Towards Safer Roads: Vehicle Detection, Counting and Classification in Challenging Foggy Conditions

By

SAMINA

Reg No: 00000364610 (MS Systems Engineering)

Supervisor

Dr. Muhammad Tariq Saeed

(Associate Professor)

Department of Sciences

School of Interdisciplinary Engineering & Sciences (SINES)

National University of Sciences & Technology (NUST)

Islamabad, Pakistan

September 2023

"To my family"

Annex A to NUST Letter No. 0972/102/Exams/Thesis-Cert dated 23 Dec 16.

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis written by Mr/Ms Samina
Registration No. 364610 of SINES has been vetted by undersigned, found complete in all aspects as per NUST Statutes/Regulations, is free of plagiarism, errors, and mistakes and is accepted as partial fulfillment for award of MS/MPhil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature with stamp: _____
Name of Supervisor: Dr. M. Tariq Saeed
Date: _____

Signature of HoD with stamp: _____
Date: 23/09/2023

(DR. MUHAMMAD TARIQ SAEED)
Assistant Professor
Research Centre for Modeling & Simulation
NUST, Sector H-12, Islamabad

Countersign by

Signature (Dean/Principal): _____
Date: 25 SEP 2023

Dr. Hammad M. Cheema
Principal & Dean
SINES - NUST, Sector H-12
Islamabad

Declaration

I, **Samina** declare that this thesis titled **"Towards Safer Roads: Vehicle Detection, Counting and Classification in Challenging Foggy Conditions"** and the work presented in it are my own and has been generated by me as a result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a Master of Science degree at NUST
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at NUST or any other institution, this has been clearly stated
3. Where I have consulted the published work of others, this is always clearly attributed
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work
5. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself

SAMINA,

Reg No: 00000364610 MSSE Fall(2021)

Copyright Notice

Acknowledgments

First and foremost I would like to extend my heartfelt thanks to my supervisor Dr. Muhammad Tariq Saeed for his guidance and unwavering moral support throughout the completion of my thesis. I am truly grateful, for the time. Valuable advice was provided by my GEC members Prof. Dr. Ilyas Ahmed and Dr. Ishrat Jabeen.

I want to express my appreciation to my parents for their love, encouragement and unwavering belief in me. Your constant support, patience and selfless acts have always been a source of strength for me.

I am deeply grateful to all those who have supported me in ways during my journey of growth – be it through moral guidance or intellectual contributions. The combined impact of your influence has left a mark, on both my achievements and personal development.

Contents

List of Figures

List of Tables

Abstract

Monitoring highway traffic is an important task for any country since it involves the transportation of people and goods across geographical boundaries. Manual monitoring of vehicles travelling via highways was frequent in the past, but thanks to technological improvements, a number of studies and activities have been carried out to improve travel safety. Previous research has significantly contributed to our understanding of foggy conditions in traffic monitoring. For instance, the DAWN dataset provided insights with 1000 images of real-world fog, shedding light on the challenges of natural foggy scenes. Additionally, the work 'Semantic Understanding of Foggy Scenes with Purely Synthetic Data' created a synthetic dataset by replicating Zurich in a virtual environment and adding synthetic fog for realism. The BDDIW dataset, an extension of BDD100K, explored fog's impact on urban highways with images depicting various foggy scenarios. However, these works have limitations, including small dataset sizes, artificial scenes, and a limited representation of fog variations in public datasets. In this study, we assess the performance of the state-of-the-art YOLOv8 algorithm in monitoring highway traffic. Our focus encompasses vehicle detection, counting, and classification to comprehensively enhance traffic management. Additionally, we recognize the significance of addressing adverse weather conditions, particularly fog, which significantly increases the risk of road traffic accidents, especially during winter. To overcome this challenge, we created a dataset of 8,000 highway traffic images with varying levels of artificial fog: low, medium, and high. This dataset includes both the original, fog-free images and those with simulated fog. We used these foggy images to train the YOLOv8 algorithm, achieving an impressive mean average precision (mAP) score of 0.942. This result clearly demonstrates the algorithm's exceptional ability to

identify and categorize vehicles, even in challenging foggy conditions. In essence, this work evaluates YOLOv8's performance on foggy images, yielding promising results. By harnessing advanced computer vision algorithms and proactively addressing the challenges presented by adverse weather conditions, this study makes a substantial contribution to improving highway travel safety and efficiency through the creation of a comprehensive dataset. The insights and discoveries presented herein provide invaluable guidance for traffic management authorities and policymakers, equipping them with the tools to enact highly effective strategies for mitigating the risks associated with foggy weather conditions.

Chapter 1

Introduction

Tracking and monitoring of vehicles through the use of technology is widely used to make informed decisions regarding safety improvements of transportation systems[1]. The data collected by them can be used to identify areas of congestion, accidents and safety risks and also help transportation authorities to develop effective strategies for managing traffic flow and improving safety[2]. With time the number of vehicles has been observed to be increasing in the urban areas. As vehicles are increasing, more chances of road accidents are likely to happen. Authorities as Pakistan National Highways and Motorway Police need a workable system that monitors the flow as number and types of the vehicles passing daily on motorways[3].

Motorways are where daily several vehicles travel, these vehicles include trucks for carrying goods to other areas, buses for passengers and many more[4]. A traffic analyzing system is needed for controlling the traffic and flow. For analyzing the traffic, traffic count provides a crucial information to the transport engineering for traffic safety and congestion conditions.

Moreover, certain weather conditions such as fog and smog bring discomfort not only to the drivers but an important concern to the authorities to monitor the traffic as these conditions increase the probabilities of accidents to happen[5].

Without any doubt, AI is trending. There have been many successful projects for making cities smarter and safer. Surveillance cameras mounted on toll plazas and

other various areas of the motorways and highways or inside cities record daily traffic. The data for daily traffic gathered fed to a model to count the number and types of cars which will give an insight for the traffic control[6].

1.1 Road types based on different factors:

1.1.1 Based on materials:

There are roads based on materials which defines the quality of the roads upon their usage. Some examples are given below ranging from low quality to high quality. Such as gravel roads, earthen roads, murrum roads, kankar roads, WBM (Water Bound Macadam), bituminous roads, and concrete roads.

1.1.2 Based on speed and accessibility:

Road types constructed on the basis of speed and accessibility of the city are as follows:

- Highways
- Freeways or controlled-access highways
- Arterials
- Collector roads
- Local streets

Freeways or controlled-access highways: Freeways are also known as motorways that are not restricted by any kind of obstruction such as traffic lights, parking lots, intersections, rails and footpaths which makes the fast traffic undisturbed with a minimum of two lanes. Bifurcation for these roads is done on road intersections using slip roads or ramps (a sloping surface that joins two different levels at the entrance to make the speed slower for safer entry).

Highways: The importance of highways is significant for a country. It connects cities, states and villages to each other. Highways are less restricted than freeways which have food plazas, petrol stations and toll plazas. There is mostly heavy traffic on highways that include cars, trucks and buses. Highways have many entry and exit points with a minimum of two lanes

Arterials: These are also known as urban roads. The traffic is within the city in the arterials that possess signals, footpaths, pedestrian crossing etc

Local streets: These are smaller streets with slow and low traffic and pedestrians. There are no specific points for road crossing. There are no restrictions on pedestrians passing the roads. These roads have white brake lines.

Collector roads: The basic function of collector roads is gathering and distributing the traffic flow from local streets to arterial roads. Traffic speed is lower than arterial roads[7][8].

1.1.3 Based on traffic volume:

1. Roads with low traffic: 400 vehicles per day, villages and less populated areas have low traffic
2. Roads with medium traffic: 400-1000 vehicles per day
3. Roads with high traffic: >1000 vehicles a day

1.1.4 Based on types of traffic:

1. Cycle tracks
2. Tracks for Pedestrian
3. Motorways

1.2 Increased vehicles:

According to a study, the number of registered vehicles in 2019 when counted was around 6,209,626 and as of the previous year it was around 5,923,280. The study clearly says there is a significant increase in the number of vehicles which can be assumed that vehicles are increasing at a fast pace, with this increase traffic congestion, pollution and road traffic accidents are also increasing in the country [9].

1.3 The importance of motorways:

Motorways and National highways of Pakistan are multilane, high-speed control access highways (freeways). Motorways have a large impact on Pakistan's "National Trade Corridor Project" and "China-Pakistan Belt Road Initiative" from Kunjerab Pass near the Chinese border to Gawadar in Balochistan[3]

1.4 Pakistan's climatic regions

Pakistan's climate regions can be divided into four categories:

1. Coastal climate
2. Arid climate
3. Highland climate
4. Lowland climate

Coastal climate The climate in these areas is mild to hot summers and cold to few freezes. Places like Karachi, Makran coast, Indus Delta, Ran of Kutch and Thatta etc. have coastal climates.

Arid climate In simple words, let's say the climate of the desert with hot dry summers and hot winds blowing. Winters are cool with a minimum of 4 degrees in the month of January. These climates have less rainfall. The areas of Kharan, Thar, Thal and Cholistan have arid climates. Kharan lies in the south-east of Balochistan, Thar lies in the south-east of Sindh, Thal lies in the north-east of Punjab, and Cholistan lies in the south-east of Punjab.

Highland climate These are the climates of high-altitude areas such as north to western highlands. These areas have long, freezing winters. These areas have mild and short warm summers with heavy rainfall in the northern mountains than the western due to the northern areas having high altitudes.

Lowland climate These are areas with hot and extreme summers with less rainfall, monsoon rainfall occurs in the months of July and August. Winters here are mild and cool. The locations of this climate are the Indus Plain in Punjab and Sindh[10].

1.5 Adverse weather conditions in Pakistan and their effect on traffic:

Weather and climate changes have remarkable effects on traffic and especially countries with fewer resources are more affected. Adverse in Pakistan include heavy rainfall and floods that can also cause land sliding in the mountainous areas.

Floods may damage bridges and make roads impassable which may cause delays in travel or make driving in those areas hazardous. Not only the road travel but these weather conditions can affect flights, even cancellations in the flights.

These were the hazardous weather conditions apart from these there are some conditions in the weather that occur in a known time at some places and make driving risky such as foggy weather which has caused many delays in travel and accidents

The effect of adverse weather conditions on road traffic accidents (RTAs)

The effects of adverse weather conditions on RTAs in Vehari, Punjab, Pakistan, both in rural and urban regions. According to the research, the occurrence of RTAs is directly correlated with rainfall, extremely cold temperatures, fog, and heat. Fog accounts for the highest percentage of accidents caused by weather-related factors, at 34 per cent, followed by rainfall at 25 per cent, temperature at 21 per cent, and other meteorological conditions at 20 per cent.

The study also shows that the driver's age significantly affects RTAs, with drivers between the ages of 40 and 60 being the most accident-prone. Additionally, accidents are more likely to occur in smaller vehicles, which is inversely related to vehicle power. With 42 per cent of incidents, motorcycles are the most often involved vehicle type in reported RTAs.

The results highlight the value of exercising caution while driving in bad weather, especially when it comes to young drivers and smaller vehicles. The research proposes that while creating laws for traffic and awareness campaigns for traffic, particularly in developing countries like Pakistan, consideration be given to elements including social ethics, traffic awareness education, vehicle size, driver maturity, conditions for roads, and environmental implications. RTA hazards can be reduced and road safety can be improved in both rural and urban settings by addressing these variables[11].

1.6 Problem statement

Travelling from one place to another has an impact on networking and the economic growth of a country, it can be within a country or outside but making sure travelling is safe is an important topic to think about. Pakistan with diverse and adverse weather conditions have caused many accidents in the past. One of the adverse weather conditions that came to my interest was foggy weather which delays the traffic or causes accidents. Although researchers have worked and addressed this issue not enough attention is given to this area in computer vision.

1.7 Objectives

The following objectives are included in this work:

- Perform vehicle detection, count and classification using the latest 'State of the art' object detection technique
- Generate a partial synthetic dataset of 8000 images with 4 fog qualities
- Train the SOTA object detection on the custom-prepared data

1.8 Thesis Organization

The remainder of the chapter is organized as follows: in Chapter 2, the literature review on vehicle object detection and foggy weather object detection is presented. The proposed methodology of this project is discussed in Chapter 3 with short code snippets and pre-requisites to perform the technique. In Chapter 4, what results we observe and what improvements can be produced on the basis of the results that are produced by the model are discussed, and in the end, chapter 5 is the conclusion and what can be done to make this work better.

Chapter 2

Previous Work

2.1 History and evolution of object detection

The advancement of machine vision is undoubtedly remarkable, if we talk in the context of object detection, it was just a few years back when object detection was considered one of the most challenging tasks but in few years later the task has produced tremendous results. Let's see the journey of object detection that evolved all the way from traditional to deep learning techniques[12], figure 2.1 is a tree description of the types of object detection given below:

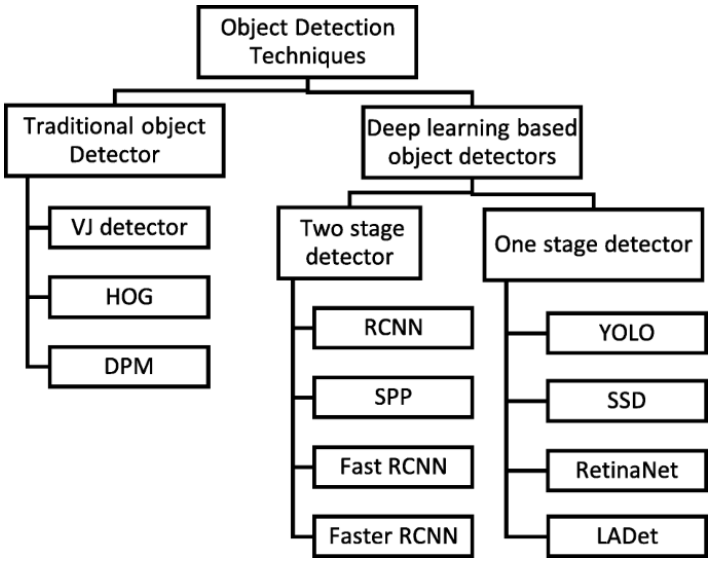


Figure 2.1: Object detection techniques

2.1.1 Traditional object detection methods

The traditional techniques were based on handcrafted methods which means that in the early days, researchers and scientists used to manually design the features of some objects from their knowledge and perspectives.

Viola-Jonas algorithm Paul Viola and Michael Jonas developed this framework back in 2001, based on Haar wavelet that detected human faces in real-time. This technique used window sliding where the sliding window goes to all possible scales and locations to find a pattern in an image that contained a face feature. Moreover, Integral image calculation and Adaboost algorithm made this technique more efficient. Viola-Jonas algorithm was the first technique that was capable of fast face detection and played a significant role in advancing the machine vision field[13].

Histogram of Oriented Gradients (HOG) algorithm HOG was presented back in 2005 by N. Dalal and B. Triggs. This technique was an advancement to the previously available techniques of 'Scale Invariant Feature Transform and Shape context'. The effectiveness of HOG is known in pedestrian detection. Similar to window sliding HOG works on dense pixel grids where gradients are computed on intensity changes in the block. Both magnitude and direction of changes in the pixel intensity are captured by the gradients. For objects having varying sizes, HOG re-scales the image several times keeping the detection window at a consistent size thus the objects of different sizes/scales are detected[14].

2.1.2 Object Detection in Deep Learning

In deep learning, the method is based on data instead of some expert's experience. From 2010 to 2012 due to the saturated performance of handcrafted object detection, the field got discouragement, in 2012 the rebirth of 'convolutional neural network' began. Convolutional neural networks and deep convolutional networks were successful at detecting robust and high-level object features. Today object detection in deep

learning is categorized into two genres; 'single-stage object detection' and 'two-stage object detection '[15]. The figure **??** below tells further types of the two types of object detection.

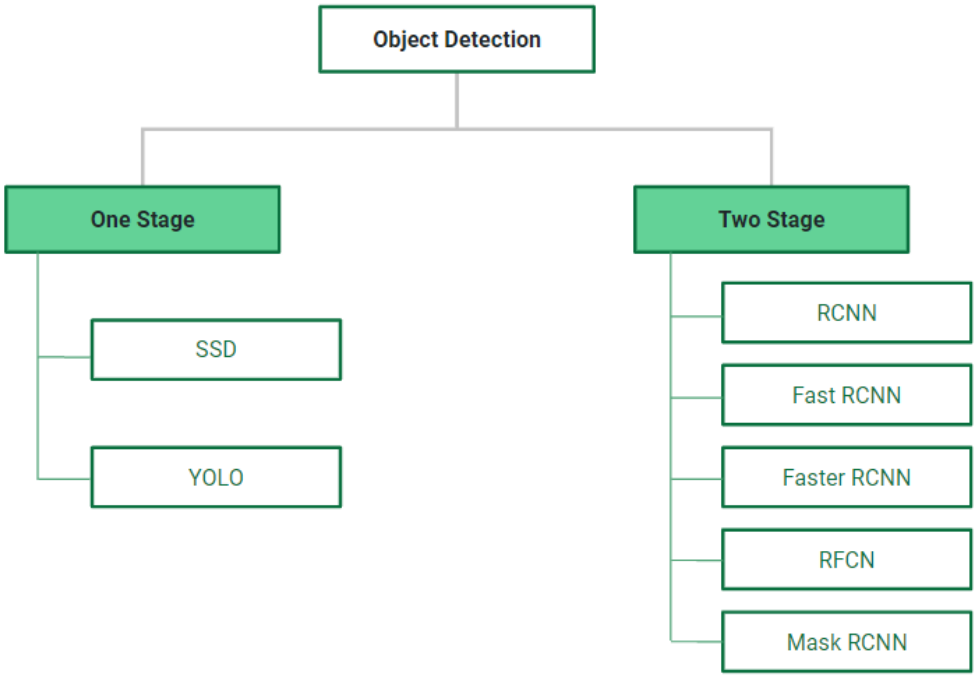


Figure 2.2: Two Types of Object Detection

Two-stage object detection: The potential of object detection was observed with the proposal of regions with CNN features where the process involves first generating regions of interest and secondly refining the bounding boxes:

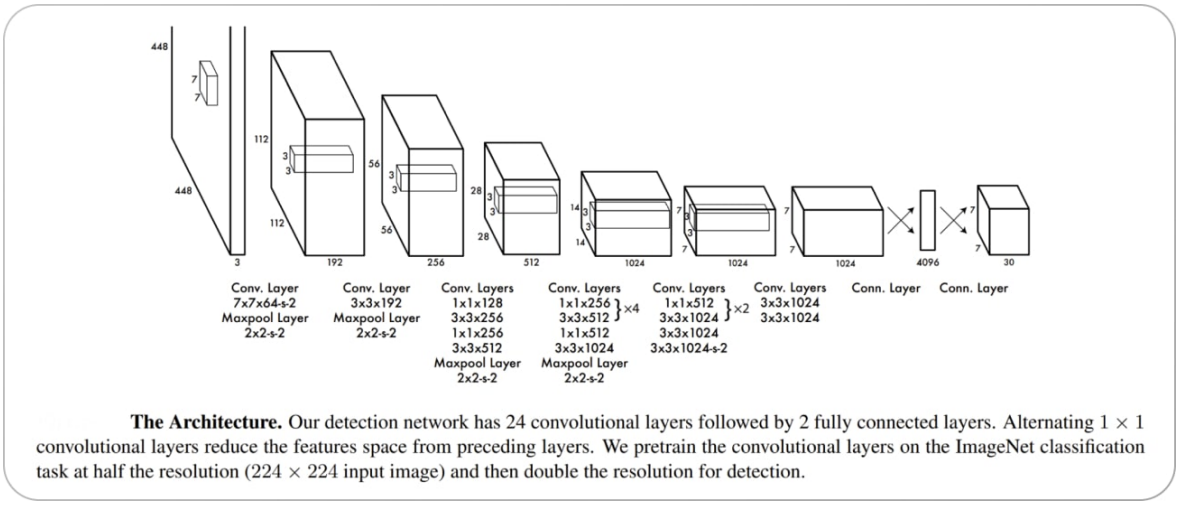
R-CNN, fast R-CNN and faster R-CNN algorithms: At the earliest RCNN which is known as a region-based Convolutional Neural Network, detects objects where it classifies an individual region proposal using a CNN, since this was one of the early approaches, it was costly too. Later, Fast RCNN was introduced that shared CNN computations for region proposals, which resulted in a faster process and more efficient object detection. It uses ROI (Region of interest) pooling to extract feature vectors of fixed sized and then performs both classification and generate bounding box regression tasks[16]. After this Faster RCNN was introduced that proposed RPN (Region Proposal Network.

one stage object detection: A single pass of the input image detects about the presence and localisation of objects in the image. To process an image this uses a fully convolutional neural network (CNN). However, this technique is less effective in detecting small objects which means this is comparatively fast but less accurate[17].

2.1.3 You Only Look Once (YOLO) algorithm

Joseph Redmon et al. Introduced a method, for object detection known as one stage or single stage detection back in 2015. YOLO, which stands for "You Look " has proven to be highly accurate and fast when it comes to real time object detection. Unlike two stage methods like RCNN YOLO processes the image in a pass using a deep convolutional neural network (CNN). It then predicts bounding boxes and class probabilities on a grid with each grid cell responsible for detecting objects. Due, to its simplicity and effectiveness YOLO is widely utilized in fields including robotics, autonomous vehicles, surveillance systems and more. Since its inception YOLO has undergone updates and improvements. .[17][18].

YOLO working It takes an image and applies deep convolutional neural network for object detection. CNN model works in the backbone of yolo.



The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Figure 2.3: YOLO backbone architecture

In the figure 2.3, There are a total of 24 convolutional layers, as can be seen. By adding

a temporary average pooling and fully linked layer, the first 20 convolution layers are pre-trained using ImageNet. Both class probabilities and bounding box coordinates are predicted by the last fully linked layer of the YOLO architecture. YOLO divides an input image into a SxS grid. An object is detected by a grid cell if its centre lies within that grid cell. B bounding boxes and their confidence scores are anticipated in each grid cell. These confidence scores represent the model's level of certainty that the box contains an object and its estimation of the model's accuracy.[19].

YOLO algorithm predicts a class of an object and its bounding box that will define the location of the object in the image. It recognizes each bounding box using four attributes:

- Center of the bounding box
- Width of the box
- Height of the box
- In addition to that, the class probability of the prediction[20].

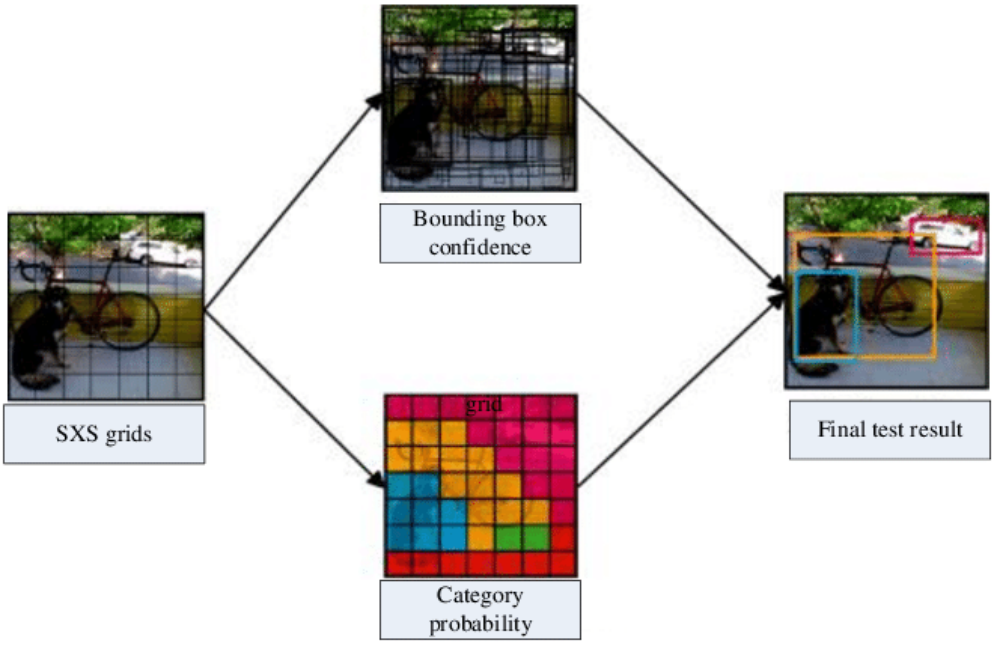


Figure 2.4: YOLO Algorithm structure

2.2 Vehicle counting and classification

In the last few decades, there has been tremendous work done for vehicle counting and classification. This work had started with the traffic management system where in the urban area [21]. Some of the recent works are mentioned below;

In year 2022,[22] Samprit and others introduced a framework on YOLOv3 for vision-based vehicle classification and counting. In the work masking detection and counting and classification of different event classes of vehicles have been used and the accuracy during day time is 93.65 and 87.68 during night time.

In [23] the work deep learning based techniques, i.e., YOLO and deep sort have been used to count and detect vehicles where vehicle counting is used to estimate the density of traffic other than this earlier techniques were focused towards speed detection, signal jumping and zebra crossing. In the work there was two steps followed; one the use of YOLOv3 for detection, counting and classification of vehicles and secondly the use of deep sort to track vehicles in video frames.

Faster RCNN has been used in real time vehicle type classification in the work [24] where images were taken at crossroads with the 2 classes of vehicles and use of GPU system.

Optimized YOLOv2 model has been used in the work [25] where for intelligent transportation system vehicles have been counted and classified, with the use of a GPU for CNN computation.

The innovative vehicle detection and counting system based on vision for highway surveillance is suggested in this research [26]. For the purpose of deep learning-based vehicle detection, a high-resolution highway vehicle dataset with approximately 57,000 annotated occurrences in 11,129 photos is developed. In order to improve vehicle recognition, the suggested system first extracts the highway road surface from the image and splits it into a remote and proximal area using a recently proposed segmentation algorithm. The suggested segmentation method enhances detection accuracy in each of these categories, especially for small vehicle objects. Overall, the management and

control of highways can benefit from this study in practice.

In another work Fuzzy neural network and YOLO have been used to monitor the traffic where vehicles are detected, counted and classified [1].

A simplified flow monitoring of vehicles was introduced by Shakir Ahmed in the year 2022 [27] where computer vision techniques were used and implemented on MATLAB with an accuracy of 90. Here they have used a camera for the collected video data. The steps in the techniques were that some portion of the video was reserved for training in which it obtained binary foreground from the initial frame and removed noise that counted the vehicles. Moreover, it has been mentioned that the technique can applied in various environments for the counting of vehicles to maintain a better traffic system.

Work on the construction vehicle detection was performed by Saeed Arabi and others [28] where they used available data by web scrapping and on SSD MobileNet object detection was performed. Three embedded devices along with a GPU was used for the work.

With the use of the KITTI dataset, a comparative analysis of contemporary deep-learning models for vehicle detection has been carried out. The comparisons of deep learning methods included Faster RCNN, R-FCN, SSD, RetinaNet, and YOLOv3. Although single-stage algorithms were faster than two-stage ones in terms of speed, it was found that the accuracy of the two-stage algorithms was higher. [29].

A traffic video analytics was introduced [30] where detection and classification of vehicles were performed with the mixture of Gaussian and SVM vehicle classification on the data of Indonesian toll plaza and MIT traffic.

2.2.1 Detection of vehicles in foggy weather

Machine vision has been successful in detecting objects in clear weather not only the detection but the classification and segmentation with the above-listed methods but the problem of detecting objects in unclear weather has got less attention. The need for solving this problem has the potential in self-driving cars or surveillance cameras

that are mounted outdoors which mostly fail to perform in foggy or dusty weather where serious problems can be caused[31][32].

As discussed earlier, in machine learning or deep learning the problem is dependent on the data. good data will provide good results and bad data will provide bad results.

2.2.2 Dataset used in vehicle detection in foggy weather

Kaggle contains a number of datasets that could be used for different machine learning problems, some famous datasets that have been used in detecting objects such as vehicles are the COCO dataset, KITTI dataset, BDD100K, CityScaper and more but these datasets have a focus on clear images and to develop a model for unclear scenes, developers cannot depend on these datasets. there are also some datasets that have initiated to highlight the need for unclear weather conditions some of the significant ones are as below:

DAWN: This dataset consists of 1000 images showing weather conditions that can cause visibility, for self driving cars on the road and around other objects. The goal of this project was to enhance the efficiency of self driving technology by enabling cars to navigate in challenging weather conditions. The images in this dataset were gathered from the internet using search engines like Google and Bing. The dataset includes images of areas, highways and interstates during weather conditions such as sandstorms, snowfall, rain and fog. To assist researchers in developing solutions for object detection under weather conditions, for self driving cars the dataset has been annotated using Labelme. [33].

Foggy Synscapes: In this work [34] a purely synthetic dataset named as Foggy Synscapes is generated which addresses attention towards understanding semantic scenes in adverse weather such as foggy weather that can be applied in self-driving cars. In the work, the dataset comprises 25,000 unique images of outdoor scenes in high quality generated purely synthetically with the help of computer graphics. The contribution

of this study is 1. To generate a dataset of synthetic images 2. To determine whether fully synthetic data performs better than partially synthetic data in grasping semantics of hazy settings. The result shows that neither pure synthetic nor partial synthetic outperform the other when training and fine-tuning the neural network but the combination of both types of dataset produces better result.

Chapter 3

Proposed Methodology

3.1 Background

Numerous methods have been applied to monitor traffic in an effort to make it safer and easier to manage. These techniques cover a variety of techniques, such as:

Sensor-Based Detection: In this technique, passing vehicles are detected by sensors like radar, or infrared sensors that are mounted on the highways. These sensors provide real-time traffic information, and are used frequently in toll plazas [35].

Computer Vision Techniques: In order to recognize and count automobiles in video footage, computer vision techniques such as background subtraction are used. Even in difficult settings, reliable vehicle counting can be accomplished by subtracting the stationary background from the moving objects [36][37] , CNNs and other deep learning models have proven to be remarkably effective at identifying and categorizing automobiles from pictures and video sources. These versions are capable of counting vehicles, classifying them by type, and even estimating their speeds [6].

In this work, we have applied the SOTA technique YOLOv8 on a custom dataset, the workflow of the methodology is explained as below in figure 3.1:

3.2 Flow Chart

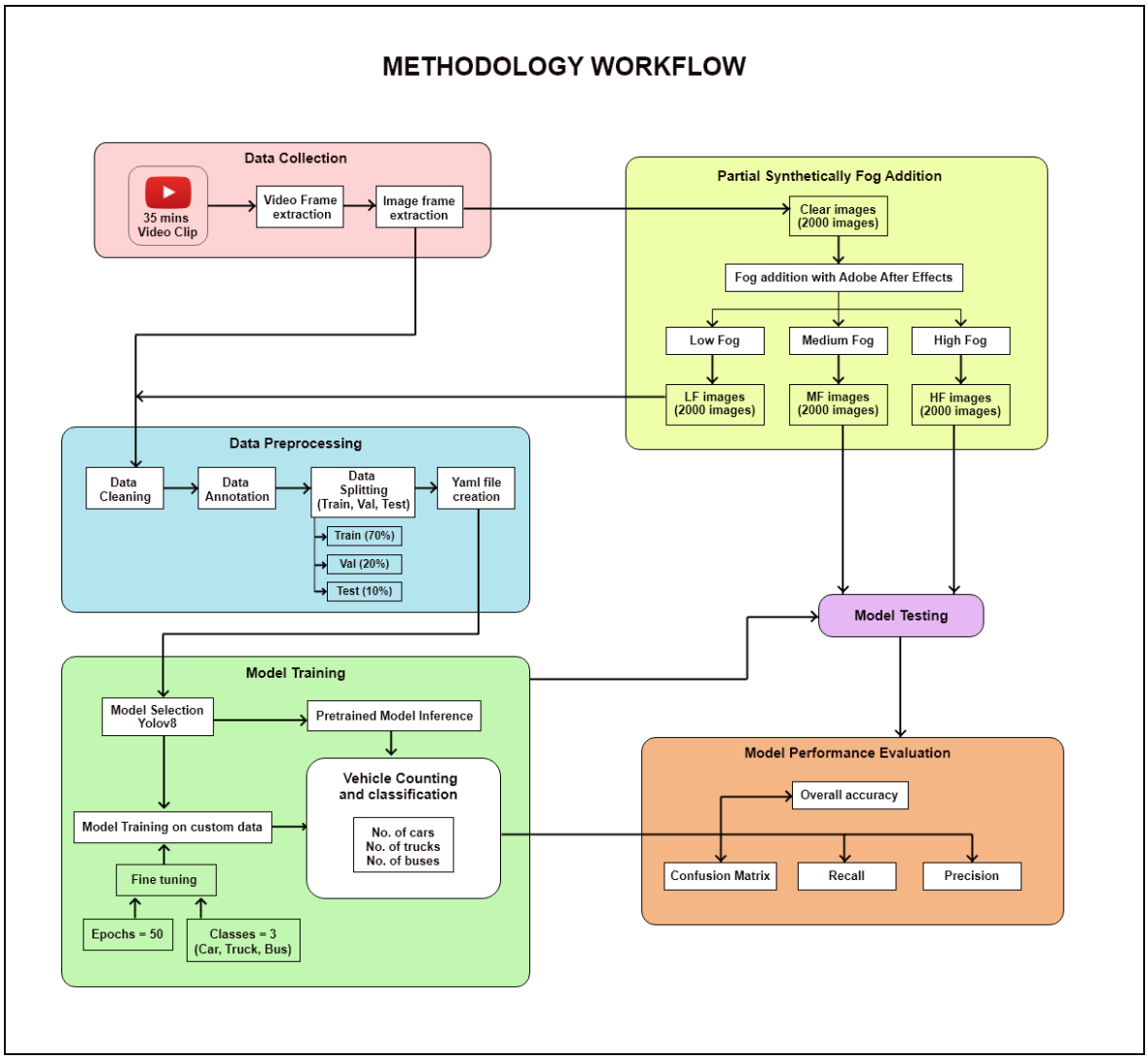


Figure 3.1: Proposed Methodology Flowchart

3.3 Data collection

A number of datasets were considered for this project such as the COCO dataset, cityscapes, BDD100K, car object detection etc. but these datasets were not according to our needs as they were mostly either too general and large such as COCO or mostly focused on self-driving cars while we needed a dataset that was captured by a surveillance camera and a footage of vehicles passing some toll plaza.

In the end, the raw data was downloaded from open source, i.e., YouTube, which was a 35-minute footage of a highway with vehicles passing in two directions. The vehicles were mostly cars, trucks and buses in daytime footage. We applied Yolov8 inference on this footage which gave good vehicle detection results. A frame extraction was applied to the 35 minutes video converting it into 82 short videos of 25 seconds in MP4 format. Later, the first 80 videos were selected and frames of 1 second were extracted in JPEG form. this made a total of 2000 images.

Partial synthetic: For this purpose of studying fog, we could get fogged images from the internet that were captured during the time of the foggy season but the aim of this research work was to compare the clear image and the fogged image of the same place and time which was not possible with the real human captured image so we decided to synthetically produce fog on the images.

3.4 Data preprocessing

Before taking data for training, data preprocessing is required for removing any kind of duplication in data, data annotation, splitting it into training and test sets and other necessary file creation or augmentation if needed. The steps for data preprocessing are listed below:

3.4.1 Data Cleaning

For this project, there was some duplication in the images that needed to be analyzed and removed. For the fog images, the sequence of the four sets was necessary. For making the images corresponding to all sets, we made sure any of the images in any set were duplicated or missed while applying the fog filter.

3.4.2 Data Annotation

Labeling data is the process of data annotation. We choose the 'Roboflow' technology to annotate our dataset. We designated three types for annotation: vehicles, trucks, and buses.

3.4.3 Data Splitting

A total of 8000 total images were prepared. 2000 of the clear/original images were taken as the ground truth images. Other 6000 images were copies of the original images with low fog filters applied, medium fog or high fog.

Photos, with low fog were utilized during the training process. The selection of images was done randomly with 70 per cent chosen for training 20 per cent, for validation and the remaining 10 per cent used for testing purposes.

3.4.4 Configuration File creation

After labelling the image dataset, there are two types of files created, the images and a file for the bounding boxes of the objects that were annotated. Above this another file that is created for all the folders is a .yaml file which will save the class names and the address of the bounding boxes and this file will be used during training the model to transfer the bounding boxes to the training data.

3.5 YOLOv8 explanation:

Yolov8 is developed by Ultralytics which had developed the previous version yolov5 model too. Yolov8 was developed for object detection, instance segmentation, and image classification.

The growth of yolov8 Yolo was introduced in 2015 by Joseph Redmond. The framework for yolo v1-v4 is based on Darknet which was maintained by the computer

vision community in C language. Later, Yolov5 was launched by Glenn Jocher at Ultralytics which became the latest state-of-the-art in Python which was based on pytorch framework. With the strong model developed, yolov5v maintainers are committed to regular improvements and issues fixation and share with the community. Later many more models were branched of yolov5 that are YOLOR, Scaled-yolov4, and yolov7. Other models were introduced based on the Pytorch framework such as YOLOX, and YOLOv6. With every one of the models, a new state-of-the-art has been introduced that is focused on better accuracy and efficiency. Yolov8 was launched by Ultralytics on 10th January 2023 with the latest SOTA with improved accuracy.

Advantages of YOLOv8 The model is trained on the COCO dataset which is a highly large versatile dataset also a high rate accuracy has been measured on COCO and RF100.

3.5.1 YOLOv8 tasks

YOLOv8 has the following tasks to perform:

- Detect
- Classify
- Segment
- Track
- Pose

Figure 3.2: YOLOv8 tasks

Let's know more about these tasks,

Detect The main duty that YOLOv8 performs is detection. It locates an object in an image or video and creates a bounding box around it. Different classes are assigned to the observed objects.

Segment Segmentation is another task performed by this model. The model uses U-Net architecture to perform the segmentation of a region in an image.

Classify In this task EfficientNet architecture is used to classify the objects of different categories.

Pose Pose estimation or movement tracking of an object in a video or image is a task in YOLOv8 that takes specific points. These keypoints are specific points in the object that consider a pose[38].

3.5.2 YOLOv8 modes

YOLOv8 has some modes for performing a task such as 'predict' or 'train'. The followings are the modes of this model:

- Predict
- Train
- Validate
- Export
- Track
- Benchmark

Predict This mode as used to detect an object such as while performing inference of pretrained YOLOv8 model for some image or video.

The code for this task is as: For image yolo task=detect mode=predict model=yolov8n.pt source="img.png"

For video yolo task=detect mode=predict model=yolov8n.pt source="vid.mp4"

Train This mode as recognized by the name trains a dataset using the specified hyperparameters. The dataset is trained to accurately predict the category and the location of the objects.

Validate After training the performance of model is evaluated by validating it on a specified set of data.

Export This mode is utilized to prepare the trained model, for deployment. Additionally the model is converted into a format, with software applications.

Track The track mode tracks objects in real time in an input video. This mode is useful in self-driving cars and surveillance cameras that need tracking

Benchmark The benchmark mode will provide some valuable information about the exported model. Such as it will profile the accuracy and speed, mAP of between 50-95 for the tasks; Object detection, pose estimation and segmentation and accuracy metrics for classification. Not only this it also displays the time in milliseconds for inference. Thus these information will give an insight for the user to choose an exporting format according to their needs[38].

3.6 Model training

Model training on custom dataset requires steps that are listed below:

1. Set an environment
2. Modules installation
3. Check the pretrained object detection result
4. Train the model on the custom dataset

3.6.1 Set an environment

Colaboratory or Jupyter Notebook, any environment that one is comfortable with can start working with. For this project we have used Google Colaboratory due to the cloud environment and free GPU service.

3.6.2 Modules installation

Before starting working on a project that uses YOLOv8, we at first place install the module that is

```
pip install ultralytics
from ultralytics install YOLO
```

This module will install the required package for the object detection.

3.6.3 Check the pretrained object detection result

Before beginning to train a dataset on the model, it will be a good practice to check the performance on the model beforehand on a single image. For this purpose we apply the detect task.

There is single line of code that is applied to check the inference:

!yolo predict model=yolov8m.pt source="/content/drive/MyDrive/thesis-final/dataset/test/medium-fog/mf-image(530).jpg"

3.6.4 Train the mode on a custom dataset

Now we will continue to training after we have checked that the YOLO modules are installed and it works on single image.

The code for training is as simple as inference, i.e.,

!yolo task=detect mode=train model=yolov8m.pt data=data.yaml epochs=50 imgsz=640 plots=True

- Task is given to detect as we are detecting objects here,
- Mode is to train the data
- Model type is yolov8m.pt
- Data.yaml is the format of the annotated data that has the information for the actual data
- Epochs or iterations is set to 50, this could be increased to 100 but 50 showed satisfactory results.

```
50 epochs completed in 0.094 hours.
Optimizer stripped from runs/detect/train5/weights/last.pt, 52.0MB
Optimizer stripped from runs/detect/train5/weights/best.pt, 52.0MB
Validating runs/detect/train5/weights/best.pt...
Ultralytics YOLOv8.0.131 Python-3.10.12 torch-2.0.1+cu118 CUDA:0
(Tesla T4, 15102MiB)
Model summary (fused): 218 layers, 25841497 parameters, 0 gradients
```

Class	Images	Instances	Box(P	R	mAP50	mAP50-95
all	20	384	0.854	0.873	0.924	0.739
bus	20	1	0.785	1	0.995	0.995
cars	20	352	0.949	0.841	0.963	0.643
trucks	20	31	0.828	0.779	0.815	0.579

Table 3.1: Training result of YOLOv8 on custom data

Speed: 0.2ms preprocess, 12.9ms inference, 0.0ms loss, 1.1ms postprocess per image
Results saved to runs/detect/train5

3.7 Performance Evaluation

3.7.1 Confusion Matrix

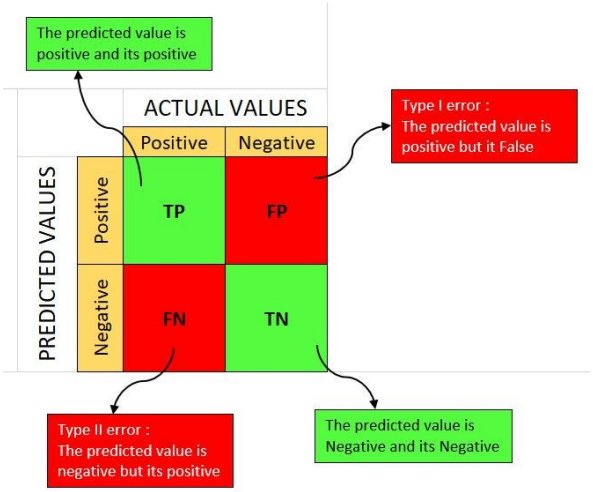


Figure 3.3: Confusion matrix in general form *(Image by Anuganti Suresh - published in Analytics Vidya)*

Figure 3.3 A confusion matrix is a performance assessment that is used in statistics and machine learning, especially in classification tasks. It is a table that lets us compare the actual and predicted classes for a piece of data to see how well a classification model performed.

The rows and columns of the confusion matrix stand in for the actual classes and the predicted classes, respectively. It breaks down the following four significant metrics:

True Positives (TP) refer to the instances where the model accurately predicts that they belong to a class, a positive class.

On the hand False Positives (FP) occur when it is mistakenly predicted that they

belong to a class despite actually belonging to a different class, namely the negative class.

True Negatives (TN) represent the proportion of examples that fall into the category and are correctly identified as, by the model.

Lastly False Negatives (FN) indicate instances that are categorized as positive but are misclassified by the model as falling into the category.

Chapter 4

Results and Discussion

The results will show the number of objects, such, as cars and trucks detected in four image qualities; low fog medium fog and high fog. These can be seen in Figures 4.1 4.2, 4.3 and 4.4 respectively.

To assess the algorithms performance in weather conditions on highway traffic we conducted the task using a total of 400 images. With 100 images, for each quality level.

Figure 4.1: Original image

Figure 4.2: Low fog image

Figure 4.3: Medium fog image

Figure 4.4: High fog image

4.1 Inference of the pretrained YOLOv8 on the dataset

Before proceeding with any step, we first applied inference of YOLOv8 on a single image, and the result was as shown in Figure 4.5:

Figure 4.5: Single image inference result

The result gives the information: 480x640 13 cars, 2 trucks, 1331.8ms Speed: 6.8ms preprocess, 1331.8ms inference, 33.9ms post process per image at shape (1, 3, 640, 640)

Let's study the output in detail;

Information about the image:

- The picture is 480x640 pixels in size.
- Results of object detection: The YOLO model found 13 automobiles and 2 trucks in the image.
- Inference Time: The analysis of this image took a total of 1331.8 milliseconds for inference.
- Time Breakdown for Inference: Preprocess Time: The image was preprocessed for 6.8 milliseconds before being sent into the model. The actual inference time for the model, which is when the detections are made, was 1331.8 milliseconds.
- Postprocess Time: After inference, it took 33.9 milliseconds to filter, threshold, and refine the discovered objects as part of post-processing.

- Shape of the Image's Input: The image's input shape when fed into the YOLO model was (1, 3, 640, 640). This shows that the image should have three colour channels (RGB), be in a batch of just one, and be scaled to 640x640 pixels.

4.2 Inference of pretrained YOLOv8 on the prepared data

Similar to the previous practice, the inference was applied to a set of 100 datasets. Out of the 8000 corresponding images, 400 were chosen for first-phase inference. The images were image 1 to image 100 of those 4 sets; 1-100 images of clear images, 1-100 low fog images, 1-100 medium fog images and 1-100 high fog images. After inference, the following results were observed. Figure 4.6 is the visual representation of the results generated.

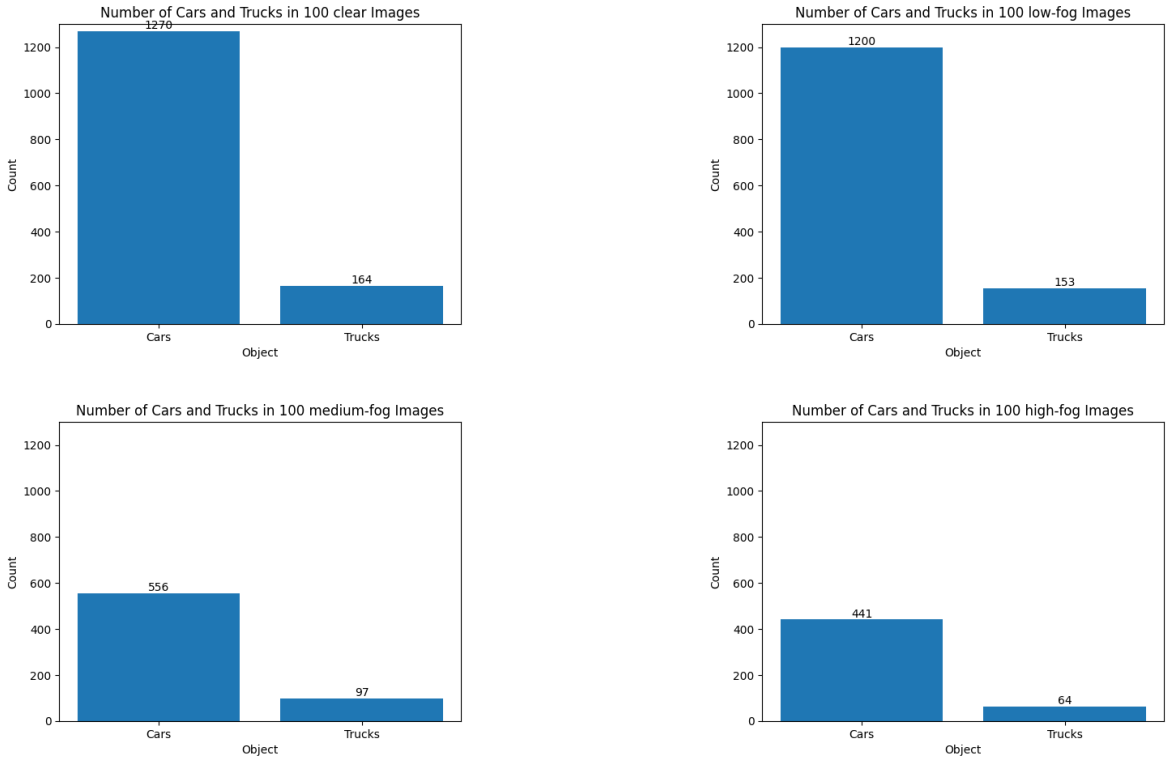


Figure 4.6: Vehicles detected in image set 1-100

After observing this we decided to try another inference on a different set of images. This time we chose images from 501 to 600. And the result of objects detected there is as below in Figure 4.7:

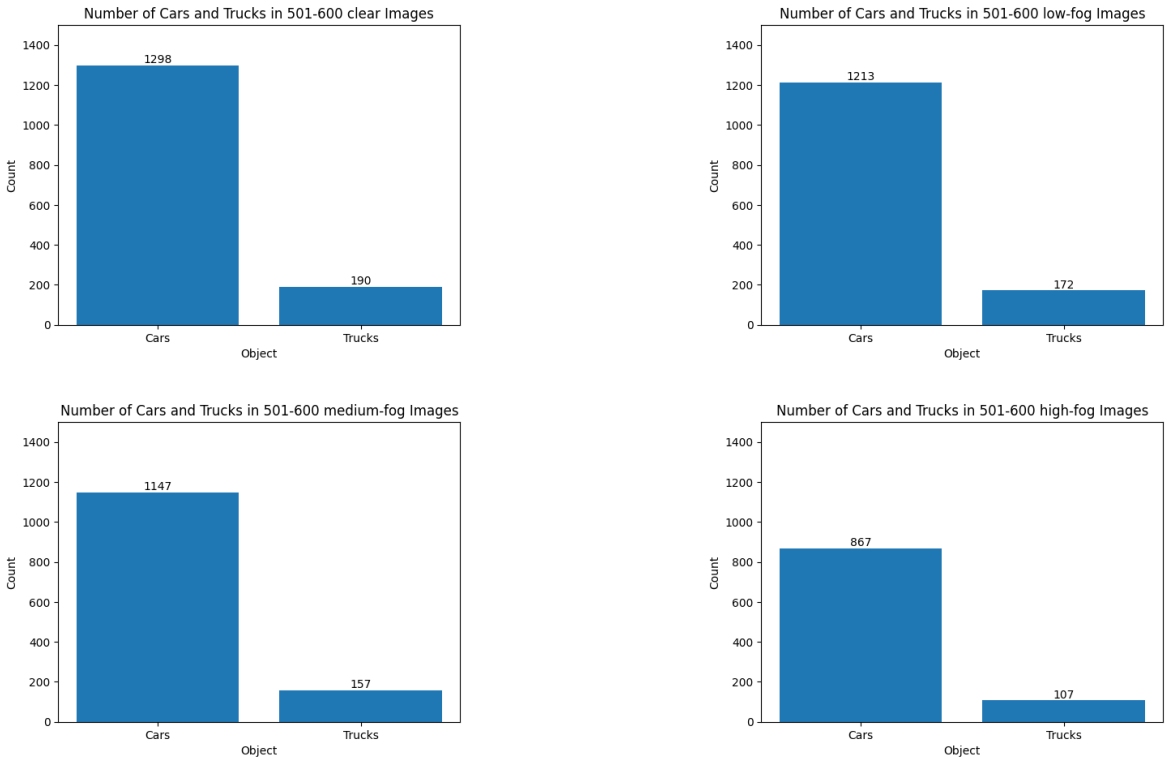


Figure 4.7: Vehicles detected in image set 501-600

Diving into the details of the above results;

Just as above, we studied the result details of the single image. Similarly applying a loop and generating a result of the whole set of selected images can be observed as below;

image 1/1 /content/drive/MyDrive/highway-traffic-folder/501-600 images/clear/image(501).jpg: 480x640 14 cars, 2 trucks, 3201.1ms Speed: 13.3ms preprocess, 3201.1ms inference, 2.3ms post-process per image at shape (1, 3, 640, 640)

Explanation The lines provide us information about the folder from where the images have been imported/read, **.jpg** is the image format, **480x640** is the dimension of the images, there have been objects of classes; cars, trucks and buses detected, after

the number of objects detected a time is given in ms which is the total time taken for inference of particular images have been produced such as **1283.4ms** is for the last image. Then the time 2.2ms was taken for preprocessing and 1.7ms for postprocessing. Later in the end the shape says there has been 1 image inferenced at the particular period of time with 3 color channels and a resolution of 640x640

4.3 Model trained and tested on the annotated data

Confusion matrix A crucial tool in machine learning for assessing the effectiveness of categorization models is the confusion matrix. To calculate metrics like accuracy and precision, it groups predictions into groups like true positives/negatives and false positives/negatives. This matrix directs decisions on model deployment and refinement.

4.3.1 Results on 50 epochs

For training the YOLOv8 model on our dataset, we chose a set of 100 low fog images, annotated on Roboflow and then trained the model on 10 epochs first but the result was unsatisfactory so we tried to train it on 50 epochs which provided better results, we also trained the model on larger epochs which we will discuss later in this document.

After training some results were generated which we will discuss below;

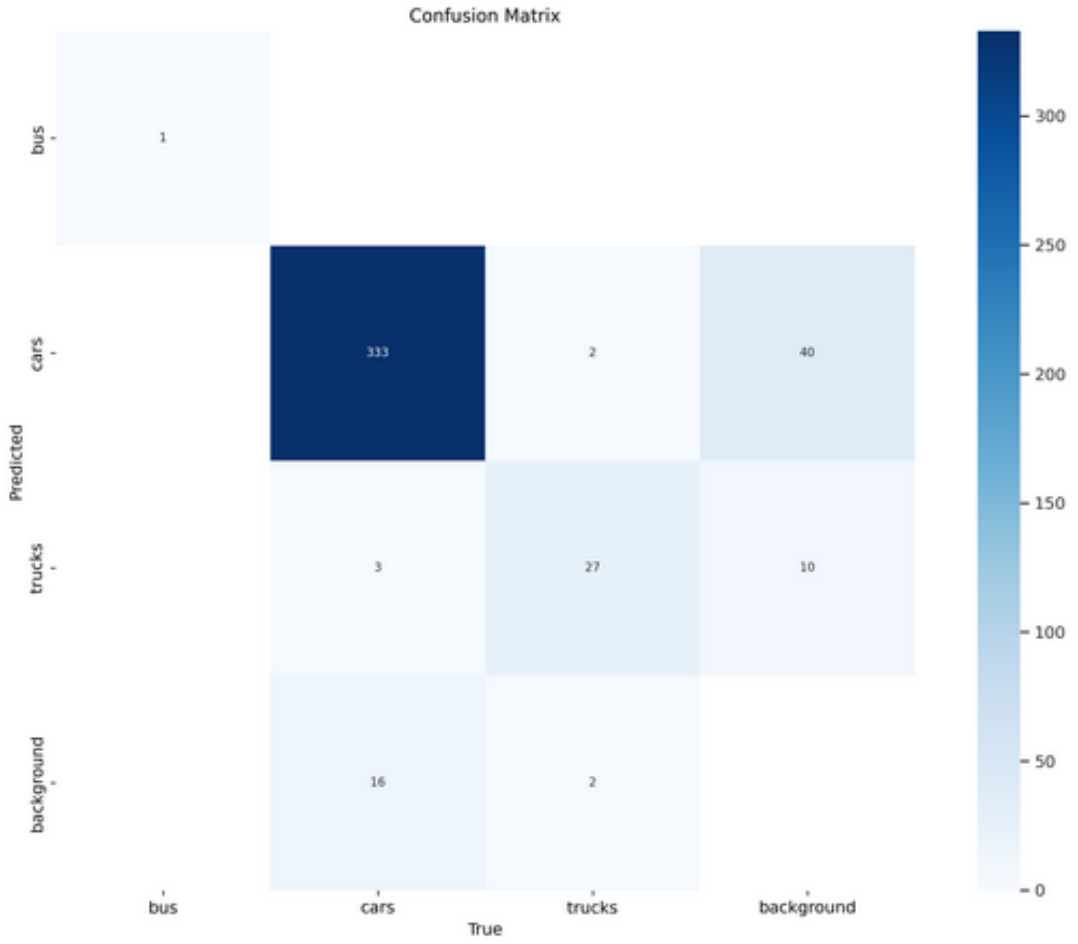


Figure 4.8: Confusion Matrix of Custom data trained on 50 epochs

Explanation of the confusion matrix We see this in the confusion matrix of the trained model. Cars have been identified as cars 333 times, trucks 27 times and buses 1 time while the background has also been identified as cars 40 times, this is because the image is blurry and secondly there are very tiny and far away objects that are confused to be vehicles of background.

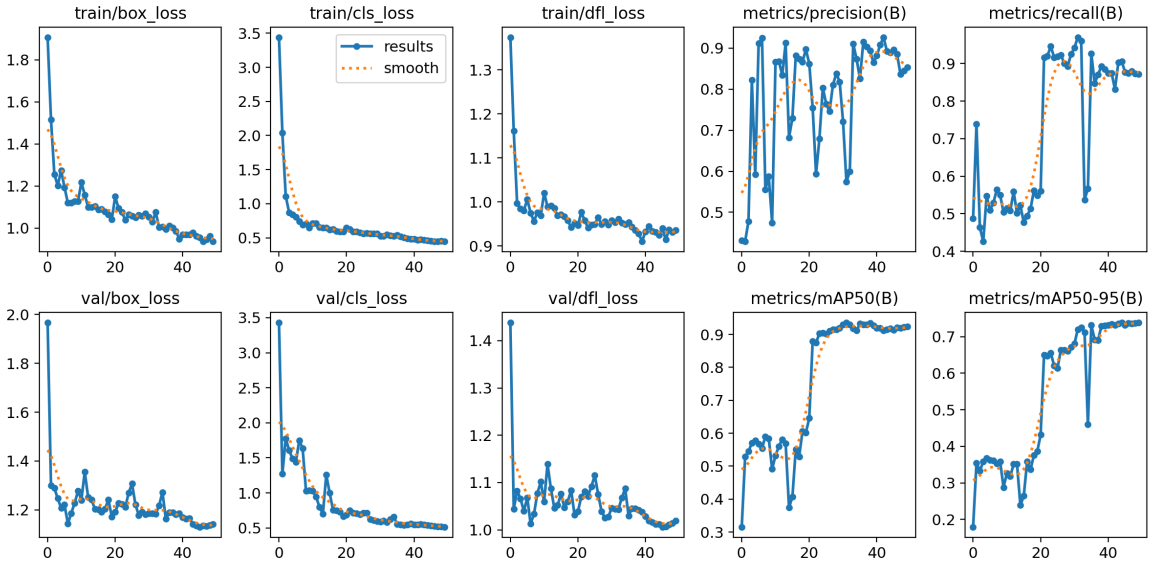


Figure 4.9: Model Performance Metrics of Custom Data on 50 epochs

Model performance metrics Let's study these metrics shown in the above figure **??**:

Train loss: Measures the effectiveness of the model's learning during training with the aim of reducing training loss.

In the case of our project, in the beginning, the loss was high but with more iterations, the loss has reduced.

mAP: Measuring overall object detection performance on unseen data is called mAP (mean Average Precision), a higher-going graph can be seen in the figure above.

Precision: Measures the accuracy of forecasts that turn out to be true positives, in our case, it was fluctuating but an improvement can be seen as the graph goes up.

Recall: Measures the model's capacity to identify every occurrence of positivity (minimize false negatives), this also is seen increasing.

NOTE: The epochs for this training were set to 50

4.3.2 Result for 100 epochs

The overall model performance and accuracy were optimized when observed by increasing the number of epochs to 100.

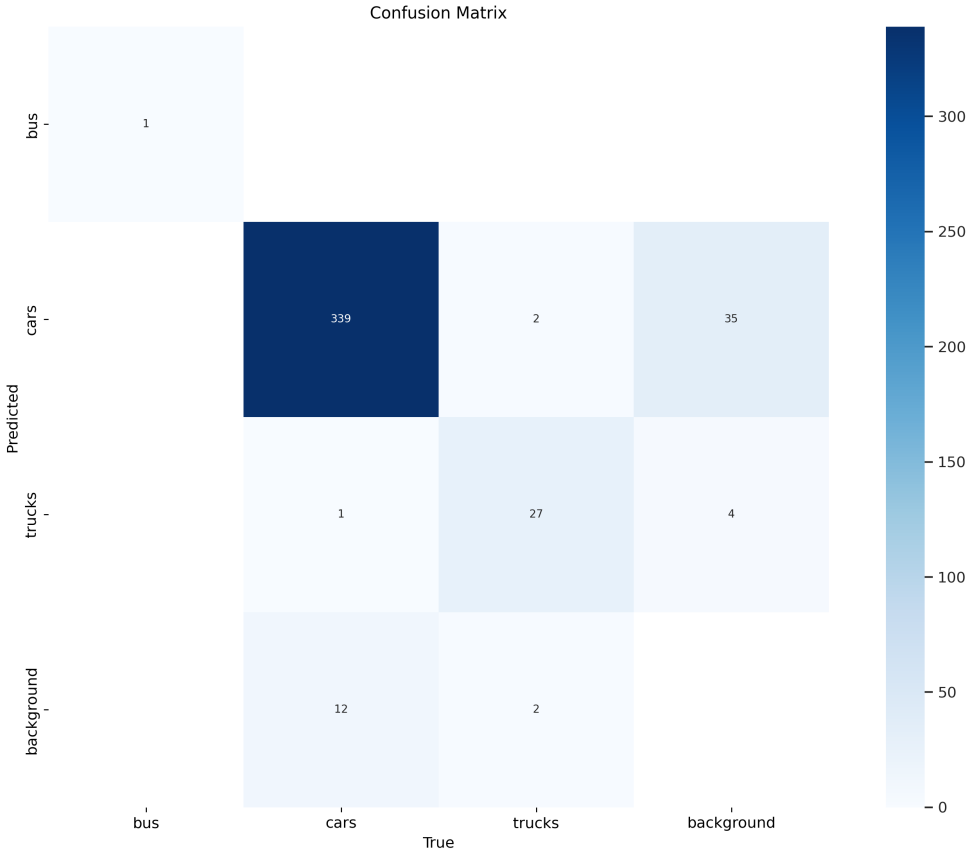


Figure 4.10: Confusion Matrix of Custom Data Trained for 100 epochs

Confusion matrix Compared to the previous matrix, figure 4.11 confusion matrix shows improved results in the number of correct classes identified. In the confusion matrix for 50 epochs, there were a number of 333 cars correlating and in the 100 epochs, there is an improvement of 339 correct cars classified.

Model performance on 100 epochs The outcomes are shown while the model's performance is visualized in Figure 4.12 and Table 4.1:

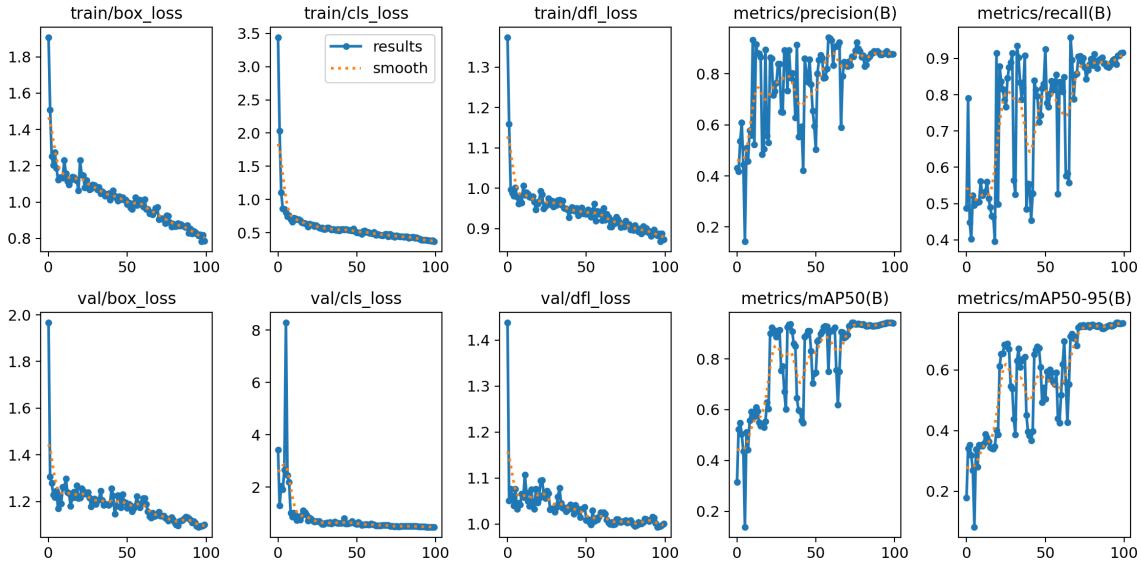


Figure 4.11: Performace metrics of Custom Data Trained on 100 epochs

Class	Images	Instances	Box(P	R	mAP50	mAP50-95
all	20	384	0.888	0.884	0.942	0.755
bus	20	1	0.812	1	0.995	0.995
cars	20	352	0.962	0.86	0.971	0.668
trucks	20	31	0.891	0.792	0.859	0.601

Table 4.1: Result for 100 epochs on custom dataset

The model appears to be functioning well overall, as seen by its impressive mAP50 score of 0.942 across all classes.

- A high Precision of 0.995 for the class "bus" indicates that practically all anticipated detections for this class are accurate.
- A high mAP50 score of 0.971 for the class "cars" indicates that the model is successfully identifying instances of vehicles in the photos.
- The model's performance on the "trucks" class is not as good as it is on the other classes, as evidenced by the class's significantly lower mAP50 score of 0.859.

4.3.3 Objects detection in the images

Figure 4.13 shows how objects were detected and displays their confidence score.

Figure 4.12: Objects detected on the test images

Chapter 5

Conclusion and Future Work

The objective and scope of this thesis work is to highlight and contribute to the problem of foggy weather object detection which has an impact on drivers, traffic control systems and authorities. Several datasets were observed but weren't thought useful. In machine learning or deep learning projects, the fuel to any problem-solving is the data. Relevant data produce relevant results. For this purpose, the very first step was to prepare a dataset that was usable and focused. A dataset of 8000 images was generated for this thesis project. The raw data was video footage of a highway where cars, trucks and buses were passing. The question may arise here that this data can be overfitting. The answer to this is that the system will be deployed for highways especially the toll plaza where only cars, buses, and trucks pass. The data of 8000 images consist of 4 types of images. A corresponding set of 2000 clear images and 2000 low fog images. Another corresponding set of 2000 images of medium fog and another 2000 images of high fog. The fog was synthetically applied. The purpose of the synthetical fog application was to have original imagery too, so we can examine the actual result compared to the fogged images result after applying object detection, so we know what is being missed to detect in the fogged images. The data after preparation was trained on YOLOv8 which is pre-trained on the COCO dataset. There was a significantly better detection of YOLOv8 on our data than the general large COCO dataset. The problem of object detection in foggy has gotten less attention in the area of computer vision and this work is an initiation for further work. The future work for this work is to apply filters

to the images and train an object detection model such as YOLOv8 or other latest techniques and examine and have an optimum solution for the detection of objects in the fog.

References

[1] Cheng-Jian Lin and Jyun-Yu Jhang. Intelligent traffic-monitoring system based on yolo and convolutional fuzzy neural networks. *IEEE Access*, 10:14120–14133, 2022.

[2] Hafiz Mohkum Hammad, Muhammad Ashraf, Farhat Abbas, Hafiz Faiq Bakhat, Saeed A Qaisrani, Muhammad Mubeen, Shah Fahad, and Muhammad Awais. Environmental factors affecting the frequency of road traffic accidents: a case study of sub-urban area of pakistan. *Environmental Science and Pollution Research*, 26: 11674–11685, 2019.

[3] NHMP. Roads of pakistan. https://nhmp.gov.pk/.

[4] Alexandra S Mueller and Lana M Trick. Driving in fog: The effects of driving experience and visibility on speed compensation and hazard avoidance. *Accident Analysis & Prevention*, 48:472–479, 2012.

[5] DJ Jeffery and ME White. Fog detection and some effects of fog on motorway traffic. *Traffic Engineering & Control*, 22(HS-032 509), 1981.

[6] Norbert Buch, Sergio A Velastin, and James Orwell. A review of computer vision techniques for the analysis of urban traffic. *IEEE Transactions on intelligent transportation systems*, 12(3):920–939, 2011.

[7] Housing news deck. Different types of roads, all you need to know. Housing.com.

[8] Sadanandam Anupoju. Classification of roads and their details. The constructor.

[9] Jagah Online. Traffic in pakistan. https://www.jagahonline.com/blog/traffic-in-pakistan, 2023. Accessed: August 2, 2023.

[10] Climatic regions of pakistan: Highland climate. https://contentgenerate.com/climatic-regions-of-pakistan-highland-climate-content-generate/. Accessed: August 10, 2023.

[11] Fulu Wei, Zhenggan Cai, Pan Liu, Yongqing Guo, Xin Li, and Qingyin Li. Exploring driver injury severity in single-vehicle crashes under foggy weather and clear weather. *Journal of advanced transportation*, 2021:1–12, 2021.

[12] Jaskirat Kaur and Williamjeet Singh. Tools, techniques, datasets and application areas for object detection in an image: a review. *Multimedia Tools and Applications*, 81(27):38297–38351, 2022.

[13] Yi-Qing Wang. An analysis of the viola-jones face detection algorithm. *Image Processing On Line*, 4:128–148, 2014.

[14] Carlo Tomasi. Histograms of oriented gradients. *Computer Vision Sampler*, pages 1–6, 2012.

[15] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.

[16] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[17] Jeremy Jordan. Object detection: One-stage methods. https://www.jeremyjordan.me/object-detection-one-stage/.

[18] Chengji Liu, Yufan Tao, Jiawei Liang, Kai Li, and Yihang Chen. Object detection based on yolo network. In *2018 IEEE 4th information technology and mechatronics engineering conference (ITOEC)*, pages 799–803. IEEE, 2018.

[19] Author(s) of the Article. Yolo object detection. https://www.v7labs.com/blog/yolo-object-detection#how-does-yolo-work-yolo-architecture, .

[20] Author(s) of the Article. Yolo (you only look once): Real-time object detection explained. https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006, .

[21] Sheeraz Memon, Sania Bhatti, Liaquat A Thebo, Mir Muhammad B Talpur, and Mohsin A Memon. A video based vehicle detection, counting and classification system. *International Journal of Image, Graphics and Signal Processing*, 10(9): 34–41, 2018.

[22] Samprit Bose, Chavan Deep Ramesh, and Maheshkumar H Kolekar. Vehicle classification and counting for traffic video monitoring using yolo-v3. In *2022 International Conference on Connected Systems & Intelligence (CSI)*, pages 1–8. IEEE, 2022.

[23] Rahul Kejriwal, HJ Ritika, Arpit Arora, et al. Vehicle detection and counting using deep learning basedyolo and deep sort algorithm for urban traffic management system. In *2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT)*, pages 1–6. IEEE, 2022.

[24] Xinchen Wang, Weiwei Zhang, Xuncheng Wu, Lingyun Xiao, Yubin Qian, and Zhi Fang. Real-time vehicle type classification with deep convolutional neural networks. *Journal of Real-Time Image Processing*, 16:5–14, 2019.

[25] N Kavitha and DN Chandrappa. Optimized yolov2 based vehicle classification and tracking for intelligent transportation system. *Results in Control and Optimization*, 2:100008, 2021.

[26] Huansheng Song, Haoxiang Liang, Huaiyu Li, Zhe Dai, and Xu Yun. Vision-based vehicle detection and counting system using deep learning in highway scenes. *European Transport Research Review*, 11(1):1–16, 2019.

[27] Shakir Ahmed Hameed Alhuthali, Muhammad Yousuf Irfan Zia, and Muhammad Rashid. A simplified traffic flow monitoring system using computer vision

techniques. In *2022 2nd International Conference on Computing and Information Technology (ICCIT)*, pages 167–170. IEEE, 2022.

[28] Saeed Arabi, Arya Haghighat, and Anuj Sharma. A deep-learning-based computer vision solution for construction vehicle detection. *Computer-Aided Civil and Infrastructure Engineering*, 35(7):753–767, 2020.

[29] Hai Wang, Yijie Yu, Yingfeng Cai, Xiaobo Chen, Long Chen, and Qingchao Liu. A comparative study of state-of-the-art deep learning algorithms for vehicle detection. *IEEE Intelligent Transportation Systems Magazine*, 11(2):82–95, 2019.

[30] Ahmad Arinaldi, Jaka Arya Pradana, and Arlan Arventa Gurusinga. Detection and classification of vehicles for traffic video analytics. *Procedia computer science*, 144:259–268, 2018.

[31] Prithwish Sen, Anindita Das, and Nilkanta Sahu. Object detection in foggy weather conditions. In *International Conference on Intelligent Computing & Optimization*, pages 728–737. Springer, 2021.

[32] Rahul Singh, Someet Singh, and Navjot Kaur. A review: Techniques of vehicle detection in fog. *Indian Journal of Science and Technology*, 9(45):1–4, 2016.

[33] Rajib Ghosh. On-road vehicle detection in varying weather conditions using faster r-cnn with several region proposal networks. *Multimedia Tools and Applications*, 80(17):25985–25999, 2021.

[34] Mario Pavlić, Heidrun Belzner, Gerhard Rigoll, and Slobodan Ilić. Image based fog detection in vehicles. In *2012 IEEE Intelligent Vehicles Symposium*, pages 1132–1137. IEEE, 2012.

[35] Walid Balid, Hasan Tafish, and Hazem H Refai. Intelligent vehicle counting and classification sensor for real-time traffic surveillance. *IEEE Transactions on Intelligent Transportation Systems*, 19(6):1784–1794, 2017.

[36] Luis Unzueta, Marcos Nieto, Andoni Cortés, Javier Barandiaran, Oihana Otaegui, and Pedro Sánchez. Adaptive multicue background subtraction for robust vehi-

cle counting and classification. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):527–540, 2011.

[37] Belmar Garcia-Garcia, Thierry Bouwmans, and Alberto Jorge Rosales Silva. Background subtraction in real applications: Challenges, current models and future directions. *Computer Science Review*, 35:100204, 2020.

[38] Glenn Jocher. Ultralytics yolov8 tasks. https://docs.ultralytics.com/tasks/, 2023. Accessed: August 2, 2023.

Appendix A

Code

This code extracts images from a video clip

```
import cv2
video_cap = cv2.VideoCapture('video.mp4')
def getFrame(sec):
    video_cap.set(cv2.CAP_PROP_POS_MSEC,sec*1000)
    hasFrames,image = video_cap.read()
    if hasFrames:
        cv2.imwrite("image"+str(count)+".jpg", image)
    return hasFrames
sec = 0
frameRate = 1 #//it will capture image in each 1 second
count=1
success = getFrame(sec)
while success:
    count = count + 1
    sec = sec + frameRate
    sec = round(sec, 2)
    success = getFrame(sec)
```

Figure A.1: Image Frame extraction

The code below will apply inference on some image files for YOLOv8

```
!pip install ultralytics

from ultralytics import YOLO

model = YOLO('yolov8m.pt')

results = model.predict('/content/fog-data/image_001.jpg')
```

Figure A.2: Inference for YOLOv8

The code below will predict the detection and classification of vehicles in a number of given images

```
from google.colab.patches import cv2_imshow
import cv2

image_paths = ['/content/fog-data/image_001.jpg', '/content/fog-data/
fogged_image_001.jpg', '/content/fog-data/fog2-image_001.jpg', '/
content/fog-data/fog3-image_001.jpg']

for image_path in image_paths:
    image = cv2.imread(image_path)  # Load the image
    results = model.predict(image_path)  # Perform the prediction
    results = results[0]
    for box in results.boxes:
        class_id = results.names[box.cls[0].item()]
        cords = box.xyxy[0].tolist()
        cords = [round(x) for x in cords]
        conf = round(box.conf[0].item(), 2)

        # Draw bounding box
        x1, y1, x2, y2 = cords
        cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)

        # Add label and confidence
        label = f'{class_id}: {conf}'
        cv2.putText(image, label, (x1, y1 - 10),
        cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)

    cv2_imshow(image)  # Display the image
```

Figure A.3: Detection and Classification of vehicles

The code will provide details about the objects detected: the class, coordinates and the probabilities

```
for box in result.boxes:
class_id = result.names[box.cls[0].item()]
cords = box.xyxy[0].tolist()
cords = [round(x) for x in cords]
conf = round(box.conf[0].item(), 2)
print('Object type:', class_id)
print('Coordinates:', cords)
print('Probability:', conf)
print('---')
```

Figure A.4: Class, coordination and probabilities of objects detected

The following results of the above code will be displayed

```
Object type: truck
Coordinates: [1, 283, 180, 479]
Probability: 0.9
---
Object type: car
Coordinates: [194, 371, 249, 437]
Probability: 0.77
---
Object type: car
Coordinates: [348, 219, 367, 239]
Probability: 0.76
---
Object type: car
Coordinates: [257, 239, 281, 263]
Probability: 0.76
---
```

Figure A.5: Bounding boxes of the objects detected

The code below will detect, count and classify the number of vehicles in the images and plot them.

```
import matplotlib.pyplot as plt

image_paths = ['/content/fog-data2/image_002.jpg', '/
content/fog-data2/fogged_image_002.jpg', '/content/
fog-data2/fog2-image_002.jpg', '/content/fog-data2/
fog3-image_002.jpg']

num_cars = []
num_trucks = []

for image_path in image_paths:
    image = cv2.imread(image_path)  # Load the image

    results = model.predict(image_path)  # Perform the prediction
    results = results[0]  # Access the first element of the results lis

    cars = 0
    trucks = 0

    for box in results.boxes:
        class_id = results.names[box.cls[0].item()]
        if class_id == 'car':
            cars += 1
        elif class_id == 'truck':
            trucks += 1

    num_cars.append(cars)
    num_trucks.append(trucks)
```

Figure A.6: Detect, count and classify and plot the objects - Part 1

```
# Create a bar plot for the number of cars
plt.figure(figsize=(16, 2))
plt.bar(image_paths, num_cars)
plt.xlabel('object')
plt.ylabel('Number of Cars')
plt.title('Number of Cars Detected in Each Image')
plt.show()

# Create a bar plot for the number of trucks
plt.figure(figsize=(16, 2))
plt.bar(image_paths, num_trucks)
plt.xlabel('Image')
plt.ylabel('Number of Trucks')
plt.title('Number of Trucks Detected in Each Image')
plt.show()
```

The result is displayed as the below, the figures is the output of the above code for counting and classification of vehicle types in the input images.

Figure A.7: Detect, count, classify and plot- Part 2

The result is displayed as the below, the figures is the output of the above code for counting and classification of vehicle types in the input images.

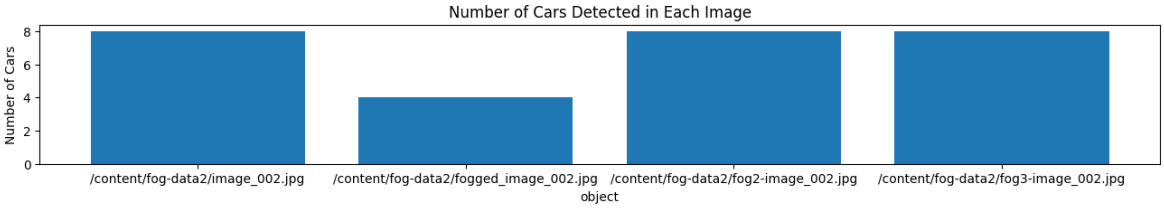


Figure A.8: Inference result

To train the YOLOv8 model on a custom dataset

```
!nvidia-smi

#Colab provides limited GPU compute units, to access it,
the above code will be used.

!pip install ultralytics

from ultralytics import YOLO
model = YOLO('yolov8m.pt')

!yolo predict model=yolov8m.pt source='/content/drive/MyDrive
/thesis-final/dataset/test/medium-fog/mf-image(530).jpg'

from google.colab import drive
drive.mount('/content/drive')

%cd /content/drive/MyDrive/thesis-final

%ls

!yolo task=detect mode=train model=yolov8m.pt data=data.yaml
epochs=100 imgsz=640 plots=True

#In the above line the model has been trained on the custom dataset,
now we will check the results.

%ls runs/detect/train6

from IPython.display import Image
```

Figure A.9: Yolov8 training on custom data, Part 1

```
Image(filename='runs/detect/train6/confusion_matrix.png', width=600)

Image(filename='runs/detect/train6/results.png', width=600)

#After this, we will validate the model by taking the weights of
the model trained and applying on validation dataset.

!yolo task=detect mode=val model=runs/detect/train3/weights
/best.pt data=data.yaml

!yolo task=detect mode=predict model=runs/detect/train5/weights/
best.pt conf=0.25 source=/content/drive/MyDrive/thesis-final/
dataset/test/medium-fog

import glob
from IPython.display import Image, display

for image_path in glob.glob('runs/detect/predict/*jpg') [:3]:
  display(Image(filename=image_path, width=600))
```

Figure A.10: Yolov8 tarining on custom data, part 2