Occluded Object Detection for Autonomous Vehicles Employing YOLOv5, YOLOX and Faster R-CNN

by

Tanzim Mostafa
18201151
Sartaj Jamal Chowdhury
18201160

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
May 2022

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Tanzim Mostafa	Sartaj Jamal Chowdhury
18201151	18201160

Approval

The thesis titled "Occluded Object Detection for Autonomous Vehicles Employing YOLOv5, YOLOX and Faster R-CNN" submitted by

1. Tanzim Mostafa(18201151)
2. Sartaj Jamal Chowdhury(18201160)

Of Spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 24, 2022.

Examining Committee:

Supervisor:
(Member)

Dr. Md. Khalilur Rhaman
Associate Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Dr. Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Autonomous vehicles [AVs] are the future of transportation and they are likely to bring countless benefits compared to human-operated driving. However, there are still a lot of advances yet to be made before these vehicles can be considered completely safe and before they can reach full autonomy. Perceiving the environment with utmost accuracy and speed is a crucial task for autonomous vehicles, ergo making this process more efficient and streamlined is of paramount importance. In order to perceive the environment, AVs need to classify and localize the different objects in the surrounding. For this research, we deal with the detection of occluded objects to help enhance the perception of AVs. We introduce a new dataset containing occluded instances of road scenes from the perspective of Bangladesh. We utilized transfer learning to train the YOLOv5, YOLOX and Faster R-CNN models, using their respective pre-trained weights on the COCO dataset. We then evaluate and compare the performance of the three object detection algorithms on our dataset. YOLOv5, YOLOX, and Faster R-CNN achieved mAP at 0.5 metric of 0.777, 0.849 and 0.688, and mAP at 0.5:0.95 of 0.546, 0.634, and 0.422 respectively in our test set. Therefore, we find YOLOX to be the best performing model on our dataset, and its high mAP scores demonstrate the effectiveness of the model as well as the dataset.

Keywords: Autonomous Vehicles; Occluded Object Detection; Object Detection; Machine Learning; Deep Learning; Supervised Learning; Occluded Objects Dataset; Transfer Learning; YOLOv5; YOLOX; Faster R-CNN

Dedication

We dedicate this thesis to our parents and siblings.

Acknowledgement

First and foremost, all praise to the Almighty Allah for whom we have managed to complete our thesis smoothly.
Secondly, we thank our supervisor Dr. Md. Khalilur Rhaman and co-supervisor Dr. Md. Golam Rabiul Alam, both of whom have helped us immensely throughout this journey. We are especially grateful towards Dr. Md. Khalilur Rhaman for always keeping us inspired and motivated. And, we are grateful towards Dr. Md. Golam Rabiul Alam for everything he has taught us through the courses we took under him and throughout our thesis.
Moreover, we sincerely express our gratitude towards all our respective teachers, companions, and staff for all the support throughout.
Finally, we are forever indebted to our parents for their continuous support and prayers. It is only because of them we got this amazing opportunity to learn and complete our undergraduate degree.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Although autonomous vehicles have become popular to most of us in recent years, the first evidence of driverless vehicles dates back to 1925, when the "American Wonder" was demonstrated by Houdina Radio Control. This was a remote-controlled car being operated by a person in a vehicle just behind it, as it was driven along Broadway in New York City [6]. Then in 1995, a major milestone was reached by Carnegie Mellon University's Navlab team when they drove from Washington, D.C., to San Diego, CA, with 98% automation using manual longitudinal control [6]. From the 1990s to the early 2000s, there were several other advancements in this field.

In 2012, for the first time, researchers around the world could evaluate their progress on the many different self-driving perception problems in an effective way since the KITTI Vision Benchmark was made accessible to the public [1]. It was also at this time that deep learning started to make a profound impact on several fields such as computer vision and robotics. This created a fundamental base for notable breakthroughs in the perception elements of AVs, in terms of robustness, accuracy, run-time, etc.

In 2014, a classification of autonomous driving systems was introduced by the Society of Automotive Engineers (SAE). It comprises six levels of SAE autonomy, where level 0 means no autonomy and level 05 means full autonomy.

From then on, self-driving cars by Mercedes, Tesla, Google's Waymo, among others, were released. Autonomous vehicles or self-driving cars have the potential to bring about a revolutionary impact on society and the way people travel. They provide many benefits such as more safety, reduced carbon emission, reduced traffic congestion, greater independence such as for people with disabilities, the elderly, and so on [10]

In recent years, we have seen numerous advances in autonomous vehicles. For example, we all know about the autopilot feature in Tesla cars through which it is able to steer, accelerate and brake automatically in its lane. Although, the car is not fully autonomous and requires some driver supervision. Similarly, there exist several other companies that are in the process of building their own self-driving cars. It is therefore without a doubt that fully self-driven vehicles are on the horizon.

Autonomous vehicles make use of certain sensors such as radar, lidar, and cameras in

order to perceive the environment around them. Radars are cheap, ultrasonic, and work effectively in extreme weather conditions. Lidar is the most powerful sensor and provides very accurate depth data, but it is very expensive [20]. Cameras work well in clear, good lighting conditions but they have substandard performance in darkness and extreme weather and are bad at depth estimation.

There exist two different philosophies among Tesla and Waymo, two companies that are at the forefront in the development of self-driving technology, regarding their main perception sensor. While Waymo's driving system is more dependent on Lidar sensors, Tesla argues that cameras are sufficient and don't use Lidars in their cars [16].

There are still several challenges that have to be tackled before self-driving cars can reach full autonomy. Among them, being able to accurately perceive the environment around them and detect occluded objects, is a significant one.

1.1 Research Problem

Deep learning is a subset of Machine Learning (ML) and Artificial Intelligence (AI). It is a multi-layered computational model that is used to automatically extract features and patterns from raw data, such as images, and predict, or take actions based on a certain reward function. Techniques such as neural networks, hierarchical probabilistic methods, supervised and unsupervised learning models, and deep reinforcement learning (DRL) are all subtypes of DL. DL has recently become extremely popular because of the remarkable results achieved in the fields of object detection, natural language processing, image classification, etc. There are mainly two causes behind the current rise of DL: the availability of large datasets and high-performance Graphics Processing Units (GPU) [9].

Autonomous vehicles have to detect various objects in the road in order to achieve a complete view of the environment. Deep learning-based perception, mainly Convolutional Neural Networks (CNNs), is the existing standard for object detection and recognition. Various types of neural network architecture are used for object detection as segmented areas in images based on pixels or 2D regions of interest, 3D bounding boxes in lidar point clouds, and 3D representations of objects using both camera and lidar data [16].

Image data is more suitable for object recognition as it contains richer information, and this is what will be used primarily in our research. Although, in this case, the 3D positions of the detected objects in the real world have to be estimated because the depth information is lost when the imaged scene is projected onto the imaging sensor. In the case of 2D object detection, single-stage and double-stage detectors are the most popular architectures [16].

According to [20], some of the perception intricacies that require further research are:

- Partial view and angled view
- Occluded object

- Many objects in the surrounding
- Similar objects at different distances
- Various lighting conditions depending on the time of day
- Road conditions such as unclear road markings and slippery road
- Adverse weather conditions such as rain, fog, snow, etc
- Changing traffic lights
- Lane markings that have faded
- Requirement for invariances such as rotational invariance, translation-invariance, size, and color invariance

To briefly explain some of these complexities and more:

When similar objects are at different distances, it seems like they are of distinct sizes, when actually they belong to the same class category [7]

Adverse weather conditions such as rain, fog, or snow, majorly degrade the performance of object detection methods. This is because the quality of visual signals that are sensed by self-driving cars is reduced in such conditions. Solving the effect of rain in the context of autonomous driving, especially, is an area that is popular among various research communities [17]

Transformations such as translation, rotation, scale, etc, of an object in an image, make it difficult to recognize or detect an object. Such a transformation problem is also known as the "invariance problem" and there is still much work to be done before we are able to find a strong and scalable solution to this problem using machine learning [5].

Illumination has a significant impact on detecting objects. For example, cameras are unable to capture good-quality images in low light conditions. Most of the current object detection methods do not have high accuracy in poor lighting conditions [13].

Detection of small objects is challenging because it is difficult to differentiate the objects from the background and similar objects. Also, the small object can be placed at a great number of possible locations on a given image, so accurately locating the objects requires more powerful computation. Moreover, knowledge on small object detection is very limited because most previous efforts were on large object detection problems [18].

Occlusion can be present in various ways and can range from partial occlusion to heavy occlusion. In the case of an autonomous driving environment, target objects can be occluded by static objects such as lampposts. Moving objects such as cars may occlude each other, or even self-occlude, such as when parts of a cyclist overlap [11]. Pedestrian detection is a part of occluded object detection and is still a complex problem because of the diversity of occlusion patterns [12].

While many of the object detection architectures have already achieved amazing accuracy and extremely low error rates of less than 5%, on datasets such as Pascal VOC and ImageNet, another problem is the speed of these architectures in producing low error rates in real-time in the case of autonomous driving [20].

So clearly, there is room for improvement in a lot of areas regarding object detection in autonomous vehicles. In this paper, we are primarily going to focus on overcoming the problem of occluded object detection while also fulfilling real-time requirements.

1.2 Research Objectives

This research aims to develop a dataset that can facilitate further advances in occluded object detection for autonomous vehicles. Furthermore, we investigate the performance of several avant-garde object detectors when applied to our dataset. The objectives of our research are as follows:

1. To deeply understand the perception problem in autonomous vehicles.
2. To intensively explore how deep learning can be used for object detection in autonomous vehicles.
3. To develop a dataset that can be leveraged to evaluate, and help improve the performance of miscellaneous deep learning models for occluded object detection.
4. To specifically evaluate and compare the YOLOv5, YOLOX and Faster R-CNN models on our dataset.
5. To further analyze and offer suggestions on how to improve the overall performance.

Chapter 2

Literature Review

Autonomous vehicles have numerous benefits and are likely to bring a far-reaching impact on the way people travel. Companies such as Tesla, Waymo, Uber, nuTonomy, etc, predict a future with self-driving cars by the next 15-20 years [20].

However, there still exist significant challenges before we can completely rely on self-driving cars and before they can reach full autonomy. One of these challenges is being able to perceive the environment with high accuracy even in complex situations. Object detection falls under perception and autonomous driving requires very high detection accuracy as well as speed in the given context [14].

2.1 Deep Learning for Object Detection

Most of the recent advances in object detection are accredited to advances in deep learning. More specifically, CNN-based object detection algorithms are most popular for detecting objects, rather than classical detection algorithms [12]. While these algorithms already work well in many cases, when it comes to occluded object detection, most of these models have been unable to produce very good results.

2.2 Deep Learning based Object Detection Architectures

Single-stage and double-stage detectors are the architectures used most often for 2D object detection [16]. Single-stage detectors only need a single pass through the neural network and predict all the bounding boxes in one go. For example, You Only Look Once (YOLO), Single Shot MultiBox Detector (SSD), RefineNet, etc. Double stage detectors divide the process of object detection into two sections: region of interest candidate proposals and bounding boxes classification. Examples include RCNN, Faster-RCNN, R-FCN, etc.

2.3 Related Works

This part aims to critically review previous relevant work regarding occluded object detection in autonomous vehicles. We analyze the different techniques used to

achieve better results and show that there are still a lot of challenges that need to be met regarding accuracy, speed, etc.

In the paper [14], the authors used Multi-Scale CNN(MS-CNN) as a baseline network and added 3 enhancements to overcome the occluded object detection and large object scale variation problem. Their network follows 2 stage object detection architecture. According to the paper, the classical non-maximal suppression (NMS) method, which is used to select a single bounding box out of overlapping proposals, has shortcomings when it comes to detecting occluded objects. Hence, they used another method called soft-NMS, at object proposals from various feature output scales so that they can achieve a balance on the quality and number of object proposals. They evaluated their system over the KITTI 2D object detection dataset and were able to achieve better object detection performance with almost no increase in detection time. For example, it was first for the pedestrian detection category "Easy" and second place for "Moderate" and "Hard".

The research work [12], proposes some improvements in the feature extraction subnet and detection subnet, in both two-stage and one-stage architectures, which can lead to better object detection. According to the authors, in terms of improving the feature extraction subnet, adding more features such as detailed and context features, can help detect small objects and occluded objects with higher precision. As an example they reviewed HyperNet to add more detailed features and Inside-Outside Net(IONNET) or Multi-Scale Deep Convolution Neural Networks(MS-RCNN), to add context features. In terms of improving the detection subnet, they propose adding output branches other than classification and localization and using Soft-NMS or IoU-Net instead of the classical NMS. To talk more specifically about occluded object detection, they also suggest that data augmentation can be used to increase the number of occlusion examples and make the system more robust to occluded object detection. Other than that, adding context features, relation features, and improving the loss function, such as using Repulsion Loss, can all help in achieving higher accuracy in detecting occluded objects.

The paper [8] uses an object detection network consisting of two modules. One of which is an object detection framework that performs classification and bounding box regression. The other one uses multiple OBB (Object Bounding Box) Critic networks which divide object areas and occlusion areas. Furthermore, they used the Faster R-CNN Object Detection framework as a base method and evaluated it on the KITTI Vision Benchmark Suite Dataset. The proposed method detected hidden or occluded objects with greater accuracy compared to the Faster R-CNN methods. However, the paper did not focus on comparing the computation time and memory consumption of the different object detection methods used, even though the running time and memory complexity are important factors for object detection in road scenes in real-time.

The research [22] designed a good dataset namely Dhaka AI, which covers a vast amount of Dhaka traffic data. However, a dataset heavy with occlusion instances from the road scenes of Dhaka, and specifically made to simulate the perception of autonomous vehicles is unattainable.

From the above discussion, it is observed that although some of the models proposed were able to achieve better detection accuracy for occluded objects, there is still space for much improvement. Another particular challenge that arises in proposing some of the enhancements to address the occluded object detection problem is that the network becomes too complex and so the detection time increases due to more expensive computation requirements. Therefore, in this research, we proceed to propose a novel dataset that can be leveraged in the evaluation of different deep learning models and measure their performance on the aforementioned problem. We evaluate two of the latest single stage detectors namely, YOLOv5 and YOLOX, which fulfills our real-time requirement needs. We also evaluate one double stage detector, namely Faster R-CNN, for comparison purposes. We then find the best performing object detector for our dataset.

Chapter 3

Methodology

In order to effectively describe our research methods, we have divided this section into three parts, namely, workflow, dataset, and model.

3.1 Workflow

Our research involves producing an adequate dataset that would aid autonomous vehicles in occluded object detection. Figure 3.1 portrays the overall workflow of our methods. First, we obtain data, then handpick images of occluded instances, annotate those images, split the dataset into train, validation and test sets, preprocess and augment them, as shown, which will be further elaborated in the dataset subsection. After that, we train the three models and tune the model parameters. We evaluate and compare their results based on the test sample. Finally, based on our results we plan to bring further enhancements to the dataset.

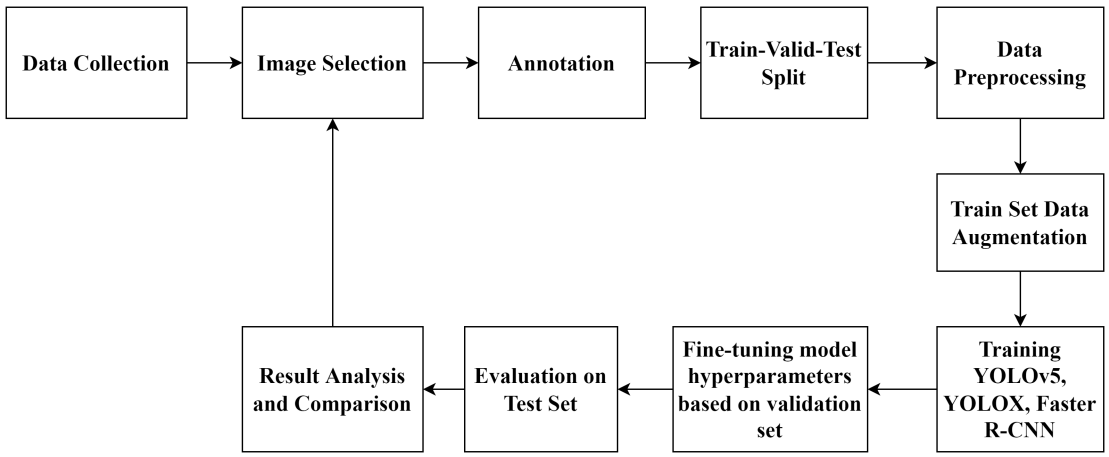

Figure 3.1: The Overall Workflow

3.2 Dataset

To design our dataset we had to carry out some groundwork for data collection. Also, we had to come up with a number of decisions on how to process the data, acquire the most relevant images, estimate a sensible volume of data to be annotated, etc.

After manually annotating the images we had to determine how to split, preprocess, and augment the data. In what follows, we describe the various steps and decisions made on designing the dataset.

3.2.1 Data Collection

At the same time-stamp, we recorded road scenes from the busy roads of Dhaka from 3 different vehicles: an SUV, a wagon, and a minivan. We recorded the road scenes for 1 hour and 20 minutes, while each of the cars maintained different lanes and remained abreast as much as possible. Also, we did maintain a unique camera angle from each vehicle, but we used the same type of camera (iPhone 11's back-facing camera in HD 30 fps mode) on each of them. Therefore, we got data from 3 different perspectives, resulting in a total of 4 hours of video data. Our goal was to simulate the perception of autonomous vehicles. Subsequently, we recorded scenes from 3 distinctive angles of the forward direction, as shown in Figure 3.2. This helped us have a more diverse dataset with miscellaneous occlusion instances of the same objects from various directions.

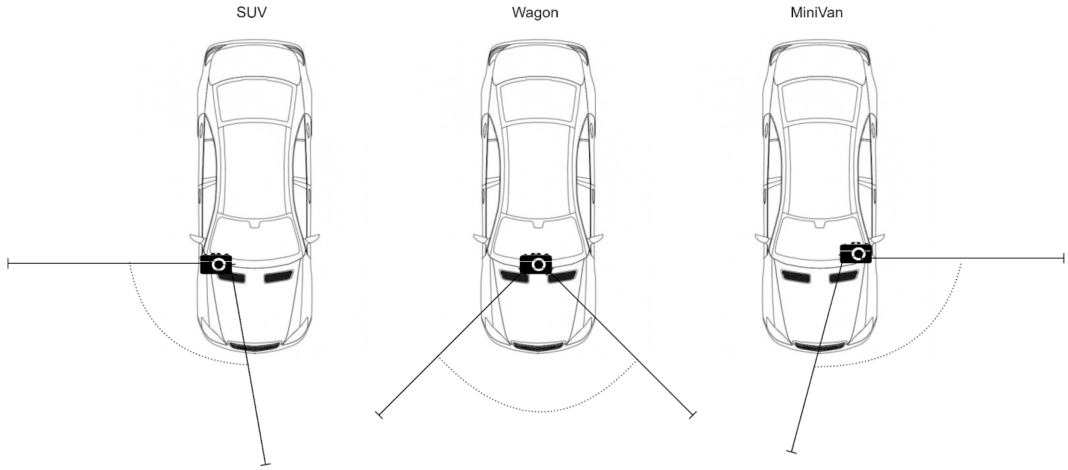

Figure 3.2: Three unique camera angles for each car used to collect data

Figure 3.3 demonstrates three images from the three different angles that were covered by each of the cars.

Figure 3.3: Three unique perspectives from the three different cars

3.2.2 Image Selection

We converted the videos to images on 10 frames per second basis. After that, we manually selected images consisting mostly of occluded instances of every class of objects. Most images for example had relatively faraway objects (cars, rickshaws, etc) blocked by more nearby objects. Moreover, we made sure that a multitude of occlusion levels was present within the dataset. We can see this in figure 3.4 below illustrating some different levels of occlusion.

(a) Lightly Occluded Bus (a) Partially Occluded Motorcycle (a) Heavily Occluded Bus

Figure 3.4: Examples of images with different levels of occlusion on certain classes of objects

3.2.3 Annotation

We used 2D bounding boxes for annotation because our goal was to detect multiple occluded objects from all the images. We annotated a total of 1,502 images, with a total 14,364 labels. We used the online tool Roboflow Annotate because it is efficient and provides miscellaneous image dataset preparation functions. After generating the dataset, an API call was made from Google Colaboratory to train the dataset using the 3 different models.

3.2.4 Labeled Classes

We annotated 8 different object classes, by maintaining a balance between the typical ones. We made sure to cover a large part of each image so that there were not many void classes, while also trying to reduce our endeavors to annotate due to time constraints. However, we did carefully annotate every instance with occlusion. Table 3.1 shows the list of different classes of objects we labeled.

Class	Description	# of Labels
car	All four-wheelers except buses.	4523
person	Both pedestrians and visible riders of certain vehicles.	3984
cng	A common three-wheeled vehicle in Dhaka that is similar to an auto-rickshaw.	1348
motorcycle	All types of motorcycles. The rider is not included in the bounding boxes but rather labeled as a "person"	1275
bus	Only buses.	1221
rickshaw	Only rickshaws.	971
bicycle	All types of bicycles. The rider is not included in the bounding boxes but rather labeled as a "person".	537
other-vehicle	Any three-wheeler or two-wheeler except CNG, rickshaw, motorcycle, and bicycle.	505

Table 3.1: List of different classes of objects we labeled

The annotation heatmaps in 3.5 illustrate the locations of all the annotations of each of the classes in our dataset so it can be seen how they are distributed, almost all over the area of the images. However, objects which are underrepresented have a heatmap with more cold colors and blank regions. It can be improved by increasing the images of the objects underrepresented. Also, the last heatmap shows the distribution of all the 8 classes of objects together.

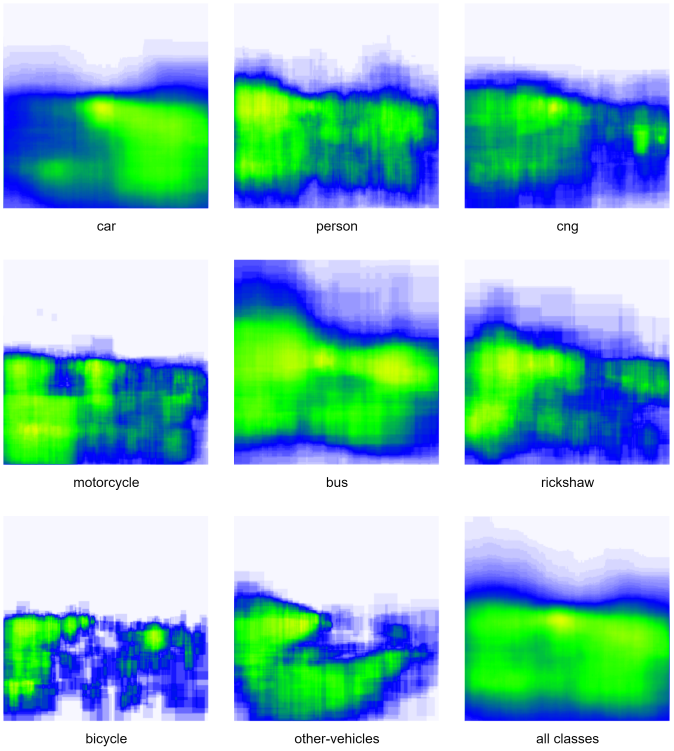

Figure 3.5: Annotation Heatmap

3.2.5 Train-Valid-Test Split

We decided to split the data into 60% training, 20% validation and 20% test. We used the validation set to help tune the hyper-parameters of our models to achieve better performance. We used the test set to find the final performance of the models.

3.2.6 Preprocessing and Training Data Augmentations

In order to reduce the training time of the model, we resize the images from 1920*1080 to 640*640. Generally, machine learning models can train faster on smaller images. However, we had to make sure the images were not too small and through experimentation, we concluded that 640*640 pixels would be a good fit.

We augmented the training data so that we could feed our model with a further variety of instances. First, by incorporating a horizontal flip to all the images. Then, adding grayscale versions of 50% of them because grayscale images can result in higher accuracy classification across a multitude of different classifiers according to [2]. We changed the brightness of the images by both brightening and darkening the original image by 25%, to make our model more resilient to different camera settings and lighting. Finally, we leveraged images with blur up to 1 pixel since [21] helps justify that manipulating color, contrast, and blur can help the model gain abilities to generalize noise patterns in the train set samples that were not seen before. After augmenting the training set, it became 4476 images. The validation set contained 301 images and the test set 303 images.

The YOLOv5 PyTorch format of the dataset used for YOLOv5 can be downloaded from: https://app.roboflow.com/ds/KfpWA5WE78?key=uNteipYfed

The Pascal VOC format of the dataset used for YOLOX can be downloaded from: https://app.roboflow.com/ds/1dVQ3pv1Tg?key=LBEJD5a5FO

The COCO JSON format of the dataset used for Faster R-CNN model can be downloaded from: https://app.roboflow.com/ds/0aEB8da395?key=12lGj1Ibzp

3.3 Model

As mentioned before, there exist several state-of-the-art deep learning-based object detection models. These models can mainly be divided into two types: two-stage detectors and one-stage detectors. Two-stage detectors include Faster R-CNN, Mask R-CNN, etc, while one-stage detectors include YOLO, SSD, and so on.

The main difference between two-stage and one-stage detectors is in accuracy and speed. In two-stage detectors, the entire training occurs in two steps. In the first step, various regions of objects are extracted and in the second step the objects are classified and the localization of the objects is made more accurate. Because of the two stages, such detectors are generally more accurate but are computationally more expensive. On the other hand, in one-stage detectors, there is no separate stage for the extraction of different regions. Instead, a fixed number of predictions are

made on the grid and the whole detection happens in one go. This makes one-stage detectors generally faster than two-stage detectors, although with a slight decrease in accuracy. Hence, they are also more suited for real-time detection tasks.

For our object detection task, we decided to use the YOLOv5, YOLOX and Faster R-CNN models. YOLOv5 and YOLOX are one stage object detectors and are the models suitable for our object detection task. This is because our object detection is for the use of autonomous vehicles where fast detection speed is crucial. While there exist other single-stage detectors like SSD, EfficientDet, we chose these two models, as they are one of the fastest models that exist right now and also very accurate. In the paper[15], it says that YOLOv4 runs two times faster than EfficientDet but with similar performance. YOLOv5 by Glenn Jocher comes after YOLOv4, but has been released without a paper alongside. Glenn Jocher first implemented a version of YOLOv3 in Pytorch and had been constantly making improvements to the architecture. Eventually, YOLOv5 was officially released by a company called Ultralytics on 25th June 2020, which has been shown to be better in terms of speed and precision than YOLOv4 in the COCO benchmark. YOLOX is a newer model than YOLOv5, released in 2021 and has been shown to perform better than YOLOv5 [19]. We also evaluated the Faster R-CNN model which was released in 2015. Although Faster R-CNN is not suitable for real-time speeds, we wanted to see how a two-stage detector performs in comparison to the other models.

3.3.1 YOLOv5

Since no paper has been released about YOLOv5, the exact architecture used is still not completely known. Hence, in the following, we give a basic idea about how the YOLO model works. Then, we explain the changes of YOLOv5 in comparison to YOLOv4, since it resembles closely to YOLOv4 architecture.

The YOLO model treats the object detection task as a single regression model. It is based on a backbone model whose role is to extract meaningful features from the image that will eventually be used by the final layers, as we can see in figure 3.6. For example, the original YOLO model used a modified version of GoogleNet as the backbone, which was changed to DarkNet networks in the later models by Joseph Redmon.

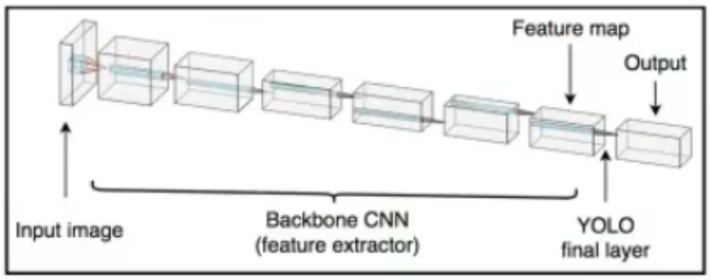

Figure 3.6: Basic structure of the YOLO model

First, the input image is divided into an S*S grid. Every grid cell is then responsible for detecting objects that appear in them. For instance, if the center of an object

appears within a grid cell, then it is the job of that cell to detect it.

Each grid cell gives a number of bounding boxes, which is a rectangular outline that is used to focus on the object present in an image. Then, the job of each cell is to predict the following for each bounding box: the center of the box, the height and width of the box, the probability that an object exists in this box, and the class of that object. By class, we mean that it could be a cat, dog, person, etc. It uses a single bounding box regression in order to predict the center, width, height, and class.

After that, it uses the concept of Intersection over Union (IoU) to remove bounding boxes lower than a specific threshold. IoU in object detection determines how the bounding boxes overlap. If the bounding box given as output is equal to the ground truth bounding box, IoU is 1 or maximum. The less the output box and ground-truth box match, the lower the IoU value. This concept is used to get rid of bounding boxes that are further apart from the real box.

The algorithm also uses a technique called Non-Max Suppression (NMS). There may still be multiple detections made for the same object. NMS technique makes sure that only one bounding box is provided for each object.

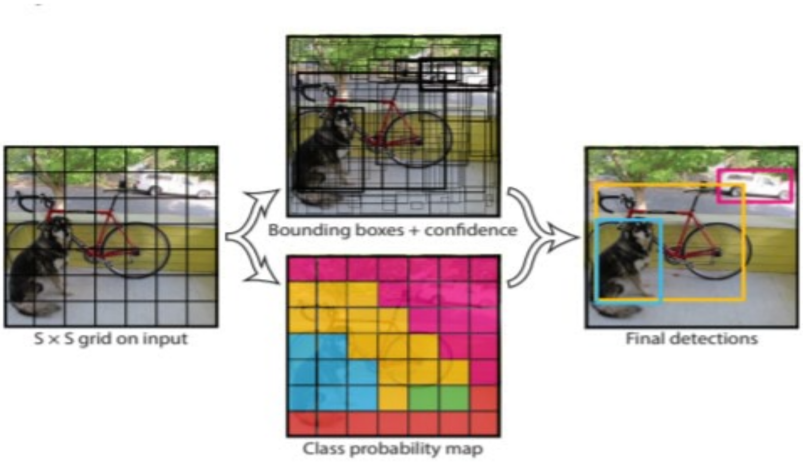

Figure 3.7: Summary of the process of YOLO algorithm

Figure 3.7 taken from [3] depicts how everything works. As we can see, in the end, we are left with perfectly fitted bounding boxes for the dog, bicycle, and car classes. It also outputs the probability with which each class is detected.

A significant part of the process is the loss function. During training, the YOLO algorithm optimizes the loss function in Eq. (3.1):

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \ \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2\right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \ [\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2\Big] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \ \left(C_i - \hat{C}_i\right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{noobj}} \ \left(C_i - \hat{C}_i\right)^2 \\
&+ \sum_{i=0}^{S^2} 1_{i}^{\text{obj}} \sum_{c \in \text{ classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2
\end{aligned}
\tag{3.1}
$$

To explain a little about the loss function, we divide it into three parts. Eq. (3.2) demonstrates the primary part.

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \ \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2\right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \ [\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2\Big]
\end{aligned}
\tag{3.2}
$$

It represents the bounding box loss. It basically aids the network to learn the weights which are used to predict the bounding box coordinates and size.

The next part is in Eq. (3.3):

$$
\begin{aligned}
&\sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \ \left(C_i - \hat{C}_i\right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{noobj}} \ \left(C_i - \hat{C}_i\right)^2
\end{aligned}
\tag{3.3}
$$

It represents the object confidence loss. This is responsible for teaching the network to learn the weights so that it can predict whether or not a bounding box contains an object.

The last part of the equation is given in Eq. (3.4):

$$
\sum_{i=0}^{S^2} 1_{i}^{\text{obj}} \sum_{c \in \text{ classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2 \tag{3.4}
$$

This represents the classification loss. This makes sure that the network learns to predict the correct class for each of the bounding boxes.

This is just a brief description of how the YOLO model works. The newer versions of the model introduced several new features that have helped the models become faster and more accurate. The YOLOv5 architecture has a lot of similarity to the YOLOv4 architecture. Therefore, in the following we describe some of the known changes in YOLOv5 compared to YOLOv4.

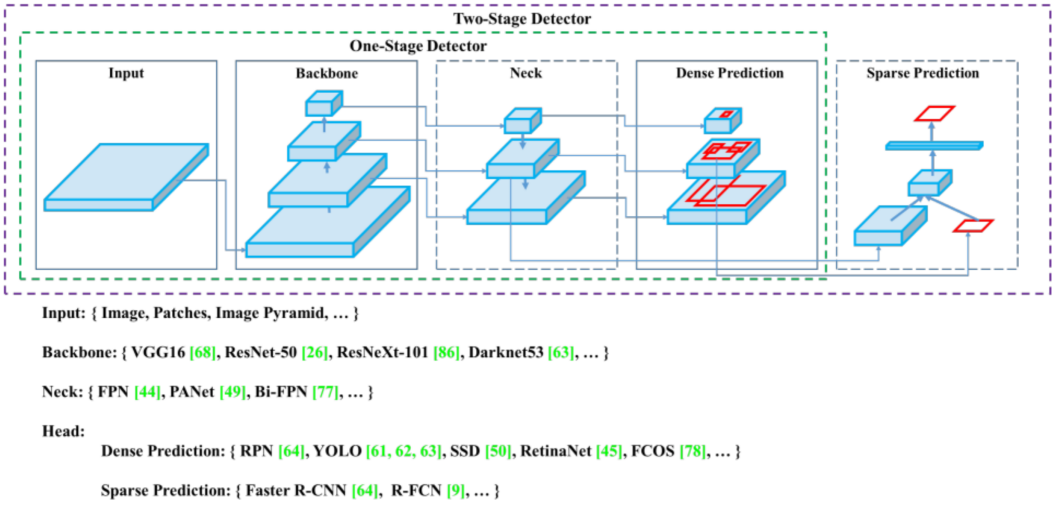

Figure 3.8: General structure of an Object Detector

Figure 3.8 taken from [15] shows the basic structure of an object detector. The backbone is used for pre-training and the head is used to predict the classes and bounding boxes. This head can be one-stage such as for YOLO, SSD, and so on, for Dense Prediction. It can also be for two-stage such as R-CNN, Faster R-CNN, etc, for Sparse Prediction, as illustrated in Figure A. Modern object detectors have some layers called the Neck in between the backbone and head, whose job is to store feature maps.

In YOLOv4 architecture, CSPDarknet53 is used as the backbone and SPP block is appended which helps increase the receptive field, isolates the features which are significant and results in almost no decrease in network operation speed. For parameter aggregation Persistent Appearance Network (PAN) is used and the same head of anchor-based YOLOv3 is used in YOLOv4. YOLOv4 also consists of new data augmentation methods. One of them, called Mosaic, mixes four training images. The other one called Self Adversarial Training (SAT), works in two forward and backward steps. In the first step, the image is altered by the network, keeping the weights the same, and in the next step, it is trained to detect an object in this modified image. Other than the ones mentioned, several existing methods such as PAN, Cross Batch Normalization (CBN) and Spatial Attention Module (SAM) have been changed which has led to better performance [15].

The YOLOv5 architecture consists of similar parts as described before of YOLOv4. A significant point in YOLOv5 is that it automatically learns anchor boxes from the distribution of bounding boxes in the custom dataset, instead of them being defined beforehand. It does so using K-means and genetic algorithms. This is significant

as the locations and distribution of bounding box sizes may be very different from the anchors defined beforehand in the COCO dataset. The main difference is that while YOLOv4 was released in the Darknet framework written in C, YOLOv5 uses the PyTorch framework. Moreover, for configuration YOLOv5 uses a .yaml file, while YOLOv4 uses a .cfg file. The primary difference between .yaml and .cfg file formatting is that, the .yaml file identifies the various layers in the network and after that the number of layers present in the block is multiplied by it.

Finally, figure 3.9, taken from the github repository of YOLOv5 by ultralytics, shows the comparison of speed and accuracy of various versions of YOLOv5 and EfficientDet. In general, the further the curve is to the left, the faster is its speed and the higher the curve is, the more is its accuracy.

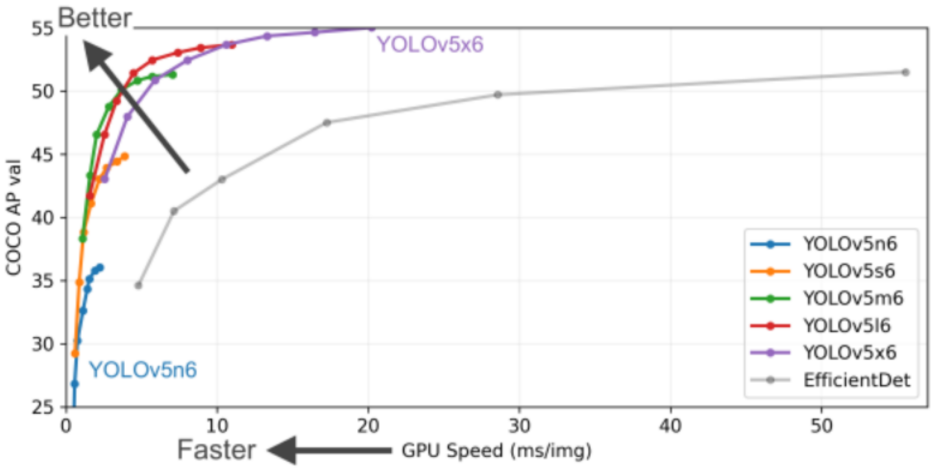

Figure 3.9: Speed-accuracy tradeoff of several YOLOv5 versions and EfficientDet.

3.3.2 YOLOX

We have already described how the basic YOLO algorithm works and about YOLOv5. So, in this part, we describe the main parts of the YOLOX model. While YOLOv5 is a relatively new algorithm, YOLOX is a newer algorithm and comes with some key changes. The algorithm comes with the recent attention in developing detectors that are anchor free, are end-to-end and have the latest label assignment strategies. It develops on the usage of YOLOv3 and Darknet53 by taking it as its baseline. Some of these changes are to be briefly defined in the following.

Data augmentation is a part of the algorithm which augments a dataset with limited data but without adding any new data. In YOLOX, in the data augmentation stage, ColorJitter, MixUp, multi-scale and RandomHorizontalFlip are used. This is in contrast to RandomResizedCrop which was used in the preceding versions of YOLO.

For feature extraction, the previous YOLO versions used a coupled head in its backbone. However, according to new research coupled heads lead to lower performance.

As a result, the YOLOX algorithm incorporated a decoupled head, where localization and classification processes are separated, and this led to a higher convergence speed. Figure 3.10 taken from [19] shows the new decoupled head architecture in contrast to the coupled head used in the previous versions of YOLO.

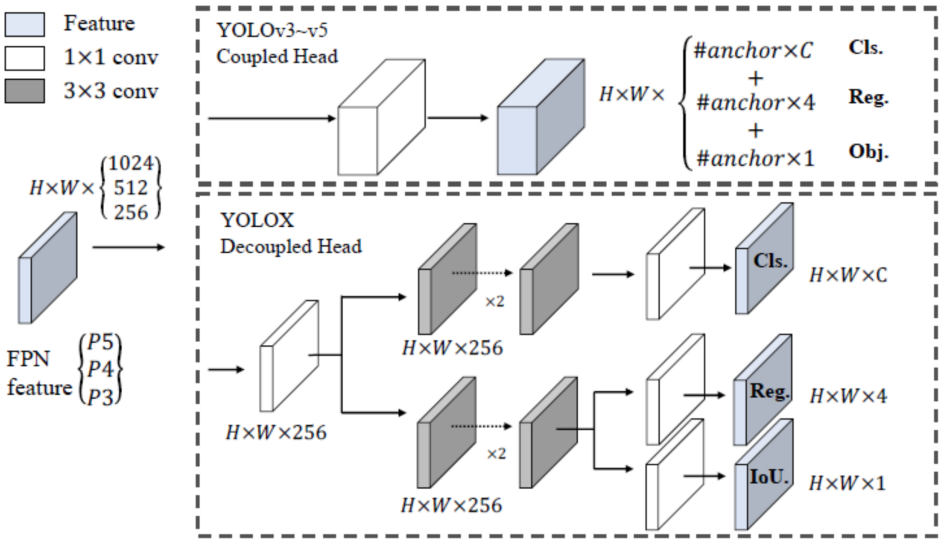

Figure 3.10: Difference between coupled head and decoupled head architecture.

Figure 3.11, also taken from [19], shows the better performance of decoupled head and its higher convergence speed compared to YOLOv3 head.

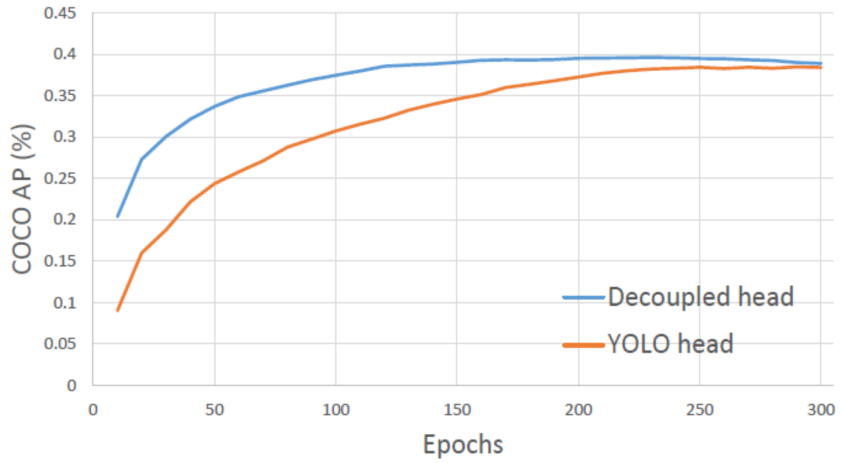

Figure 3.11: Training graphs for detectors with YOLOv3 head and decoupled head on COCO dataset.

Previous versions of YOLO such as YOLOv3,YOLOv4 and YOLOv5 all have anchor-based structures. This has been found to have several drawbacks which eventually leads to wastage of time. YOLOX instead implemented an anchor free system which has helped it increase its speed and also reduce the number of parameters it

requires. The algorithm makes only one prediction for each location instead of three like before and chooses one positive sample for each object keeping consistency with the YOLOv3 baseline. Therefore, it disregards any other predictions, no matter if they are of higher quality.

In the development of YOLOX, they found that there are four crucial parts in advanced label assignment: loss or quality aware, center prior, dynamic number of positive anchors for each ground-truth or dynamic top-k, and global view. To this end they found Optimal Transport Assignment(OTA) fulfills all these requirements and developed SimOTA.

Finally, in order to guarantee that the YOLOX model works in an end-to-end fashion, it appends two extra convolutional layers. The first of this is a one-to-one label assignment and second is stop gradient. Nevertheless, these are left as optional additions and are not in the actual model as they were found to reduce the inference speed as well as performance to some degree.

In this way, YOLOX is the brand new high-performance object detection algorithm. In short, it consists of three significant innovations: a decoupled head, anchor-free structure and advanced label assignment procedure. It has achieved better results in terms of both speed and accuracy than all of its counterparts. Figure 3.12 shows a comparison of YOLOX with some of the other state-of-the-art object detectors in terms of speed and accuracy. It also attained first place in Streaming Perception Challenge which is a part of the Conference on Computer Vision and Pattern Recognition (CVPR) 2021 Workshop on Autonomous Driving (WAD) [19].

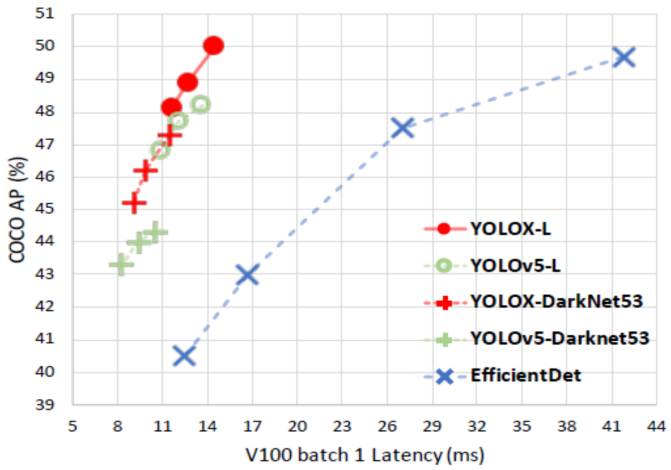

Figure 3.12: Speed-accuracy tradeoff of YOLOX and some other state-of-the-art models.

3.3.3 Faster R-CNN

Faster R-CNN is a deep convolutional neural network that appears to the user as a single, end-to-end, and fully integrated network for object detection. In order

to locate an object in an image using a hypothetical approach, Region Proposal algorithms are commonly used by models such as the Fast R-CNN. However, such algorithms require heavy computation. According to [4], Faster R-CNN blends the Region Proposal Network and Fast R-CNN to form one quality, and inexpensive network. It can propose regions with higher accuracy by using an "attention" technique, which makes the process of localizing faster.

From [4] we also learnt that this model was a big breakthrough as it achieved highest accuracies on the MS COCO, PASCAL VOC 2007, and 2012 datasets. In 2015, it also contributed to winning first places in many categories of ILSVRC and COCO competitions.

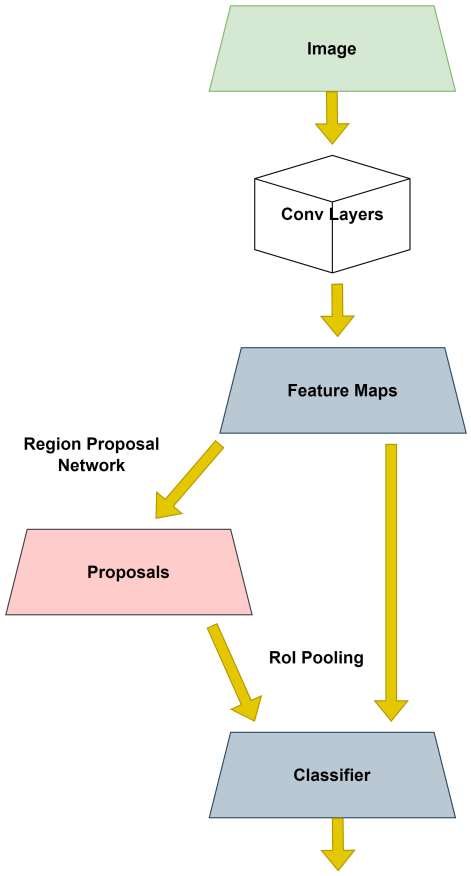

Figure 3.13: Faster R-CNN Method

As seen in 3.13 , the initial step of the Faster R-CNN requires a Fully Convolutional Network which helps narrow down the regions. Next, a Fast R-CNN detects the objects utilizing the suggested areas of the image. The entirety of this process is one end-to-end network which uses the 'attention' mechanisms and RPN to find out where to find the object, and finally detect it.

Chapter 4

Implementations

This section describes the implementation of the three models, namely YOLOv5, YOLOX, and Faster R-CNN. Since we are going to make a final comparison between the three models, we kept the main parts of the implementation constant for all three cases. We used the PyTorch framework for our implementations and all the code was carried out in Google Colaboratory notebook. We applied transfer learning to train our models, which is a famous approach in deep learning where pre-trained versions of the models are used as the starting point for another related task. Transfer learning has been shown to help improve the performance of the models and is especially useful given the huge compute resources required to train deep learning models. We used the Tesla P100-PCIE-16GB GPU for training across the three models.

YOLOv5 required annotations for each image in form of a '.txt' file and a specific PyTorch YOLOv5 format of the dataset was used for it. The Pascal VOC format was required for YOLOX, and COCO JSON format for Faster R-CNN.

We used the smallest versions of YOLOv5 and YOLOX, called the YOLOv5s and YOLOX-s respectively. This is because these versions are the fastest to train and detect, which is suitable for our use in autonomous vehicles. We implemented Faster R-CNN using Detectron2, which is a famous computer vision library of models based on PyTorch. We used the X101-FPN version of Faster R-CNN, since it has a good speed-accuracy tradeoff compared to the other versions.

In the training script of YOLOv5, we mention that we are using images of size 640. We use a batch size of 32, and train it for 100 epochs. It took us 2 hours and 29 minutes to train our model for 100 epochs. After training, we choose the best weights for evaluation. Image size of 640 and confidence threshold of 0.5 is set, which is the same values used in evaluation of our other models. Finally inferences on the test set are made using the best weights and confidence threshold of 0.5.

During training of YOLOX, we also set the batch size of 32 and mention floating point 16. We set the devices parameter, which is the number of GPUs our model will train with, to 1. The training occurs for a maximum of 300 epochs. However, we found that our model started to converge within 61 epochs, so we ended the training. It took a total of 8 hours and 41 mins to train YOLOX on our dataset for 61 epochs. After training, we evaluate YOLOX on the test set, using batch

size of 32, 'd' parameter set to 1 which is the device for training, and confidence threshold of 0.5. Finally, to make inferences on the test set, we mention the test image size of 640, set the confidence threshold to 0.5, 'nms' threshold to 0.45 and 'device' parameter to gpu.

We had to set up a training configuration file for training Faster R-CNN using Detectron2, where a lot of hyperparameters had to be set. Here, we had to mention that we are going to use the X101-FPN version of Faster R-CNN. Another significant part we set here was the max number of iterations to 18000, which is something we have fine tuned by looking at our validation set results and observing when our model converges. Moreover, we set:

- '`cfg.DATALOADER.NUM_WORKERS`' to 4
- '`cfg.SOLVER.IMS_PER_BATCH`' to 4
- '`cfg.SOLVER.BASE_LR`' to 0.001
- '`cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE`' to 64
- '`cfg.MODEL.ROI_HEADS.NUM_CLASSES`' to 9
- '`cfg.TEST.EVAL_PERIOD`' to 500

It took a total of 6 hours and 21 minutes to train Faster R-CNN for 18000 iterations. After training, we evaluated the model on the test set and then visualized the predictions, setting the '`cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST`' to 0.5.

Chapter 5

Results

5.1 Results of YOLOv5

We got various visualizations of different metrics for YOLOv5 with the help of Tensorboard. Fig. 5.1 shows precision, recall and mean Average Precision (mAP) metrics over the training epochs on our dataset, and how the model converged within 100 epochs. One indicator of a machine learning model's performance is precision, which measures the quality of a positive prediction made by the model. Recall measures the model's capability to detect positive samples. Mean Average Precision is a common and popular metric used to measure the accuracy of object detectors. Here, mAP at 0.5 refers to mAP calculated at Intersection over Union (IoU) threshold of 0.5 and mAP at 0.5:0.95 refers to mAP calculated by averaging over different IoU values from 0.5 to 0.95 with a step of 0.05.

Fig. 5.2 shows three types of loss graphs, namely, box loss, objectness loss and classification loss. The box loss refers to how accurately our model was able to locate the center of the objects and how well the bounding box covered the objects. Objectness basically measures the probability that an object is present in a region proposed by the algorithm. Finally, classification loss measures how effectively the algorithm predicted the correct class of any object. The box, objectness and classification loss of the training set reached very low values by the end of 100 epochs. The box and classification loss of the validation decreased to a lower value by the end of 100 epochs, but the objectness loss of the validation set kept increasing.

After evaluating in the validation set, YOLOv5 scores precision of 0.831, recall of 0.589, mAP at 0.5 of 0.732, and mAP at 0.5:0.95 of 0.503. Table 5.1 shows the breakdown of Average Precision (AP) at IoU threshold of 0.5 over our eight classes in the validation set, as well as their corresponding precision and recall values.
In our test set, YOLOv5 achieved precision of 0.865, recall of 0.639, mAP at 0.5 of 0.777 and mAP at 0.5:0.95 of 0.546. Table B shows the breakdown of AP at IoU threshold of 0.5 over our eight classes in the test set, as well as their corresponding precision and recall values.

From Table 5.2, we observe that the YOLOv5 model has a particularly hard time detecting the person class as it has the lowest AP. The rickshaw and bicycle class also have relatively lower AP values compared to other classes.

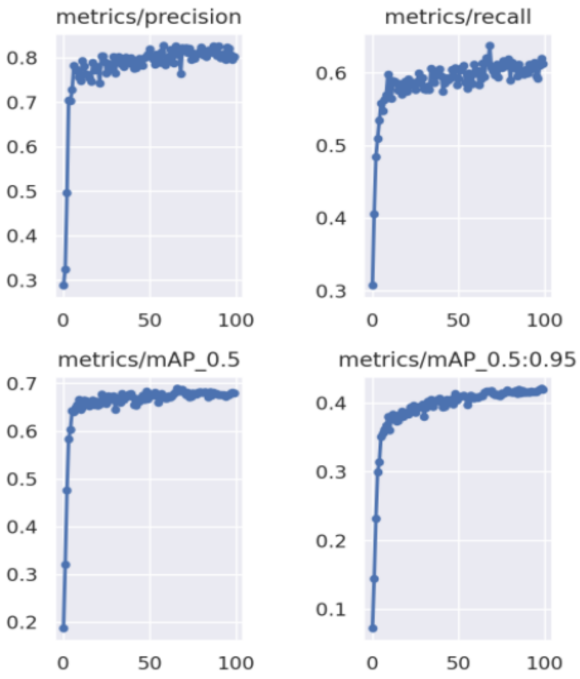

Figure 5.1: Plots of YOLOv5 Precision, Recall and mAP metrics over the training epochs on our dataset

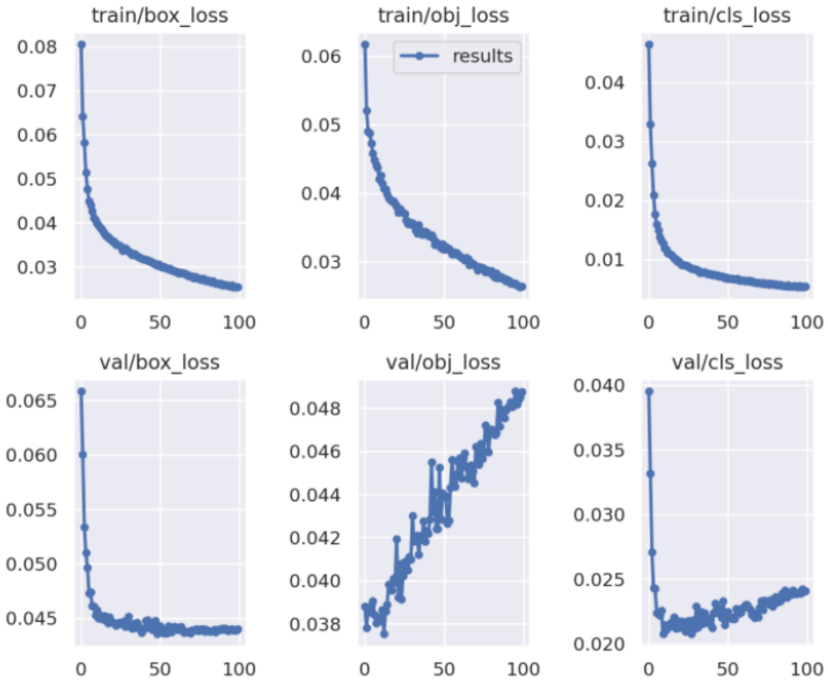

Figure 5.2: Plots of YOLOv5 box loss, objectness loss, and classification loss over the training epochs for the training and validation set

Now, we look at some of the detected images from our test set. From Fig. 5.3 (a), we can see that the YOLOv5 model has detected all the objects correctly, including the partially occluded bus and the small car behind it. Fig. 5.3 (b) shows that

Class	Precision	Recall	Average Precision at 0.5
car	0.916	0.730	0.845
person	0.779	0.487	0.637
cng	0.820	0.651	0.778
bus	0.883	0.583	0.745
motorcycle	0.858	0.602	0.758
rickshaw	0.865	0.683	0.801
bicycle	0.667	0.375	0.549
other-vehicle	0.860	0.598	0.741

Table 5.1: YOLOv5 Precision, Recall, and AP at 0.5 IoU values of the eight classes in the validation set.

Class	Precision	Recall	Average Precision at 0.5
car	0.894	0.726	0.845
person	0.821	0.529	0.683
cng	0.913	0.713	0.828
bus	0.89	0.638	0.788
motorcycle	0.922	0.663	0.812
rickshaw	0.776	0.556	0.716
bicycle	0.783	0.551	0.702
other-vehicle	0.922	0.734	0.844

Table 5.2: YOLOv5 Precision, Recall, and AP at 0.5 IoU values of the eight classes in the test set

YOLOv5 failed to detect one partially occluded bus, but the rest of the objects were detected correctly. Fig. 5.3 (c) shows YOLOv5 detected all the objects correctly, including the heavily occluded car and the bus. Finally, Fig. 5.3 (d) shows the model detected all the objects correctly, containing varying levels of occlusion.

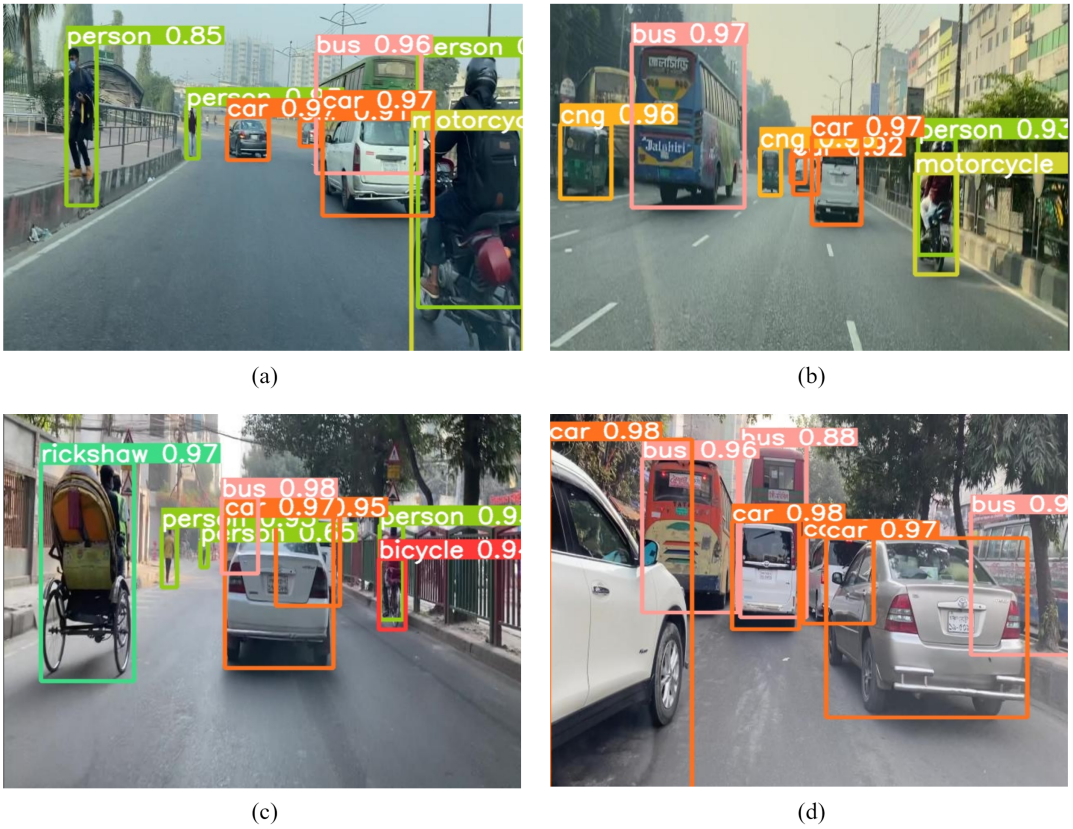

Figure 5.3: Example of occluded objects detected by YOLOv5 on four test set images

5.2 Results of YOLOX

The YOLOX model scored an mAP at 0.5 of 0.8847 and mAP at 0.5:0.95 of 0.6554 in the validation set. Table 5.3 shows the breakdown of AP at IoU threshold of 0.5 over our eight classes in the validation set.

Class	AP at 0.5
car	0.9080
person	0.8471
cng	0.9041
bus	0.9084
motorcycle	0.8136
rickshaw	0.8960
bicycle	0.8936
other-vehicle	0.9067

Table 5.3: YOLOX AP at 0.5 IoU values of the eight classes in the validation set.

In our test set, YOLOX achieved mAP at 0.5 of 0.8488 and mAP at 0.5:0.95 of 0.6344. Table 5.4 shows the breakdown of AP at IoU threshold of 0.5 over our eight classes in the test set.

Class	AP at 0.5
car	0.9080
person	0.7223
cng	0.9041
bus	0.9084
motorcycle	0.7273
rickshaw	0.8960
bicycle	0.8175
other-vehicle	0.9067

Table 5.4: YOLOX AP at 0.5 IoU values of the eight classes in the test set.

From Table 5.4 we can see that the YOLOX model is accurately detecting most classes corresponding to their high AP values. However, it has some trouble detecting the person and motorcycle class as they have relatively lower AP values.

From Fig. 5.4 (a), we can see that the YOLOX model has correctly detected the heavily occluded bus to the far left of the image, among the other objects. 5.4 (b) shows that YOLOX detected the motorcycle partially occluded by the railings in front of it correctly, among the other objects. 5.4 (c) shows that YOLOX failed to detect two instances of partially occluded cars, but the rest of the objects were detected correctly. Finally, 5.4 (d) shows another instance of the model accurately detecting all the objects with varying occlusions.

(a) (b)

(c) (d)

Figure 5.4: Example of occluded objects detected by YOLOX on four test set images

5.3 Results of Faster R-CNN

With the help of TensorBoard we get some visualizations for Faster R-CNN AP curves over the training iterations. In Fig. 5.5, AP refers to mAP at 0.5:0.95 and AP50 refers to mAP at 0.5, and the AP values are in percentage. It shows how the model converged within 18000 iterations.

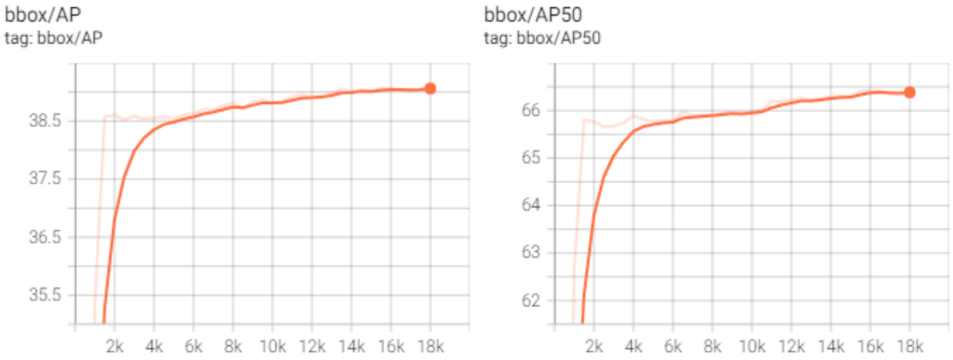

Figure 5.5: Faster R-CNN Average Precision vs number of iterations graphs.

The Faster R-CNN model scored an mAP at 0.5 of 0.6641 and mAP at 0.5:0.95 of 0.3909 in the validation set. Table 5.5 shows the breakdown of AP at IoU threshold of 0.5:0.95 over our eight classes in the validation set.

Class	**AP at 0.5:0.95**
car	0.5692
person	0.3524
cng	0.4666
bus	0.4598
motorcycle	0.3766
rickshaw	0.4053
bicycle	0.2019
other-vehicle	0.2956

Table 5.5: Faster R-CNN AP at 0.5:0.95 IoU values of the eight classes in the validation set.

In our test set, the Faster R-CNN achieved mAP at 0.5 of 0.6883 and mAP at 0.5:0.95 of 0.4218. Table 5.6 shows the breakdown of AP at IoU threshold of 0.5:0.95 over our eight classes in the test set.

Class	AP at 0.5:0.95
car	0.5676
person	0.3841
cng	0.5559
bus	0.4543
motorcycle	0.3914
rickshaw	0.3279
bicycle	0.3458
other-vehicle	0.3476

Table 5.6: Faster R-CNN AP at 0.5:0.95 IoU values of the eight classes in the test set

From Table 5.6 we can see that the Faster R-CNN model is relatively good at detecting the car and cng class, corresponding to their higher AP values.

(a) (b)

(c) (d)

Figure 5.6: Example of occluded objects detected by Faster R-CNN on four test set images

From Fig. 5.6 (a), we can see that the Faster R-CNN model has detected all the objects correctly, including the lightly occluded bus to the right side of the image. Fig. 5.6 (b) shows that the Faster R-CNN model misclassified the cng as a car, but correctly detected all the other objects. Fig. 5.6 (c) shows that Faster R-CNN correctly detected all the objects containing some light occlusions. Finally, Fig. 5.6 (d) shows the model misclassified the partially occluded car to the left as a cng but

correctly detected all the other objects.

5.4 Final Comparison

We have already discussed the results of the three models separately in detail. In this section, we make a final comparison of the three models based on how they performed on the test set of our dataset, as this gives us an unbiased estimate of the performance of our final models. We use mAP at 0.5 and mAP at 0.5:0.95 as the metrics for comparison and the details are shown in Table 5.7.

Model	**mAP at 0.5**	**mAP at 0.5:0.95**
YOLOv5	0.777	0.546
YOLOX	0.849	0.634
Faster R-CNN	0.688	0.422

Table 5.7: Test set performance of YOLOv5,YOLOX, and Faster R-CNN based on mAP at 0.5 and mAP at 0.5:0.95 metrics

From Table 5.7, we can observe that YOLOX has the highest mAP at 0.5 and mAP at 0.5:0.95 score while Faster R-CNN has the lowest. This shows that YOLOX is the best performing model on our dataset.

Chapter 6

Conclusion

In this work, we studied extensively the perception problem of autonomous vehicles and decided to tackle the occluded object detection problem. To this end, we created a dataset of occluded objects from the perspective of Bangladesh road scenes to help leverage this dataset for the detection of occluded objects. We evaluated and compared three different object detection algorithms namely, YOLOv5, YOLOX, and Faster R-CNN, on our datset. We found the best performing detector to be YOLOX, achieving mAP at 0.5 of 84.9% and mAP at 0.5:0.95 of 63.4% on our test set. Although these are good scores, it still has scope of further improvements. One of the things we observed is that all three of the models scored relatively worse AP scores for the person class, because of which our overall mAP metrics went down. One reason for this could be because pedestrian detection is an open research problem and most detectors still face difficulty in accurately detecting people. Additionally, not achieving higher mAP scores could be the result of not having a larger dataset. Although we had 4476 images in our training set, deep learning based models such as YOLOX perform better with larger datasets.
In the future we plan to further increase the size of our dataset. Also, experiment further by incorporating model fusions, utilizing vision transformers, and try to achieve higher accuracy. Finally, we also aim to use a federated approach in the long run, where various vehicles and roadside units communicate with each other to share information. Later, using this data computations get executed, and model trained. We strongly believe that this approach also has a high potential to further improve the performance of occluded object detection for autonomous vehicles.

Bibliography

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361. DOI: 10.1109/CVPR.2012.6248074.

[2] H. M. Bui, M. Lech, E. Cheng, K. Neville, and I. S. Burnett, "Using grayscale images for object recognition with convolutional-recursive neural network," in *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*, IEEE, 2016, pp. 321–325.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[4] L. Zhang, L. Lin, X. Liang, and K. He, "Is faster r-cnn doing well for pedestrian detection?" In *European conference on computer vision*, Springer, 2016, pp. 443–457.

[5] E. J. Barrow, "The use of deep learning to solve invariance issues in object recognition," Ph.D. dissertation, Coventry University, 2017.

[6] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art," *CoRR*, vol. abs/1704.05519, 2017. arXiv: 1704.05519. [Online]. Available: http://arxiv.org/abs/1704.05519.

[7] A. Uçar, Y. Demir, and C. Güzeliş, "Object recognition and detection with deep learning for autonomous driving applications," *Simulation*, vol. 93, no. 9, pp. 759–769, 2017.

[8] J. U. Kim, J. Kwon, H. G. Kim, H. Lee, and Y. M. Ro, "Object bounding box-critic networks for occlusion-robust object detection in road scene," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, IEEE, 2018, pp. 1313–1317.

[9] A. R. Pathak, M. Pandey, and S. Rautaray, "Application of deep learning for object detection," *Procedia computer science*, vol. 132, pp. 1706–1717, 2018.

[10] S. Pettigrew, L. Fritschi, and R. Norman, "The potential implications of autonomous vehicles in and around the workplace," *International journal of environmental research and public health*, vol. 15, no. 9, p. 1876, 2018.

[11] S. Gilroy, E. Jones, and M. Glavin, "Overcoming occlusion in the automotive environment—a review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 1, pp. 23–35, 2019.

[12] J. Lu, S. Tang, J. Wang, H. Zhu, and Y. Wang, "A review on object detection based on deep convolutional neural networks for autonomous driving," in *2019 Chinese Control And Decision Conference (CCDC)*, IEEE, 2019, pp. 5301–5308.

[13] Y. Qu, Y. Ou, and R. Xiong, "Low illumination enhancement for object detection in self-driving," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, 2019, pp. 1738–1743.

[14] J. Wei, J. He, Y. Zhou, K. Chen, Z. Tang, and Z. Xiong, "Enhanced object detection with deep convolutional neural networks for advanced driving assistance," *IEEE transactions on intelligent transportation systems*, vol. 21, no. 4, pp. 1572–1583, 2019.

[15] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[16] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[17] M. Hnewa and H. Radha, "Object detection under rainy conditions for autonomous vehicles: A review of state-of-the-art and emerging techniques," *IEEE Signal Processing Magazine*, vol. 38, no. 1, pp. 53–67, 2020.

[18] K. Tong, Y. Wu, and F. Zhou, "Recent advances in small object detection based on deep learning: A review," *Image and Vision Computing*, vol. 97, p. 103 910, 2020.

[19] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.

[20] A. Gupta, A. Anpalagan, L. Guan, and A. S. Khwaja, "Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues," *Array*, vol. 10, p. 100 057, 2021.

[21] D. Mu, W. Sun, G. Xu, and W. Li, "Random blur data augmentation for scene text recognition," *IEEE Access*, vol. 9, pp. 136 636–136 646, 2021.

[22] S. Das, N. Tasnim, M. I. H. Shihab, M. F. Khan, and R. A. Mahmud, "Team passphrase: Dhaka ai vehicle detection,"