure such as vehicle-to-infrastructure communication is in place, the burden of detecting the traffic lights and signs and reacting to them accordingly falls on the ADAS system.

With a tonne of autonomous cars offering lane change assistance, cruise control and automatic parking, most of these systems can be classified as Level 2. The system reacting to the surrounding becomes an essential part for it to be classified as Level 3, and we believe traffic light and sign detection is the first step towards achieving Level 3. Moreover, a single model must be deployed to perform all the tasks such as traffic light

detection, sign detection, lane assist and change, collision avoidance to reduce the overhead on the hardware.

Thus we design and implement a single model to detect traffic lights and signs and to test the extent of our model's ability to accommodate various other ADAS components, we couple the model with an anterior car distance measurement system for collision avoidance.

1.3 Literature Survey

Traffic light detection and traffic sign detection are not new-age problems that emerged right after the arrival of driver-assist systems like Tesla Autopilot. Several works have been published on these problems for over a decade now. Earlier works consisted mainly of image processing techniques such as colour thresholding [1], template matching [2], and prior maps [3]. Ever since Krizhevsky et al. published [4], deep learning has been the go-to method for most of the computer vision tasks.

A traffic light detector that uses a heuristic ROI detector for proposing regions and a deep CNN classifier for traffic light classification is presented in [5]. Behrendt et al. [6] proposed a YOLO based traffic light detector that uses a small classification network at the end to classify the states of the light and in [7], a comparative study between various versions of YOLO on LISA traffic light dataset is presented.

In the case of traffic sign detection, a modified Convolution Neural Network is used in [8]. Multi-scale convolutional network is proposed in [9] and a CNN with dilated convolutions in [10].

Even though several works have been presented for traffic light detection and traffic sign detection separately, there isn't a combined work except for [11]. And even in [11], no other major ADAS component is added as we intend to do. As the first step in this process, we had to narrow down to a single object detection algorithm that tends to perform better on the traffic light, sign and car datasets. In [12], the performance trade-

offs of Faster RCNN, R-FCN, and SSD models have been discussed and the performance of these models on the MS COCO dataset is also studied. Since the average width of traffic lights in Bosch Small Traffic Light dataset is 10 pixels, and Faster RCNN performs the best when it comes to detecting smaller objects, we choose Faster RCNN over SSD. Since no prior works have performed a comparative study between YOLO and Faster RCNN, we do it to single out an object detection algorithm.

CHAPTER 2: ALGORITHMS AND DATASET

2.1 Algorithms:

2.1.1 Faster RCNN:

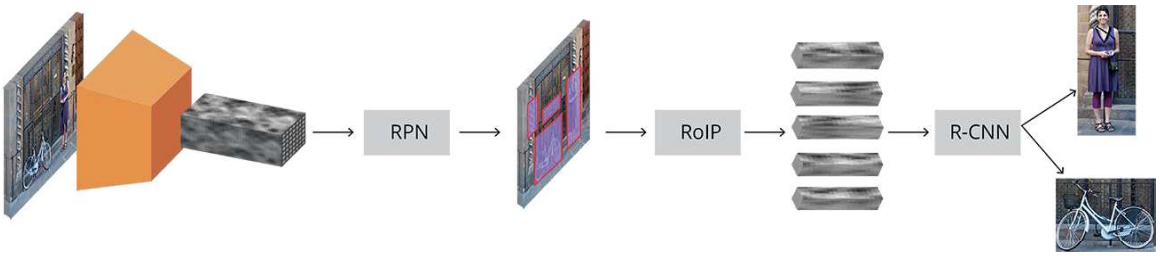

Fig 2.1 Complete Faster RCNN architecture

The Faster RCNN [13] algorithm replaced selective search used in Fast RCNN by a more efficient Region Proposal Network (RPN) that can be trained end-to-end along with the classifier network, thereby leading to improved results at an increased speed. The core part of the algorithm can be explained with the following steps:

1. Generation of the feature map using a pre-trained Convolutional Neural Network (CNN).
2. Using the feature map obtained in the previous step, the RPN outputs a set of region proposals with the help of different sized anchor boxes.
3. The final stage, which consists of the Region-based Convolutional Neural Network (RCNN), carries out the classification task by a fully connected layer that predicts a score for each class.

Region Proposal Network (RPN):

The Region Proposal Network's primary objective is to predict bounding boxes that might contain objects in an image. It takes the convolutional feature map generated by the base network as input. An object detector is only efficient when it generates bounding boxes of varying lengths that better fit the objects, as they differ in size. Faster RCNN tackles this problem by using anchors, which are reference bounding boxes placed uniformly throughout the image. These anchor boxes take the feature maps extracted from CNN as the input. The dimensions of

anchor boxes are the same as the height and width of the feature map. The anchor boxes generated have two outputs based on the Intersection over Union (IoU) metric; one is the objectness score (possibility of the anchor boxes for being an object if IoU is greater than 0.5 or not an object if IoU is lesser than 0.1) and the other is the bounding box regression problem (adjust anchor boxes to better fit the object). Using non maximal suppression algorithms, duplicate bounding boxes are eliminated. Then the final proposal coordinates along with their objectness scores are generated.

Region-based Convolutional Neural Network (RCNN):

The next stage in the Faster RCNN pipeline consists of RCNN. The architecture reduces computational complexity by re-using the existing convolutional feature map to get a fixed size feature map for each proposal region given by the RPN, which is done using the Region-of-Interest (RoI) pooling. Thus the classifier network serves two purposes: classify the proposals obtained by RoI pooling to one class and also adjust the proposal according to the predicted class. Similar to the RPN, bounding boxes are classified into multiple classes and a score for each class is generated by using IoU. If $0.1 < IoU < 0.5$ then the proposal is classified as the background class and if $IoU > 0.5$ then the proposals are classified as the object class. After these steps, the bounding boxes and objects class score are obtained and the object is detected in that location.

It can be seen that the model as a whole consists of four different losses: two for RPN and two for RCNN, and the total loss represents a weighted sum of these losses. The base network can be fine-tuned according to the dataset with the popular ones being Inception V2, ResNet-50 and ResNet-101. The standard implementation of the Faster RCNN model as in [13] is trained using a stochastic gradient descent optimizer function with a momentum value of 0.9. Initially, the learning rate is set to 0.001 which then decreases to 0.0001 after 50k steps.

Since the Inception V2 network coupled with Faster RCNN performs the best when it comes to the speed vs. accuracy trade-off, we implement this architecture for the combined detector. The main motivation behind Inception was to reduce computation and obtain state-of-the-art performance at the same time. The original Inception network (also known as GoogLeNet) has multiple sized filters on the same level and shaped the network to be a bit wider instead of deeper. It uses convolutions of different sizes to capture details at varied scales (5x5, 3x3, 1x1). The computation cost was decreased by the 1x1 convolutions, which is called the bottleneck layer. The architecture was made more efficient by introducing Batch Normalization which enabled higher learning rates and regularized the model better, which is named Inception V2. In Inception V2, the reasoning was that reducing the input dimension drastically during convolution might lead to poor performance, so the convolutions are made efficient in terms of computational complexity. 5x5 convolutions were converted to two 3x3 convolutions to improve speed (a 5x5 convolution is 2.78 times more expensive than a 3x3 convolution).

2.1.2 YOLO

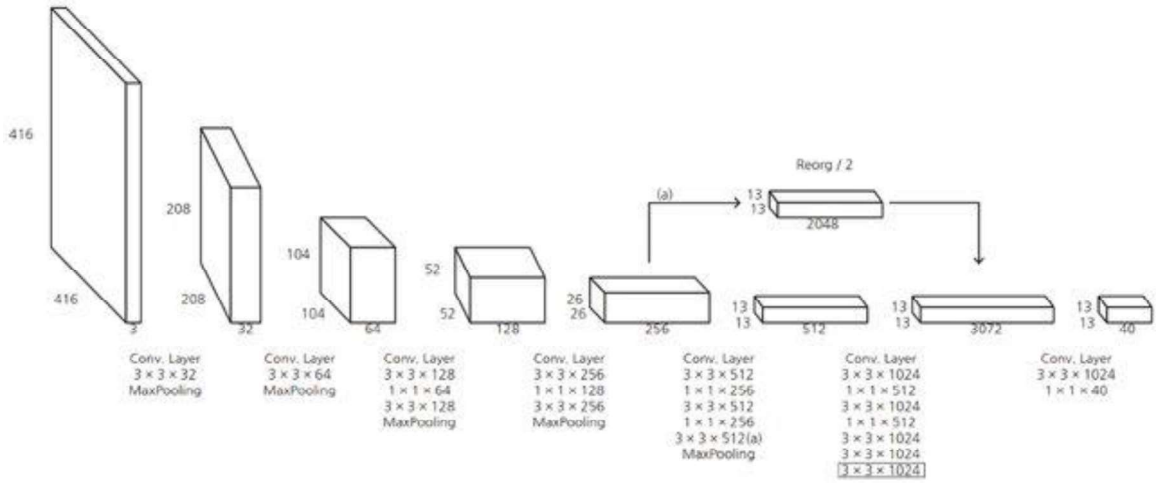

Fig 2.2 YOLOv2 architecture

YOLOv2 divides the input image into NxN (typically, 13x13) grids and at each grid, the network predicts whether the centre of an object falls within the gird with the help of five pre-determined bounding boxes called anchor boxes and attaches a confidence score to each box. The anchor boxes are calculated with the help of the k-means clustering algorithm and the ground truth boxes. To finalize on the predictions, the algorithm removes the boxes with no objects and the boxes which have a confidence score less than the threshold. And finally, to avoid multiple predictions of the same object, non-max suppression is used.

YOLOv2 uses darknet-19 as its base feature extractor. Darknet-19 is a set of Convolutional and MaxPooling layers, enabled with batch normalization. Typically, the input resolution of the image is set to 416x416.

The reorg layer just reshapes the feature map by reducing the size and increasing the number of channels without changing the numerical values. If the stride of the reorg layer is mentioned as two, the height and width of the feature map would be reduced by a factor of two and the number of channels will be increased by a factor of 2x2. So reorg layer reorganizes or changes the output tensor of the middle level to concatenate it with the higher level tensor outputs. The reason for this is to take into account the fine grained features. This is achieved by stacking the low resolution features into different channels.

The objective function for training YOLOv2 is provided below,

$$loss_t = \sum_{i=0}^{W}\sum_{j=0}^{H}\sum_{k=0}^{A}\left(\begin{array}{c} 1_{MaxIOU<Thresh}\,\lambda_{noobj} * \left(-b_{ijk}^{o}\right)^2 \\ +1_{t<12800}\lambda_{prior} * \sum_{r\,\epsilon\,(x,y,w,h)}\left(prior_k^r - b_{ijk}^k\right)^2 \\ +1_k^{truth}(\lambda_{coord} * \sum_{r\,\epsilon\,(x,y,w,h)}\left(truth^r - b_{ijk}^k\right)^2 \\ +\lambda_{obj} * \left(IOU_{truth}^k - b_{ijk}^o\right)^2 \\ +\lambda_{class} * (\sum_{c=1}^{c}\left(truth^c - b_{ijk}^c\right)^2) \end{array}\right)$$

The function defines the loss function for a single iteration t. The first loss term represents the reduction in the objectness score if there is no object in a grid. The second loss term reduces the difference between the predicted bounding box and the chosen anchor boxes after a few iterations. The λ values are pre-defined weightages. The third loss term is responsible for the predictions to be aligned with true values.

YOLOv3 is pretty similar to YOLOv2 with just minor incremental updates. The architecture and the working of YOLOv3 is similar to that of YOLOv2 with just these changes,

1. Feature extractor: YOLOv2 uses Darknet-19 whereas YOLOv3 uses Darknet-53 as its feature extractor. Darknet-53 is as accurate as ResNet-101 but is 1.5x times faster.

2. Feature Pyramid Network: (FPN) Even though YOLOv2 used residuals from previous higher layers and concatenated it with lower layers, YOLOv2 still struggled with detecting smaller objects. So YOLOv3 makes detections in three different layers similar to a Feature Pyramid Network (FPN).
    1. In the last convolutional layer
    2. Feature map from 2 layers previous to the last convolutional layer is taken and upsampled by 2x.
    3. A feature map from earlier in the network is also taken and concatenated. Convolutional layers are then used to process these concatenated tensors to provide meaningful semantic information

3. Anchor boxes: YOLOv2 required a single set of anchor boxes but since YOLOv3 predicts objects at three different layers, YOLOv3 requires three different (or same) set of anchor boxes. These set of anchor boxes can be found using k-means clustering.

2.1.3 Semi Global Block Matching (SGBM)

The depth information from stereo images can be obtained by mapping the correspondence of objects in both left and right images. The disparity map represents the horizontal shift in pixels and it is important that there are no distortions while generating disparity. The algorithm should be able to perform well in terms of speed as well as quality. The overhead of stereo matching techniques increases considerably with respect to the resolution of images, and therefore efficient ones which achieve a decent performance while being computationally less expensive, are preferred. Semi Global Block Matching (SGBM) [14] is such an algorithm used in real-time applications like autonomous driving. It combines aspects of both local and global stereo methods for accurate pixel-wise matching at a lower runtime. SGBM is a single pass algorithm that makes use of a modified Mutual Information (MI) matching cost, whilst optimizing from several paths, usually eight directions, to the pixel considered. The cost function is given below,

$$E(D) = \sum_{P} \begin{pmatrix} C(p, D_p) + \sum_{q \epsilon N_p} P_1 T|\left|D_p - D_q\right| = 1| + \\ \sum_{q \epsilon N_p} P_2 T|\left|D_p - D_q\right| > 1| \end{pmatrix}$$

Here, $C(p, D_p)$ is the pixel-wise matching cost for a particular pixel p with disparity D_p, T is a function that returns 1 if the argument inside the modulus operator is true, and 0 if the value is false. N_p is the neighboring pixel of p that has a cost P_1, while the parameter P_2 is adjusted according to the local intensity gradient between the pixels p and q.

The number of paths affects the speed of the algorithm and a higher number always leads to better quality. The disparity with the lowest cost value is then chosen. The OpenCV implementation of SGBM gives a decent amount of flexibility to better adjust the algorithm with respect to the task at hand by varying

the disparity range, block size and disparity smoothness. To improve the disparity map generated, certain pre- and post-processing steps such as filtering are done.

2.2 Dataset:

2.2.1 Bosch Small Traffic Light Dataset:

The Bosch Small Traffic Light Dataset consists of 5,093 images for training and 8,334 images for testing with a resolution of 1280x720 pixels, containing 10,756 and 13,493 annotated traffic lights respectively. The training set has 13 classes such as “red”, “yellow”, “green”, “off”, “green straight right”, “red straight” whereas the testing set has only 4 classes (“red”, “yellow”, “green”, “off”). Despite this inconsistency, the hand-labelled small traffic lights present a challenging task that makes it suitable for evaluating and comparing various object detectors. To mitigate this inconsistency and to overcome the uneven categorization of classes, we limit the number of classes to 4 by grouping similar labels as “red”, “yellow”, “green”, or “off” in the training set.

2.2.2 Tsinghua-Tencent 100K Dataset:

The Tsinghua-Tencent 100K (TT100K) dataset consists of 100,000 images, out of which 10,000 images contains 30,000 traffic sign instances. Even though there are around 221 labelled classes, only 45 classes have more than 100 examples. Thus we limit the number of classes in the TT100K dataset to 45 as in [15], [16]. The number of images in train and test sets is 6107 and 3071 respectively.

2.2.3 Udacity Dataset:

The Udacity Dataset annotated by CrowdAI consists of three classes namely car, truck, and pedestrian. For our purpose, we reduce the number of classes to one i.e., car. The dataset consists of 9,423 images which were hand separated into test and train sets. The images with instances of traffic lights and traffic signs were pushed to the test set whereas images with just instances of cars were pushed to

the train set. The train set consists of 5487 and the test consists of 3936 images respectively.

2.2.4 KITTI Stereo Vision Dataset:

The KITTI Vision Benchmark Suite [17], a joint venture by the Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago, consists of datasets for tasks such as stereo, optical flow, visual odometry, object detection and tracking. The setup consists of a standard station wagon equipped with two high-resolution colour and grayscale video cameras. We found the stereo datasets to be best suited for the project as it provides accurate ground truth obtained by a Velodyne laser scanner combined with a GPS localization system. Also, there are many instances of cars in the stereo images, which becomes useful for developing our vehicular distance estimation model. The Stereo Evaluation 2012 benchmark consists of 194 image pairs with a resolution of 1241x376 pixels and each frame has a calibration file with the specifications.

CHAPTER 3: METHODOLOGY AND EVALUATION

A bird's eye view of our system to detect traffic light, traffic sign and anterior car distance measurement is provided in the figure below,

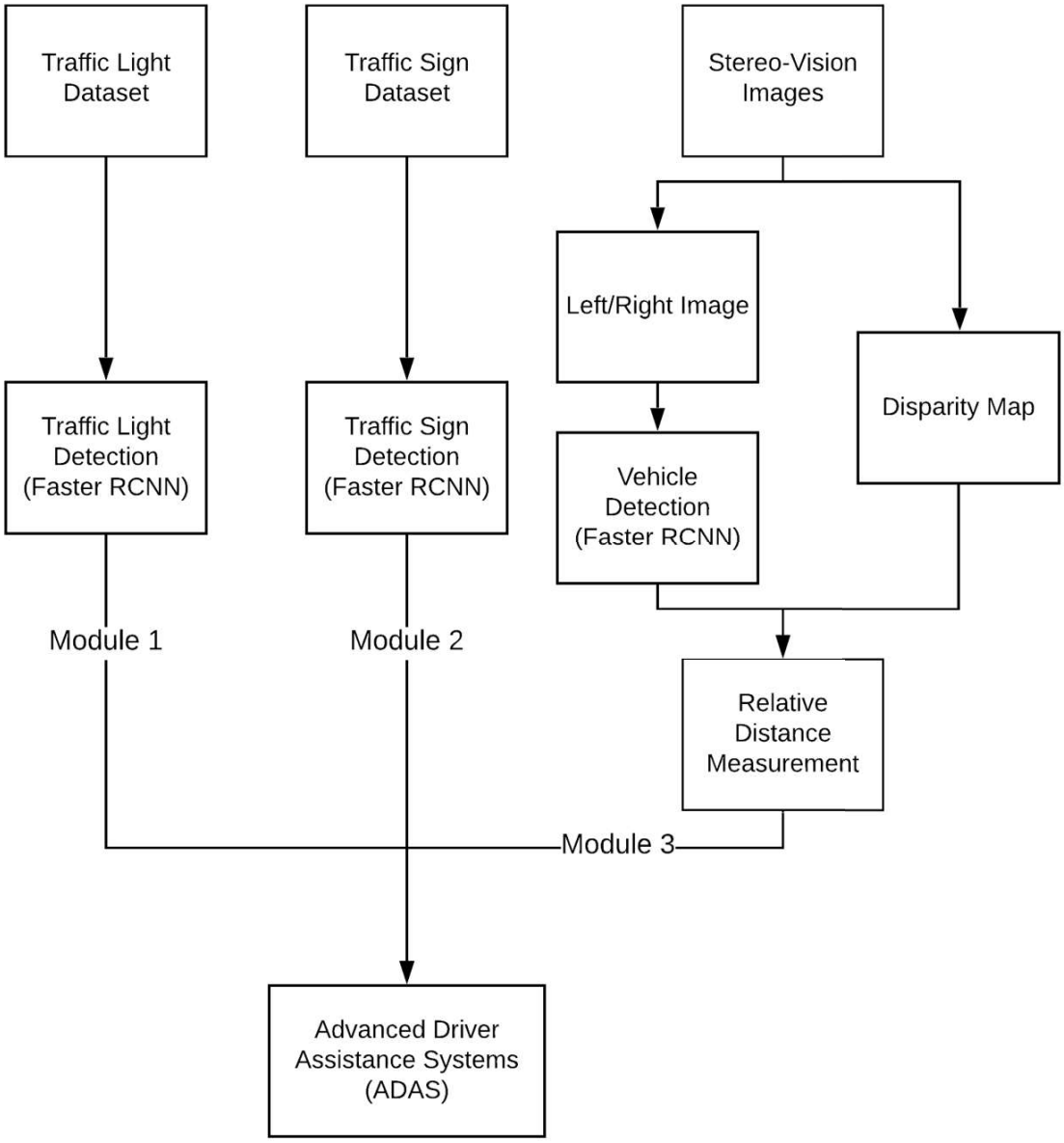

Fig 3.1 A flowchart of our system

3.1 Background Thresholding

When it comes to combining large datasets (consisting of similar classes) to build a joint detector, a drop in performance arises primarily due to the issue of overlapping objects. As in our case, there are traffic sign instances in the Bosch Small Traffic Light dataset and traffic light instances in the TT100K dataset. These turn out to be negative examples while training as there are no annotations for traffic signs in the traffic light dataset and vice-versa. As suggested in [11], we make use of the background thresholding model to bypass this problem by altering the RPN mini-batch selection in the Faster RCNN pipeline. It tries to reduce the possibility of unlabelled objects of interest being considered as background. The reasoning behind background thresholding is that traffic lights and traffic signs are generally separated by a few metres and are not found adjacent to each other. The standard Faster RCNN mini-batch selection gives negative proposals (background) when the IoU is less than 0.3 and positive proposals when the IoU is greater than 0.7. We alter the criteria for predicting negative proposals as such: 0.01<IoU<0.3. Thus, by including the lower threshold value of 0.01, it is made sure that the negative proposals have a small overlap with the ground truth.

In the original Faster R-CNN mini-batch selection, proposals are separated into a positive set B_+ and a negative set B_- for loss computation according to:

$$B_+ = \{b_i | 0.7 \leq IoU(b_i, g_i) \leq 1\},$$
$$B_- = \{b_i | IoU(b_i, g_i) \leq 0.3\},$$
$$\forall b_i \in B, g_i \in G$$

The background thresholding alters the mini-batch selection by adding a threshold as mentioned above,

$$B_+ = \{b_i | 0.7 \leq IoU(b_i, g_i) \leq 1\},$$
$$B_- = \{b_i | 0.01 \leq IoU(b_i, g_i) \leq 0.3\},$$
$$\forall b_i \in B, g_i \in G$$

To better understand this, consider Fig 3.2. The ground truth is shown in green, whereas the positive and negative proposals are shown in blue and red respectively. It can be observed that the negative proposals consist of an unlabelled object of interest (in this

case, a traffic sign). But in Fig 3.3 it can be seen that introducing a background threshold minimizes the possibility of negative proposals containing unlabelled objects of interest.

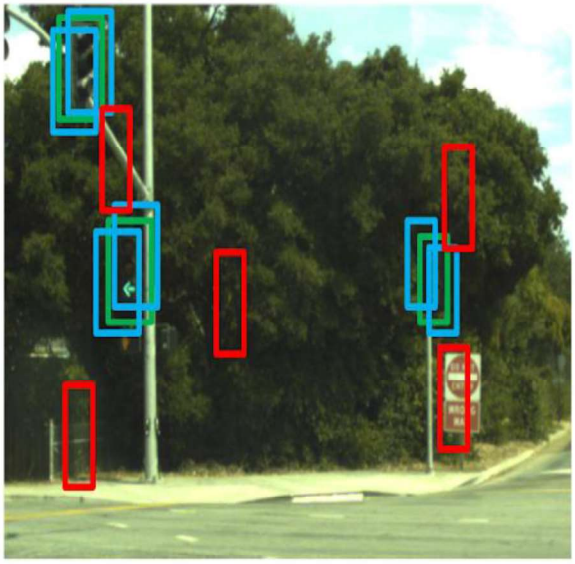

Fig 3.2 Region Proposals without background thresholding

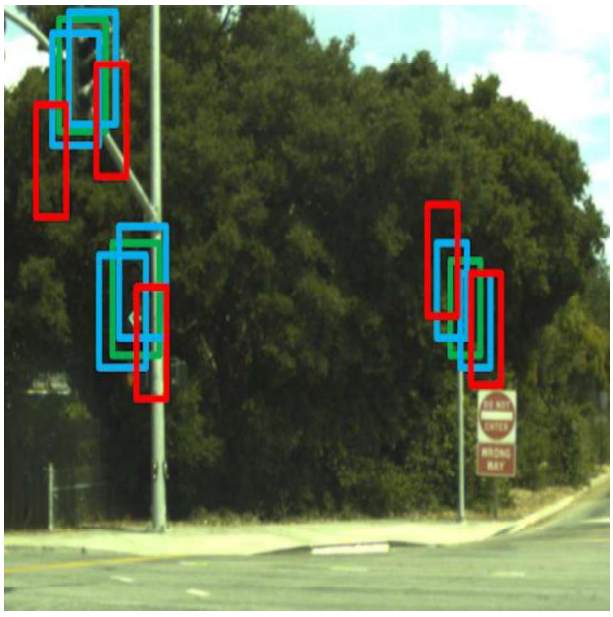

Fig 3.3 Region Proposals with background thresholding

3.2 Stereo Triangulation

The object depth can be interpreted from images with the help of the horizontal shift, known as disparity in stereo images. Triangulation refers to the process of obtaining the 3D position of an object from images using the disparity map generated.

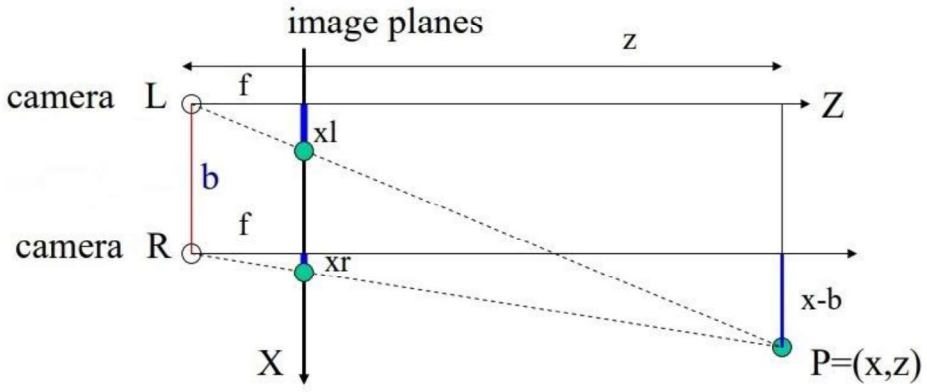

Fig 3.4 Stereo Triangulation

The standard model of triangulation is as shown in the figure. The left camera is considered to be the reference point or origin. The point P (x, z) is our object of interest and the objective is to determine the depth (z). The left and the right cameras are separated by a distance referred to as the baseline width (b) and (f) is the focal length. For ease of understanding, the image planes (xl and xr) are depicted in front of the cameras.

$$Disparity, d = xl - xr$$

By the property of similar triangles,

$$\frac{z}{f} = \frac{x}{xl} \tag{1}$$

$$\frac{z}{f} = \frac{x - b}{xl} \tag{2}$$

From (1) and (2),

$$z = \frac{b - f}{xl - xr}$$

$$Depth = \frac{baseline * focal\ length}{disparity}$$

3.3 Evaluation Metrics:

In a typical dataset, there will be many classes and their ways of distribution might be non-uniform. So a basic method of accuracy based metric will introduce bias, and it is important to assess the risk of misclassification. Thus there is a need to assign a confidence score or model score with each bounding box detected and to assess the model at its various levels of confidence. This is achieved by incorporating IoU as it measures the overlap between the ground truth and detected bounding box.

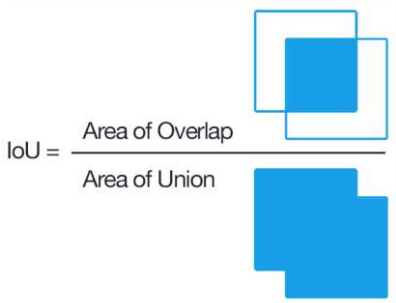

Fig 3.5 Intersection Over Union

Generally, for most cases, the threshold for IOU can be set to 0.5. This can vary from problem to problem. Normally, IoU>0.5 is considered a good prediction. This type of binary classification is what makes computing accuracy straightforward for object detection tasks.

Precision and recall are two commonly used metrics to judge the performance of a given classification model. To better understand mean Average Precision (mAP), we would need to first know about precision and recall. In data retrieval, precision refers to the percentage of correct detections, whereas recall (also known as sensitivity) refers to the percentage of total relevant detections correctly classified with respect to the ground truth.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Usually, for any given model, a precision vs. recall curve is plotted by varying the IoU thresholds in order to understand the trade-off between these two quantities. An ideal object detection model should have the highest precision and recall possible which enables it to return detections with a minimal number of false positives and false negatives. In other words, the area under the precision vs. recall curve must be equal to one in the ideal case.

The generic definition for the Average Precision (AP) is the area under the precision vs. recall curve obtained by integrating or summing up precision values at different recall points that are evenly spaced (101-point interpolated AP as in the case of COCO mAP). The mean average precision or mAP score is calculated by taking the mean of AP over all the classes. Thus mAP serves as a single metric that characterizes the performance of a model. It gives a fair understanding of the model's ability to balance both precision and recall.

CHAPTER 4: EXPERIMENTS AND RESULTS

The models were trained on a machine equipped with Intel Core i7-7700, 32GB RAM and NVIDIA P1000 whereas the tests were carried out on a PC equipped with Intel Core i7-65002.5GHz processor, 8GB RAM, and NVIDIA GeForce 930M. The experiments involving Faster RCNN were performed with the help of Tensorflow Object Detection API, whereas the experiments involving YOLO were performed with the help of Darknet.

4.1 Traffic Light and Sign Detection

In the case of object detection, as mentioned earlier, deep learning-based methods yield impressive results. With numerous object detectors out there, selection of an object detector becomes a paradox of choices. Roughly, the object detectors can be classified as two-stage detectors (Fast RCNN, Faster RCNN) and single-stage detectors (SSD, YOLO). In **[18],** the performance trade-offs between SSD, R-FCN and Faster RCNN models have been discussed. When it comes down to detecting smaller objects, Faster RCNN performs better than SSD. Given the fact that the average width of traffic lights in Bosch Small Traffic Light dataset is only 10 pixels, we choose Faster RCNN over SSD. YOLO, another single shot detector is quite known for its fast detections. Even though **[7]** presents a comparative study between various versions of YOLO and **[18]** presents a comparison between Faster RCNN and SSD, no prior works have performed a comparative study between YOLO and Faster RCNN.

The models were trained for 50,000 iterations with a learning rate of 0.0003 and momentum of 0.9. In order to speed up the greedy non-maximal suppression algorithm in Faster RCNN and reduce the overall training time, the post-processing score and IoU thresholds are increased to 0.7, the maximum detections per class are set to 5 and the maximum total detections are decreased from 300 to 50. In the case of YOLOv2 and YOLOv3, anchor boxes are chosen with the help of the k-means clustering algorithm.

There exists a problem of bias towards classes “red” and “green” in the Bosch Small Traffic Light Dataset as they are over-represented. We make use of data augmentation options such as resizing and cropping with a probability of 0.5 to optimize the model. The dataset does not account for extreme variations in brightness. So we adjust the

brightness randomly to improve the model accuracy. Traffic lights are mounted in different positions depending on the country and in a few cases, the traffic lights have different shapes to aid the colour blind. In order to bypass this problem, we add jitter in the dataset which makes the network generalize better.

Some sample detections by Faster RCNN + Inception V2 is presented below,

Fig 4.1 Faster RCNN detection from Bosch dataset

Fig 4.2 Faster RCNN detection from Bosch dataset

Fig 4.3 Faster RCNN detection from a video

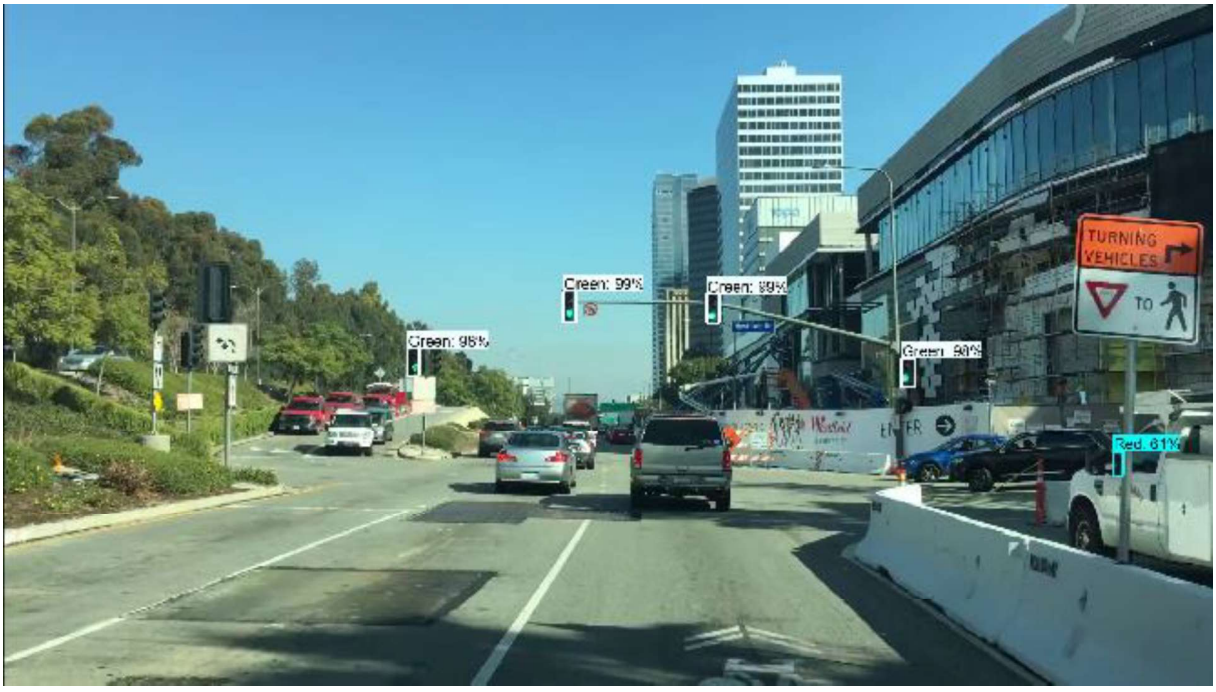

Fig 4.4 Faster RCNN detection from a video

Some sample detection from YOLO is presented below,

Fig 4.5 YOLO detection from Bosch dataset

Fig 4.6 YOLO detection from Bosch dataset

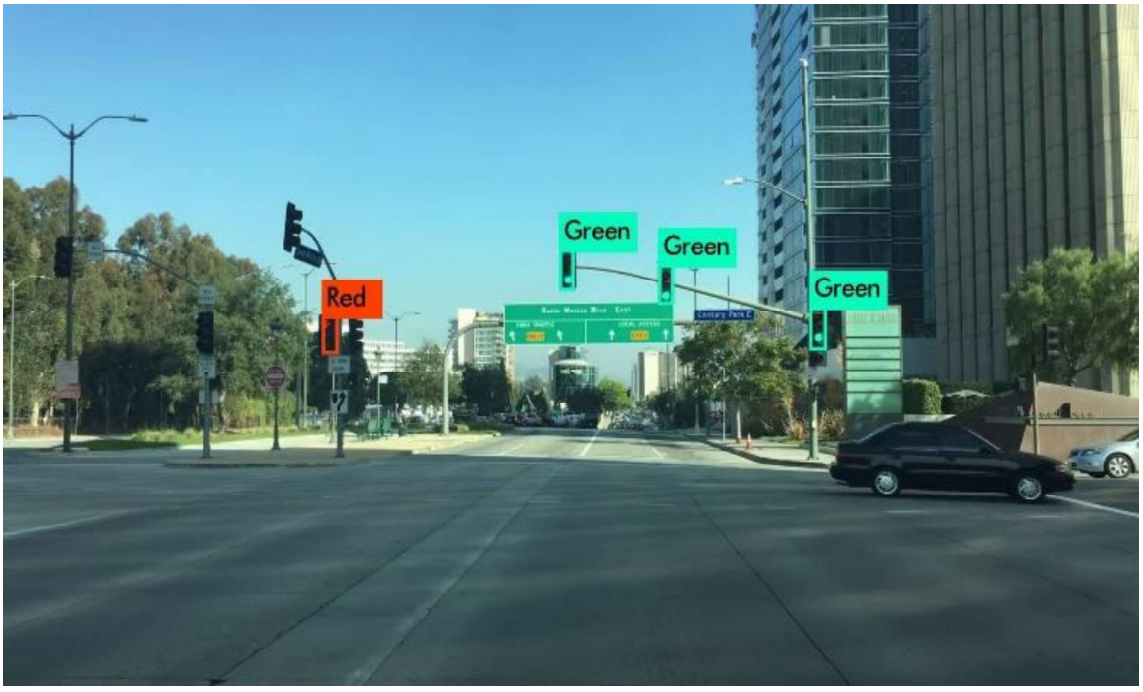

Fig 4.7 YOLO detection from a video

Fig 4.8 YOLO detection from a video

TABLE 4.1 Average Detection Time and mAP

Models	Time (ms)	mAP (MS COCO AP50)
Faster RCNN + Inception V2	743.21	48.64%
Faster RCNN + ResNet 50	2123.48	32.84%
Faster RCNN + ResNet 101	2638.73	45.07%

YOLOv2 (416x416)	159.90	14.92%
YOLOv2 (862x862)	543.32	30.63%
YOLOv3 (462x462)	376.92	29.47%

A similar comparative study was performed for traffic sign detection with TT100K dataset and the results were comparable to that of the results of traffic light detection with Bosch Small Traffic Light Dataset

4.2 Anterior Car Distance Measurement

The first step in-depth estimation involves obtaining the bounding box coordinates of the objects, cars in this case. For our baseline model, we make use of the COCO object detection pre-trained weights and all classes other than 'car' are masked. The centre pixel coordinates of detected objects are extracted and given as input to the next stage.

Before generating the disparity map, certain pre-processing steps are done to make sure that the output is without distortions. Stereo matching algorithms may lead to flawed results sometimes in areas that are occluded, texture-less areas and regions near depth discontinuities. The stereo images are subjected to Gaussian filters with a kernel size of 3. Then the disparity map is generated by the Semi-Global Block Matching (SGBM) algorithm. By comparing the disparity map with the ground truth, we were able to optimize the SGBM parameters in order to get better results. The minimum disparity value is set to default, while the disparity range is specified as 96 and block size set to 7. The output is then passed to the Weighted Least Squares (WLS) filter to smoothen the image while preserving the edges, as a post-processing step. Optimizing the images using a WLS filter turns out to be effective in reducing noise as noted in **[19].** From the

disparity map, the centre disparity values of objects are used to compute the depth of that particular object by using the triangulation geometry discussed above. The distance values (in metre) are then projected on each image frame for real-time depth estimation. The focal length is 718.856 pixels and the baseline width is 0.54 m for the image shown below. Therefore, the formula to obtain distance becomes,

$$Distance = \frac{0.54 * 718.856}{disparity}$$

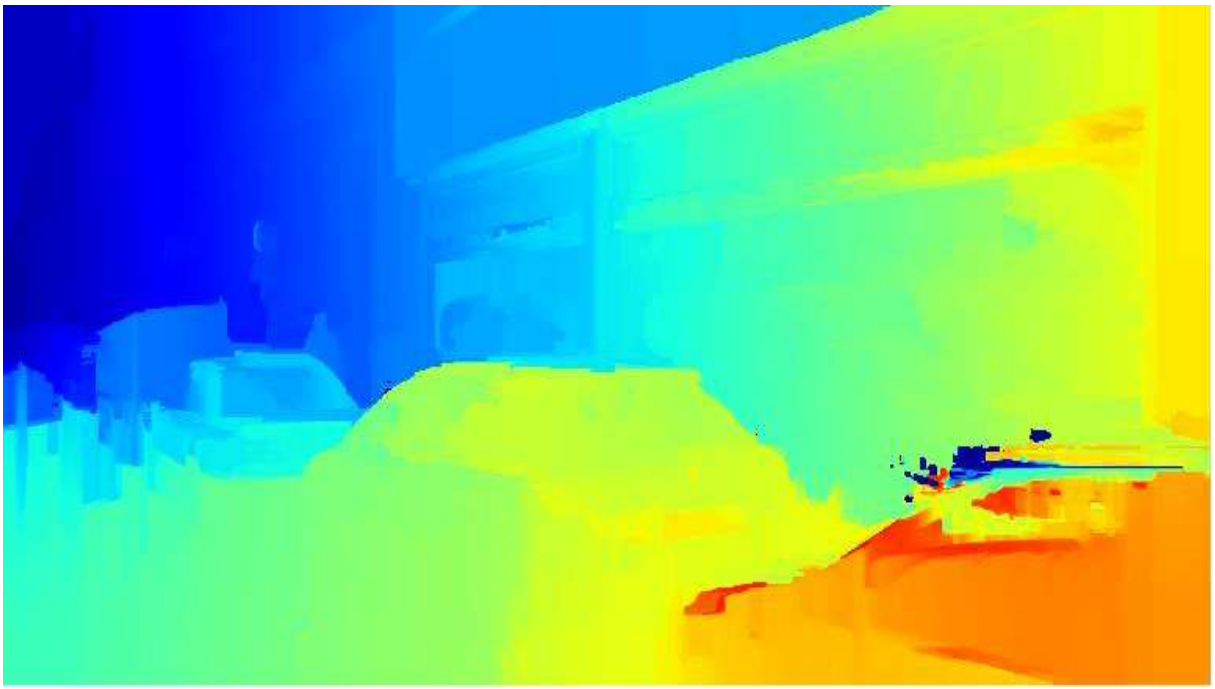

Fig 4.9 Disparity map - RGB

Fig 4.10 Disparity map -Grayscale

Fig 4.10 Ground truth distances

Fig 4.11 Output distances of the algorithm

4.3 Combined Traffic Light, Sign and Vehicle Detection:

Faster RCNN coupled with Inception V2 provides the best speed vs accuracy trade-off. Hence we choose it as our base network for providing us with the detections. The images from the TT100K dataset are resized to 1024x1024 from the original resolution of 2048x2048. Whereas the Bosch Small Traffic Light Dataset is trained with the images in their original resolution (1280x720). In order to compare the performance of the background threshold model, we choose the naive approach of training the model with the combined dataset as our baseline. We first train the baseline model until the loss reaches 0.1 (350k steps). The mAP for the two datasets is found separately. This is done by suppressing the traffic sign detections when measuring the mAP of traffic light detection and vice versa. The mAP for the baseline model is presented in the table below.

Table 4.2 mAP for naïve approach

Dataset	mAP
Bosch Small Traffic Light Dataset	32.16
TT100K Dataset	27.02

The background threshold model too was trained till the loss reached 0.1 (650k steps). The mAP for the background threshold model is presented in the table below.

Table 4.3 mAP for background thresholding approach

Dataset	mAP
Bosch Small Traffic Light Dataset	33.24
TT100K Dataset	22.88

The background threshold model seems to perform better than the baseline model in traffic light detection but fails to achieve the performance of the baseline model in traffic sign detection. But the numbers here might be misleading. Taking a look at the sample detections,

Fig 4.11 Detection by background threshold model

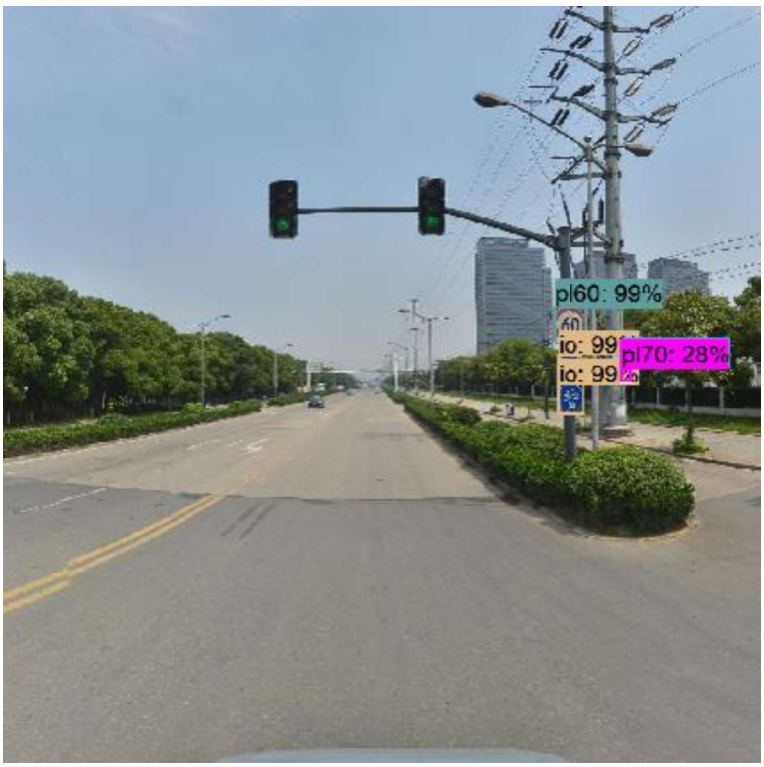

Fig 4.12 Detection by baseline model

The figures above are sample detections by baseline model and background threshold model. The images were taken from the TT100K dataset. The baseline model is good in detecting traffic lights in Bosch Small Traffic Light dataset and is good in detecting traffic signs in TT100K dataset but fails in detecting traffic lights in TT100K dataset and fails in detecting traffic signs in Bosch Small Traffic Light dataset. The background threshold model, on the other hand, produces good cross detections by losing some performance in traffic light detection.

We extend the background threshold model to detect cars as well. To avoid over-representation of the car class, we reduce the number of images from the Udacity dataset to 1000. We then train the background threshold model with the 1000 images for detecting cars. The sample detection is presented below.

Fig 4.13 Combined detection of light, sign and car

Since the main aim of detecting cars is for calculating the distance of the car in front for collision avoidance, the model was trained only to detect the cars right in front, so calculating mAP for this model over the Udacity dataset seems redundant.

CHAPTER 5: CONCLUSION AND FUTURE SCOPE

Our proposed model detects three objects (traffic lights, traffic signs, cars), over fifty classes, which was trained with three different datasets. The background threshold model not only makes detections as accurate as the baseline model but also makes cross-detections i.e., traffic signs in traffic light dataset and vice versa, which the baseline model fails to do. This combined with the fact that our system can measure the distances of the car in front for collision avoidance makes the system stand-out.

We believe with further training by employing data-augmentation techniques, the background threshold will perform better than the baseline model in both cases, rather than just in traffic lights. The same model can also be extended to detect many other objects like trucks, pedestrians, etc., for deploying it as a full-scale autonomous driving system.

References:

[1] Haltakov, Vladimir, Jakob Mayr, Christian Unger, and Slobodan Ilic. "Semantic Segmentation Based Traffic Light Detection at Day and at Night." *Lecture Notes in Computer Science Pattern Recognition*, 2015, 446–57.

[2] "Traffic Lights Detection In Adverse Conditions Using Color, Symmetry And Spatiotemporal Information." *Proceedings of the International Conference on Computer Vision Theory and Applications,* 2012.

[3] Fairfield, Nathaniel, and Chris Urmson. "Traffic Light Mapping and Detection." 2011 *IEEE International Conference on Robotics and Automation*, 2011.

[4] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." *Communications of the ACM 60, no. 6 (2017): 84–90.*

[5] Ouyang, Zhenchao, Jianwei Niu, Yu Liu, and Mohsen Guizani. "Deep CNN-Based Real-Time Traffic Light Detector for Self-Driving Vehicles." *IEEE Transactions on Mobile Computing* 19, no. 2 (January 2020): 300–313.

[6] Behrendt, Karsten, Libor Novak, and Rami Botros. "A Deep Learning Approach to Traffic Lights: Detection, Tracking, and Classification." *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[7] Jensen, Morten B., Kamal Nasrollahi, and Thomas B. Moeslund. "Evaluating State-of-the-Art Object Detector on Challenging Traffic Light Data." *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017.

[8] Luo, Hengliang, Yi Yang, Bei Tong, Fuchao Wu, and Bin Fan. "Traffic Sign Recognition Using a Multi-Task Convolutional Neural Network." *IEEE Transactions on Intelligent Transportation Systems* 19, no. 4 (2018): 1100–1111.

[9] Sermanet, Pierre, and Yann Lecun. "Traffic Sign Recognition with Multi-Scale Convolutional Networks." *The 2011 International Joint Conference on Neural Networks*, 2011.

[10]Aghdam, Hamed Habibi, Elnaz Jahani Heravi, and Domenec Puig. “A Practical Approach for Detection and Classification of Traffic Signs Using Convolutional Neural Networks.” *Robotics and Autonomous Systems* 84 (2016): 97–112.

[11]Pon, Alex, Oles Adrienko, Ali Harakeh, and Steven L. Waslander. “A Hierarchical Deep Architecture and Mini-Batch Selection Method for Joint Traffic Sign and Light Detection.” *2018 15th Conference on Computer and Robot Vision (CRV)*, 2018.

[12]Huang, Jonathan, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, et al. “Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors.” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[13]Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, no. 6 (January 2017): 1137–49.

[14]Hirschmuller, H. “Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information.” 2005 *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (*CVPR*05).

[15]Li, Jianan, Xiaodan Liang, Yunchao Wei, Tingfa Xu, Jiashi Feng, and Shuicheng Yan. “Perceptual Generative Adversarial Networks for Small Object Detection.” 2017 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[16]Meng, Zibo, Xiaochuan Fan, Xin Chen, Min Chen, and Yan Tong. “Detecting Small Signs from Large Images.” 2017 *IEEE International Conference on Information Reuse and Integration (IRI)*, 2017.

[17]Geiger, A., P. Lenz, and R. Urtasun. “Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite.” 2012 *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[18]Huang, Jonathan, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, et al. “Speed/Accuracy Trade-Offs for Modern

Convolutional Object Detectors." *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[19] Liu, Wei, Xiaogang Chen, Chuanhua Shen, Zhi Liu, and Jie Yang. "Semi-Global Weighted Least Squares in Image Filtering." *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.