Advancing Autonomous Navigation: YOLO-Based Road Obstacle Detection and Segmentation for Bangladeshi Environments

by

Ishtiaque Mahmud
20301311
Sumaia Arefin Ritu
24141196
Zaki Zawad Mahmood
23241139

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

Department of Computer Science and Engineering
School of Data & Sciences
Brac University
May 2024

Declaration

It is hereby declared that

1. The thesis submitted is my/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:

Ishtiaque Mahmud
20301311

Sumaia Arefin Ritu
24141196

Zaki Zawad Mahmood
23241139

Approval

The thesis/project titled "Advancing Autonomous Navigation: YOLO-Based Road Obstacle Detection for Bangladeshi Environments" submitted by

1. Ishtiaque Mahmud (20301311)
2. Sumaia Arefin Ritu (24141196)
3. Zaki Zawad Mahmood (23241139)

Of Spring, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science in May 2024.

Examining Committee:

Supervisor:
(Member)

Amitabha Chakrabarty, PhD
Professor
Department of Computer Science and Engineering
School of Data & Sciences
BRAC University

Program Coordinator:
(Member)

Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
School of Data & Sciences
Brac University

Head of Department:
(Chair)

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

Abstract

The advancement of autonomous vehicles requires a fast and effective object detection and segmentation system handling a wide range of road environments. The goal of this research is to improve autonomous navigation by applying fast and popular YOLO-based models for road obstacle detection and segmentation in South Asian countries, especially Bangladesh. Our team has compiled an extensive collection of videos taken on Bangladeshi streets using a smartphone camera that shows a variety of road conditions such as potholes, speed bumps, barricades, and normal roads taking into account rainy, sunny, day and night environments. By using Roboflow annotation and sampling tools, these videos were sampled into images and annotated with both bounding boxes and bounding masks.

Using our custom annotated dataset, we trained and refined YOLO-based object detection and segmentation models such as YOLOv5, YOLOv7, and YOLOv8. The YOLOv5x model trained on our custom dataset shows better results with the highest mAP50 and mAP50-95 scores of 0.876 and 0.647 respectively. However, the YOLOv7x model trained on our custom dataset gives the lowest performance outcome with mAP50 and mAP50-95 scores of 0.583 and 0.331. Also, while comparing the models trained with custom dataset with models trained with a benchmark dataset, our dataset shows major improvements in results. We deployed the models on our local computer to allow real-time object detection with a camera, upon which our prototype car can take decisions using a microprocessor. This implementation reflects the feasibility of effective object detection and segmentation with limited resources.

This research intends to optimize autonomous vehicle navigation in Bangladeshi road environments quickly and effectively. The outcomes of our experiments suggest that our approach offers a viable way to improve the security and efficiency of autonomous navigation in these kinds of settings. By addressing the unique challenges with road infrastructure in developing nations, our research advances the area of autonomous driving and creates opportunities for more customized and adaptive navigation systems.

Keywords: automated vehicles, YOLOv5, YOLOv7, YOLOv8, custom dataset, segmentation, road obstacles, hardware implementation, Bangladeshi road environments.

Acknowledgement

Firstly, all praise to Almighty Allah for whom our thesis has been completed without any major interruption.
Secondly, to our Supervisor Dr. Amitabha Chakrabarty sir for his kind support and advice in our work. His guidance and advice helped us to make our thesis satisfactory.
And finally, to our parents. Without their support, it would not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The inauguration of autonomous vehicles has led industries to sense their environment and utilize it without human implications. Autonomous vehicles anticipate complex algorithms, sensors, effective processors, etc to implement the software. With the advancement of technology, autonomous vehicles have become of substantial interest to engineers, researchers, and also industry experts in the upcoming years [14].

The scheme for convenience and improvements of human life is unbounded. The real promises of autonomous vehicles have great potential in many sectors to ease daily life. According to research, autonomous vehicle (AV) systems are considered to be able to reduce traffic congestion, improve safety, reduce transportation in terms of infrastructure, reduce urban emissions worldwide, and improve livability [7]. In addition, AV technologies have been introduced to minimize road accidents and also with many advantages, like resolving security issues, social, ethical, and technological problems [5], and so on. Stepping into new technologies, autonomous vehicles can lead the way to a new conveying environment with safer driving [12]. From the initial years, AVs were only able to handle a few driving situations. But nowadays, advancement in real-time data processing have led to realistic autonomous vehicles.

The continuous development of autonomous technology has gained a good opportunity for the research community to utilize some technologies such as computer vision, GPS, sensors, mapping systems, and object detection systems to traverse [27]. These technologies are used for detecting obstacles in the routes, determining the weather, and finding suitable routes for a location. The AV systems aim to solve all the existing problems to avoid negligence. But in many cases, with object detection and in real-time decision-making action, there are some shortcomings in making a proper visualization and having the solution for autonomous vehicles. In this research, we look forward to having proper solutions for object detection and segmentation and also in the decision-making process in a more efficient way for autonomous vehicles' functionality in the Bangladeshi context.

Although convolutional neural networks are the most common in terms of detecting objects in automated vehicles. Usually, the neural networks look for patterns in the data, which are taken input by the machine learning algorithms [[34]. But in terms

of efficiency and accuracy in computation, CNN models are ahead of the current state-of-the-art. According to the particular YOLO model which is a framework of the CNN model that allows the models to learn image formation without any dependency, the images are processed as a series of patches, with each patch compressed into a single vector by integrating the channels of all pixels in a dot, followed by linearly projecting it to the suitable input dimension[36]. Therefore, we can say that the CNN-based models are faster to train given the fact that there is a large dataset for them to train. To detect unidentified objects and image segmentation, the roads of the city faster which will help the vehicle make decisions faster and save it and its passengers from hazards or accidents, YOLO models show the best potential if trained with the appropriate amount of data.

1.2 Research Problem

We have divided our research problem into three parts. They are as follows:

- **Detection and Segmentation Issues:** Current models must be improved to accurately detect and segment road obstacles in real-time, specifically for the diverse and challenging road conditions in Bangladesh.
- **Lack of Context-Specific Data:** There is a lack of training data and research tailored to the Bangladeshi environment that hampers the performance of existing models.
- **Efficiency and Accuracy:** For properly detecting and segmenting obstacles in the Bangladeshi context,CNN models like YOLO need improvements.

Unexpected and unidentified objects on roads can cause vehicles, sometimes the deadliest accidents, if not taken into account. Detecting objects properly and acting accordingly in the minimum amount of time is essential for autonomous vehicles to save themselves and their passengers. We have selected objects like Speed bumps, potholes, and obstacles (e.g., construction signs, and barricades) for our models to detect and segment in this research.

1.2.1 Speed Bumps

Speed bumps are used on roads to reduce the speed of overspeeding vehicles. They are especially used on places such as road crossings and intersections for secure crossings of vehicles and pedestrians. According to a recent study [9], out of 2000 drivers, 22 percent of them experienced damage to the vehicles that they drive when crossing a speed hump and it cost them 141 Euros to repair the vehicles. Speed bumps can be effective in reducing road accidents but can cost emergency vehicles some valuable time and increase the effects of different hazards[30]. Therefore, the need to detect speed bumps for autonomous vehicles and slow down accordingly is essential for a smooth riding experience for the passengers in terms of the Bangladeshi environment.

Nevertheless, unplanned speed bumps with an unusual height or curve can be problems for the vehicles and can result in damage to the vehicles. According to England

and Wales Highways Regulations, speed bumps shall be no less than 20mm in height and no more than 100mm high. The highest gradient recommended for speed bumps is a 1:10 gradient[42].

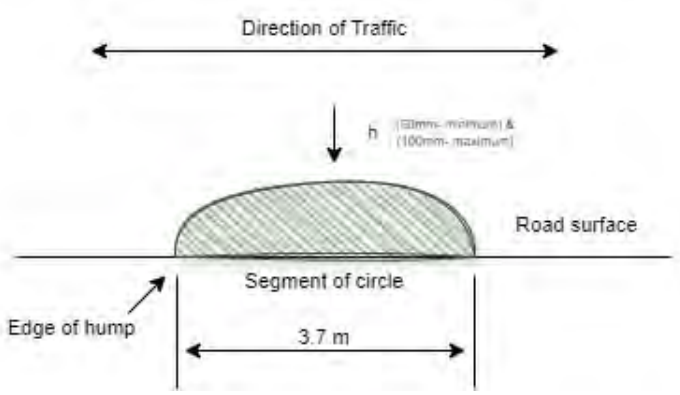

Figure 1.1: Ideal height of speed bump,(Jersey) Regulations 2005.

Figure 1.1 represents the ideal height of a road bump according to the States of Jersey (Amendments and Construction Provisions No. 3) (Jersey) Regulations 2005. An unnecessary amount of speed bumps are also a reason for damage and accidents. From 2009 to 2011, 41-speed bumps were installed along a 40-kilometer section of road between Dhaka and Mawa in Bangladesh[2].

1.2.2 Potholes

Potholes are another kind of deadly reason for accidents, both in cities and rural areas. Heavy usage of the city and highway roads without repair results in uneven roads with potholes. Around 25% of the roads in Bangladesh are in a "poor, bad, or very bad" state, according to the Roads and Highways Department (RHD)[10]. The RHD assessed 17,452 kilometers and discovered that 4,247 kilometers (24.34 percent) of them are in poor, terrible, or extremely bad condition[10]. Completing different works under the roads and not repairing them afterward can also result in different kinds of potholes. During the rain, these potholes go underwater, and the drivers of the vehicles lose visibility of them and cannot decide their actions properly. Therefore, it is very important for autonomous vehicles to detect these potholes properly and act accordingly. Small potholes are 25 mm in depth and 200 mm in width, whereas medium potholes are 25 to 50 mm in depth and 500 mm in width, and large potholes are the ones that are more than 50 mm in depth and 500 mm in width[8]. The damage to the vehicle depends on the size and depth of the potholes. If these types of potholes are not detected beforehand, then not only damage but also deadly accidents can occur to the autonomous vehicles.

1.2.3 Obstacles (e.g., construction signs, barricades, cones)

There can be many unidentified objects on the road that can work as obstacles for vehicles and cannot be determined before facing them. A sudden construction project can happen, which may require a single lane of the whole road. Often, these types of construction works have only a signboard in front of them, which has to be detected by the vehicle to determine the next step it has to take. If there are no

signboards, the vehicle has to detect the road cuts and find its way.

The city roads are often blocked or diverted by barricades by the traffic control authority, which should be detected by the autonomous vehicle upon viewing. These types of obstacles require slight changes of lanes sometimes or a rerouting depending on the type of barricade. Often, the barricade could be implemented by the authority directing a check post or sometimes can be used to block one of the multiple entries to bridges or flyovers.

1.3 Research Objective

Autonomous vehicles (AVs) have revolutionized transportation as well as healthcare, space computing, agriculture, military, and supply chain management by now and they are projected to be the future of smart and intelligent transportation [23]. The United States wins the AV title with a score of 8.62/10, topping second-place Japan by 1.03 points, the greatest margin in the top five as well[28]. The leading electric vehicle companies offer different features like autopilot mode for highway, auto-parking, lane-assist, and auto lane change with their EVs, which use multiple sensors like camera, radar, ultrasound, infrared, and lidar[40]. Nevertheless, completely self-driving automobiles are not yet available for purchase as of late 2022[16]. We aim to detect unidentified objects and obstacles on the roads faster and more accurately with the help of YOLO-based models which can lead to better decision-making of the driver-less vehicles. Our research objectives:

- Gathering a diverse dataset of road environments in urban settings of Bangladesh to cover a wide range of obstacles in different conditions and weather. These obstacles and roads vary in different countries and continents. Our aim is to create a custom dataset that can train object detection models that can work in South Asian road environments, specifically Bangladesh.
- Annotating the dataset with detailed annotation labeling considering the size, color, and shape of the objects like barricades, normal roads, potholes, and Speed bumps.
- Identifying the obstacles that are unidentified to the vehicle, such as speed bumps, potholes, and barricades, along with normal roads, and creating training set data. We aim to identify these objects in context with Bangladeshi and South Asian road environments and situations.
- Training and fine-tuning the existing Convolutional Neural Networks object detection models like Yolo with the custom dataset to detect the objects on the road to improve the object detection speed and accuracy.
- Validating and classifying the objects to help real-time detection processing of the Convolutional Neural Network models.
- Implementing the best-performing model into a proposed hardware system to detect road objects fast and accurately in real-time.

Chapter 2

Related Work

This literature review section represents all the relevant papers to our research data from various research works for understanding several aspects of concerns. It is very effective for future research work in a particular domain. Here, we tried to recruit some good papers that have been done previously that are relevant to our domain.

2.1 Deep learning/Convolutional neural network (CNN) Classifier

In a paper [4] regarding real-time object detection using a small and low-powered convolutional neural network model named SqueezeDet, the researchers aimed to create the model in such a way that it becomes highly accurate and fast in detection, being able to guarantee prompt vehicle detection in real-time. The researchers adopted a single-stage detection pipeline that was inspired by the You Only Look Once (YOLO) learning model. The SqueezeDet is enabled by the ConvDet to generate numerous regions with few parameters. The faster R-CNNs deploy a 4-state training strategy, but the SqueezeDet can be trained in an end-to-end style that was inspired by the characteristics of YOLO. SqueezeNet is composed of Fire Modules. These modules include a compressed layer as input and two parallel enlarged layers as output. Tensorflow, assembled with the cuDNN computational kernels, was used to create this model's training, error analysis, evaluation, and visualization pipeline.

A recent paper [20] talks about detecting objects from videos for autonomous driving using an end-to-end framework named MFCN or Motion-aid Feature Calibration Network, which is also a deep learning framework. The proposed method's entire architecture is end-to-end, which considerably results in improving the model training efficiency along with the inference efficiency. The model takes into account the influence of feature movement, allowing it to learn at both the pixel and instance levels. The MFCN or motion-aid Feature calibration Network, adaptive weight calculation, pixel-feature calibration, and instance-feature calibration are used in this research to calibrate image features, improve the object features in the current frame, amass calibrated feature maps at pixel level across multiple frames to enhance instance-feature recognition across frames. The method's design contains three contributing networks, consisting of the feature extraction network, the motion estimation network, and the object detection network, which use ResNet, FlowNet2, and R-FCN,

respectively. When compared to top approaches based on KITTI benchmark testing, the model proved to be competitive with cutting-edge approaches.

In the paper [15], the researchers introduced a model named Mobilenet-SSDv2, which does not require a high-end GPU to detect objects but rather is lightweight and capable of working in embedded systems. To improve detection accuracy and stability, the researchers used an attribute pyramid network with the model. Their proposed backbone network, Mobilenet-v2, incorporates a standard convolutional layer along with 17 inverse residual modules. After training the model with 200 epochs in the SGD method accuracy improved by about 0.6% mAP compared with the Mobilenet-SSD. Moreover, the Mobilenet-SSD method provided 17 fps in processing speed whereas the proposed Mobilenet-SSDv2 can go up to 23 fps processing a 720 video with Nvidia Jetson AGX Xavier. During daylight at a crossroad environment, during low light in a lane environment, and during daylight in an environment with many pedestrians, this model performed quite well and could detect accurately.

The paper [31] talks about identifying cars, pedestrians, and traffic lights on roads using the YOLO or You Only Look Once version 5 model taking into consideration that there might be bad weather. The researchers used street recordings in rainy and regular weather conditions and utilized Roboflow datasets in their proposed system to train. They used pre-trained COCO weights on a Roboflow self-driving car dataset based on YOLOv5. The whole annotated dataset was then divided into three sections. The sections are training (959 images), validation (239 photos), and testing (302 images). The suggested model improves precision, recall, and mAP and manages to reach a peak of 17, 93, and 99 epochs, respectively. According to the results of the experiments, the YOLOv5 is 72.3% accurate for automobile recognition and 57.3% for truck identification for mAP (0.5).

According to the paper [37], based on 3D object detection focusing on autonomous driving, different types of sensor setups were used in autonomous vehicles. Object detection methods, like single image-based detectors, have been used for classical computer vision. Then we tried other detectors named two-stage detectors to predict a casual number of object detection proposals. Further, the BEV Depth method is used for depth estimation, which allows the object depth to divine determined. This paper initiates the development of work around vision-based object detection. In addition, this work category for around-view detection focuses on situated networks for autonomous vehicles.

In another paper [6], it is fully about the new ways of object detection. They introduced deep learning and deep convolutional neural networks (DCNN). They aim to develop a model that works for real-time video object detection with a new framework based on a neural network. They tried to develop deep convolutional neural networks to reduce the complexity of object detection. On the detection side, the DCNN model is faster to predict local regions for the objects. In this paper, the major contribution is they tried to send the knowledge from a large network that was pre-trained. Here, by turning down the computational complexity they designed a feature to reduce network capacity. In their experimental result, they used a large dataset from existing research. And they compared various algorithms on the dataset. In addition, they proposed another method of cross-network features,

choosing linear projection for the mapping. DCNN-ROI method is also used to detect objects in images. Also, SVM models are introduced for extended work for frameworks. So, in this work, they used projection frameworks for neural networks and the issues of the large dataset in the domain. Moreover, they could compute the solution for the region and their target objects from videos, making it faster.

According to a paper [24], their implementation on detecting a person, potholes, or wetland using the YOLO algorithm, computer vision, and Convolutional neural network (CNN). They have used Raspberry pi4 to detect potholes on Indian roads. This is a great method for embedded computer boards for detecting objects. This also solves real-time detecting problems. In this paper, they tried to develop a system using Raspberry pi4 a with high-performance ARM cortex A72 4x 1.5GHZ quad-core processor. This supports processing speed and frame rate as this is very important for object detection in real-time. Then they used TensorFlow in the local disk and calculated the datasets that were provided by them for the detection of multiple objects. Moreover, the YOLO algorithm is used for detecting objects like potholes, wetlands, and traffic lights with computation power. They got a very good result for detecting the potholes and wetlands in autonomous vehicles.

In a recent paper [3], a free space detection system is proposed that can detect speed humps on the road using a medium-cost lidar sensor and a low-cost camera. The research took place in a Spanish environment with a massive number of speed humps ranging from 4 to 9 meters in length and spanning the width of the road and also working as zebra crossings in that particular area. The camera, with a resolution of 640*480 pixels at 30 fps, detects the stripe markings on the zebra crossing, and the lidar calculates accurate free space. A Sick LD-MRS HD laser is used to scan the next 30 meters in front of the vehicle, which works in extreme weather conditions. The lidar calculates the space under the vehicle when a speed hump approaches in a range of 15 meters in front of the vehicle using the curve angle and position vectors of the surface. The researchers managed to detect 100% of the speed humps 12 meters before the car reached the humps and were able to reduce the speed to 20 km/h.

Another recent paper [16] reported different technologies and approaches for speed breaker detection using different data acquisition techniques like laser scanning, vibration-based detection, and computer vision-based detection such as image processing. The study found that vibration-based detection (smartphone) was the best approach. Moreover, properties from this type of sensor were extracted via time, frequency domain features, and transformation filters/wavelets. They used machine learning techniques such as SVM and ANN and added that the future of these types of research should have involvement of the already mentioned techniques as well as the control systems of vehicles through Controller Area Network (CAN) communication.

A published paper [11] researched the identification, detection, and analysis of potholes using 4 different image segmentation techniques, such as Image thresholding, fuzzy C-means clustering, K-means clustering, and canny edge detection. The paper suggested that further research can be done and can be used in automobiles with a camera to identify potholes in developed countries with relatively good infrastructure. Further, the Canny Edge Detector had an 82 percent accuracy and

was advised in such cases. The paper concluded with the suggestion that image segmentation techniques can be upgraded with a support vector machine or neural networks that can filter the disturbance intelligently and therefore can be executed for autonomous applications such as autonomous driving all over the world.

In this paper [26], they implemented the object detection algorithm based on the model YoloV4. The researchers compared the accuracy of the yoloV4 model with the dataset by 2.06%. They have used the KITTI and BDD100k datasets for autonomous research. There were 7481 training sets in the KITTI and in BDD100k there were 70000 training data sets. They have used them to train the model. The research was carried out on NVIDIA RTX 2080Ti. Moreover, they also mentioned that they tried to keep the detection accuracy from changing so that the speed of the algorithms could be increased by 9.14%. They have shown that this model can detect objects at a speed of more than 58.47 FPS in real time. In their research, they could show the result that this model has higher accuracy and performance.

In this paper[1], an artificial vision method is introduced that can detect obstacles in complicated, unstructured settings in real-time. With a computing time of 64 milliseconds for 340*240 pixel images, this approach works in cases such as thin poles and trees. To estimate the camera's pitch oscillation, the suggested method uses a low computational burden to calculate a V-disparity image between the left and right matching images. To locate the discrepancy, the system created a Discrepancy Space Image or DSI, and the DSI assisted in aggregating the detection within the local region. Obstacles are detected and mapped into real-world coordinates, which aids in course planning and navigation. The program was successful in recognizing obstructions that other sensors, such as lasers, were unable to detect. For textureless obstacles, the system can detect only the edges.

In another published paper [17], convolutional neural networks (CNN) were used to detect potholes for autonomous vehicles, where the model showed truly impressive results. The accuracy, sensitivity, and F-measure of the suggested model, which were experimentally analyzed on a variety of performance metrics, were 99.02%, 99.03%, and 98.33%, respectively. These values are comparable to those of current state-of-the-art methods, outperforming various methods.

Small cars and pedestrians are the main topics of a paper's [33] discussion of an improved YOLOv5 algorithm for quick and precise object recognition in autonomous driving. To increase detection speed and accuracy, new techniques such as structural re-parameterization, neural architecture search, and coordinate attention mechanisms are used. On the KITTI dataset, the enhanced YOLOv5 algorithm outperformed standard methods with a 96.1% detection accuracy at 202 FPS. For autonomous cars to be safe and effective, the study highlights the need of striking a balance between detection speed and accuracy.

The research paper [18] proposes a deep learning strategy to detecting potholes on Indian roadways, which employs a novel dataset and YOLO (You Only Look Once) algorithms. A 1500-image dataset of varied road conditions in India was developed and annotated before being used to train the YOLO models. The study examined YOLOv3, YOLOv2, and YOLOv3-tiny models using mean average precision (mAP),

precision, and recall measures. The article proposes future real-time deployment in automobiles and interaction with GPS for road maintenance.

The research paper [13] describes a system that uses visual data and convolution neural networks (CNNs), especially ResNet-50, to categorize road surfaces as potholes, speed bumps, or regular roads. The study's first phase classified road surfaces with a true positive rate of 88.9%. In order to pinpoint the exact placement of speed bumps, the YOLO object identification method was used in the second phase. The goal of the project is to build on this work by warning drivers and modifying the suspension of the car in response to the state of the road, improving comfort and safety.

Another paper [25] makes use of deep learning architectures and photographs taken by a cellphone placed on a windshield to detect potholes in real-time. It shows that YOLOv4-CSPDarknet53 outperforms SSD TensorFlow, YOLOv3-Darknet53, and YOLOv4-CSPDarknet53 in pothole identification. With an image resolution of 832x832, YOLOv4 processed at 20 frames per second and obtained good recall (81%), accuracy (85%), and mean Average accuracy (mAP) of 85.39%. By alerting government agencies to potholes and assisting drivers and autonomous cars in identifying potholes in real time, the technology seeks to increase road safety. Adding to the dataset and distinguishing between potholes and manholes are two areas of future effort.

In order to enhance traffic management on highways, the study [35] presents a real-time vehicle recognition, tracking, and counting system that uses YOLOv7. It discusses the shortcomings of YOLOv7 in terms of car detection on city streets and suggests improvements for increased precision and real-time tracking. The suggested method can help with traffic analysis and planning by classifying various vehicle kinds and providing statistical data on traffic flow. By providing a holistic system that includes detection, tracking, and counting in a range of illumination and environmental circumstances, the study seeks to close research gaps.

The study [41] tackles problems such as low detection accuracy and resilience in infrared image identification by introducing an improved infrared road object recognition method, YOLOv8-EGP, which is based on YOLOv8s.The enhancements include employing a dyhead detection head for improved expression capability, substituting the C2f module with SCConv to increase local feature perception, and including a tiny target detection layer to decrease missed detections. By increasing mAP50 by 6.1% and accuracy and recall by 5.8% and 1.6%, respectively, ablation tests conducted on the FILIR dataset demonstrate the efficacy of YOLOv8-EGP in infrared target identification. The improved model is useful for car assistance and intelligent transportation, particularly for nighttime road identification.

2.2 Research Models and Metrics

Ref	Task	Model	Object	Dataset	Performance Metrics
[4]	Real-Time Object Detection	SqueezeDet (CNN)	Car, Cyclist	7381 images	mAP 76.7%.
[20]	Video object detection	Motion Aid Feature	Car, Motorcycle and Bicycle	KITTI and ImageNet VID	mAP 87.4% .
[15]	Real-time object detection	Mobilenet-SSDv2	20 Objects VOC Dataset	Pascal VOC Dataset	mAP 75.9%,
[31]	Detection of street level objects	YOLO v5	vehicles, traffic signals	Roboflow Dataset (1500 images)	mAP(0.5) 0.258, Precision 0.474, Recall 0.337.
[37]	3D detection methods	BEVPOOLV2, BEVSTEREO, BEVDEPTH	multiple objects	Waymo Open Dataset	mAP(0.338 to 0.586).
[6]	Video object detection	Fast deep neural network	multiple objects	ImageNet	Accuracy: 92.37%.
[24]	Uses YOLOv3 Algorithm	YOLOv3, CNN	Motorcycle, people, pothole, car	PASCAL VOC image dataset	mAP(0.4).
[3]	A real-time free space detection speed humps	LIDAR,Vision	Speed Humps,zebra crossing	LIDAR and Camera used	100%(speed humps),94% (zebra crossing).
[16]	Identify speed bumps	3D vision-based contemporary methods	Speed bumps	Online dataset	average 89% and above detection rate among all models
[11]	4 segmentation techniques	Image Thresholding, Canny Edge Detection	Potholes	Online:51 img, Custom:68 img	Accuracy:82%

Ref	Task	Model	Object	Dataset	Performance Metrics
[13]	Categorize road surfaces	ResNet-50, YOLO	Potholes, Speed-bumps	Online data	True positive rate of 88.9%
[26]	Object detection	YOLOv4	Person,car, bike, traffic-light	KITTI, BDD100k dataset	Real-time rate: 58.47FPS.
[17]	Detect potholes	CNN	Potholes	Kaggle dataset	Accuracy: 99.02%
[33]	Object detection	YOLOv5 (improved)	Small cars and pedestrians	KITTI dataset	Accuracy 32% higher than normal version.
[18]	pothole detection on Indian roadways	YOLOv2, YOLOv3, YOLOv3-tiny	potholes	novel dataset of 1500 images	highest mAP@0.25 in YOLOv3-tiny
[35]	Vehicle Detection	YOLOv7	cars	MS COCO dataset	Improvement results on TensorBoard
[41]	Road object detection	YOLOv8-EGP	person, bike, car, bus, light and sign	FILIR dataset	increased mAP@0.50 by 6.1% and 5.8%, 1.6% for accuracy and recall

Table 2.1: Research Matrix

2.3 Research Purposes and Literature Inadequacy

From the existing research reviewed above in Table 2.1, we see a gap in the context of the Bangladeshi road environment as previous work has not been done focusing on a YOLO-Based model to autonomously detect and segment our specific road obstacles - potholes, barricades, speed bumps, and normal roads. Additionally, our newly made custom dataset collected directly from the roads of Bangladesh gives us an upper hand in these segmentation and detection tasks to tailor it towards the improvement of autonomous detection in Bangladesh which is still lagging behind compared to others. Consisting of various types of images, our dataset is diversified with various augmentation techniques and light and weather conditions which play a vital role in highlighting the challenging road environment of Bangladesh effectively taking part in the performance improvement of the trained models. Furthermore, a comparison among the mentioned YOLO models is missing in respect to detecting and segmenting road obstacles. In this research, we are trying to fill the gap by bringing forward the above mentioned aspects into one paper.

Chapter 3

Methodology and Workflow

In this section, we represent the detailed methodology of our proposed analysis for detecting real-time objects like potholes, speed bumps, and barricades for autonomous vehicles. Moreover, we are providing a detailed description of our custom dataset that we have collected and tried to apply more techniques for the visualization of the dataset.

3.1 Convolutional Neural Network (CNN) Workflow

Our research depends on our real-time dataset. We used CNN-based multiple Yolo models for unidentified objects or obstacles such as construction signs, barricades, speed bumps, potholes, and cones and also considered the weather conditions. We have collected real-time video data and samples and divided the objects into three parts, such as collecting real-time video for various types of speed bumps, gathering all the data for potholes, and cumulating real-time video for the obstacles. In key terms of making a decision or coming to a conclusion, we examine the four main parts: Localization, Perception, Prediction, and Decision-making.

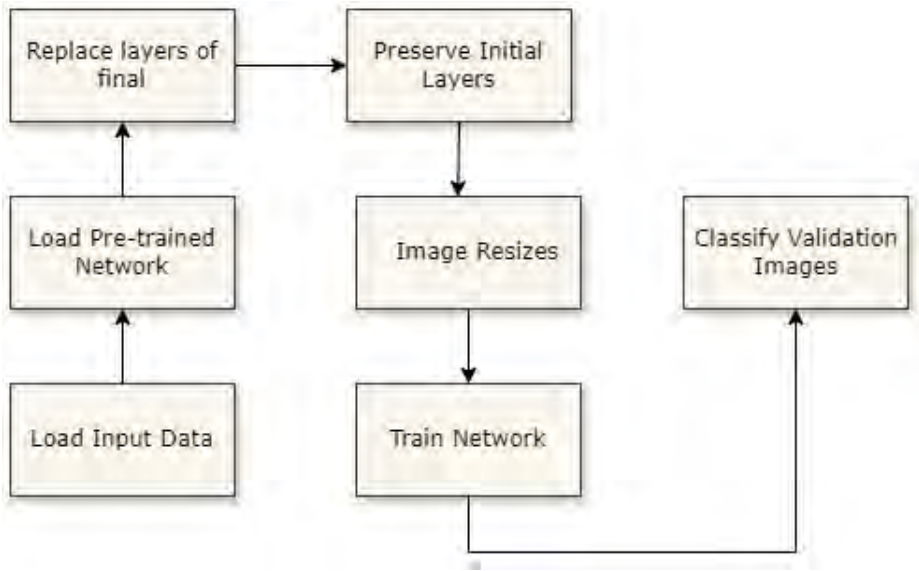

Figure 3.1: Workflow of CNN model

Figure 3.1 shows the workflow of CNN models. Here we can find out all the basic steps of the workflow and how this model works in detecting images layer by layer [19]. Deep learning has become very powerful nowadays. We know one of the most popular deep learning neural networks is the convolutional neural network, especially for computer vision applications [22]. One of the best series of CNN models is YOLO (You Only Look Once). YOLO models are real-time object detection algorithms that support and show end-to-end testing, training, and detecting target objects in images with particular accuracy.

3.1.1 Dataset acquisition and preprocessing

Training CNN models requires many crucial steps. First, begin with data preprocessing techniques, which are focused on preparing the input data for better and more effective model training. We create a custom dataset and test it using state-of-the-art models to detect unidentified objects from the streets and roads of Dhaka city areas in Bangladesh. In our case, we have used CNN models to train our custom dataset to detect potholes, barricades, and speed bumps on the roads of Dhaka city. To analyze the data more specifically and in detail, the images of the dataset are annotated first. After splitting, the images are divided into three sections, including training (75%), validation (20%), and testing (5%). Further, we get the classification of the dataset. In our preprocessing techniques, it includes tasks such as normalization which scales pixel values to a common range, and resizing. This gives data consistency to ensure the image dimensions. Furthermore, we have used data augmentation techniques which are contrived to increase the diversity of the training of our custom dataset. Besides, these augmentation techniques are applied by transformations, and translations to input images. By doing this, it can mitigate overfitting. Furthermore, we have set our own custom hyperparameters, such as learning rate, batch size, and optimization algorithms, which help us improve the generalization performance of the models.

3.1.2 CNN model training and detection

CNN model implies iteratively bringing up to date its parameters to minimize a predefined loss function, especially in using labeled data. In this training process begins with feeding input images through convolutional layers, which draw out features hierarchically and also apprehend patterns of increasing complexity with our custom data. Moreover, through the classifier, the error signal is generated backward, Adjust the weights to minimize the loss between predicted and ground truth labels. After training our custom dataset, CNN models were deployed in object detection. In object detection, we integrated into frameworks like YOLO, where they scan regions of interest within an image to localize and classify objects. Besides, this initializes using anchor boxes to generate candidate regions, which are then clarified and classified by CNN. So these techniques increase the accuracy and efficiency of the models in identifying objects of interest within complex scenes in autonomous vehicles.

3.1.3 Integration and Testing

Integrating and testing a CNN model includes ensuring its seamless affiliation with the purposeful system. This is followed by a meticulous evaluation to assess its

performance and reliability. In our model, integration helped to entails adapting the trained model to the target environment. We faced some issues, such as hardware constraints, software compatibility, and deployment infrastructure. However, we overcame the constraints, which helped us optimize the model for inference speed and memory usage. After integration, thorough testing is essential to validate the model's functionality and validity across diverse scenarios and inputs. So basically, it involves estimating its accuracy, speed, and resource employment under real-world conditions. After that, we evaluate its performance against established ground truth data. Therefore, we can ensure effectiveness in delivering real-world solutions across a wide range of applications.

3.1.4 Methodology

In our methodology, the methods we used for testing and training by YOLO models, which is a framework model of CNN. We created a custom dataset and tested it using state-of-the-art models like YOLO to detect unidentified objects from the streets and roads of Dhaka city areas in Bangladesh. In our case, we have used different versions of the YOLO (you only look once) models to train our custom dataset to detect potholes, barricades, and speed bumps on the roads of Dhaka city. To analyze the data more specifically and in detail, the images of the dataset are annotated first. After splitting, the images are divided into three sections, including training (75%), validation (20%), and testing (5%). Further, we get the classification of the dataset. After data pre-processing and data augmentation, we get to the next step, which is model training. Here, figure 3.2 represents an overview of the prototype model.

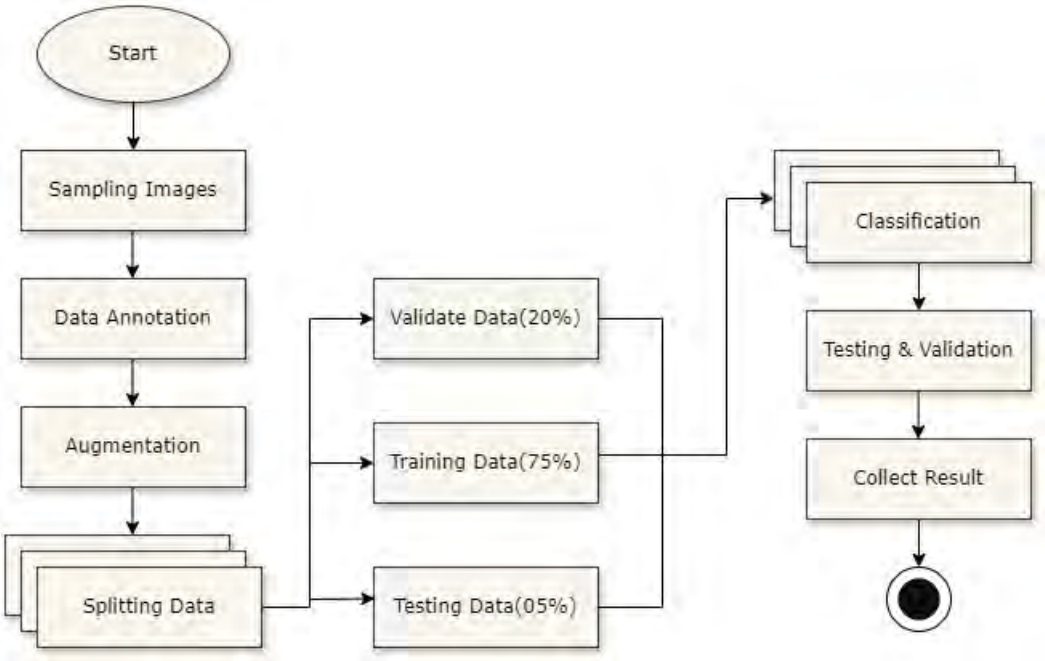

Figure 3.2: Methodology

Chapter 4

Models and Architecture

In this section, we are presenting a detailed description of the model architecture that we have used in our research. Additionally, we showed every architecture workflow and the techniques for understanding how these models work for object detection purposes.

4.1 Convolutional Neural Network(CNN) Models

Convolutional Neural Networks (CNN) are a class of deep learning neural networks [21]. This field of the deep neural network is dynamic. The choices of CNN architectures and models depend on the resources, objectives, and datasets. CNN models are mainly used for object detection, segmentation, classification, and computer vision objectives. Various architectures and principles have been developed for the different tasks. Convolutional Neural Network (CNN) models have found large-scale applications across various domains in computer vision tasks. One of the most popular applications is image classification, where the model automatically identifies objects within images with incredible accuracy. Figure 4.1 represents the basic architecture of the CNN model.

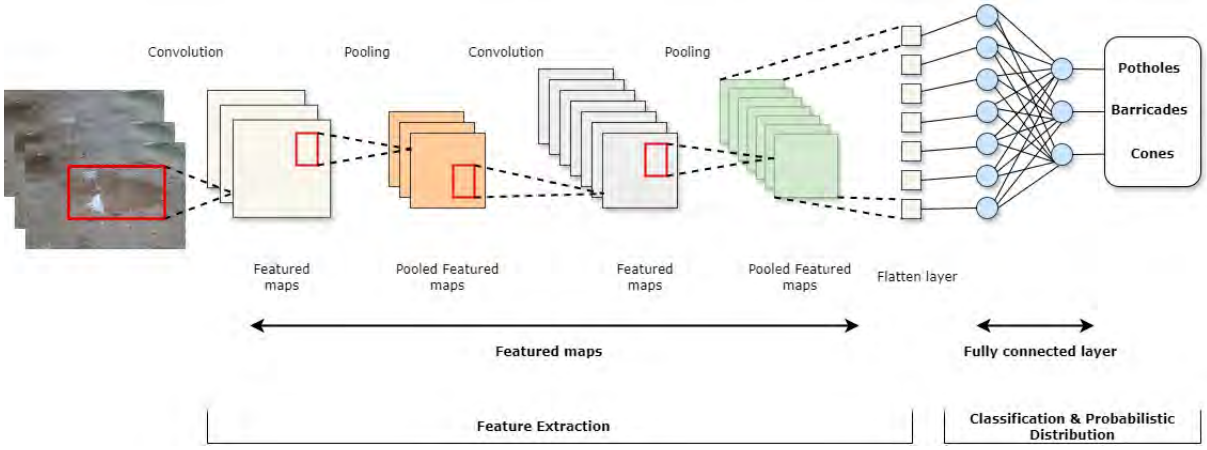

Figure 4.1: Basic Architecture of CNN model

Additionally, CNNs are generally used in object detection, enabling real-time detection and locating of multiple objects within complex scenes. These models are invaluable in autonomous driving, inception systems, and augmented reality applications. Moreover, CNNs also work to ease the precise portrait of object boundaries

and semantic segmentation. Beyond conservative computer vision tasks, CNN models are also employed in image generation tasks, as like generating realistic images by transforming images into different artistic styles.

4.1.1 YOLO(You Only Look Once) Models

The YOLO (You Only Look Once) framework represents an advance in object detection, renowned for its remarkable speed and accuracy [29]. In contrast to traditional detection methods that depend on region proposal algorithms followed by classification but YOLO embrace a single neural network architecture that at the same time forecasts bounding boxes and class probabilities for multiple objects within an image in a single forward pass. Moreover, YOLO enables real-time performance and makes it appropriate for applications where speed is analytical, such as autonomous driving and video surveillance. Besides, this framework is utilized by dividing the input image into a grid and predicting bounding boxes and related class probabilities for objects within each grid cell. These predictions are clarified based on learned feature rendering extracted from the entire image. It also allows YOLO to capture context and dimensional relationships between objects. Moreover, this model employs anchor boxes to improve localization accuracy with confidence estimation for objects of multiple shapes and sizes.

4.1.2 Performance Evaluation Metrics

The YOLO models we have trained our dataset upon all use the same performance metrics. The precision or accuracy in YOLO-based models is determined by the Mean Average Precision (mAP) metric value which is calculated over recall values of 0 to 1. The formula is based on the following sub metrics -

Intersection Over Union (IOU)

This model anticipates the bounding box, which shows the area of the object in an image. For this prediction, Intersection Over Union (IOU) is used for determining the bounding box and this shows how many bounding boxes are overlapped.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{4.1}$$

Precision and Recall

Precision is basically the capability of a model to identify real-time objects, and recall is one of the capabilities of a model that finds the ground-bounding box, which is truth. A model, when it shows no false negatives, has a recall of 1.0. Even if the model detects the wrong bounding box, the recall will still be 1.0. We can say that when a model has low precision and high recall, it will determine most of the positive samples, but there can be false positives that determine the negative test as positive. On the other hand, when a model has low recall and high precision, it works on a positive sample.

$$\text{Precision} = \frac{\text{True positive}}{\text{True Positive + False Negative}} \tag{4.2}$$

$$\text{Recall} = \frac{\text{True positive}}{\text{True Positive + False Negative}} \tag{4.3}$$

Confusion Matrix

The confusion matrix is defined as:

$$\begin{bmatrix} \text{TP} & \text{FP} \\ \text{FN} & \text{TN} \end{bmatrix} \tag{4.4}$$

where:

- TP is the number of true positives,
- FP is the number of false positives,
- FN is the number of false negatives,
- TN is the number of true negatives.

True positive(TP) attribute predicted a label and then tested correctly for the ground truth. True Negative(TN) stands for the model that is not a part of the ground truth and can not be predicted. On the other hand, False positive (FP) means the model predicts but is not a part of the ground truth and False negative (FN) denotes the model is part of the ground truth but does not predict.

Mean Average Precision(mAP)

To analyze the performance of the object detection and segmentation system Mean Average Precision(mAP) is used. mAP formula is based on metrics such as recall, precision, intersection over union and confusion matrix. So this is calculated by the average precision(AP) for each class and takes the average over classes.

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i \tag{4.5}$$

F1 score

F1 score mainly computes the balance between recall and precision. Basically when recall and precision give the highest F1 score, this finds the most favorable confidence score threshold.

$$\text{F1 score} = \frac{2(\text{Precision x Recall})}{\text{Precision + Recall}} \tag{4.6}$$

4.1.3 YoloV8

The model we have used is YoloV8[39], which is a one-stage object detection model for computer vision. YoloV8 is the latest version of Yolo, which has built on the success of previous versions. The model holds up a full range of AI tasks involving detection, tracking, segmentation, and classification. Moreover, this model goes along with both developer experiments and architectural improvements. Compared to the other versions of the Yolo model, this version is easier to use, faster, and has improvements in features. We have used the resources and documentation of the model for training, prediction, validation, and deployment. In a wide range of object detection, image classification, and custom tasks YoloV8 is the choice of the highest quality. In figure 4.2, this is an image of an architecture model of the YoloV8. In this model, there are a total of 53 convolutional layers. There are 5 different variants: nano (n), small (s), medium (m), large (l), and extra large (x) in the YoloV8 model. Figure 4.2 represents the architecture of the YOLOv8 model.

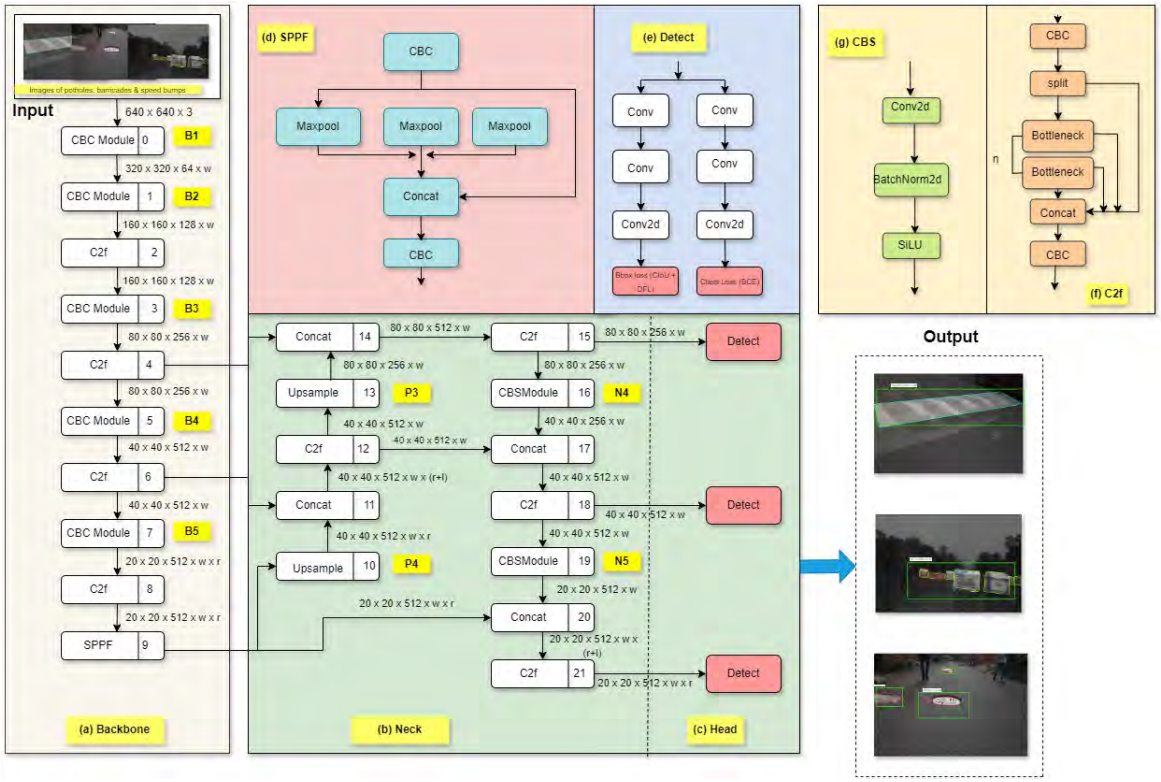

Figure 4.2: YOLO v8: Model Architecture

Backbone:

In the architecture of the YOLOv8 model, the backbone is the initial part, which works as draw-out features from the input images. From its convolutional layers, it does down sample the input image and, meanwhile, conserves important visual data.

Neck:

In the part of the neck, the architecture follows the backbone. It also works as an additional layer to magnify feature representation. In this layer, there are some

techniques included, like spatial pyramid pooling (SPP) and feature pyramid networks(FPN), to capture multiscale features.

Head:

In this part, the final predictions of the dataset are made. This layer is responsible for class probabilities for every grid, predicting the bounding boxes in the feature map.

SPPF:

SPPF (Spatial Pyramid Pooling Fusion) is a component that comprises spatial pyramid pooling. Without losing spatial information, it helps to capture factor information at many scales. It also merges these multiscale features for better performance.

The architecture represents that this is an extension of the object detection model, which performs semantic segmentation. This segmentation is for the input images. The backbone of the model is a CSPDarknet53 feature separator; this C2f module follows the Yolo traditional architecture. This is a faster implementation of the C2 module. The C2 module stance for the CSP (Cross Stage Partial) Bottleneck with 2 convolutions. In the module of the C2f, these two segmentation heads help train the model semantic segmentation masks for each image. In addition, the module of C3 takes the output of the last module of bottleneck. The bottleneck module works to reduce the cost of computation, which helps speed up the time of training. Further, there are five detection modules and a prediction layer, which helps to show the result more efficiently and bring the state-of-the-art on different segmentation and object detection by maintaining higher efficiency, less time, and high speed.

4.1.4 YoloV5

YOLO models are state-of-the-art [39], and YoloV5 is another success of the previous versions, built with more features and improvements. This version is well known for its speed. Besides, this model is optimized for deployment on various hardware platforms, including CPUs, and GPUs, also suitable for both real-time and batched scenarios. YoloV5 comes in five different sizes: nano (n), small (s), medium (m), large (l), and extra large (x). This model has three parts: the backbone, neck, and head. Figure 4.3 represents the architecture of the YOLOv5 model.

Backbone:

The model YOLOv5 typically takes on a convolutional neural network (CNN) as its backbone. This backbone is designed for efficiency and accuracy. This stage includes architectures like CSPDarknet53, which is a modified version of Darknet and extracts high-level features.

Neck:

The neck of the YOLOv5 architecture plays a pivotal role in feature fusion and resolution adjustment. Besides, this module combines features, often incorporating

modules like PANet (Path Aggregation Network) or FPN (Feature Pyramid Network) from different scales, and increases the model's ability to detect objects of various sizes.

Head:

The final predictions are made in the head module based on the features brought out by the backbone and neck. This layer is a combination of detection layers which include features like bounding boxes, and objectness scores in the feature map. This module not only includes improvements like focal loss but also improves accuracy and validation in object detection tasks.

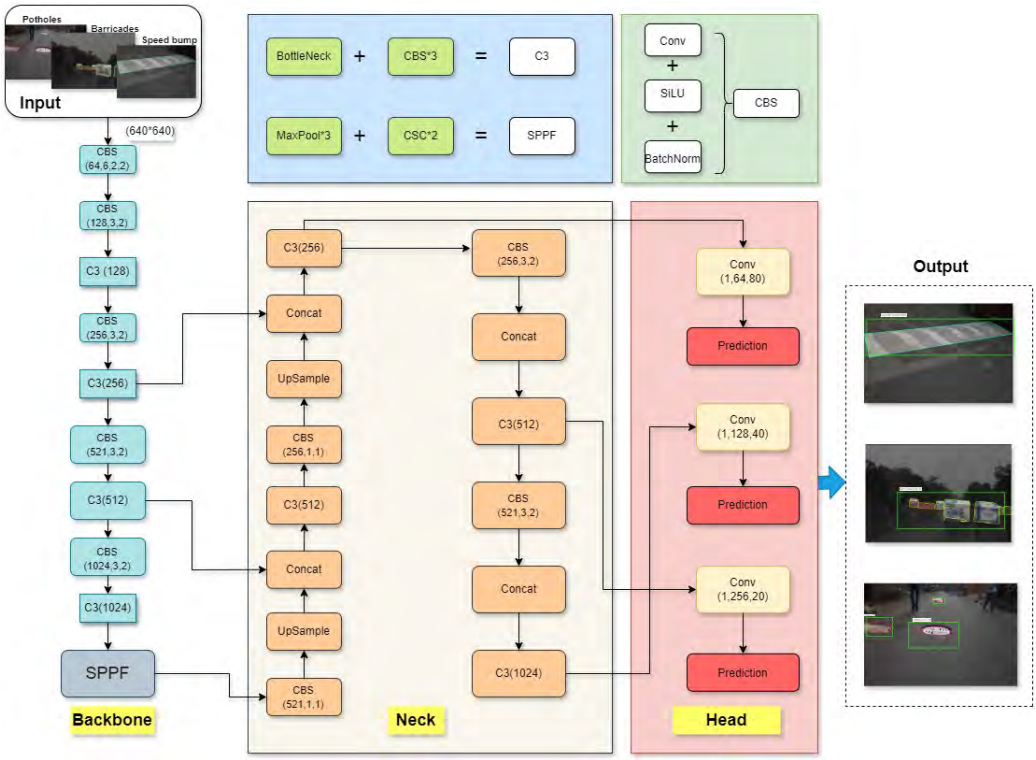

Figure 4.3: YOLO v5: Model Architecture

In this model, the backbone is CSPDarknet53, and the CBC module is used for feature extraction. The backbone is a pre-trained network, so it helps reduce the spatial resolution. Further, the Spatial Pyramid Pooling (SPPF) module performs the assembly of the data that is received from the input images and returns output with a fixed length. This module works to improve the model's running time and speed. This model uses the methods of the Path Aggregation Network (PAN) and Feature Pyramid Networks (FPN). Moreover, FPN does the up-sampling of the output with feature maps C3, C4, and C5, and PANet is for feature combination. The multiple convolution sampling operations generate these. Besides, this network generates a new map for detecting objects at different scales, as YoloV5 utilizes the CSPNet strategy for the separation of the feature map. The input images go through an input layer. The backbone requires feature extraction by different maps in the dimensions, which are conveyed with the sizes of 80×80, 40×40, and 20×20 for

detecting objects in the images accordingly. Then the YOLO layers can determine the outputs of detecting images; as a result, it shows class, size, location, and score. Therefore, YoloV5 gives three outputs: the classification of the detected samples, the sample bounding box, and their scores. It uses binary cross entropy (BCE) to calculate the class loss and object loss.

Overall, the model YOLOv5 represents a state-of-the-art solution for object detection tasks and offers a balance between accuracy, and efficiency, along with mitigation of deployment and customization for object detection.

4.1.5 YoloV7

The YOLOv7 model [39] is a notable advancement in the field of object detection. It is also known for exceptional performance, and remarkable all known real-time object detectors in terms of speed and accuracy with 30 FPS or higher on GPU V100. Basically, in the short dataset, this model can work faster without any pre-trained weights. Another advantage is it optimized the loss function to get better performance. In computer vision, this model can perform various tasks like segmentation, pose estimation, object detection, and classification [39]. It works in various domains beyond the COCO dataset. However, there is a significant need to fine-tune the settings for particular tasks. The next official iteration of the YOLO family, the YOLOv7 model, is a modified version. Therefore, YOLOv7 is a powerful tool for real-time object detection in terms of less expensive hardware, which makes it obtainable for a wide range of applications in computer vision tasks. Figure 4.4 represents the architecture of the YOLOv7 model.

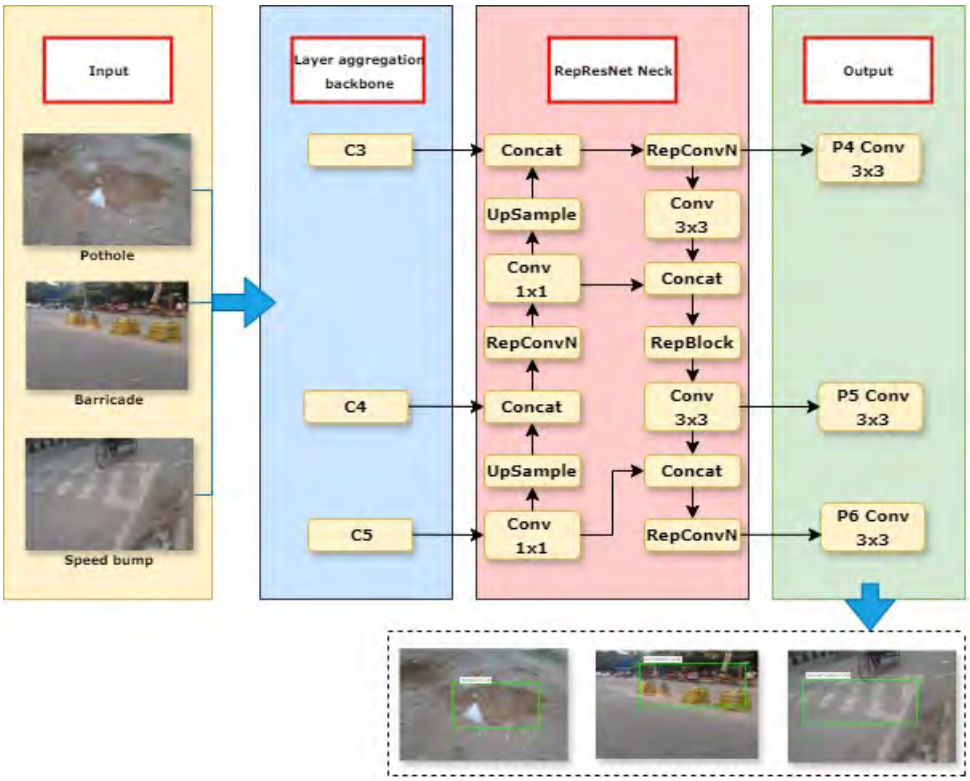

Figure 4.4: YOLO v7: Model Architecture

Backbone:

The model make use of an improved version of Darknet architecture with the CSP-Darknet53 backbone. CSP means Cross-Stage Partial connections. CSP amplifies gradient flow and information exchange between layers and extracts features with the help of max-pooling and skips connections from the input image.

Neck:

The neck module merges features from different layers of architecture to increase better representation. As its neck, this architecture uses PANet (Path Aggregation Network) which combines features from different spatial resolutions.

Head:

YOLOv7 has three detection heads small, medium, and large. So, each of them is responsible for detecting objects at different scales. Also, it predicts bounding boxes using anchor boxes. The module applies non-maximum conquering to filter out not-required detections. Moreover, the head module anticipates bounding boxes, objectness scores, and class probabilities.

The model YOLOv7 makes use of anchor boxes because this helps to predict object sizes and locations. These anchor boxes can be different in particular ratios and scales. In the feature map, this model predicts class probabilities, which determines the chances of an object belonging to each class. Then this model determines the objectness score and calculates the confidence that an object has in each class. Moreover, for the final predictions, the outputs from all detection heads are combined. The loss function calculates the accurate, confident forecast and the localization. Like all other YOLO models, this model comes in different sizes small(S), medium(M), large(L), and extra large(X) which are varied with complexities and inference speeds. Therefore, this model is a combination of efficient feature extraction, precise predictions, and feature integration to bring off real-time object detection and segmentation.

4.1.6 Models' architecture comparison

In this section, we are giving an overview of the architecture matrix and comparison for the models YOLOv5, YOLOv7, and YOLOv8. Table 4.1 represents the models' architecture comparison.

Feature	**YOLOv5 Model**	**YOLOv7 Model**	**YOLOv8 Model**
Release date	June 2020	July 2022	January 2023
Architecture	CSPDarknet backbone, PANet head, SPP, CSP	Improved CSP-Darknet, ELAN	Enhanced CSP-Darknet, PANet head, SPP, ELAN
Backbone	CSPDarknet53	Modified CSP with ELAN	More improved CSPDarknet with ELAN
Neck	PANet (Path Aggregation Network)	PANet with additional modification	More improvement in PANet
SPPF	Yes	Yes	Yes
Model Variants	YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x	YOLOv7-tiny, YOLOv7, YOLOv7-x	YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, YOLOv8x
ELAN	No	Yes	Yes
Parameters	7.5m (YOLOv5s), 21.2m (YOLOv5m), 46.5m (YOLOv5l), 86.7m (YOLOv5x)	6.2m (YOLOv7-tiny), 37m (YOLOv7), 71m (YOLOv7-x)	6.2m (YOLOv8n), 12m (YOLOv8s), 25m (YOLOv8m), 43m (YOLOv8l), 68m (YOLOv8x)
Anchor-free	No	No	Yes
Training time	Shorter	Moderate	Faster training with new features
Performance	Better balance of speed	Better optimal speed-accuracy	Diverse detection in scenarios

Table 4.1: Models' architecture comparison

Chapter 5

Description of Data

We have created a custom dataset for our model to detect and segment unidentified objects on the road (Barricade, Normal Road, Pothole, Speed Bump), etc. We have taken real-time video data and then sampled all the video into images. After sampling, we annotated the objects inside the images and created a source dataset. From the source dataset, we have pre-processed the data using augmentation techniques to increase the size of our dataset keeping the ratios of the classes similar to each other. Then we split the dataset into train, test, and valid data so that every class contributes the same ratio in this split.

5.1 Source and Collection

5.1.1 Collection Process

In this research, we have used a Samsung Android mobile phone camera as a filming device to collect video samples of the roads and streets containing our mentioned classes. At first, the mobile phone was placed in a horizontal manner which enabled us to record the videos in a landscape view. Videos were recorded by hand but from a height which is suitable for the camera placement in the front of the average car keeping in mind the real-life application scenarios. The videos were in 720p and 1080p in resolution. The video file formats were in (.mp4). The video footages were collected both in the daytime and the nighttime.

5.1.2 Data Collection Location

For the data collection, we have chosen roads and streets from both inside and outside Dhaka City, Bangladesh. Keeping in mind the real time implementation, our videos capture various types of the same class representing the challenging and diverse road scenario of Bangladesh. The videos generated within Dhaka city, Bangladesh are examples of the city scenario of Bangladesh whereas videos collected from outside Dhaka are representations of Bangladeshi good highways. Thus, the collected data locations in our custom-made dataset are divided into 2 parts -

- **Inside Dhaka City:** Mohammadpur (Noorjahan Road, Housing Limited, Chanmia Housing), Manik Mia Avenue, Dhanmondi, Khilgaon (Bashabo, Goran), Uttara (Sector 4, Sector 7).

- **Outside Dhaka City:** Dhaka-Mawa highway, Padma Bridge.

For the potholes and speed bumps, we have taken video footage from the streets inside Dhaka city. Most of these streets are made of concrete and are light gray in color. For normal roads and barricades, we have taken video footage from the main roads around Dhaka city and the highways where the roads are paved and have darker colors with proper road markings.

5.1.3 Source Footage Scenarios

Various weather and light conditions were taken into consideration when gathering the video material in order to increase the diversity of the dataset and improve the efficacy of the training models. We took video during the day in both clear and cloudy conditions, as well as in the light of sunset. We have also recorded footage at night in both clear and rainy conditions, right after sunset. The footage shot in a variety of weather conditions and at different times of the day makes our dataset more diversified and ideal for training the algorithms for maximum efficiency. In a variety of lighting and weather scenarios, the training models can identify and categorize potholes, barricades, regular roads, and speed bumps from these videos.

Clear Daylight

The daylight videos which were shot during clear weather, were taken around 11 AM to 2 PM so that the sunlight is present in the environment and the objects in the videos are clearly visible. The images extracted from the video show a clear representation of these objects and has no blurry or shaky effects so that the training models can detect and segment the objects in more accuracy. Figure 5.1 shows a picture of speed bump extracted from the footage of sunny clear weather. In Figure 5.2, a pothole is found under clear sunlight which shows no noise or blurriness.

Figure 5.1: SpeedBump in sunny weather

Figure 5.2: Pothole in sunny weather

Clear Nighttime

We have gathered different video footage during the nighttime in clear weather so that our model can be trained with images that have low light conditions. During low-light situations, generic phone cameras try to gather more light while capturing by slowing the frame rate which makes the video footage blurry and noisy. The integrated camera used in most hardware components is not high quality and can

generate blurry footage during the night light. To replicate these kinds of situations, we have tried to gather night light footage under the roadside lampposts which are mostly clear, but have very little blurry and noisy effect to them. Figure 5.3 and Figure 5.4 are some photographs of barricade and speed bump respectively under low light but clear condition.

Figure 5.3: Barricade at Nighttime

Figure 5.4: Speed Bump at nighttime

Rainy Daylight

During the daytime, rainy weather can create unique situations like potholes filled with water, wet roads, and blurry vision that can be an issue for the vehicles to make dynamic decisions. Our dataset consists of footage taken in bad rainy weather that has water-filled potholes, wet normal roads that can reflect sunlight, barricades, and speed bumps. Here, Figure 5.5 and Figure 5.6 show two daytime scenarios of a speed bump in rainy weather.

Figure 5.5: SpeedBump at Rainy Daylight

Figure 5.6: SpeedBump in rainy weather

Rainy Nighttime

The nighttime video footage itself has low light and a blurry effect to them, we have tried to add the rainy situation to our dataset by capturing night roads in rainy weather. The footages are taken after sunset under street lights that have wet roads and objects in them. The night street lights reflect on the roads and the objects which create unique situations for our model to learn in a more effective way. Figure 5.7 shows a scenario of normal road after rain under nightlight and Figure 5.8 is a scenario of pothole filled with water after rain at nighttime.

Figure 5.7: Normal Road at Rainy Night

Figure 5.8: Pothole at Rainy Night

Dusk time

The lighting situation before and after the sun sets can create different effects on the camera integrated into any vehicle or hardware device. Our dataset consists of footage taken at Dusk time that has a little sunlight and street light effects on them that can replicate these kinds of situations for the model to understand the road environments during dusk time. As shown in Figure 5.9 and Figure 5.10, the photographs of barricades and speed bumps under low light at dusk time show a bit of blurriness and noise.

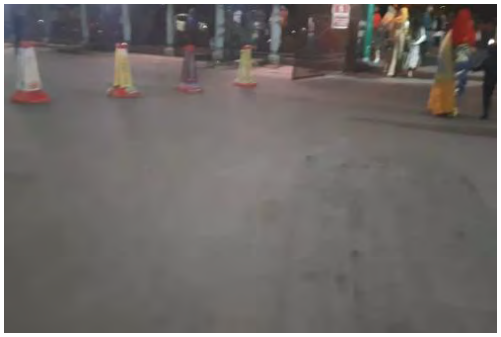

Figure 5.9: Barricade at Dusk

Figure 5.10: Speed Bump at Dusk

5.2 Classes of Data

5.2.1 Barricade

The video footage of the barricades is taken from the main roads and highways around the city consisting of a total of seconds. The barricades are of different sizes, types, and colors. They include the roadblocks that are used by law enforcement agencies, the triangular cones used in parking spaces (figure 5.11), and some other kinds of barricades that are used as road dividers (figure 5.12). The video footage is taken from different angles for a better understanding of the model's training.

Figure 5.11: Triangular Cone Barricade

Figure 5.12: Road Divider Barricade

5.2.2 Normal Road

For this research, we have decided to classify the plain roads as normal roads that have no potholes or barricades in the region. As we have taken footage of potholes, barricades, and speed bumps, there were also parts of the streets that had no objects like those and were safe to drive at a constant speed. We have decided to mark those parts as normal roads. Also, there was a lot of footage having a plain highway road which was shot for the Normal Road class. In total, we have gathered seconds of footage for the Normal Road class. Figures 5.13 and 5.14 represent the normal road.

Figure 5.13: Normal road inside the city

Figure 5.14: Normal road in highway

Figure 5.15: Pothole in Concrete Road

Figure 5.16: Pothole in Paved Road

Figure 5.17: Concrete Speed Bump

Figure 5.18: Speed-Bump of Paved Roads

5.2.3 Pothole

We have taken seconds of video footage of potholes from around Dhaka city. Some of the potholes are filled with water and some of them are not. The potholes vary in their diameter where some are around 8-12 inches in diameter, and very few of them are up to 30 inches. Figures 5.15 and 5.16 represent the potholes in the road.

5.2.4 Speed Bump

We have taken the Speed Bump footage from the main city roads and also from the narrow streets of Dhaka, Bangladesh. The speed bumps are from 10 to 20 cm in width and 4 to 8 cm in height. The speed bumps from the main roads around the city are marked with white color markers but the speed bumps from the narrow streets do not contain any color and are made of concrete (Figures 5.17 and 5.18).

5.3 Image Sampling from Video Footages

From the collected video footage from various parts of the city, we created our custom dataset. A new project was opened in Roboflow under instance segmentation where we uploaded our videos. Then, we sampled the videos in 1 FPS (frame per second) meaning we took 1 frame from each second of the videos. The videos had a grand total of 2302 seconds of footage, which was sampled into 2302 images. The sampling was conducted using Roboflow which resulted in 2302 images of different

conditions of the Barricade, Normal Road, Pothole, and Speed Bump in (.jpg) format.

5.4 Data Pre-Processing

We have annotated the objects present in the whole source image dataset using the in-built annotation tool of Roboflow. We then used augmentation techniques to increase the dataset. The augmentation was run keeping in mind to avoid class imbalance across the dataset. Then we have split the data using Python into train, valid, and test in 75%, 20%, and 5% respectively. We have pre-processed 2302 total source images into 9630 total augmented images and created our custom dataset.

5.4.1 Data Annotation

Roboflow, being a popular site for annotation and efficient dataset management has various types of tools for annotation. We used their smart polygon tool and polygon tool. The smart polygon annotation tool auto-annotates a known object or class according to its understanding using multiple points. We also used the polygon tool which enabled us to manually pinpoint the points around our class objects to accurately specify the position. Using these tools we accurately annotated our custom dataset of 2302 images. Four classes named "Barricade", "Normal Road", "Pothole", and "Speed Bump" was created and these objects in the images were marked and annotated. Roboflow generates the annotated files according to the needs of the models. For our YOLO models, the labels were generated in a separate (.txt) file format for each corresponding image in Pytorch format. Table 5.1 represents the class distribution after annotation.

Annotated Class	**Number of Images**
Barricade	556
Normal Road	626
Pothole	614
Speed Bump	650
Total Source Images	**2302**

Table 5.1: Class Distribution after Annotation

5.4.2 Data Augmentation

We employed several augmentation techniques on our dataset to expand the number of images with diverse effects so that our models could be trained with a larger number of images from different settings. We used ten different ways to produce a new custom dataset that was more than four times larger than the source image dataset. The augmented images are designed such that they may duplicate various hardware and environmental circumstances, allowing the models to be introduced to new situations. The enhanced photos were combined with the original images

retrieved from the video clip, yielding a total dataset of 9630 images. Geometric movement of the objects was avoided during the augmentation process to preserve the label file data of the source images, therefore the annotation data was kept and duplicated across all the enhanced images for each source image. The augmented photos were separated into train, test, and valid directories to ensure that no class imbalance occurs in our custom dataset. Figure 5.19 shows multiple augmented images after being processed with multiple augmentation techniques.

Figure 5.19: Images after Augmentation

Augmentation Parameters

- **Brightness Jittering:** We have used values from 0.5 to 1.5 to change the image brightness level to enhance the model's ability to handle varying lighting conditions.

- **Contrast Adjustment:** We have used linear contrast adjustment by scaling pixel values in the range from 0.5 to 1.5 so that the model can perform better in underexposed or overexposed images.
- **Gaussian Blur:** We have used values from 0 to 3.0 to mitigate overfitting and to help generalize the detection models to noisy and less focused objects.
- **Noise Injection:** We have injected noise into the augmented images to prevent overfitting to clean training data and enhance the models' robustness.
- **Hue Adjustment:** We have used values from -50 to 50 in Hue Adjustment so that the models' color invariance improves.
- **Saturation Adjustment:** This augmentation technique was used in our dataset to improve the models' ability to handle variation in color saturation and color intensities.
- **Value (Brightness) Adjustment:** We have adjusted the brightness of the third color channel using linear contrast adjustment so that the model performs well in uneven lighting.
- **Sharpness Adjustment:** We have applied alpha value from 0 to 1.0 to sharpen the image which can improve the model's sensitivity to fine details.
- **Cutout:** Random rectangular areas within the images were cut out and replaced with zero values so that the models focus on different parts of the image and avoid overfitting.
- **Random Erasing:** Random erasing was done to enhance the robustness of the models and regularize the prevention of overfitting.

Implementation Details

We implemented the augmentation process in Python, using several key libraries for image processing and augmentation. Table 5.2 represents class distribution after augmentation.

- **Python:** We have used Python 3.12 as our primary language for the augmentation.
- **Python Imaging Library:** To load the source images and to save the output images, PIL(Python Imaging Library) was used in the process.
- **NumPy:** We imported NumPy for array manipulation and conversion between the images and arrays.
- **Imgaug:** Importing imgaug was necessary because it enables to use of a wide range of augmentation techniques.
- **os:** os was used for directory and file manipulation during the process.
- **shutil:** shutil library was used for copying files from the source directory to the destination directory.

Class	Images after Augmentation
Barricade	2605
Normal Road	2265
Pothole	2324
Speed Bump	2436
Total Augmented Images	**9630**

Table 5.2: Class Distribution after Augmentation

Simulation of Real World Environment

Our augmented dataset ensured that one source image goes through three different versions of augmentation and can create real-world scenarios artificially. We have used augmentation methods that can replicate hardware malfunction, vision difficulties, and environmental hostility.

Figure 5.20: Low Light Simulation

Figure 5.21: Blurry Vision Simulation

In Figure 5.20, a low-light environment is simulated by the Brightness Jittering augmentation method for the model to be introduced with low-light real-world conditions that can replicate night-time and dawn-time scenarios. Figure 5.21 is an augmented image that has been processed with Gaussian Blur and Noise Injection that can replicate blurry images taken by generic integrated cameras which helps our models to be familiar with low-resolution photos and low-powered hardware.

Figure 5.22: Vision Blockage Simulation

Figure 5.23: Color Limitation Simulation

Here, in Figure 5.22, the augmented images were created using multiple augmentation methods like Noise Injection, Cutout, and Random Erasing that can simulate water drops blocking the vision during rainy weather. Figure 5.23 shows an augmented image using Grayscale and Cutout which makes our Machine Learning model

more familiar with the low-resolution images taken by the integrated camera modules.

5.4.3 Dataset Splitting

We have decided to have a split of 75%, 20%, and 5% as Train, Validation, and Test data respectively. To preserve the class balance, we have used Python to divide the images from each class, keeping the same ratio. Here, Figure 5.24, Figure 5.25 and Figure 5.26 shows the class-wise image distribution for train, valid and test of the dataset

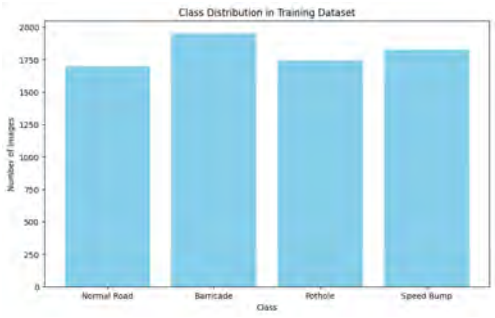

Figure 5.24: Distribution for Train

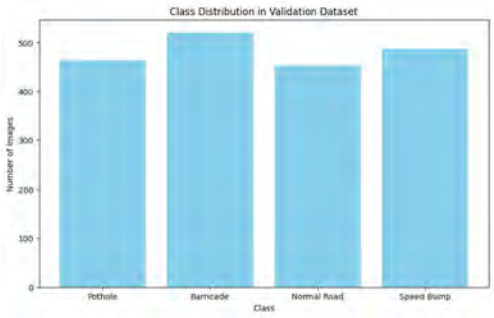

Figure 5.25: Distribution for Valid

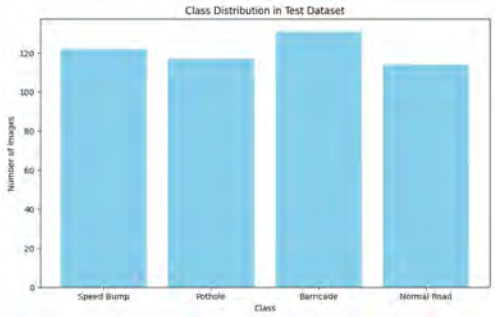

Figure 5.26: Distribution for Test

5.5 Dataset Limitations

We created our dataset from the streets of Bangladesh. All of the processes were done manually by our team to enrich the dataset. However, our new dataset has some limitations.

- **Dataset Diversity:** Although we have tried to diversify our dataset to uphold the challenging road scenarios of Bangladesh, it may not adequately represent all the variances that exist across different locations. We have enriched the dataset with images mainly from Dhaka city and nearby areas, missing out on rural roads and other places of Bangladesh. Different places may have varied sorts of the same classes (potholes, barricades, speed bumps), which we attempted to handle, but our dataset may not cover every potential difference.

- **Use of Roboflow:** Roboflow was selected as the primary platform for sampling and annotating our dataset due to its wide range of tools and options. However, roboflow has some constraints that were a setback. Initially, we selected "instance segmentation" as the task, which covers both detection and segmentation since it was our primary goal. But, the generated dataset label files were not compatible with object detection models and worked only for instance segmentation models.

- **Model Compatibility:** The annotated label files were not compatible with models solely designed for object detection. YOLOv5, YOLOv7, and YOLOv8 are all models that support instance segmentation and object detection which aligned with our needs but limited us to fewer models.

- **Annotation format challenges:** many detection models use JSON COCO annotation format, we also converted to the mentioned annotation format from our initial PyTorch annotation files. We then discovered that there are fundamental differences between file formats, class structures and label parameters. So, despite our efforts, we could not train our custom dataset with a range of state-of-the-art object detection models.

Chapter 6

Implementation and Results

6.1 Convolutional Neural Network(CNN) models

Our custom dataset has been evaluated using 3 instance segmentation YOLO model variations. After using YOLOv5, YOLOv7, and YOLOv8 to train our dataset, we saw that the model was able to accurately identify and segment the aforementioned groups.

6.1.1 Implementation

Two distinct iterations of the YOLOv5, YOLOv7, and YOLOv8 models were used to train the custom dataset. YOLOv5 was implemented in two variations: YOLOv5s and YOLOv5x. YOLOv7 was trained with its base variant YOLOv7s and a larger YOLOv7x variant. YOLOv8 was trained using YOLOv8s and YOLOv8x. In all models, the standard form denoted by the letter "s" strikes a decent balance between speed and accuracy. However, the "x" option requires a lot more time and computational resources in order to achieve better accuracy.

The experiments were conducted in various environments including Google Colab, kaggle, and high-configuration PCs. The dataset had a total of 4 classes namely - Barricade, Normal Road, Pothole, and Speed Bump. For the prepared custom dataset, performance is measured in the following terms:

- mAP50: Mean Average Precision of an IoU (Intersection over Union) threshold of 0.5 which is a common evaluation metric.

- mAP50-95: Over a range of IoU thresholds from 0.5 to 0.95, the mean average precision.

6.1.2 Training Parameters

The training for the models was done using various training parameters. The parameters changed as per the model requirement and resource availability.

YOLOv5s

YOLOv5s is a relatively smaller and faster model which is a good choice for real time object detection and segmentation having limited computational resources. It is good for smaller datasets where simpler models are generalized well.

Parameter	Value
Layers	165
Parameters	7406513
Gradients	0
GFLOPs	25.7
Image Size	640
Batch Size	32
Epochs	20
GPU	2xNVIDIA Tesla T4
Training time	1.010 hours
Optimizer	SGD

Table 6.1: YOLOv5s Training parameters

YOLOv5x

This model of YOLOv5 is larger and more accurate comparatively. However, these have increased computational requirements. The model is suitable for larger datasets where there are complex and diverse scenarios and a higher level of accuracy is required.

Parameter	Value
Layers	330
Parameters	88263041
Gradients	0
GFLOPs	264
Image Size	640
Batch Size	32
Epochs	20
GPU	2xNVIDIA Tesla T4
Training time	2.416 hours
Optimizer	SGD

Table 6.2: YOLOv5x Training parameters

YOLOv7

The base variant of YOLOv7 strikes a balance between efficiency and performance with its optimized architecture and training techniques making it suitable for various applications such as surveillance, autonomous driving, etc. The YOLOv7 offers improved speed and accuracy for real-time object detection and segmentation tasks.

Parameter	Value
Image Size	640
Batch Size	32
Epochs	20
GPU	NVIDIA GeForce RTX 4090
Training time	1.953 hours
Optimizer	Adam

Table 6.3: YOLOv7 Training parameters

YOLOv7x

YOLOv7x is an expanded version of YOLOv7 that is designed to suit specific criteria or handle certain use cases. It contains tailored designs, sophisticated features, and unique training techniques to improve performance in cases where traditional YOLOv7 may not be sufficient, providing versatility and adaptability across a wide range of applications.

Parameter	Value
Image Size	640
Batch Size	32
Epochs	20
GPU	NVIDIA GeForce RTX 4090
Training time	1.154 hours
Optimizer	Adam

Table 6.4: YOLOv7x Training parameters

YOLOv8s

This variation of YOLOv8 is the standard size which provides a reasonable balance between the speed and accuracy measured in mAP along with model size.

Parameter	Value
Layers	195
Parameters	11781148
Gradients	0
GFLOPs	42.4
Image Size	640
Batch Size	32
Epochs	20
GPU	NVIDIA L4
Training time	0.769 hours
Optimizer	AdamW

Table 6.5: YOLOv8s Training parameters

YOLOv8x

The largest and most complex variant among the YOLOv8 models, YOLOv8x provides the highest accuracy but with the cost of high computational resources and model size.

Parameter	Value
Layers	295
Parameters	71724508
Gradients	0
GFLOPs	343.7
Image Size	640
Batch Size	32
Epochs	20
GPU	A100
Training time	0.990 hours
Optimizer	AdamW

Table 6.6: YOLOv8x Training parameters

6.1.3 Results for YOLOv5s

The training results after 20 epochs for YOLOv5s are showcased below -

Class	Box(Precision	Recall	mAP50	mAP50-95)
All	0.833	0.734	0.805	0.529
Barricade	0.772	0.725	0.766	0.473
Normal Road	0.866	0.89	0.94	0.821
Pothole	0.766	0.566	0.661	0.375
Speed Bump	0.926	0.756	0.854	0.448

Table 6.7: YOLOv5s Training Results after 20 epochs

As can be seen from Table 6.7, the YOLOv5s model showed good results for all classes which are over 80%. Although the normal road was above 90%, we can also see that potholes had the lowest mAP50 among the classes.

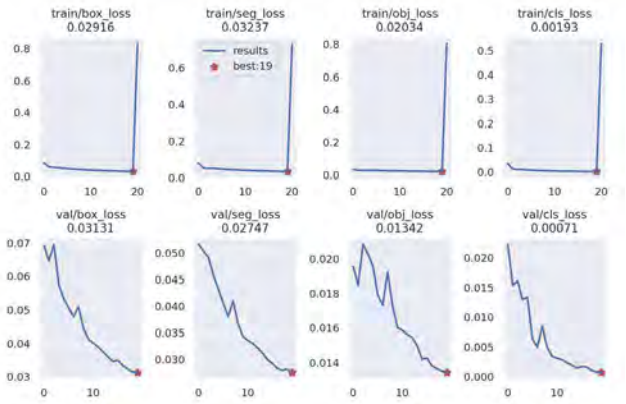

Figure 6.1: Training results for YOLOv5s

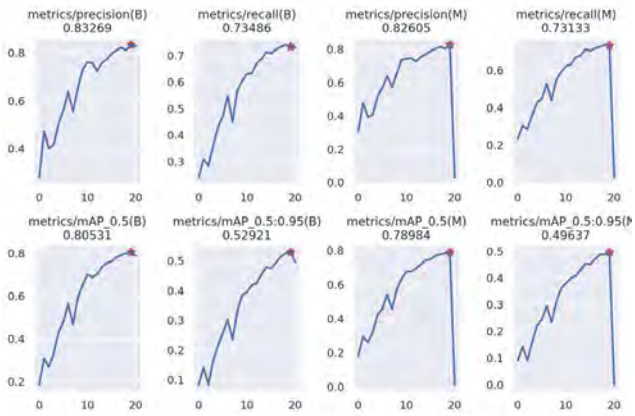

Figure 6.2: Training results for YOLOv5s

Figure 6.1 and Figure 6.2 showcase the train vs loss, validation vs loss, and metrics graphs. It can be seen from the graphs that the training and validation losses show a gradual decrease in all classes until epoch 19 and rise up at epoch 20. The final training losses show values 0.02916 for box loss, 0.03237 for segment loss, 0.02034

for object loss and 0.00193 for class loss. The validation losses display a decrease with the values 0.03131 for box loss, 0.02747 for segment loss, 0.01342 for object loss, and 0.00071 for class loss. The overall values and the corresponding graphs suggest that the model's performance has improved despite having an increase in losses at the last epoch.

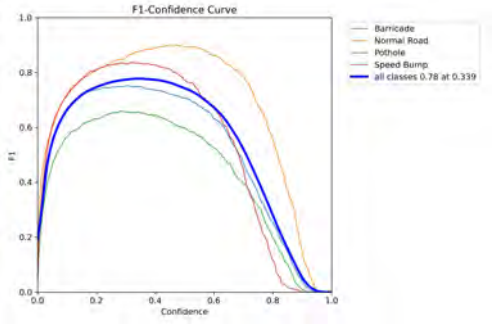

Figure 6.3: F1 Curve Yolov5s

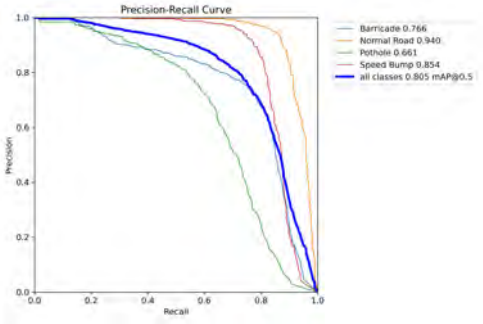

Figure 6.4: PR Curve Yolov5s

Figure 6.3 and 6.4 visualize the F1 confidence curve and the Precision vs Recall curve respectively. The F1 confidence graph shows a higher curve for normal road class and a lower curve for potholes. All classes' curve maintains an average form of curve in the PR curve graph. The PR curves show how successfully the model separates true and false positives at various recall levels for each class. In this case, the graph shows high precision and recall for normal road and speedbump classes whereas it has low performance for potholes.

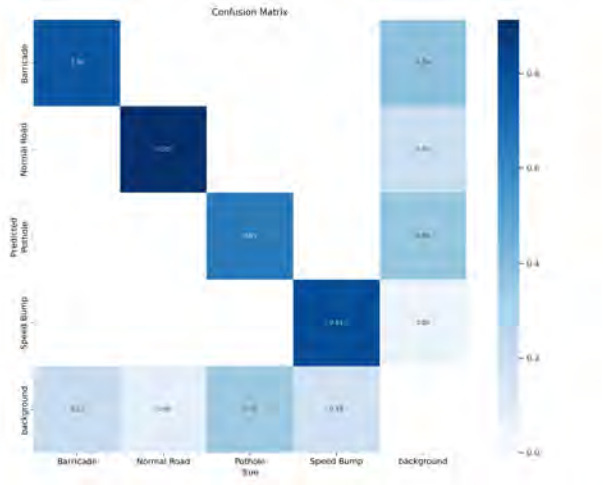

Figure 6.5: Confusion Matrix for YOLOv5s

Figure 6.5 visualizes the confusion matrix. The occurrences where the predicted labels match the actual labels are represented by the diagonal cells, which go from top left to bottom right. Better performance for that particular class is indicated by higher values in these cells. The matrix shows higher accuracy diagonally, denoting a good true positive scenario.

6.1.4 Results for YOLOv5x

After training for 20 epochs, the following results for YOLOv5x were generated -

Class	Box(Precision	Recall	mAP50	mAP50-95)
All	0.85	0.822	0.876	0.647
Barricade	0.799	0.799	0.838	0.582
Normal Road	0.854	0.941	0.954	0.865
Pothole	0.822	0.688	0.788	0.519
Speed Bump	0.927	0.86	0.926	0.622

Table 6.8: YOLOv5x training results after 20 epochs

Table 6.8 shows that YOLOv5x got very good results for all classes which is over 87%. Normal Road and Speed Bump classes got 95% and 92% mAP50 values respectively which are great results.

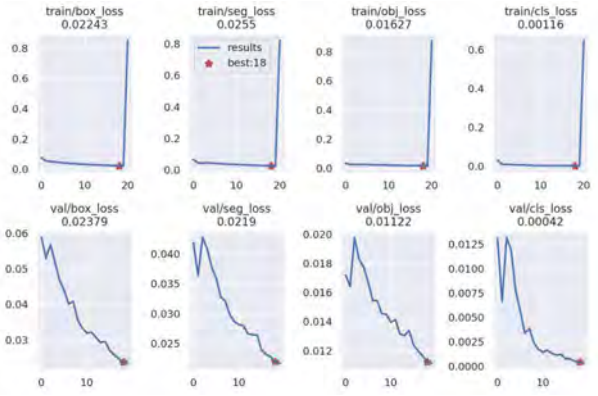

Figure 6.6: Training results for YOLOv5x

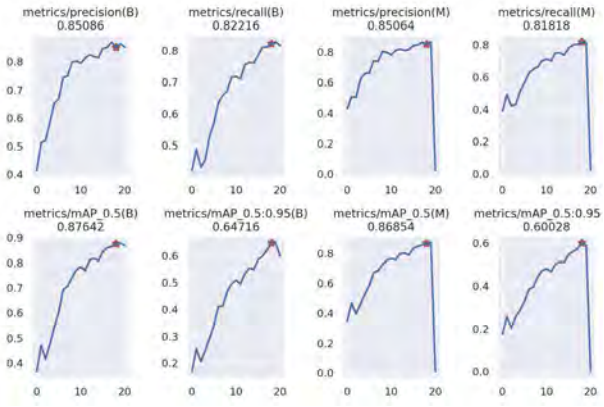

Figure 6.7: Training results for YOLOv5x

Figures 6.6 and 6.7 visualize the train vs loss, validation vs loss and metrics graphs for YOLOv5x. The graphs denote that the best was found at the 20th epoch. The

graphs show the trend for box-loss, segment loss and object loss for train and validation.

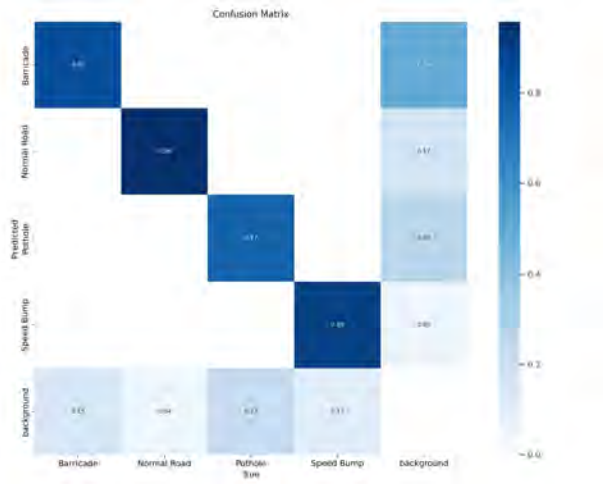

Figure 6.8: Confusion Matrix for YOLOv5x

The confusion matrix for the YOLOv5x shown in Figure 6.8 gives a deeper color diagonally which represents a good true positive ratio for all the classes.
The F1 confidence curve and the Precision vs Recall curves showcased in Figures 6.9 and 6.10 indicate a higher curve for the normal road class and speed bump class in both graphs indicating great performance for those classes. The model is able to perform better for those classes.

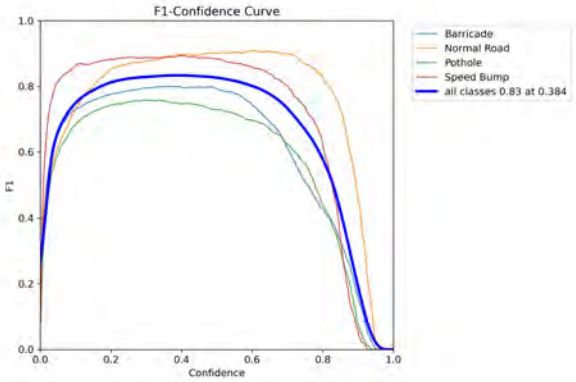

Figure 6.9: F1 Curve Yolov5x

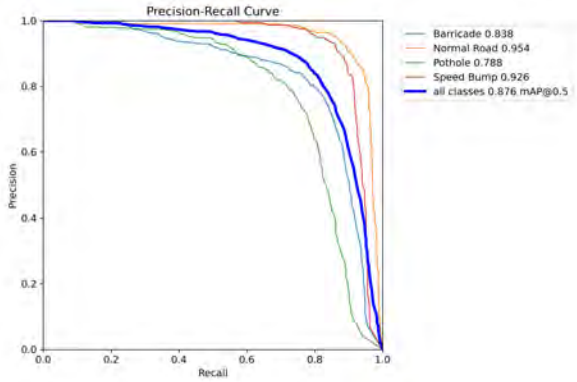

Figure 6.10: PR Curve Yolov5x

6.1.5 Results for YOLOv7s

Class	Box(Precision	Recall	mAP50	mAP50-95)
All	0.825	0.725	0.8	0.557
Barricade	0.774	0.735	0.78	0.531
Normal Road	0.883	0.87	0.925	0.816
Pothole	0.839	0.47	0.646	0.396
Speed Bump	0.802	0.821	0.849	0.484

Table 6.9: YOLOv7s training results after 20 epochs

The base YOLOv7s model showcased good results for all classes which is 80% as shown in table 6.9. Although the normal road was above 92%, we can also observe a speed bump giving a good mAP above 84%. It can be seen that the pothole had the lowest mAP.

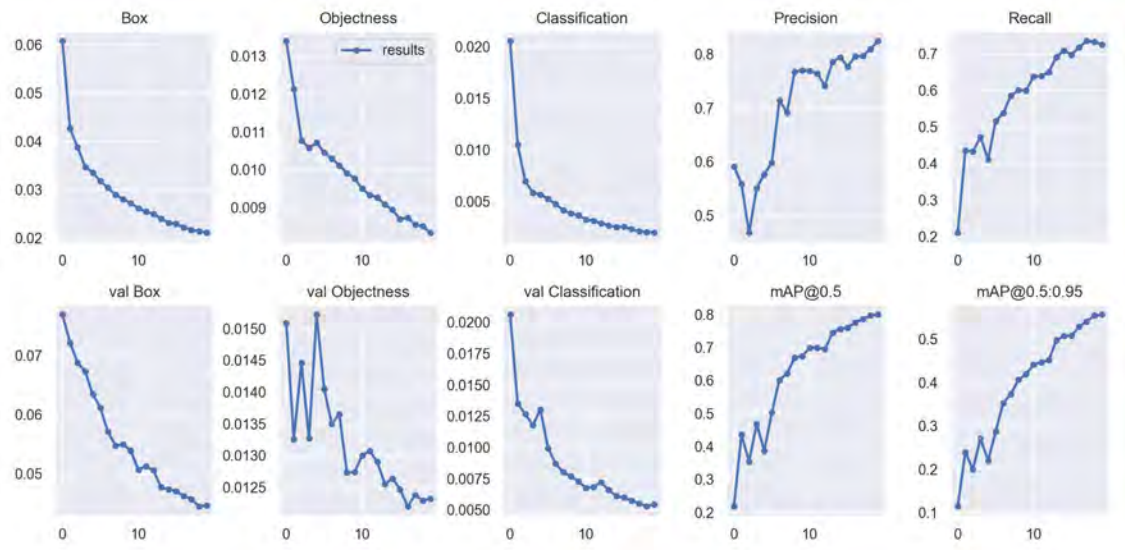

Figure 6.11: Training results for YOLOv7

Figure 6.11 showcases the result curves for train, validation, and metrics. The graphs have epochs along the x-axis, and respective loss or performance metric values along the y-axis. The graphs help visualize the results efficiently for the mentioned model.

The confusion matrix displayed in Figure 6.12 showcases a good true positive ratio forming a dark path along the diagonal of the matrix.

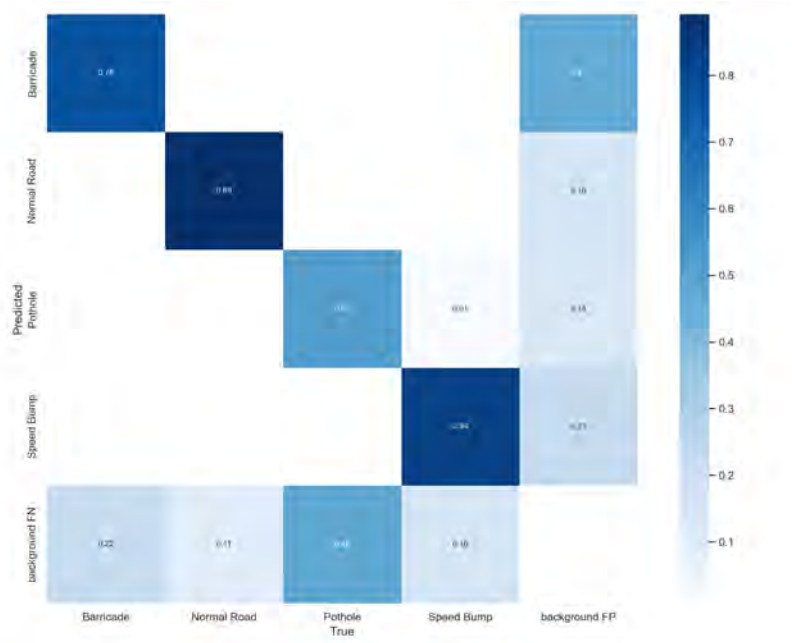

Figure 6.12: Confusion Matrix for YOLOv7s

The F1 confidence curve and the precision-recall curve are shown in Figure 6.13 and Figure 6.14 respectively. The F1 confidence curve reveals that the normal road and speed bump class curves have higher values when compared to others denoting good results for those classes. The PR curve produces comparable results across all classes. Similar to the F1 curve, the PR curve shows normal road and speedbump classes give better results, while pothole underperforms among the classes.

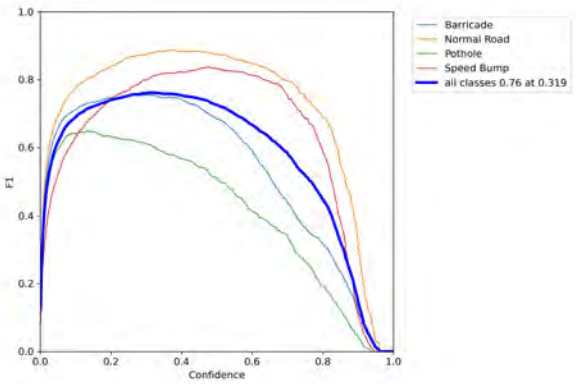

Figure 6.13: F1 Curve Yolov7

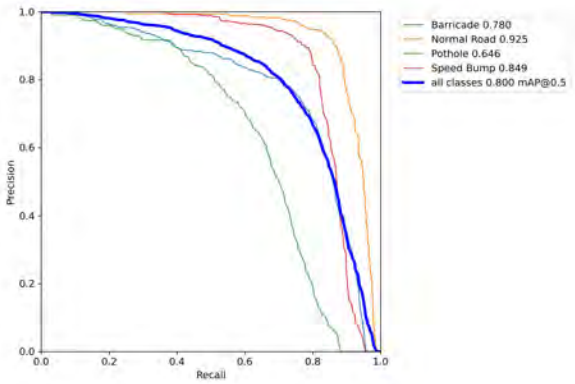

Figure 6.14: PR Curve Yolov7

6.1.6 Results for YOLOv7x

Class	Box(Precision	Recall	mAP50	mAP50-95)
All	0.678	0.53	0.583	0.331
Barricade	0.641	0.591	0.58	0.333
Normal Road	0.793	0.663	0.765	0.564
Pothole	0.618	0.297	0.379	0.183
Speed Bump	0.661	0.567	0.609	0.243

Table 6.10: YOLOv7x training results after 20 epochs

The YOLOv7x model displayed moderate results for all classes which is around 58% as shown in table 6.10. The table discloses a disparity among the classes as the normal road class has around 76% mAP50 value whereas the pothole class has the lowest of around 37%.

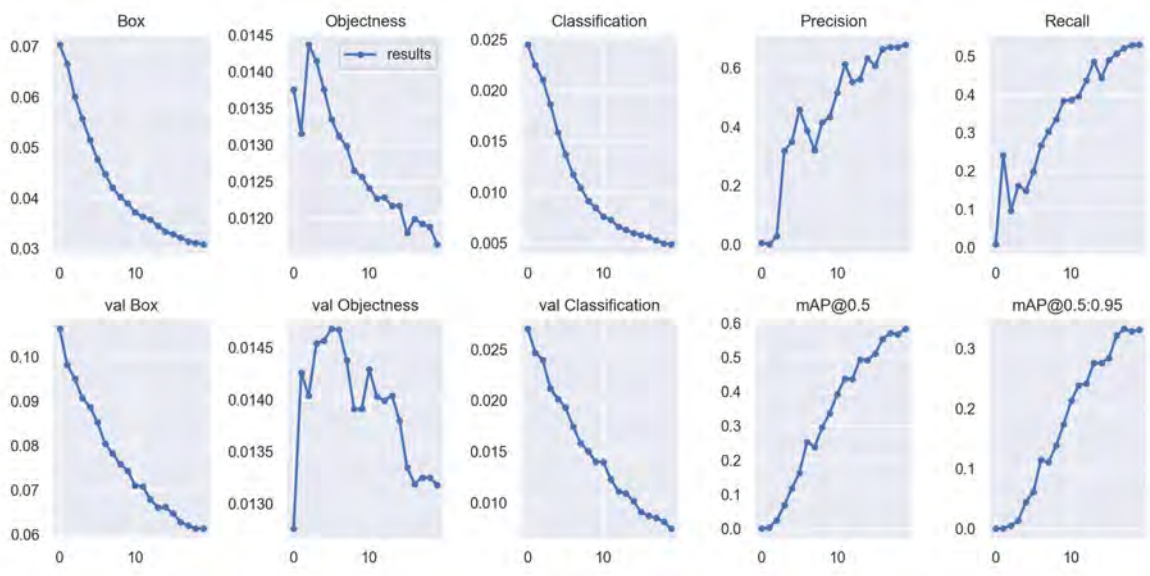

Figure 6.15: Training results for YOLOv7x

Figure 6.15 showcases the result curves for train, validation, and metrics. Along the X-axis, epochs are calculated whereas along the Y-axis respective loss value or performance metrics are measured in all the graphs. The graphs help visualize the training curve results for the YOLOv7x model.

The confusion matrix displayed in Figure 6.16 showcases a good true positive ratio forming a dark path along the diagonal of the matrix.

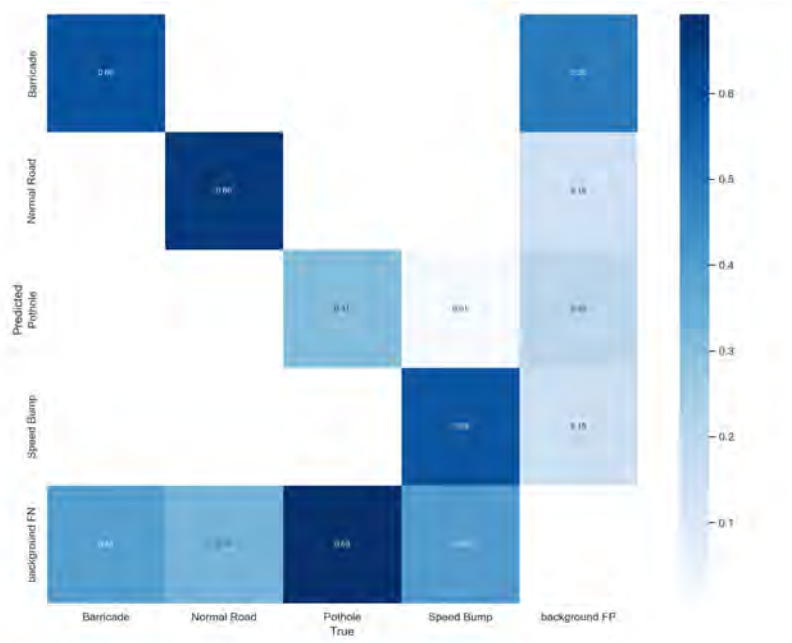

Figure 6.16: Confusion Matrix for YOLOv7x

The F1 confidence curve and the precision-recall curve are shown in Figure 6.17 and Figure 6.18 respectively. The F1 confidence curve reveals that the normal road class curve has a larger curve when compared to others while the pothole class has the lowest curve. The PR curve produces comparable results across all classes showing that pothole and normal road curves are most deviant from all classes curve.

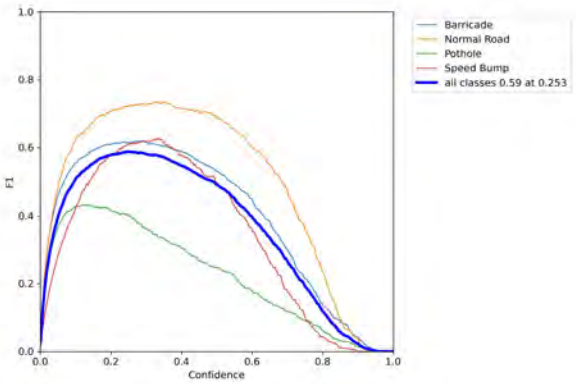

Figure 6.17: F1 Curve Yolov7x

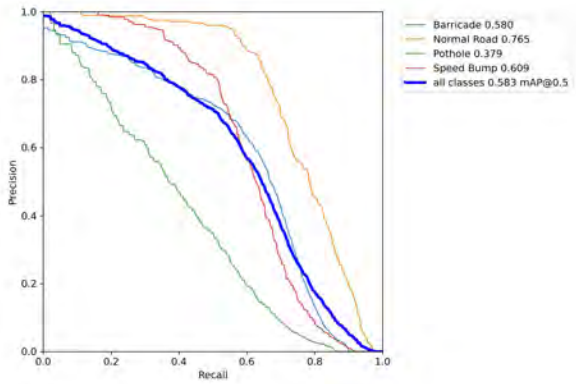

Figure 6.18: PR Curve Yolov7x

6.1.7 Results for YOLOv8s

Training completed after 20 epochs generated the following results -

Class	Box(Precision	Recall	mAP50	mAP50-95)
All	0.835	0.781	0.856	0.639
Barricade	0.792	0.759	0.822	0.587
Normal Road	0.884	0.907	0.956	0.883
Pothole	0.791	0.643	0.747	0.498
Speed Bump	0.873	0.814	0.899	0.586

Table 6.11: YOLOv8s training results after 20 epochs

The YOLOv8s model showed great results as seen in figure 6.11 for all classes which is nearly 86%. The normal road class had the most mAP50 value which is above 96%. We can also see that the pothole had the lowest mAP50 among all the classes which are displayed in Table 6.11.

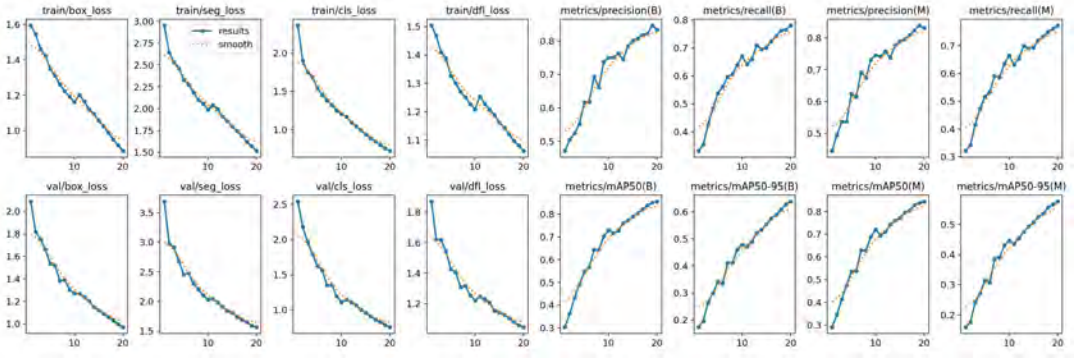

Figure 6.19: Training results for YOLOv8s

Figure 6.19 visualizes the result curves for train, validation, and metrics where epochs are placed along the X axis and the Y axis holds the loss and performance metrics value. The graphs show result curves going along with the smooth line denoting good results for the training.
In the confusion matrix for YOLOv8s shown in Figure 6.20, we see that the true positives for all classes have a high ratio, forming a darker color diagonally in the matrix.

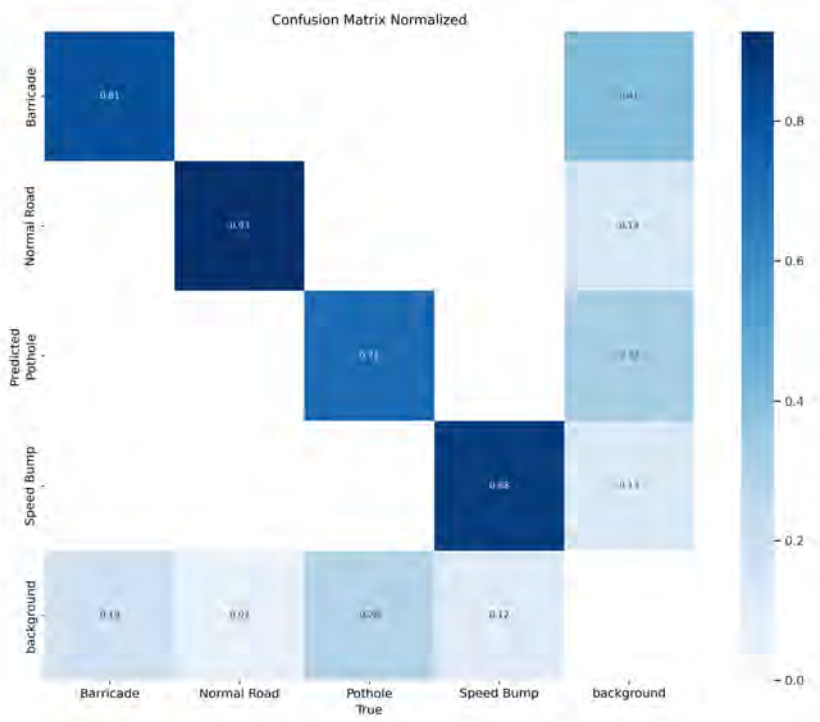

Figure 6.20: Confusion Matrix for YOLOv8s

The F1 confidence curve and the precision-recall curve are shown in Figure 6.21 and Figure 6.22 respectively. The F1 confidence curve shows the normal road curve to have much higher values compared to others. The PR curve also shows similar results across all classes where normal road and speedbump have higher curves.

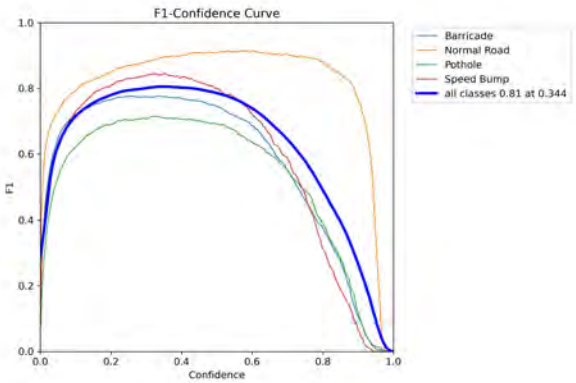

Figure 6.21: F1 Curve Yolov8s

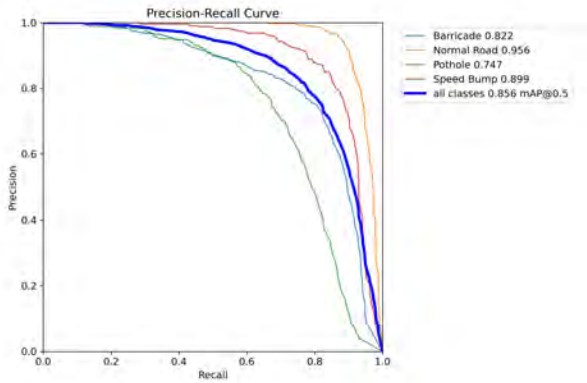

Figure 6.22: PR Curve Yolov8s

6.1.8 Results for YOLOv8x

Class	Box(Precision	Recall	mAP50	mAP50-95)
All	0.838	0.803	0.863	0.652
Barricade	0.791	0.785	0.828	0.606
Normal Road	0.846	0.903	0.953	0.889
Pothole	0.825	0.675	0.773	0.526
Speed Bump	0.891	0.847	0.898	0.586

Table 6.12: YOLOv8x training results after 20 epochs

The YOLOv8x model showed impressive results for all classes which is nearly 86% as shown in table 6.12. The normal road class had almost 95% mAP50 value and the speed bump gave a good mAP of almost 90%, we see that the pothole had the lowest mAP.

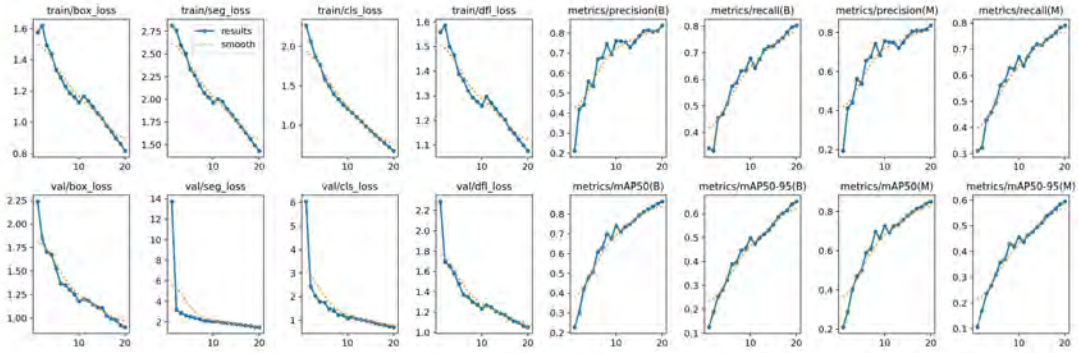

Figure 6.23: Training results for YOLOv8x

Figure 6.23 showcases the result curves for train, validation, and metrics. The X-axis holds the epoch count and the Y-axis shows the corresponding loss value or performance metrics. The graphs help visualize the results efficiently for the mentioned model.

The confusion matrix displayed in Figure 6.24 showcases a good true positive ratio forming a dark path along the diagonal of the matrix.

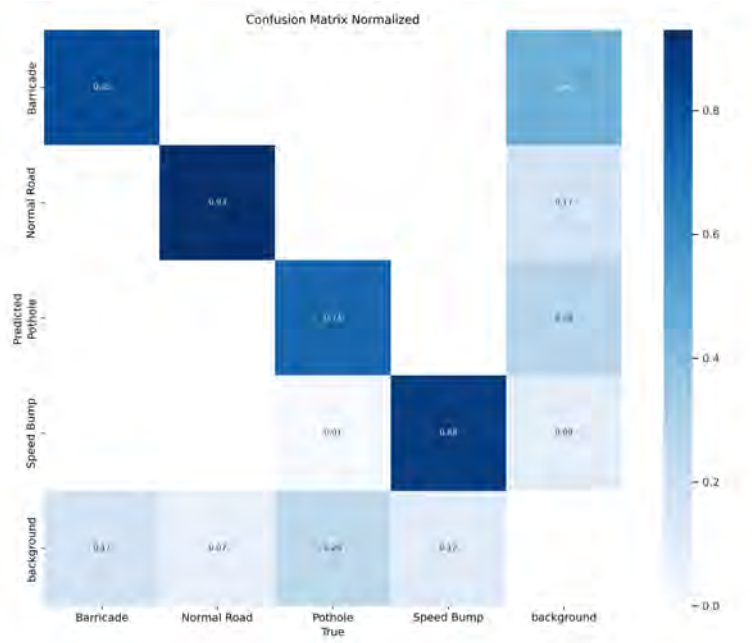

Figure 6.24: Confusion Matrix for YOLOv8x

The F1 confidence curve and the precision-recall curve are shown in Figure 6.25 and Figure 6.26 respectively. The F1 confidence curve reveals that the normal road class curve has high values when compared to others. The PR curve produces comparable results across all classes.

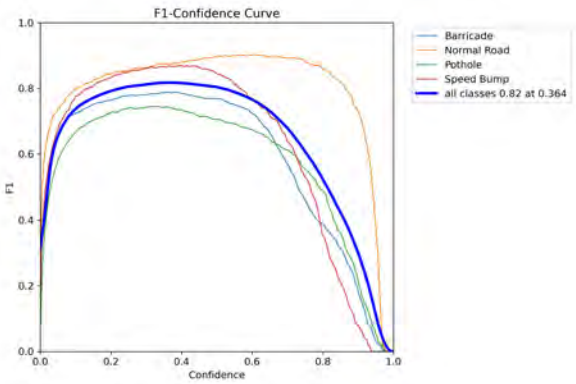

Figure 6.25: F1 Curve Yolov8x

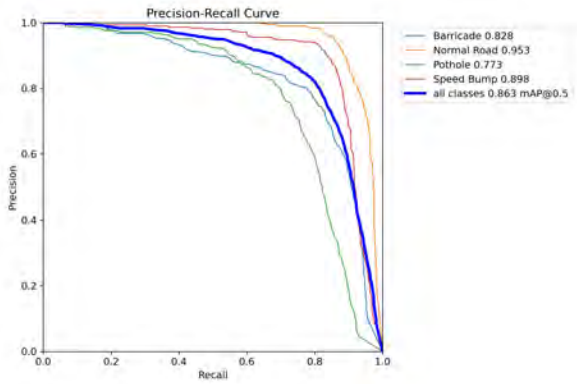

Figure 6.26: PR Curve Yolov8x

6.2 Segmentation

In this research, we have used our trained models that can segment the objects from the photographs which enables to partition the digital images into discrete segments of pixels. Our models have shown very good performance in parsing the images' complex visual data to differently shaped segments in all four classes that were selected for this research. As shown in the figures below, our trained model was able to detect as well as segment the objects in the photographs. The segmentation masks covered the edges of the detected objects with a colored overlay, which enabled the model to validate and test.

6.2.1 YOLOv5

In this section, The Figures 6.27, 6.28, 6.29, 6.30, 6.31 and 6.32 shows the segmentation of classes (Barricades, potholes, speed bumps), which is calculated by the model.

Figure 6.27: Class segmentation

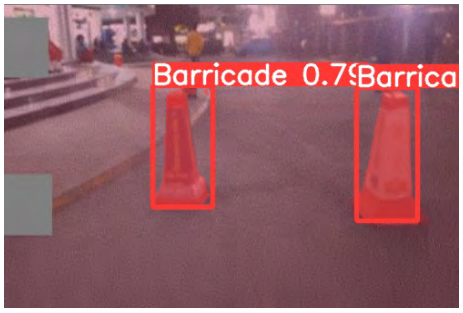

Figure 6.28: Barricade segmentation

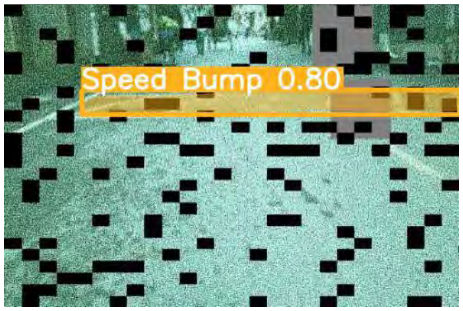

Figure 6.29: Speed bump segmentation

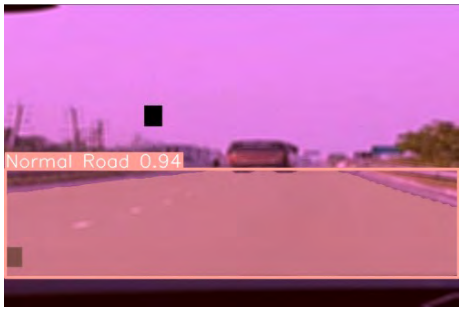

Figure 6.30: Normal road segmentation

Figure 6.31: Speed bump segmentation

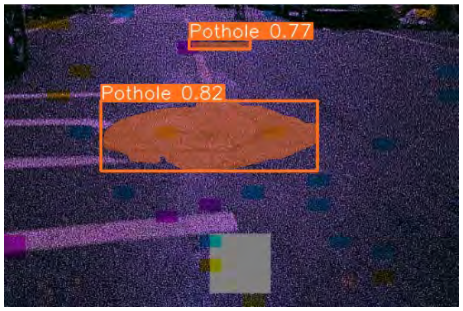

Figure 6.32: Pothole segmentation

6.2.2 YOLOv8

The Figures 6.33, 6.34, 6.35, 6.36, 6.37 and 6.38 represent the segmentation of classes, which is calculated by the model.

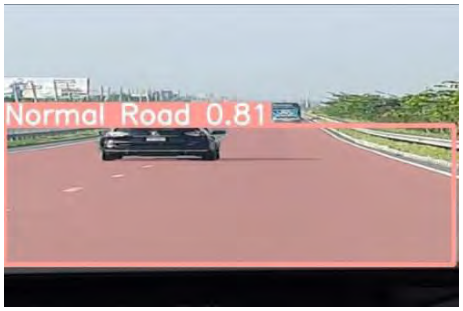

Figure 6.33: Normal Road segmentation

Figure 6.34: Class segmentation

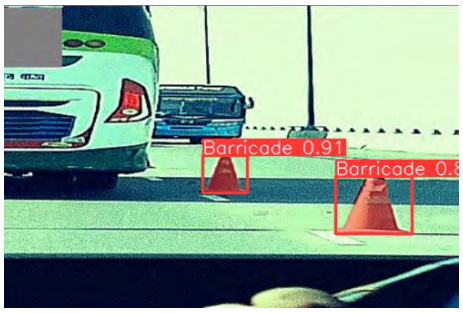

Figure 6.35: Barricade segmentation

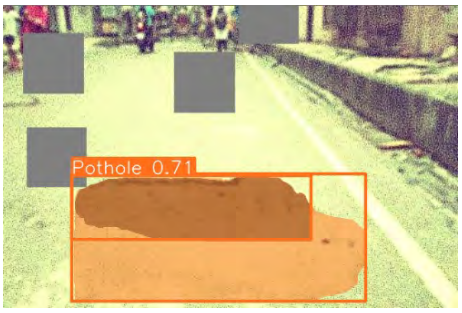

Figure 6.36: Pothole segmentation

Figure 6.37: Speed bump segmentation

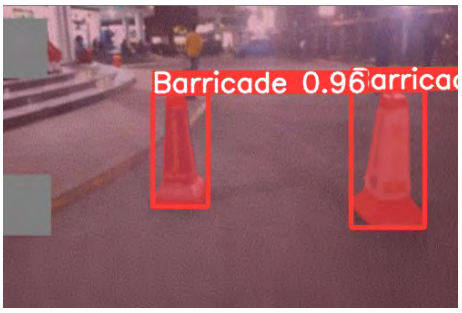

Figure 6.38: Barricade segmentation

6.2.3 YOLOv7

The Figures 6.39, 6.40, 6.41, 6.42, 6.43 and 6.44 shows the segmentation of our determined classes, which is calculated by the model.

Figure 6.39: Barricade segmentation

Figure 6.40: Pothole segmentation

Figure 6.41: Pothole and Normal road segmentation

Figure 6.42: Pothole segmentation

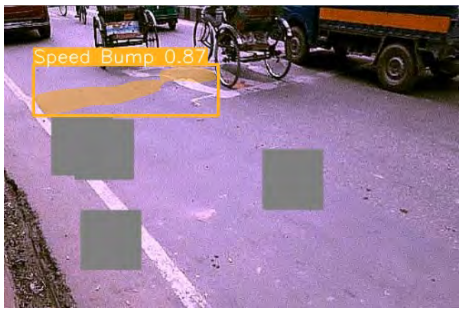

Figure 6.43: Speed bump segmentation

Figure 6.44: Barricade segmentation

6.3 Comparative Numerical Analysis

6.3.1 Comparison among our models trained with our dataset

With our newly created custom dataset of more than 2300 images which was then augmented with 10 different techniques, the final dataset size was over 9600 images. The images were collected from the streets of various places in Dhaka city, Bangladesh. we trained the popular YOLO models specifically YOLOv5, YOLOv7, and YOLOv8 with a standard and large variant for all the models. While some parameters and resources varied for the models; image size, number of epochs, and batch size were kept the same. The results for the models varied from each other and so did their performance. The table 6.13 below resonates with the mAP50 and mAP50-95 performance evaluation metric values for all classes of all the models we have trained.

Model	mAP50	mAP50-95
YOLOv5s	0.805	0.529
YOLOv5x	0.876	0.647
YOLOv7	0.8	0.557
YOLOv7x	0.583	0.331
YOLOv8s	0.856	0.639
YOLOv8x	0.863	0.652

Table 6.13: Comparative Numerical Analysis

We can present a comparative analysis from the table according to the mean Average Precision (mAP) metrics at IoU thresholds of 0.5 (mAP50) and across a range

from 0.5 to 0.95 (mAP50-95).

YOLOv5
The YOLOv5s achieve a mAP50 value of 0.805 and 0.529. This smaller and faster variant of YOLOv5s provides good performance with lower computational requirements.
On the other hand, YOLOv5x gets the highest mAP50 and mAP50-95 values among all models which are 0.876 and 0.647 respectively. The largest variant in the YOLOv5 series benefits from the more complex architecture leading to greater precision-recall and making it suitable for the detection of small or obscured objects.

YOLOv7
The YOLOv7 has a mAP50 of 0.8 and mAP50-95 of 0.557. While it performs moderately, it is significantly below the YOLOv5x and YOLOv8 versions in terms of overall performance.
Moreover, YOLOv7x has a mAP50 of 0.583 and a mAP50-95 of 0.331. This extended version underperforms compared to the other models, suggesting that the added complexity may not be as successful for this particular dataset and job.

YOLOv8
Showcasing mAP50 and mAP50-95 values of 0.848 and 0.624 respectively, the YOLOv8s strikes a balance between computational efficiency and performance which makes it a strong candidate for deployment where both accuracy and speed are important.
Again, the YOLOv8x scores a mAP50 value of 0.844 and a mAP50-95 value of 0.632. The model provides robust performance results close to YOLOv5x, showcasing the capability to handle complex detection and segmentation tasks with high precision.

In summary, YOLOv5x comes out with the highest mAP scores, indicating superior detection and segmentation capabilities for all specified road obstacles. However, YOLOv8s and YOLOv8x offer a good balance of accuracy and computational efficiency with high mAP scores. As the results show, according to the mAP values, YOLOv5x can be considered the best model among all the six models that were trained. Considering computational constraints, YOLOv8s and YOLOv8x can be considered as strong alternatives. This analysis emphasizes the need to choose the appropriate model and variation based on the unique accuracy, recall, and processing power necessities for autonomous road obstacle detection and segmentation in Bangladesh provided through the custom dataset.

6.3.2 Comparison among different online datasets

We have created our custom dataset from the streets of Bangladesh keeping in mind the Bangladeshi road scenarios so that we can train YOLO models that identify Bangladeshi road conditions accurately and promptly which would contribute to Bangladesh's autonomous driving scenario. Our dataset contains 4 classes in total named: pothole, speedbump, barricade, and normal road, and has a total of 9630 images.

Additionally, we have collected a publicly available benchmark dataset[38] containing 3243 images. This dataset has a total of 2 classes namely: pothole and speed-

bump. The dataset was created to be utilized in self-driving car technology to provide efficient navigation.

In a comparison between the YOLOv8 models run on our custom dataset and run on the benchmark dataset[38], we find some differences that show our dataset brings out better results that can help detect the objects more efficiently.

Model	**Custom Dataset**	**Benchmark Dataset**
YOLOv8s	0.856	0.728
YOLOv8x	0.863	0.701

Table 6.14: Comparison for All Classes

Model	**Custom Dataset**	**Benchmark Dataset**
YOLOv8s	0.791	0.556
YOLOv8x	0.873	0.531

Table 6.15: Comparison for Pothole Class

Model	**Custom Dataset**	**Benchmark Dataset**
YOLOv8s	0.825	0.899
YOLOv8x	0.891	0.87

Table 6.16: Comparison for Speed Bump Class

Here, Table 6.14 shows the differences in mAP50 scores in all classes, Table 6.15 shows the differences in mAP50 scores in the Pothole class, and Table 6.16 shows the differences in mAP50 scores for the Speed Bump class between the custom dataset and the benchmark dataset. Here, our custom dataset trained model shows better accuracy and consistency than the models trained on the benchmark dataset.

Chapter 7

Hardware Deployment

7.1 Proposed Device Implementation

In our proposed device, we have implemented hardware with our trained model that can make necessary decisions to run autonomously taking into account all the road objects from our dataset. We first planned the device implementation and deployed the model into the computer creating necessary commands for a prototype car to run autonomously avoiding the obstacles.

7.1.1 Pre-Deployment Planning and Strategy

We planned for a constructive deployment of an object detection system using a laptop computer and a microprocessor. Besides, we have to go through the strategy plans for how we can assess the specific hardware requirements based on their complexity and scale.

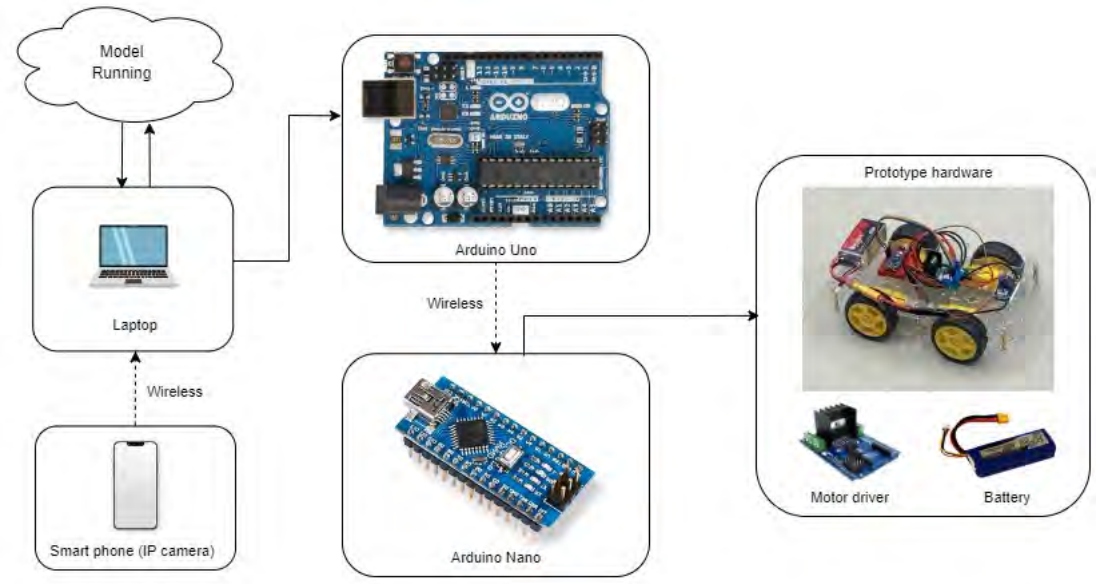

Figure 7.1: System Block Diagram

Moreover, after a cost analysis, we collected the necessary components, such as a camera, some additional sensors, wires, a microprocessor, and a prototype car. The

most pivotal part is selecting the right hardware.

- **Processor:** We have selected Ryzen 5 3500X 3.6GHz as the processing unit that can process the necessary computation.

- **Graphics Processing Unit** To properly run the algorithm, we needed a GPU that could be used to show the live feed and create bounding boxes on the feed to show the detection. We have used GTX 1650 Gaming OC as the GPU which is capable of handle the necessary graphics processing.

- **Microprocessors:** We have used an Arduino Nano in the prototype car and connected the car to an Arduino Uno wirelessly which was connected to our processing unit, the laptop computer.

- **Bluetooth Module** We have used two units of HC05 Bluetooth module to connect the prototype car and the laptop wirelessly.

- **Mobile Phone Camera** We have used a mobile phone camera with an application named 'IP Camera' that enables to send the video feed wirelessly to the laptop computer.

7.1.2 Configuration and Deployment

We have customized the hardware setup for the object detection task according to the requirements. This involves configuring the local system with necessary libraries and choosing the ideal processing unit for the model to be run properly like OpenCV, TensorFlow, and Ultralytics, NumPy, and Pandas. After that, we tried to ensure the proper setup of cameras and proper integration into the system. As we know, security configurations are essential for handling object detection algorithms.

- The mobile phone application 'IP Camera' was connected to the laptop through a local network and sent the video feed from the prototype car to the computer.

- The laptop receives the real-time video feed and shows the video on screen using the OpenCV library installed and the detection model takes the frames and detects objects on the frame. The OpenCV creates bounding boxes on the live feed and shows the detected objects.

- As soon as the computer detects any object, it sends the object information to the prototype car which is connected with Bluetooth and microprocessors.

- The microprocessor built in the prototype car receives the detection information and takes navigation decision for the car and executes in a way that the car changes its course to avoid the obstacles on the road.

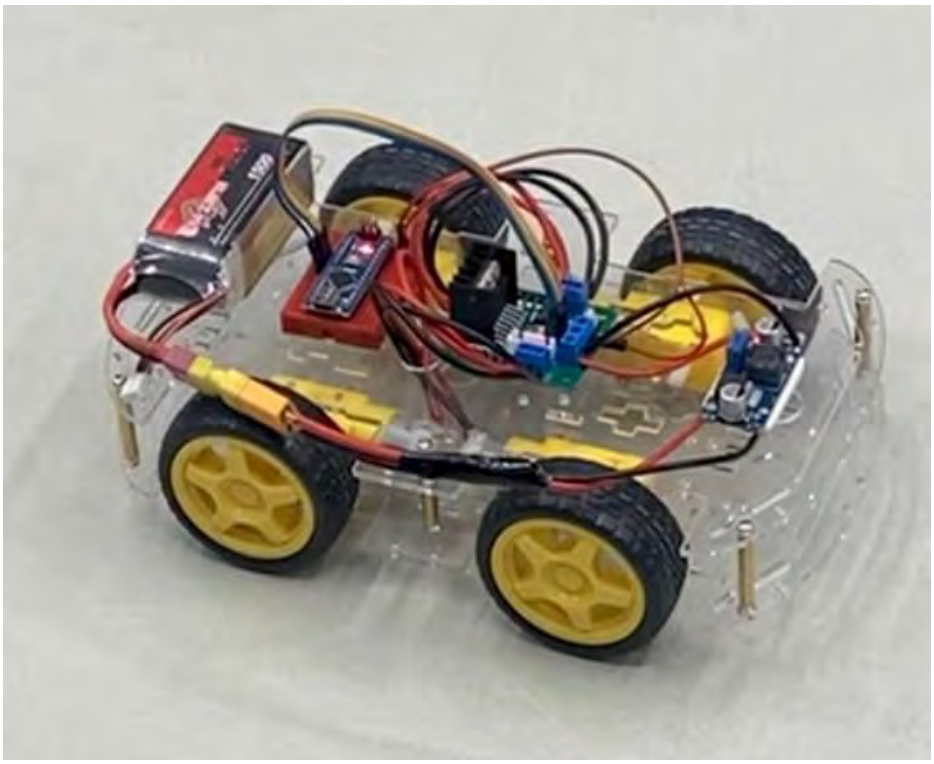

Figure 7.2: Protoype Car Unit

Moreover, figure 7.2 shows the prototype car unit of the hardware. Figures 7.3 and 7.4 show the bounding box generated by our trained model over the live camera feed from the laptop computer.

Figure 7.3: Detection interface

Figure 7.4: Detection interface

7.2 Hardware limitation

We have tried to implement our trained model in a hardware device as shown above. The proposed hardware works as a prototype. We faced various challenges a limitations for the hardware as we had limited resources. Additionally, the hardware faced various issues during assembling and working.

- **Processing Power Limitation:** The restricted processing capability of the Arduino Uno prevents the learned model from being executed, requiring an alternative hardware solution or improvement. In this case, the local computer.

- **Functionality Constraints:** Although our hardware is capable of detecting classes, it cannot segment or mask the detected object. This limits its application and full functionality.

- **Connectivity Dependence:** The Arduino uno and local computer is required to be connected at all times. This impacts the portability of the prepared device.

- **Image Stability Issues:** The webcam that is used as camera for the device is integrated with the local machine. As it is not integrated in the hardware prototype, the camera consists stability issues contributing to lower image stability, affecting the object detection capability.

7.3 Scalability and Future-Proofing

We have trained our model with our custom dataset keeping in mind the conditions of Bangladesh. Proper steps like increased processing power, cloud integration, modular architecture, AI model evaluation can help this system for scalability and future proofing.

- **Increased Processing Power:** Upgrading the GPU to more powerful models like NVIDIA RTX series and a higher-core CPU can significantly improve the processing capability of our hardware integration. This will enable the system to handle more complex models and large datasets.

- **Cloud Integration:** Integrating cloud computing can increase the storage system and processing power of the integrated system. Combining local processing with cloud resources can increase the system's ability to handle peak loads and large-scale data analysis without a reduction in the speed of the processing.

- **Modular Architecture:** Designing the whole system with a more modular architecture can improve the system for the real-life implementation of our

proposed system. If the system can be designed in such a way that can be integrated into motor vehicles using less power and more processing ability, it will bring a revolutionary change to the current autonomous vehicles.

- **AI Model Evaluation:** Continuously improving the object detection models with new data will help the system to be more fined in a short time. Implementing a feedback loop in the system to learn from the real-life environment and user interactions can enhance the predictive capabilities of the hardware system over a short period of time.

With proper research and implementation with greater resources, this can be scaled to be installed in full-sized vehicles after careful inspection and evaluation which will require a step-by-step process and go through multiple multi-layered developments.

Chapter 8

Future Work and Conclusion

8.1 Future Work

This research represents our ongoing efforts to increase road object detection and segmentation performance in the Bangladeshi context by creating a custom dataset and initial experiments using some of the most well-known Convolutional Neural Network models. Furthermore, a very extensive and comprehensive plan for future research and development can be formed with the findings of our experiments.

- **Dataset Improvement:** Our custom dataset contains 9630 images in total, Though the state-of-the-art Convolutional Neural Networks models work well in small datasets. Our custom dataset can be expanded and refined to contain more diverse objects and situations for the models to perform better in terms of the Bangladeshi road environment.
- **Model Refinement:** Extending the experiment into more well-known object detection and segmentation models like Yolact, Mask R-CNN, etc. to upgrade the detection and classification.
- **ViT models Integration:** We propose that the Vision transformer model, which is a higher version of the CNN model can be used in image segmentation and object detection in various domains. ViTs perform better in cases of bigger datasets. As the ViT models are able to compete with the well-known CNN models in terms of accuracy and speed, the results can open up scopes for the ViTs to be implemented in road object detection in the future.
- **Hardware and Software Integration:** This model can be implemented into a user interface for further and extensive use of the used algorithms. Also, the fine-tuned and smaller versions of the models implanted in this research can be deployed into low-powered devices which can help to guide drivers of vehicles.

As we have a custom dataset, we have tried to implement it in the well-known YOLO models. With a dataset specially prepared in the challenging Bangladeshi context, we believe that a lot of work can be done. Training our dataset in multiple other object detection and segmentation models and comparing the results might lead to better findings. Furthermore, the model showing good accuracy can be deployed to test in the real-time scenario. Further extensive research may lead to a model with great accuracy that may give excellent results in the Bangladeshi road scenario and boost the autonomous vehicle industry in the country.

8.2 Conclusion

Gradually the world is being introduced to newer technology in everything including the transportation sector with the introduction of Electric vehicles. While the near future is likely to be more focused on saving time for productivity using these artificially intelligent vehicles, improving the detection system will allow them to run in any kind of environment on any kind of surface. These can operate autonomously by effectively detecting the necessary things while driving using various neural network models for this detection. CNN-based models are commonly used models in this case. YOLO is a popular model in this segment. However, as newer integrations are introduced in the computer vision segment, prospects for improvement are always there. New and improved models can revolutionize this sector completely. In fact, the relatively new vision transformer model might be able to prove more effective. Through our research, we intend to provide insight and show the use of the newer model in this case. An expected $300 billion dollar market by 2027 [32] is still hesitant to enter the Bangladeshi market due to the challenging road scenarios of the country. Our custom-prepared dataset can be used to look for better accuracy and search for new models that provide better accuracy and precision in the South Asian road context with our hardware prototype. Therefore, extensive research and proper implementation might pave the way for the billion-dollar industry to make its mark on the roads of Bangladesh.

Bibliography

[1] A. Broggi, C. Caraffi, R. I. Fedriga, and P. Grisleri, "Obstacle detection with stereo vision for off-road vehicle navigation," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, 2005, pp. 65–65. DOI: 10.1109/CVPR.2005.503.

[2] M. A. Khan, *Speed breakers the accident makers*, en, https://www.thedailystar.net/news-detail-181157, Accessed: 2023-5-24, Apr. 2011.

[3] C. Fernandez, M. Gavilan, D. F. Llorca, *et al.*, "Free space and speed humps detection using lidar and vision for urban autonomous navigation," in *2012 IEEE Intelligent Vehicles Symposium*, Alcal de Henares , Madrid, Spain: IEEE, Jun. 2012.

[4] B. Wu, A. Wan, F. Iandola, P. H. Jin, and K. Keutzer, "SqueezeDet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Honolulu, HI: IEEE, Jul. 2017.

[5] J. Złotowski, K. Yogeeswaran, and C. Bartneck, "Can we control it? autonomous robots threaten human identity, uniqueness, safety, and resources," en, *Int. J. Hum. Comput. Stud.*, vol. 100, pp. 48–54, Apr. 2017.

[6] W. Cao, J. Yuan, Z. He, Z. Zhang, and Z. He, "Fast deep neural networks with knowledge guided training and predicted regions of interests for real-time video object detection," *IEEE Access*, vol. 6, pp. 8990–8999, 2018.

[7] M. A. Javed, E. B. Hamida, A. Al-Fuqaha, and B. Bhargava, "Adaptive security for intelligent transport system applications," *IEEE Intell. Transp. Syst. Mag.*, vol. 10, no. 2, pp. 110–120, 2018.

[8] N. Namala, "A study on potholes and its effects on vehicular traffic," Feb. 2018.

[9] *Speed bumps causing extensive damage to cars, according to new research*, en, https://www.honestjohn.co.uk/news/driving-1/2018-09/speed-bumps-causing-extensive-damage-to-cars-according-to-new-research/, Accessed: 2023-5-24, Sep. 2018.

[10] T. S. Adhikary, *4,200km of roads in bad condition*, en, https://www.thedailystar.net/frontpage/news/4200km-roads-bad-or-very-bad-1764895, Accessed: 2023-5-24, Jul. 2019.

[11] A. Ahmed, S. Islam, and A. Chakrabarty, "Identification and comparative analysis of potholes using image processing techniques," in *2019 IEEE Region 10 Symposium (TENSYMP)*, Kolkata, India: IEEE, Jun. 2019.

[12] J. Lee, D. Lee, Y. Park, S. Lee, and T. Ha, "Autonomous vehicles can be shared, but a feeling of ownership is important: Examination of the influential factors for intention to use autonomous vehicles," en, *Transp. Res. Part C Emerg. Technol.*, vol. 107, pp. 411–422, Oct. 2019.

[13] S. Shah and C. Deshmukh, "Pothole and Bump detection using Convolution Neural Networks," *2019 IEEE Transportation Electrification Conference (ITEC-India)*, Dec. 2019. DOI: 10.1109/itec-india48457.2019.itecindia2019-186. [Online]. Available: https://doi.org/10.1109/itec-india48457.2019.itecindia2019-186.

[14] M. Alawadhi, J. Almazrouie, M. Kamil, and K. A. Khalil, "Review and analysis of the importance of autonomous vehicles liability: A systematic literature review," en, *Int. J. Syst. Assur. Eng. Manag.*, vol. 11, no. 6, pp. 1227–1249, Dec. 2020.

[15] Y.-C. Chiu, C.-Y. Tsai, M.-D. Ruan, G.-Y. Shen, and T.-T. Lee, "Mobilenet-SSDv2: An improved object detection model for embedded systems," in *2020 International Conference on System Science and Engineering (ICSSE)*, Kagawa, Japan: IEEE, Aug. 2020.

[16] C. S. Choong, A. F. Ab. Nasir, A. P. P. Abdul Majeed, M. A. Zakaria, and M. A. Mohd Razman, "Machine learning approach in identifying speed breakers for autonomous driving: An overview," in *Lecture Notes in Mechanical Engineering*, ser. Lecture notes in mechanical engineering, Singapore: Springer Singapore, 2020, pp. 409–424.

[17] D. K. Dewangan and S. P. Sahu, "Potnet: Pothole detection for autonomous vehicle system using convolutional neural network," *Electronics Letters*, vol. 57, no. 2, pp. 53–56, 2020. DOI: 10.1049/ell2.12062.

[18] D. J, S. Dhakshana, A. S. A, K. R, and L. Parameswaran, "Deep Learning based Detection of potholes in Indian roads using YOLO," *2020 International Conference on Inventive Computation Technologies (ICICT)*, Feb. 2020. [Online]. Available: https://doi.org/10.1109/icict48043.2020.9112424.

[19] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial intelligence review*, vol. 53, no. 8, pp. 5455–5516, Apr. 2020. DOI: 10.1007/s10462-020-09825-6. [Online]. Available: https://doi.org/10.1007/s10462-020-09825-6.

[20] D. Liu, Y. Cui, Y. Chen, J. Zhang, and B. Fan, "Video object detection for autonomous driving: Motion-aid feature calibration," en, *Neurocomputing*, vol. 409, pp. 1–11, Oct. 2020.

[21] L. Alzubaidi, J. Zhang, A. J. Humaidi, *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of big data*, vol. 8, no. 1, Mar. 2021. DOI: 10.1186/s40537-021-00444-8. [Online]. Available: https://doi.org/10.1186/s40537-021-00444-8.

[22] D. Bhatt, C. Patel, H. Talsania, *et al.*, "CNN variants for Computer Vision: history, architecture, application, challenges and future scope," *Electronics*, vol. 10, no. 20, p. 2470, Oct. 2021. DOI: 10.3390/electronics10202470. [Online]. Available: https://www.mdpi.com/2079-9292/10/20/2470.

[23] S. Jain, N. J. Ahuja, P. Srikanth, *et al.*, "Blockchain and autonomous vehicles: Recent advances and future directions," *IEEE Access*, vol. 9, pp. 130 264–130 328, 2021.

[24] K. R and N. S, "Pothole and Object Detection for an Autonomous Vehicle Using YOLO," *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*, May 2021. DOI: 10.1109/iciccs51141.2021.9432186. [Online]. Available: https://doi.org/10.1109/iciccs51141.2021.9432186.

[25] A. A. Shaghouri, R. Alkhatib, and S. Berjaoui, "Real-Time pothole detection using deep learning," *arXiv (Cornell University)*, Jan. 2021. DOI: 10.48550/arxiv.2107.06356. [Online]. Available: https://arxiv.org/abs/2107.06356.

[26] R. Wang, Z. Wang, Z. Xu, *et al.*, "A real-time object detector for autonomous vehicles based on yolov4," *Computational Intelligence and Neuroscience*, pp. 1–11, 2021. DOI: 10.1155/2021/9218137.

[27] G. Bathla, K. Bhadane, R. K. Singh, *et al.*, "Autonomous vehicles and intelligent automation: Applications, challenges, and opportunities," en, *Mob. Inf. Syst.*, vol. 2022, pp. 1–36, Jun. 2022.

[28] S. Doll, *The top five best-equipped countries to support autonomous vehicles – who's leading the self-driving revolution?* en, https://electrek.co/2022/03/04/the-top-five-best-equipped-countries-to-support-autonomous-vehicles-whos-leading-the-self-driving-revolution/, Accessed: 2023-5-24, Mar. 2022.

[29] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of Yolo algorithm developments," *Procedia computer science*, vol. 199, pp. 1066–1073, Jan. 2022. DOI: 10.1016/j.procs.2022.01.135. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050922001363.

[30] K. Kosakowska, "Evaluation of the impact of speed bumps on the safety of residents - selected aspects," en, *Transp. Res. Procedia*, vol. 60, pp. 418–423, 2022.

[31] T. Sharma, B. Debaque, N. Duclos, A. Chehri, B. Kinder, and P. Fortier, "Deep learning-based object detection and scene perception under bad weather conditions," en, *Electronics (Basel)*, vol. 11, no. 4, p. 563, Feb. 2022.

[32] E. T. Auto, *EV charging revenue likely to exceed USD 300 billion globally by 2027*, en, https://auto.economictimes.indiatimes.com/news/industry/ev-charging-revenue-likely-to-exceed-300-billion-globally-by-2027/98118533, Accessed: 2023-5-24, Feb. 2023.

[33] X. Jia, Y. Tong, H. Qiao, M. Li, J. Tong, and B. Liang, "Fast and accurate object detector for autonomous driving based on improved YOLOv5," *Scientific Reports*, vol. 13, no. 1, Jun. 2023. DOI: 10.1038/s41598-023-36868-w. [Online]. Available: https://doi.org/10.1038/s41598-023-36868-w.

[34] B. Lutkevich, *Self-driving car (autonomous car or driverless car)*, en, https://www.techtarget.com/searchenterpriseai/definition/driverless-car, Accessed: 2023-5-24, Jan. 2023.

[35] M. A. Rouf, Q. Wu, X. Yu, Y. Iwahori, H. Wu, and A. Wang, "Real-time vehicle detection, tracking and counting system based on yolov7," *Embedded Selforganising Systems*, vol. 10, no. 7, pp. 4–8, 2023.

[36] D. Shah, *Vision transformer: What it is & how it works [2023 guide]*, en, https://www.v7labs.com/blog/vision-transformer-guide, Accessed: 2023-5-24, Apr. 2023.

[37] A. Singh and V. Bankiti, "Surround-view vision-based 3D detection for autonomous driving: A survey," Feb. 2023. arXiv: 2302.06650 [cs.CV].

[38] detection system, *Humps/bumps potholes detection dataset*, https://universe.roboflow.com/detection-system/humps-bumps-potholes-detection, Open Source Dataset, visited on 2024-05-30, May 2023. [Online]. Available: https://universe.roboflow.com/detection-system/humps-bumps-potholes-detection.

[39] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of YOLO architectures in computer vision: from YOLOV1 to YOLOV8 and YOLO-NAS," *Machine learning and knowledge extraction*, vol. 5, no. 4, pp. 1680–1716, Nov. 2023. DOI: 10.3390/make5040083. [Online]. Available: https://www.mdpi.com/2504-4990/5/4/83.

[40] C. Threewitt, J. DiPietro, and N. Parsons, "What does Tesla's Full-Self driving mode do?" *US News World Report*, Aug. 2023. [Online]. Available: https://cars.usnews.com/cars-trucks/advice/tesla-full-self-driving.

[41] Z. Luo and Y. Tian, "Infrared road object detection based on improved yolov8.," *IAENG International Journal of Computer Science*, vol. 51, no. 3, 2024.

[42] T. R. People, *Speed Bump Regulations - The Ramp People*. [Online]. Available: https://www.theramppeople.co.uk/speed-bump-regulations.