

**DEVELOPMENT OF VEHICLE DETECTION AND COUNTING SYSTEM FOR  
TRAFFIC ANALYSIS USING COMPUTER VISION**

**BY**

**Kenny Saw Wei Wen**

**A REPORT**

**SUBMITTED TO**

**Universiti Tunku Abdul Rahman**

**in partial fulfillment of the requirements**

**for the degree of**

**BACHELOR OF COMPUTER SCIENCE (HONOURS)**

**Faculty of Information and Communication Technology**

**(Kampar Campus)**

**JUNE 2024**

## REPORT STATUS DECLARATION FORM

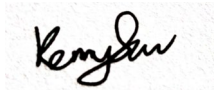
**Title:** Development of Vehicle Detection and Counting System for Traffic Analysis using Computer Vision

**Academic Session:** 2024/6

I KENNY SAW WEI WEN declare that I allow this Final Year Project Report to be kept in Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



(Author's signature)



(Supervisor's signature)

**Address:**

8, Leboh Bercham (S) 2/2,  
Taman Desa Impian,  
31400 Ipoh, Perak

Supervisor's name

Muhammad Syaiful Amri bin Suhaimi

**Date:** 9/9/2024

**Date:** 09/09/2024

<b>Universiti Tunku Abdul Rahman</b>			
Form Title : <b>Sample of Submission Sheet for FYP/Dissertation/Thesis</b>			
Form Number: <b>FM-IAD-004</b>	Rev No.: <b>0</b>	Effective Date: <b>21 JUNE 2011</b>	Page No.: <b>1 of 1</b>

**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**  
**UNIVERSITI TUNKU ABDUL RAHMAN**

Date: 9/9/2024

**SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS**

It is hereby certified that Kenny Saw Wei Wen (ID No: 21ACB06312) has completed this final year project entitled “Development of Vehicle Detection and Counting System for Traffic Analysis using Computer Vision” under the supervision of Dr Muhammad Syaiful Amri Bin Suhaimi (Supervisor) from the Department of Computer Science, Faculty of Information and Communication Technology.

I understand that University will upload softcopy of my final year project in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

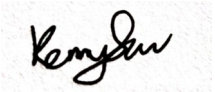


Kenny Saw Wei Wen  
(*Student Name*)

\*Delete whichever not applicable

## DECLARATION OF ORIGINALITY

I declare that this report entitled “**DEVELOPMENT OF VEHICLE DETECTION AND COUNTING SYSTEM FOR TRAFFIC ANALYSIS USING COMPUTER VISION**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : 

Name : Kenny Saw Wei Wen

Date : 9/9/2024

## **ACKNOWLEDGEMENTS**

I would like to express my sincere thanks and appreciation to my supervisors, Dr Muhammad Syaiful Amri Bin Suhaimi and Ts Dr Chai Meei Tyng who has given me this bright opportunity to engage in a Computer Vision project. It is my first step to establish a career in Computer Vision. A million thanks to both of you.

To a very special person in my life, Hooi Lok Li, my beloved mother for her patience, unconditional support, and love, and for standing by my side during all the hard times. Finally, I must say thanks to my family for their love, support, and continuous encouragement throughout the course. Without their support, I believe it will be much harder and more stressful throughout the entire project's journey.

## **ABSTRACT**

In traffic analysis, a real-time vehicle detection and counting system is definitely important, as existing systems only provide the ability to detect or count, but not all of them. In this project, the aims are to achieve a video-based vehicle detection and counting system via implementing various computer vision techniques using python language and several libraries including OpenCV, PyTorch and etc. Initially, the image and video dataset of road traffic is gathered from various online sources and undergoes categorization before loaded into the system for model training. Then two model training process begins, one is for vehicle model detection to learn the feature of a vehicle and the image background, while another training is for binary weather classification for rainy and sunny conditions. Once the trained weights are ready a deep learning algorithm detection model is applied to detect and draw a bounding box on the vehicles. Lastly another deep sort algorithm is utilized to keep track of the detected vehicles and count them accordingly. In conclusion, this project aims to produce a robust yet effective and accurate video-based vehicle detection and counting system.

# TABLE OF CONTENTS

<b>TITLE PAGE</b>	<b>i</b>
<b>REPORT STATUS DECLARATION FORM</b>	<b>ii</b>
<b>FYP THESIS SUBMISSION FORM</b>	<b>iii</b>
<b>DECLARATION OF ORIGINALITY</b>	<b>iv</b>
<b>ACKNOWLEDGEMENTS</b>	<b>v</b>
<b>ABSTRACT</b>	<b>vi</b>
<b>TABLE OF CONTENTS</b>	<b>vii-x</b>
<b>LIST OF FIGURES</b>	<b>xi-xii</b>
<b>LIST OF TABLES</b>	<b>xiii</b>
<b>LIST OF SYMBOLS</b>	<b>xiv</b>
<b>LIST OF ABBREVIATIONS</b>	<b>xv</b>
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 Problem Statement and Motivation	1
1.1.1 Problem Statement	1
1.1.2 Motivation	2
1.2 Objectives	3
1.3 Project Scope and Direction	3
1.4 Contributions	3
1.5 Report Organization	4

<b>CHAPTER 2 LITERATURE REVIEW</b>	<b>5</b>
2.1 Previous work on Vehicle Detection and Counting System for Traffic Analysis	5
2.2 Review of the Existing Systems	5
2.2.1 Vision-based vehicle detection and counting system using deep learning in highway scenes (2019)	5-7
2.2.2 A real-time vehicle detection system under various bad weather conditions based on a deep learning model without retraining (2020)	8-11
2.2.3 Real-time vehicle classification and tracking using a transfer learning-improved deep learning network (2022)	12-14
2.3 Strength and Weakness	15
2.3.1 Vision-based Vehicle Detection and Counting System Using Deep Learning in Highway Scenes	15
2.3.2 A Real-Time Vehicle Detection System Under Various Bad Weather Conditions Based on a Deep Learning Model Without Retraining	15-16
2.3.3 Real-time Vehicle Classification and Tracking Using a Transfer Learning-Improved Deep Learning Network	16-17
2.4 Summary	17
<b>CHAPTER 3 SYSTEM METHODOLOGY/APPROACH (FOR DEVELOPMENT-BASED PROJECT)</b>	<b>18</b>
3.1 Data Acquisition and Preparation	18-19
3.2 Data Preprocessing	19
3.3 Machine Learning Models	19
3.3.1 Vehicle Detection Model	19-20
3.3.2 Weather Classification Model	20
3.4 Model Application and System Integration	20
3.4.1 Video Enhancement Using Weather Classification Model	20-21
3.4.2 Vehicle Detection in Processed Video	21
3.4.3 Vehicle Tracking and Counting with Deep Sort	21
3.5 System Evaluation	22



<b>CHAPTER 4 SYSTEM DESIGN</b>	23
4.1 System Block Diagram	23-24
4.2 System Components Specifications	24
4.2.1 Data Acquisition	24
4.2.2 Data Segregation	24
4.2.3 Data Labelling and Annotation	24-25
4.2.4 Vehicle Detection Model Training	25
4.2.5 Weather Classification Model Training	25
4.2.6 Video Processing	25-26
4.2.7 Vehicle Detection	26
4.2.8 Vehicle Tracking and Counting	26
<b>CHAPTER 5 SYSTEM IMPLEMENTATION (FOR DEVELOPMENT- BASED PROJECT)</b>	<b>27</b>
5.1 Hardware and Software Setup	27
5.2 Setting and Configuration	27-28
5.3 Code Implementation and Explanation	28
5.3.1 Vehicle Detection Model Training	28-30
5.3.2 Weather Classification Model Training	30-33
5.3.3 Video Processing	33-35
5.3.4 Vehicle Detection and Counting	36-39
5.4 System Operation	39
5.4.1 Vehicle Detection Model Training Dataset	39-41
5.4.2 Weather Classification Model Training Dataset	41-43
5.4.3 Vehicle Model Detection Training	43-44
5.4.4 Weather Classification Model Training	45
5.4.5 Video Processing	45-47
5.4.6 Vehicle Detection	48
5.4.7 Vehicle Detection with Tracking	49
5.4.8 Vehicle Detection, Tracking and Counting	50
5.5 Implementation Issues and Challenges	51

<b>CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION</b>	<b>52</b>
6.1 System Testing and Performance Metrics	52
6.1.1 Accuracy	52
6.1.2 Reliability and efficiency	53
6.2 Testing Setup and Result	54
6.2.1 Detection Model Comparison with Alternative Methods (Faster-RCNN)	54-56
6.2.2 Accuracy Assessment of Video Enhancement Effects	56-59
6.2.3 Reliability Assessment	59-60
6.3 Objective Evaluation	60
6.4 Project Challenges	60-61
<b>CHAPTER 7 CONCLUSION AND RECOMMENDATION</b>	<b>62</b>
<b>REFERENCES</b>	<b>63</b>
<b>APPENDIX</b>	
<b>WEEKLY LOG</b>	<b>65</b>
<b>POSTER</b>	<b>69</b>
<b>PLAGIARISM CHECK RESULT</b>	<b>70</b>
<b>FYP2 CHECKLIST</b>	<b>81</b>

## LIST OF FIGURES

<b>Figure Number</b>	<b>Title</b>	<b>Page</b>
Figure 2.2.1.1	Block diagram of vehicle detection system	6
Figure 2.2.1.2	Overall block diagram of road surface segmentation	6
Figure 2.2.1.3	Track counting result	7
Figure 2.2.2.1	Overall system flowchart	8
Figure 2.2.2.2	Left: Original glare image Right: Corrected glare image	9
Figure 2.2.2.3	Left: Original haze image Right: Corrected haze image	9
Figure 2.2.2.4	Left: Original rainy image Right: Corrected rainy image	10
Figure 2.2.2.5	Comparison of non-corrected image and corrected image	10
Figure 2.2.3.1	A lane-based count and speed detection technique for multi-vehicle tracking	12
Figure 2.2.3.2	Performance achieved by various YOLO network	13
Figure 3.0	Overview of system methodology	18
Figure 4.1.1	System Block Diagram of Vehicle Detection and Counting System	23
Figure 5.4.1.1	Vehicle detection model training folder structure	39
Figure 5.4.1.2	Vehicle model training set images	40
Figure 5.4.1.3	Vehicle model validation set images	40
Figure 5.4.2.1	Weather classification model training folder structure	41
Figure 5.4.2.2	Rainy training set images	41
Figure 5.4.2.3	Sunny training set images	42
Figure 5.4.2.4	Rainy validation set images	42
Figure 5.4.2.5	Sunny validation set images	43
Figure 5.4.3.1	Confusion matrix of the training model	43
Figure 5.4.3.2	Output weights of the vehicle model detection training process	44
Figure 5.4.4.1	Weather classification training process	45
Figure 5.4.5.1	Original sunny video	45
Figure 5.4.5.2	Enhanced sunny video	46

Figure 5.4.5.3	Original rainy video	46
Figure 5.4.5.4	Enhanced rainy video	46
Figure 5.4.5.5	Output of videoprocessing.py	47
Figure 5.4.6.1	Current processing frame of video	48
Figure 5.4.6.2	Vehicle detected in the video frame with bounding box	48
Figure 5.4.7.1	Vehicle detected and tracked in the video frame	49
Figure 5.4.8.1	Vehicle detected, tracked and counted in the video frame	50
Figure 5.4.8.2	Console output of vehicle detection and counting	50
Figure 6.2.1.1	Faster-RCNN with Deep Sort	54
Figure 6.2.1.2	YOLOv5 with Deep Sort	55
Figure 6.2.1.3	Execution time of Faster-RCNN	55
Figure 6.2.1.4	Execution time of YOLOv5	55
Figure 6.2.2.1	Original rainy video output	56
Figure 6.2.2.2	Enhanced rainy video output	56
Figure 6.2.2.3	Example of misidentification	56
Figure 6.2.2.4	Example of correct identification	57
Figure 6.2.2.5	Example of false detection	57
Figure 6.2.2.6	Example of correct detection	58
Figure 6.2.2.7	Example of heavy traffic conditions	58
Figure 6.2.3.1	1 <sup>st</sup> iteration of vehicle detection process	59
Figure 6.2.3.2	2 <sup>nd</sup> iteration of vehicle detection process	59
Figure 6.2.3.3	3 <sup>rd</sup> iteration of vehicle detection process	59

## LIST OF TABLES

<b>Table Number</b>	<b>Title</b>	<b>Page</b>
Table 2.3.1.1	Strength and description	15
Table 2.3.1.2	Limitation and description	15
Table 2.3.2.1	Strength and description	16
Table 2.3.2.2	Limitation and description	16
Table 2.3.3.1	Strength and description	16
Table 2.3.3.2	Limitation and description	17
Table 5.1.1	Specifications of laptop	27
Table 6.2.1	Performance comparison of Faster-RCNN and YOLOv5	56
Table 6.2.2.1	Vehicle detection accuracy across video conditions	59
Table 6.2.3.1	Vehicle counted and execution time across iterations	60

# LIST OF SYMBOLS

## LIST OF ABBREVIATIONS

<i>YOLO</i>	You Only Look Once
<i>YOLOv5</i>	You Only Look Once Version 5
<i>VGG16</i>	Visual Geometry Group 16
<i>FASTER-RCNN</i>	Faster Region Convolutional Neural Network
<i>CPU</i>	Central Processing Unit
<i>GPU</i>	Graphics Processing Unit

# Chapter 1

## Introduction

In modern society today, efficient yet accurate traffic management has become a challenge to achieve with the increasing of road traffic in the cities. The main focus of this system development is to resolve the problems encountered by road users, such as road congestion that leads to a waste of time. With the help of advanced computer vision technologies, this project focuses on developing an integrated vehicle detection and counting system. First, the system will take the input video and undergo processing, and several algorithms are applied to clearly identify a variety of vehicles and record them down. By the end of the development, the results generated are capable of assisting several purposes such as urban planning, traffic flow monitoring, traffic management and insights into road traffic. The outcome will benefit both the traffic administrator and road user in a sense that traffic administrators are more easily implement traffic management systems such as building traffic lights in the most congested areas to smooth the traffic, and road users can choose the best route to reach their desire destination without involving in congested traffic.

### 1.1 Problem Statement and Motivation

#### 1.1.1 Problem Statement

According to the authors of [12], due to the advancement of different models and sizes of vehicles being introduced every year, intelligent vehicle detection and counting systems are becoming a challenge to overcome as the difficulty of achieving high accuracy for detecting and counting vehicles in the traffic. This system contains two main processes, detection of moving vehicles in the road traffic and counting each of the detected vehicles and recording them. However, there are a few main issues that exist in the detection and counting system. First, the problem statement of this project is the **limited dynamic adaptability of the detection and counting system**. The main problem that arises is that most of the existing system can only detect and count the vehicle for specific road structures. This is because most of the developer pay too much attention to the accuracy of the system and forgot the system's robustness tradeoff.



The next problem that arises is that **environmental conditions impair detection reliability**. In [13], Xiu-Zhi Chen et al. point out that one of the crucial factors that affected vehicle detection accuracy is the various bad weather conditions. Some of the external factors, such as lighting, different weather conditions and shadows, have significantly affected the performance of vehicle detection systems. Additionally, vehicle counting logic fully depends on the vehicle detection algorithm, and once the system is unable to detect a certain vehicle, a particular vehicle will not be counted. In another way, this will also affect the accuracy of the counting logic.

Lastly, the third problem is the **lack of integration of detection and counting system**. Most of the existing system in the market are mainly focus on one objective whether specializing in vehicle detection, counting or classification as unified approach are most challenging and non-beneficial. Hence by building an integrated system that is able to execute both task in a single system will definitely ease the process of traffic management.

### **1.1.2 Motivation**

The motivation behind this thesis is to produce an efficient yet dynamic vehicle detection and counting system. The capacity of the system to adjust dynamically to different road types is essential for precise vehicle identification and counting, which subsequently able to guide traffic management strategies and infrastructure construction.

Other than that, the existing system falls short of achieving high accuracy in detecting vehicles in various traffic conditions due to the system's lack of robustness. For example, suppose the system misses count due to the inability to detect the vehicle in heavy traffic and report it as fast-moving traffic. In that case, the traffic administrator might plan for a road maintenance schedule that causes unnecessary traffic congestion. Hence, an accurate vehicle detection system could dramatically improve traffic analysis and planning.

Moreover, this project aims to develop an integrated system that consists of both vehicle detection and counting functions. The existing system mostly provides fragmentation of function, which makes it hard for traffic analysis to analyze traffic flow precisely. In addition,

an integrated system able to provide an insight of the total car volume in a certain intersection of road traffic and ease the process of traffic management.

## **1.2 Objectives**

In this project, the aim is to develop an integrated system consisting of a vehicle detection model and counting logic, particularly focusing on applying the most effective algorithm to achieve the highest accuracy of detecting moving vehicle in a road traffic. Other than that, challenges such as varying light condition, weather and angle of vehicle captured by camera also normally impair the accuracy of the detection mechanism. This thesis proposes leveraging advanced computer vision techniques and machine learning models to enhance the reliability of detection and counting precision. With the help of existing vast ecosystem of libraries this thesis aims to develop an effective vehicle detection and counting system of real-time video footage of a moving traffic.

## **1.3 Project Scope and Direction**

In this thesis, the project has progressed from the initial phase, building upon the foundational work of a robust image-based vehicle detection system. The current stage has successfully expanded into video-based, where the model is able to detect, track, and finally count all the vehicles across video frames. While developing the system, the main goal of this project is to ensure that the system can work effectively even under diverse weather conditions. The next phase of the project will focus on optimizing the processes and exploring any other way of refining the system's performance.

## **1.4 Contributions**

This research aims to identify the most suitable algorithm for developing the video-based vehicle detection and counting system. Several detection algorithms are taken into account to find the best algorithm that produces the highest accuracy while detecting the moving vehicles in the road traffic. Furthermore, this research employs transfer learning to improve the efficiency and efficacy of the vehicle detecting system. By utilising pre-trained models tailored to the objective of vehicle identification and weather classification, the system advantageously exploits previously acquired characteristics and patterns, therefore greatly minimising the requirement for substantial data from the beginning. This methodology not only expedites the

training procedure but also enhances the precision and resilience of the model, especially in identifying vehicles under diverse weather conditions.

## **1.5 Report Organization**

The details of this project will be shown in the upcoming chapters. In chapter 2, some new studies have been made regarding video-based vehicle detection and counting systems and each of the advantages and limitation of the proposed solutions are compared and reviewed. Furthermore, in chapter 3 the system methodology is discussed clearly with the provided overall system approach diagram. On top of that, chapter 4 describes the system design and discussed the system components in detail for better understanding of the entire system. Chapter 5 mainly focus on the implementation of the system including explanation of the code and the figure included for each output produced. Where in chapter 6, system evaluation is performed to discuss the system's accuracy and overall performance with the applied methods. Lastly, chapter 7 concludes the entire project and discusses on future approaches.

# Chapter 2

## Literature Review

### 2.1 Previous work on Vehicle Detection and Counting System for Traffic Analysis

In order to improve traffic management, real-time video processing and advanced imaging technologies are being integrated into vehicle recognition and counting systems. Vehicle detection systems augmented with visibility complementing modules have demonstrated significant potential in reducing the negative effects of environmental conditions like rain and fog on identification accuracy, according to a study by Xiu-Zhi Chen et al. In a similar vein, Bipul Neupane et al. talked about how transfer learning can be used to enhance the capabilities of deep learning networks especially YOLO models for more effective real-time vehicle tracking and categorization. These studies demonstrate how sophisticated and capable today's traffic monitoring systems are becoming, but they also show enduring difficulties including correct vehicle classification in a variety of traffic and weather scenarios and dynamic backdrop adaptability. In order to find possible areas for more innovation and improvement in traffic analysis systems, this research segment lays the groundwork for a thorough investigation of each strategy.

### 2.2 Review of the Existing Systems

#### 2.2.1 Vision-based vehicle detection and counting system using deep learning in highway scenes (2019)

A method using deep learning technology has been proposed by Huansheng Song et al. The authors proposed this vision-based vehicle detection and counting system specifically tailored for highway scenes. This study focusses on the complexity of detecting various sizes of vehicle that have direct impact on the accuracy of the vehicle counts on highways. According to Huansheng Song et al. [12], conventional vehicle detection methods such as background subtraction and frame difference are effective but fall short under different road conditions such as lighting and bad weather. Hence, they point out that with the help of deep learning, particularly Convolutional Neural Networks (CNNs) has shown the increase of performance in accurately detecting vehicles due to their robust extraction feature. Figure 2.2.1.1 shows the overall flow of the method proposed by the authors.

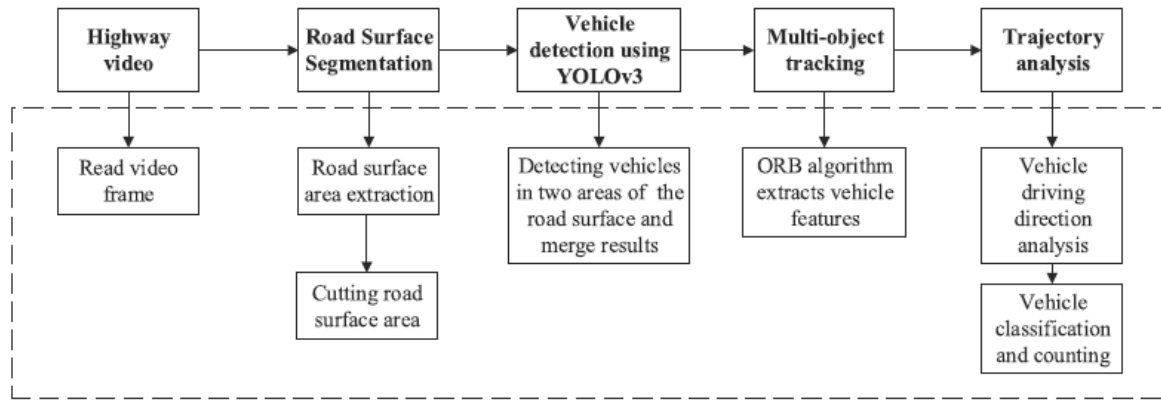


Figure 2.2.1.1 Block diagram of vehicle detection system

The system starts with the segments of video frame captured from the highway to identify the road surface area. This is crucial because precise segmentation guarantees high accuracy of the detection algorithm on relevant part of the image. Upon successful segmentation, the YOLOv3 algorithm is applied. This deep learning technique is utilized for its robustness in detecting objects in varied conditions and is efficient in processing video frames in real-time. Next, ORB algorithm is applied to extract features from the detected vehicle to track multiple objects across frames. The authors also stated that this step is vital for maintaining the continuity of vehicle detection and counting accuracy. Finally, the system analyses the trajectory of each tracked vehicle, categorizing each of the vehicle by type and counting them as they pass through the designated areas of the road. In the following Figure 2.2.1.2 shows the overall process of the road surface segmentation.

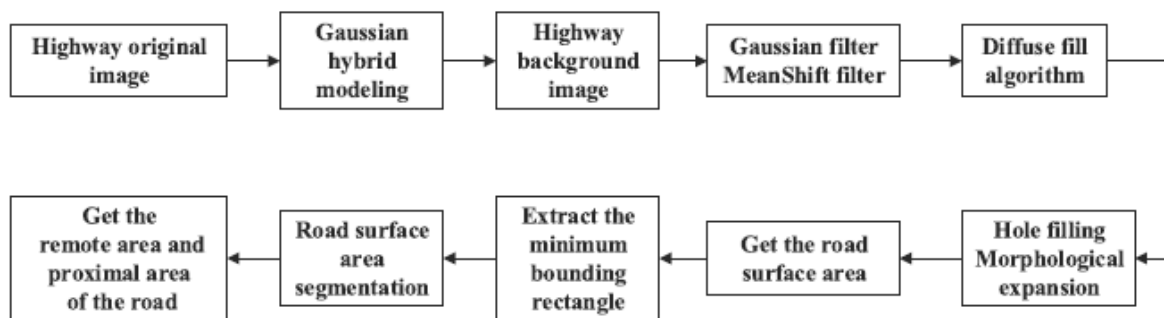


Figure 2.2.1.2 Overall block diagram of road surface segmentation

According to Huansheng Song et al. [12], the road surface extraction process starts with original input of highway image, applying Gaussian hybrid modeling to the image to differentiate the road surface from other attributes. Then, a MeanShift filter is applied to the

image to refine the original image quality, which will help in better of distinguishing the road surface. After that, hole filling morphological expansion is applied to make the road surface representation continuous and complete. These techniques address any gaps or discontinuities in the segmented road surface resulting a better area for accurate vehicle detection stated by the authors. Once the road surface is segmented, the system then proceeds with extracting the minimum bounding rectangle. This feature is to encapsulates the road surface optimizing the region of interest for vehicle detection. Lastly the system segments the road into remote and proximal areas for further analysis.

Scenes		Scene 1			Scene 2			Scene 3			Direction correct rate
Video frames		11000			22500			41000			
Vehicle category		Car	Bus	Truck	Car	Bus	Truck	Car	Bus	Truck	
Direction A	Our method	29	21	3	110	40	21	287	141	22	
	Actual number of vehicles	32	21	3	117	43	22	297	150	24	
	Extra Number	3	0	0	8	3	2	15	13	3	0.92
	Missing number	0	0	0	1	0	1	5	4	1	
	Correct rate	0.906	1	1	0.923	0.930	0.864	0.933	0.887	0.833	
Direction B	Our method	41	37	4	117	69	13	300	168	15	
	Actual number of vehicles	43	38	4	125	77	13	311	172	17	
	Extra Number	2	2	0	11	10	0	15	8	2	0.931
	Missing number	0	1	0	3	2	0	4	4	0	
	Correct rate	0.953	0.947	1	0.888	0.844	1	0.939	0.930	0.882	
Real time rate		1.27			1.35			1.48			
Average correct rate		0.967			0.911			0.917			0.932

Figure 2.2.1.3 Track counting result

The Figure 2.2.1.3 above is the table of the track counting result of the vehicle detection and counting system proposed by the authors. In [12], Huansheng Song et al. concludes that the integration of these technologies not only outperforms traditional detection methods but also offers scalability and adaptability for real-world scenarios. The authors suggest that such system can significantly contribute to intelligent traffic management and control, especially in managing highway traffic.

### 2.2.2 A real-time vehicle detection system under various bad weather conditions based on a deep learning model without retraining (2020)

In [13], Xiu-Zhi Chen et al. has proposed a vehicle detection system that operates effectively under diverse adverse weather condition by using a complementation module that enhances the accuracy of the detection. The authors stated that most of the traditional vehicle detection system available in the industry rely too much on machine learning algorithms with manually defined features such as Haar-like and histogram of gradients (HoG). These features work fine under normal weather condition but not in bad weather conditions like raining or low-light traffic. Hence, Xiu-Zhi Chen et al. proposed a new system that employs YOLOv3, which is a deep learning model that is well known for its performance and efficiency in real-time object detection. Below Figure 2.2.2.1 is the flowchart of the proposed system.

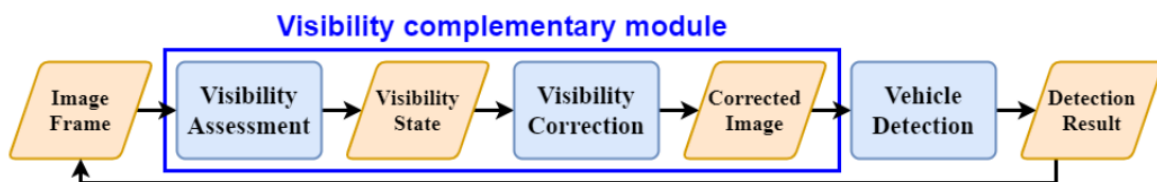


Figure 2.2.2.1 Overall system flowchart

The proposed system begins with acquisition of video frames from highway cameras. These frames collected will act as primary input for the entire detection process. Each of the collected video frames undergoes a visibility assessment phase, where the system evaluates the visibility conditions of the video. This assessment is important for determining the specific challenges that contain inside the image such as fog and rain. Next, the system categorizes the visibility into predefined states and each state corresponds to a particular type of visibility impairment commonly encountered. After the state being determined, the system applies a series of visibility correction algorithms to enhance the image by improving contrast, brightness and clarity. Once the output from the visibility correction stage is completed, the image will be then been optimized once again for better vehicle detection. With the enhanced image, the system employs the YOLOv3 learning model to detect the vehicles in the traffic.



Figure 2.2.2.2 Left: Original glare image Right: Corrected glare image

From the Figure 2.2.2.2 above, Xiu-Zhi Chen et al. explained that the visibility correction of glare is performed by using an advanced algorithm to adjust the image's contrast and brightness dynamically, ensuring that the glare does not obscure the view. From the corrected glare image, the vehicle detection system is able to detect vehicles in the traffic more accurately even under harsh sunlight conditions.



Figure 2.2.2.3 Left: Original haze image Right: Corrected haze image

For haze conditions, in the Figure 2.2.2.3 above shows the visibility correction module employs techniques to clarify and enhance the image by reducing the foggy overlay that caused by haze weather. The authors state that first the system analyzes the color and saturation levels to adjust the image clarity, effectively cutting through the haze to reveal a clearer scene. From the figure above, the correction process successfully enhances the image and eases the detection process.





Figure 2.2.2.4 Left: Original rainy image Right: Corrected rainy image

Xiu-Zhi Chen et al. also explained on the rainy visibility correction as shown in the Figure 2.2.2.4 above. The correction method proposed for rainy conditions focuses on removing distortion caused by the raindrops on the camera lens and reducing visual impairment caused by heavy rainfall in the traffic. By applying the filter that identifies and isolates the distortion typical of rain, the authors claim that the system able to construct the affected areas of the image to restore visibility. Figure above show how the technology applied capable of recovering the visibility of the original rainy image, ensuring that the vehicle detection remains functional even in adverse weather conditions.

	Scene	Bounding Box Amount	Positive	False Detection	False Classification
Glare	Without proposed	5702	4859 (85.22%)	4 (0.07%)	839 (14.71%)
	With proposed	8093	7269 (89.82%)	17 (0.21%)	807 (9.97%)
Haze	Without proposed	97	80 (82.47%)	10 (10.31%)	7 (7.22%)
	With proposed	760	602 (79.21%)	102 (13.41%)	65 (8.55%)
Rainy	Without proposed	650	435 (66.92%)	172 (26.46%)	43 (6.62%)
	With proposed	998	871 (87.27%)	52 (5.21%)	75 (7.52%)

Figure 2.2.2.5 Comparison of non-corrected image and corrected image

Based on the above Figure 2.2.2.3 Xiu-Zhi Chen et al. conclude their findings by stating that in glare condition the proposed system increases the positive detection rate from 85.22% to 89.82% while reducing false classification from 14.71% to 9.97%. In haze, positive detection rise from 82.47% to 79.21% with a notable reduction in false detection which showed the system's effectiveness. Whereas in raining condition the improvement is most outstanding with positive detection increasing from 66.92% to 87.27% and a significant decrease in false detection from 26.46% to 5.21%. The authors mentioned that by integrating right and image

processing techniques and robust deep learning algorithms, this system ensures reliable and accurate vehicle detection without the need for frequently retaining or manual adjustment of parameters.

### 2.2.3 Real-time vehicle classification and tracking using a transfer learning-improved deep learning network (2022)

In the recent study by Bipul Neupane et al. the authors proposed a sophisticated vehicle classification and tracking system by utilizing an advanced deep learning network enhanced by transferring learning techniques to achieve high accuracy and efficiency in real-time applications. For intelligent transport systems (ITSs), where quick and accurate vehicle recognition is crucial this unique system is extremely helpful. The demand for large training datasets, the domain-shift issue, and the integration of a real-time multi-vehicle tracking method with deep learning frameworks are three major obstacles that are frequently encountered in deep learning applications for vehicle identification. The authors concentrated on solving these three main issues. Figure 2.2.3.1 displays the comprehensive diagram of a multi-vehicle tracking algorithm designed for lane-based vehicle count and speed detection.

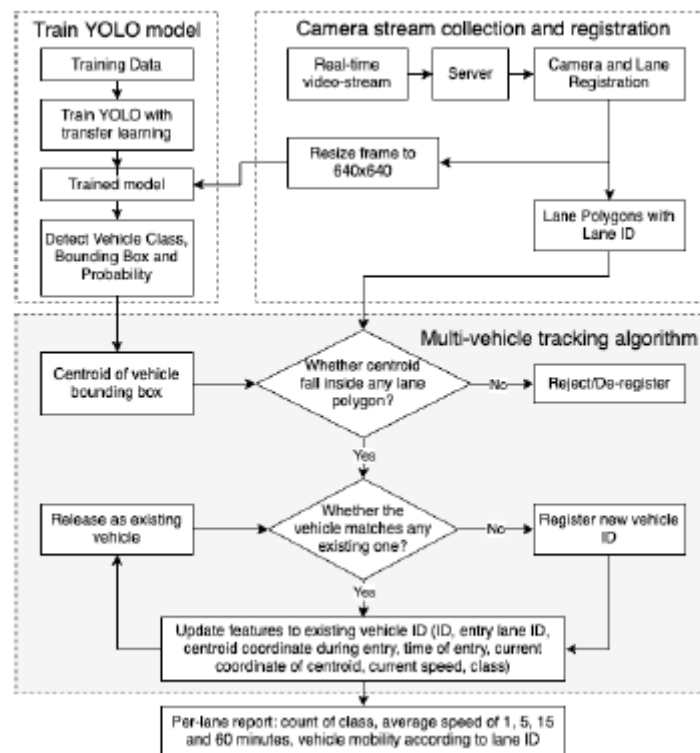


Figure 2.2.3.1 A lane-based count and speed detection technique for multi-vehicle tracking

Bipul Neupane et al. explained that the system they proposed starts with real-time acquisition of video, and the video will then be processed to identify and classify vehicles using YOLO model. In order for better processing, the authors resized the video dimension to standard (640x640) to maintain consistency and optimize computational efficiency. Then the video feed

will be analyses to delineate lane polygons which allow the system to track the vehicle movement across specific lanes. Once the processed video frames enter the multi-vehicle tracking module, each vehicle’s centroid inside the video is calculated based on the bounding box identified by the YOLOv5 model. After that, the system then verifies if the centroid of the vehicle falls inside any registered lane polygon. To ensure accuracy the authors also proposed a checking mechanism to check whether any previously detected vehicle same as the one already in the system. Lastly, the detected vehicle will be then tracked and counted by taking counts of each vehicles entering and existing the camera’s point of view.

Classes	Average Precision (AP)			
	YOLOv3	YOLOv3t	YOLOv5l	YOLOv5s
car	0.764	0.755	0.722	0.745
bus	0.861	0.761	0.762	0.750
taxi	0.660	0.687	0.733	0.673
bike	0.792	0.724	0.817	0.784
pickup	0.757	0.719	0.686	0.711
truck	0.688	0.584	0.654	0.628
trailer	0.447	0.457	0.499	0.492
mAP_50	0.710	0.670	0.695	0.683
Class Loss	0.015	0.028	0.033	0.036
Train time (hours)	7.83	2.83	7.98	2.55
Model Size (MB)	120.59	17.02	91.61	14.09

Figure 2.2.3.2 Performance achieved by various YOLO network

Additionally, in [14] Bipul Neupane et al. also discussed their findings on applying different YOLO models and the precision are shown in the Figure 2.2.3.2 above. In place of using pre-trained weights at the beginning of the training process, the authors' robust training methodology included random data augmentation techniques like scaling, translation, and flipping. The goal of this strategy is to improve the model's ability to generalize across different vehicle detection settings.

In addition, the performance of the YOLO models is assessed according to how well they can manage different lighting situations as well as different object sizes and shapes, all important aspects of real-time vehicle recognition systems. Using a k-means clustering approach, the models' anchor sizes which are critical to the detection process were determined, optimizing their response to various vehicle types and sizes seen in traffic scenes.

Subsequently, these YOLO models are incorporated into a highway real-time video processing system, highlighting their operational effectiveness under dynamic and uncertain environmental settings. This integration demonstrates how the YOLO models are used in real-world traffic management systems, greatly improving vehicle recognition, classification, and tracking precision.

## 2.3 Strength and Weakness

### 2.3.1 Vision-based Vehicle Detection and Counting System Using Deep Learning in Highway Scenes

<b>Strengths</b>	<b>Description</b>
Real-Time	Implements efficient real-time processing for immediate results
Robustness	Utilizes deep learning (YOLOv3) for high accuracy
Adaptive	Enhanced by using ORB algorithm for tracking vehicles

Table 2.3.1.1 Strength and description

<b>Limitation</b>	<b>Description</b>
Over-segmentation	Potential inaccuracy if vehicles are close to each other
Environmental Impact	Performance of detection might degrade due to bad weather condition

Table 2.3.1.2 Limitation and description

### 2.3.2 A Real-Time Vehicle Detection System Under Various Bad Weather Conditions Based on a Deep Learning Model Without Retraining

<b>Strengths</b>	<b>Description</b>
Weather Resilience	Specially designed to maintain high accuracy under various weather conditions.

Enhanced Detection Algorithm	Incorporates YOLOv3 deep learning model for robust detection.
------------------------------	---

Table 2.3.2.1 Strength and description

<b>Limitation</b>	<b>Description</b>
Complex Implementation	Require precise calibration for visibility evaluations, which could be difficult to set up and maintain.
Processing Intensity	System might demand high computational power for real-time and robust detection.

Table 2.3.2.2 Limitation and description

### 2.3.3 Real-time Vehicle Classification and Tracking Using a Transfer Learning-Improved Deep Learning Network

<b>Strengths</b>	<b>Description</b>
Transfer Learning	Utilize transfer learning to enhance YOLO model, improving accuracy and efficiency.
Comprehensive Tracking	Employs multi-vehicle tracking algorithm to track heavy traffic

Table 2.3.3.1 Strength and description

<b>Limitation</b>	<b>Description</b>
Dependency on Data Quality	Highly relied on quality and variability of training data

Resource Intensive	Require substantial computational resources of training devices.
--------------------	--

Table 2.3.3.2 Limitation and description

## 2.4 Summary

After reviewing existing systems in the market, each of the proposed methods by the authors are yet to perfect and still has room for improvement such as handling dynamic backgrounds, over-segmentation and sensitivity to environmental conditions. Other than that, weak formation of boundaries and shadow related errors could also impact the system's accuracy. This project aims to further develop the fundamental image-based approach used in previous work to produce an integrated vehicle detection and counting system. The objective is to detect vehicles, track them and count them in moving traffic.



# Chapter 3

## System Methodology/Approach

This project process was mainly categorized into two phases throughout the entire development, which were the preprocessing stage and the postprocessing stage. In preprocessing stage, the primary focus was on filtering and training the data to enhance the efficiency and accuracy of the postprocessing stage later. Whereas in the postprocessing stage, several algorithms are applied to detect, track and count all the moving vehicles in the traffic. Machine learning, specifically supervised learning is applied in the preprocessing stage to train two separate model one is for vehicle model detection, and another is for weather classification. Lastly, in the postprocessing stage transfer learning is applied with the implementation of detection algorithm to detect and track moving vehicles in the traffic. Below diagram 3.0 is the overview diagram of the system methodology.

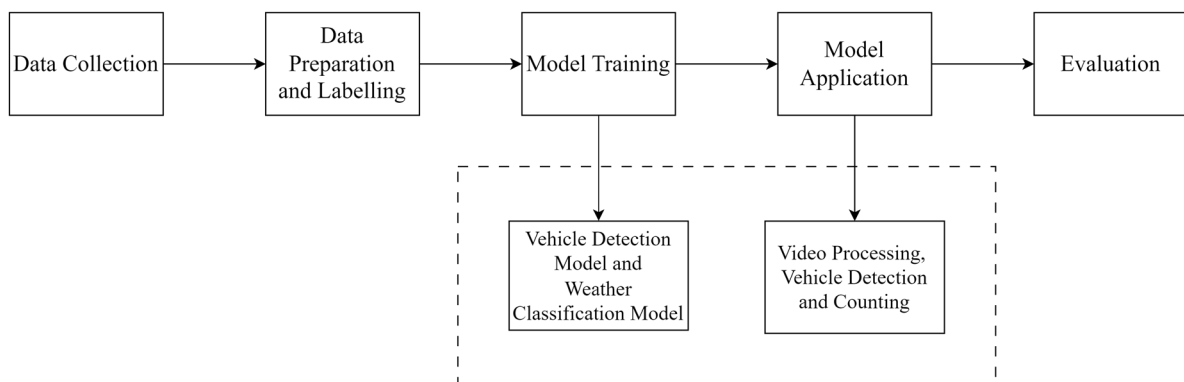


Figure 3.0 Overview of system methodology

### 3.1 Data Acquisition and Preparation

The input data that are used in this project are mainly two types which are image and video. First, the dataset of images and videos is gathered across multiple online resources, including Kaggle and GitHub. The image datasets will be used for training purposes, while the video will be used for postprocessing, where detection and counting of moving vehicles in the moving traffic. While sourcing for data, only two categories of data that were mainly focused on, sunny weather and rainy weather, and the quality of the dataset are in the acceptable range where too

low quality of data will be filtered out, such as images with few kilobytes are normally images that are overly blurred. Other than that, some of the processed images are filtered out. For example, images that are converted to grayscale are removed, and only the original images in RGB remain. Moreover, the datasets are then undergoing categorization where images are copied into two sets, one set is used for vehicle model training and another set is used for weather classification training. For vehicle model training, the images are separated into two folders, which are the 'train' and 'valid' folders. Meanwhile, for weather classification, images are differentiated into 'train' and 'valid' folders, with further categorization into 'rainy' and 'sunny' subfolders. This organized approach guarantees that the training and validation processes are correctly supported, allowing for the construction of strong models capable of accurately detecting vehicles and analyzing weather conditions.

### **3.2 Data Preprocessing**

Before stepping into the training process, the image dataset undergoes a preprocessing process, which is labelling to tailor the training of the vehicle detection model using a supervised learning approach. Each of the images were labeled with bounding boxes indicating the vehicle class and normalized coordinates (center\_x, center\_y, width, height) and stored in label files. To ensure the model scalability across different resolutions, these coordinates were normalized against the image dimensions to increase the object detection accuracy. Besides that, the dataset was also divided into training and validation sets to facilitate the training process and stated inside the 'data.yaml' file, together with the number of classes involved in the model training. While the vehicle detection model focuses on detailed annotations, the preprocessing for the weather classification model was comparatively straightforward, where only basic image transformation and normalization are needed to meet the model's input needs. For example, resizing the image to a fixed dimension and standardizing the pixel values in the image using mean and standard deviation for each color channel.

### **3.3 Machine Learning Models**

#### 3.3.1 Vehicle Detection Model

You only look once (YOLO) is an object detection system that use convolutional neural network to detect object from the input background. While conventional detection systems apply the model to different areas of an image, YOLO treats detection as a singular regression

issue. It predicts bounding boxes and class probabilities for these boxes over the whole picture in a single evaluation. For the training of the vehicle detection model, transfer learning was applied using the YOLOv5m model to achieve high performance in real-time object detection tasks, which is crucial for dynamic environments, typically in vehicle detection in moving traffic. This medium variant of the YOLOv5m is particularly suited for handling the medium size of the image datasets. On top of that, the reason for selecting this model is the capability of the model to perform multi-scale predictions in detecting vehicles across various sizes and distances in road traffic from the background. Throughout the project, this pre-trained model weight was fine-tuned on a custom dataset specifically compiled and annotated for this system. The training process involved adjusting hyperparameters like the batch sizes and epochs to maximize the model's precision and recall, which is important for minimizing false vehicle detections and enhancing robust accuracy in vehicle detection in a later stage.

### 3.3.2 Weather Classification Model

VGG16 is a deep convolutional neural network that developed by the visual graphics group from Oxford that can perform image recognition and classification tasks. For weather classification, the VGG16 model was employed as this model was originally designed to classify among 1000 categories of data by extracting important features from the input images. The reason for selecting this model in this project is to perform a binary classification task on the dataset, focusing on identifying rainy and sunny weather conditions. With a good structure and distinctly labelled as sunny and rainy, this model can distinguish the visual patterns with each weather condition effectively. By utilizing a pre-trained model, the weather classification training process is able to reduce the extensive computational resources and save time. In the classification process, the model's training parameters were tuned to increase the model's accuracy in distinguishing weather conditions in the dataset. The fine-tuning involves modifying the network's final layers and training settings, such as learning rate, to refine the model's performance and accuracy toward weather classification.

## **3.4 Model Application and System Integration**

### 3.4.1 Video Enhancement Using Weather Classification Model

With the weather classification training model output from the training in the previous stage, the video dataset will undergo processing where the video quality will be enhanced based on the weather conditions to establish optimized conditions for subsequent vehicle detection. The

weather classification model, based on a modified VGG16 network, evaluates the video frame to determine whether the weather will be ‘sunny’ or ‘rainy’. After that, the model employs a series of transformations to standardize the input frames, including resizing the pixels, normalizing the pixel values and converting the frame from BGR (blue, green and red) uncommon variant to RGB (red, green and blue). Lastly, the system dynamically adjusts the video’s contrast and brightness using gamma correction for sunny weather or histogram equalization for rainy conditions with the stated parameters, improving the detection algorithm's accuracy.

### 3.4.2 Vehicle Detection in Processed Video

In the post-processing stage, the trained vehicle model weight from the previous vehicle detection model and the processed video is input into the YOLOv5 detection model to detect and locate vehicles inside each enhanced frame from the background. The utilization of YOLOv5 ensures that the detection process is highly accurate under the dynamic nature of the road traffic structure and various weather conditions. The detection output includes coordinates representing each of the detected vehicles. These coordinates are essential for tracking vehicle movement across all the frames in the subsequent processes.

### 3.4.3 Vehicle Tracking and Counting with Deep Sort

Whereas in order to achieve high accuracy in tracking the detected vehicle in the moving traffic, this system utilized the Deep Sort algorithm for its robustness and precise object tracking. With the help of transfer learning and a pre-trained object tracking checkpoint, this algorithm can adapt and track vehicles in the dynamic road traffic. This pre-trained model offers a strong basis for tracking all the vehicles detected by the YOLOv5. The motivation behind implementing transfer learning instead of training the object tracking from scratch is that this Deep Sort algorithm is one of the famous objects tracking algorithms available and widely used by many object tracking systems, with a high accuracy guarantee. The tracking process begins with taking the coordinates and confidence scores from the YOLOv5 detections and then fed into Deep Sort to keep track of the moving vehicles. This tracking algorithm is crucial to maintaining consistent, unique vehicle identities across video frames, and with the tracking data, vehicle counts and traffic density can be generated and ease the traffic flow analysis. Finally, for each of the uniquely tracked vehicles, the system will take into account one vehicle counted in the moving traffic.

### **3.5 System Evaluation**

Once all the video frames are processed, each of the vehicles within each frame is uniquely counted. The accuracy of the system is then compared by changing the input video from processed video to non-processed video, which is the original video. This is to showcase that the enhancing of video with the trained weather classification did assist in increasing the accuracy of the detection and counting system. Besides that, another detection algorithm, Faster-RCNN, is also implemented as a benchmark testing to compare against the YOLOv5 in terms of the processing speed and the accuracy, providing a comprehensive assessment of the overall system's performance.

# Chapter 4

## System Design

### 4.1 System Block Diagram

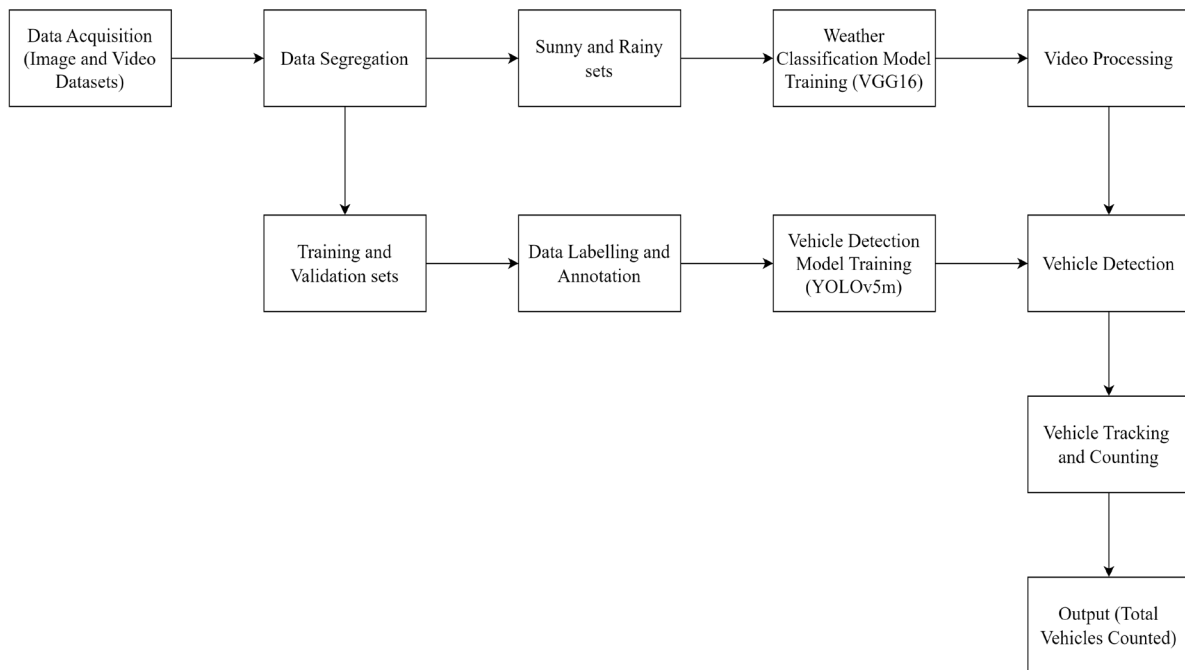


Figure 4.1.1 System Block Diagram of Vehicle Detection and Counting System

The figure above represents the comprehensive system block diagram for the vehicle detection and counting system. The system first begins with data acquisition, a dataset of images and videos is downloaded from several online resources and particularly for rainy and sunny weather conditions. Then the dataset undergoes a segregation process where the images are duplicated and separated into mainly two set, one is used for vehicle detection model training that contains training and validation set in a subfolder and another set is for weather classification model training that includes sunny and rainy sets in a subfolder. The sunny and rainy images are then trained using VGG16 for binary weather classification model training. The output of the trained weather classification is used to process the original video to enhance the video frame's feature according to the weather conditions. Whereas for the training and validation, image sets will be labelled in a text file and placed into each train and valid subfolder and used for vehicle detection model training using YOLOv5m. The labelling and annotation involve annotating the images with bounding boxes that specify the location and class of

vehicles within each image. The trained weight will then be used for vehicle detection to perform detection on the enhanced video. After that, each of the detected vehicles will be tracked across all the video frames and then counted. Lastly, the total vehicles counted will be displayed in the output upon finishing the execution of the system.

## **4.2 System Components Specifications**

### **4.2.1 Data Acquisition**

The datasets are downloaded from several online resources, including Kaggle and GitHub, and only videos and images that are under sunny and rainy weather will be taken. In contrast, others will be filtered out as this system mainly focuses on two weather conditions that are most common to occur. Besides that, images and videos that are too low quality, for example blurry images or videos, will be removed. This is because training low-quality datasets will affect the overall system's accuracy as too many noises and unnecessary features are included. The total dataset comprises 375 JPG images and 4 MP4 videos for both sunny and rainy weather conditions.

### **4.2.2 Data Segregation**

For data segregation, the datasets are organized into structured sets that optimize the training process in the later stage. First, the images are duplicated and categorized into two folders, which are 'Model Dataset' and 'Weather Dataset'. The model dataset folder contains two sub folders named 'Images' and 'Labels', and inside each of the sub folder have two child folder named 'Train' and 'Valid'. Whereas for the weather dataset it contains two sub folders named 'Train' and 'Valid', and inside each of the sub folders the child folders are categorized as 'Sunny' and 'Rainy'. The model dataset will be used for the vehicle detection model, while the weather dataset will be used for weather classification. This segregation is important as it provides systematic access to the directory structure and easier to manage.

### **4.2.3 Data Labelling and Annotation**

Before the images undergo the model detection training, each of the validation and training images is labeled in detail with the bounding boxes and labels and stored in text file format, which defines the vehicle locations in the image and also the type for the detection model. Annotation of the label text file includes the class id as a unique identifier for the object type, center x, and y are the normalized coordinates of the bounding box center. At the same time,

width and height are the values that indicate the dimensions of the bounding box relative to the image size. These detailed annotations provide essential training data for the training for the YOLOv5 model to recognize and locate vehicles accurately among the images.

#### 4.2.4 Vehicle Detection Model Training

Once all the label and image datasets are ready, another yaml file is configured before going into the model training process. This yaml file states the number of classes and the directory of all the training and validation images located. After that, the yaml file was loaded into the YOLOv5m, which is the medium weight of the YOLOv5 model for training. The setup of the model detection training included the use of computing resources and detailed hyperparameter settings such as image size, batch size and epochs. The image size is set to 640x640 pixels, the batch size is set at 16 for optimization for available GPU memory without compromising the training speed, and the model undergoes 20 training epochs to improve the accuracy of vehicle detection in the dataset. The output of the model training weight will be saved as 'best.pt', which will be used in the later vehicle detection stage.

#### 4.2.5 Weather Classification Model Training

The weather classification model training employs the VGG16 architecture, which is used for binary classification to distinguish sunny and rainy weather conditions. This pre-trained model is first loaded into the system and fine-tuned according to the needs. The parameters include image size pixel, batch size, learning rate, momentum, and epochs. The images are resized to 224x224 pixels, then converted to tensors and normalized to fit the VGG16 training rule. Then, the batch size is set to be 32 for each of the trains and valid set, with a learning rate of 0.001 and momentum of 0.9. Due to the limitation of the computational resources, the epochs are set to 10 for optimal performance. Lastly, the trained weather classification model is saved as 'weather\_classification.pth', which will be used in the video processing stage.

#### 4.2.6 Video Processing

First, the trained weather model is loaded into the system together with the original video, which is processed frame by frame to determine the prevailing weather conditions. For sunny weather, the video frame will be enhanced by applying gamma correction to 0.75 to increase the brightness, whereas for rainy weather, histogram equalization is performed across all color channels to enhance the contrast for better visibility. These enhancements are applied using the



OpenCV library, and the output of the processed frame is stored as a processed video. This process is crucial to ensure the video quality is optimized for the detection process and improve the overall system's accuracy in different weather conditions.

#### 4.2.7 Vehicle Detection

Once the processed video is ready, the YOLOv5 model is loaded with the training weight 'best.pt' for the detection process. Each of the video frames will be extracted from the enhanced video and evaluated, and bounding rectangle boxes will be outputted around the detected vehicles with the confidence scores stated. The detection is filtered by a confidence threshold of 0.3, where only confidence scores above this threshold are considered valid detection to reduce false positives and higher accuracy. Lastly, the bounding box coordinates are configured in the format of center x, center y, width, and height, and the values of the bounding box are calculated based on the model's predictions.

#### 4.2.8 Vehicle Tracking and Counting

In the tracking process, this system employs the Deep Sort algorithm to track each of the detected vehicle across the entire video frames. From the detection data provided by the YOLOv5 model, the bounding boxes that represent the coordinates of the vehicle's position and size are extracted and used as input to maintain consistent tracking of each vehicle frame by frame. Deep Sort uses data association technique to enhance the tracking accuracy of the object which is vehicle in this case. For each of the tracked vehicle, a unique ID will be then assigned and used to count each vehicle appear in the entire video frames. Finally, once all the frames have been processed, the system will calculate and output the total number of unique vehicles counted based on the unique IDs.

# Chapter 5

## System Implementation

### 5.1 Hardware and Software Setup

The hardware that used in this system development is a laptop with the minimum requirements of 8 GB RAM is required to produce the optimal performance when training the dataset and executing the deep learning algorithm.

Description	Specifications
Model	Aorus 15P KC
Processor	Intel Core i7-10870H
Operating System	Windows 10
Graphic	NVIDIA GeForce RTX 3060 GPU 6GB GDDR6
Memory	16GB DDR4 RAM
Storage	512GB SSD

Table 5.1.1 Specifications of laptop

Dataset sources: <https://www.kaggle.com>, <https://github.com>

Deep Sort algorithm repository: [https://github.com/ZQPei/deep\\_sort\\_pytorch](https://github.com/ZQPei/deep_sort_pytorch)

Platform: Microsoft Visual Studio 2019

Language: Python 3.9

### 5.2 Setting and Configuration

Before starting to develop the vehicle detection and counting system there are several setting and configuration needed. These are the libraries packages that needed to be installed into the system OpenCV, PyTorch, YOLOv5, NumPy and Deep Sort algorithm is clone from the GitHub for transfer learning with the command 'git clone repository link'. Each of the libraries were installed in the python environment with the command 'pip install'.

OpenCV – Used for image processing and video manipulation process.

PyTorch – Used for the backbone of all neural network operations, including loading the and running the detection and classification models.

YOLOv5 – Primary object detection algorithm used for detecting vehicles.

Bachelor of Computer Science (Honours)

Faculty of Information and Communication Technology (Kampar Campus), UTAR

NumPy – Used for supporting numerical operations.

Deep Sort – Algorithm used for tracking all the detected vehicles.

For training vehicle detection model, a yaml configuration file need to be setup by stating in the following format.

train: directory path to train data

(C:\Users\kenny\source\repos\VehicleDetectionAndCounting\Model dataset\images\train)

val: directory path to validation data

(C:\Users\kenny\source\repos\VehicleDetectionAndCounting\Model dataset\images\valid)

nc: number of classes (1)

names: ['Vehicle']

## 5.3 Code Implementation and Explanation

### 5.3.1 Vehicle Detection Model Training

Modeltraining.py

```
# Import all the necessary libraries
import os
import torch
import yaml
from yolov5 import train

# Function to ensure the required directories for training and validation exist
def setup_directories(data_yaml_path):
    # Open and read the YAML configuration file that contains the directory paths
    with open('C:/Users/kenny/source/repos/VehicleDetectionAndCounting/data.yaml', 'r') as
file:
        data_config = yaml.safe_load(file)
        assert os.path.isdir(data_config['train']), f'Training directory not found:
{data_config['train']}'
        assert os.path.isdir(data_config['val']), f'Validation directory not found:
{data_config['val']}'
        print("Dataset directories are set up correctly.")
```

```

# Function to configure and run the model training
def train_model(data_yaml_path):
    if torch.cuda.is_available():
        device = 'cuda'
    else:
        device = 'cpu'
    print("CUDA is not available, training will proceed on CPU but may be slow.")

# Start the training process
print("Starting training process...")
train.run(
    data=data_yaml_path,
    imgsz=640,
    batch_size=16,
    epochs=20,
    weights='yolov5m.pt',
    device=device,

project='C:/Users/kenny/source/repos/VehicleDetectionAndCounting/Vehicle_Model_Training',
    name='Vehicle_Model',
    exist_ok=False
)

# Main execution block to set up directories and start model training
if __name__ == "__main__":
    data_yaml_path =
'C:/Users/kenny/source/repos/VehicleDetectionAndCounting/data.yaml'
    setup_directories(data_yaml_path)
    train_model(data_yaml_path)

```

First, import all the necessary libraries and define all the functions needed. The function 'setup\_directories' opens the yaml configuration file and reads the path inside to check whether the stated directory inside the file is correct and accessible for both training and validation datasets. Next, the 'train\_model' is the function to run the vehicle detection model training process. This function begins with checking for the availability of Compute Unified Device Architecture (CUDA) and performing GPU acceleration if possible else the CPU is used for the processing which is comparatively slower. Then the function initiates the training with specific parameters such as imgsz for image size, batch size is the number of images processed simultaneously, epochs refer to one complete pass through the entire training dataset, weight is the pre-trained model's weight. The 'train.run' will run the imported YOLOv5 library with the parameters set and, start the training process and save the output into the stated 'project' directory. Inside the 'Vehicle\_Model\_Training' the 'best.pt' weight will be used in later stage of detection process. For the main execution the path of yaml configuration file was set.

### 5.3.2 Weather Classification Model Training

Weathertraining.py

```
import torch
import torchvision
from torchvision import datasets, transforms, models
from torch.utils.data import DataLoader
from torch import nn, optim
import numpy as np

# Transformations applied on each image
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
# Load the image datasets for train and valid
train_dataset = datasets.ImageFolder('C:/Users/kenny/source/repos/VehicleDetectionAndCounting/Weather dataset/Train', transform=transform)
```

```

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)

valid_dataset =
datasets.ImageFolder('C:/Users/kenny/source/repos/VehicleDetectionAndCounting/Weather dataset/Valid', transform=transform)
valid_loader = DataLoader(valid_dataset, batch_size=32, shuffle=True)

# Load a pre-trained model (VGG16) and modify it for binary classification
model = models.vgg16(pretrained=True)
model.classifier[6] = nn.Linear(model.classifier[6].in_features, 2)

# Define loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)

# Function to compute accuracy
def compute_accuracy(outputs, labels):
    _, predicted = torch.max(outputs.data, 1)
    total = labels.size(0)
    correct = (predicted == labels).sum().item()
    return 100 * correct / total

# Track best validation accuracy
best_accuracy = 0

# Train the model
model.train()
for epoch in range(10):
    running_loss = 0.0
    for i, data in enumerate(train_loader, 0):
        inputs, labels = data
        optimizer.zero_grad()
        outputs = model(inputs)

```

```

loss = criterion(outputs, labels)
loss.backward()
optimizer.step()
running_loss += loss.item()
if i % 10 == 9:
    print(f'Epoch: {epoch + 1}, Batch: {i + 1}, Loss: {running_loss / 10:.4f}')
    running_loss = 0.0

# Validation loss
model.eval()
with torch.no_grad():
    valid_loss = 0.0
    valid_accuracy = 0.0
    for inputs, labels in valid_loader:
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        valid_loss += loss.item()
        valid_accuracy += compute_accuracy(outputs, labels)

valid_loss /= len(valid_loader)
valid_accuracy /= len(valid_loader)
print(f'Epoch: {epoch + 1}, Validation Loss: {valid_loss:.4f}, Validation Accuracy:
{valid_accuracy:.2f}%',)
if valid_accuracy > best_accuracy:
    best_accuracy = valid_accuracy
    torch.save(model.state_dict(), 'Weather_Classification.pth')
    print("Saved Best Model")
    print("Finished model training")

```

The weather classification training process begins by applying transformation to each of the images in the dataset, resizing to 224x224 pixels, converting to tensor format, and normalizing the colors with the predefined mean and standard deviation. The dataset's directory is loaded using PyTorch's 'ImageFolder', and the batch size is set to 32. Then, a pre-trained VGG16

model is loaded, and the classifier is from 1000 classes to only 2 classes, which are for rainy and sunny weather. The model is set to be trained using the Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.001 and momentum of 0.9. The cross-entropy loss function is used to visualize the training model performance. For better visualization during training, the validation loss and accuracy will be calculated and output at the end of each epoch of training. The best possible model configuration is maintained by tracking the highest validation accuracy gained and saving the model state whenever a new best is discovered, and the file is saved as 'Weather\_Classification.pth'.

### 5.3.3 Video Processing

Videoprocessing.py

```
import cv2
import numpy as np
from weather_model import WeatherClassificationModel
import torch
import warnings

warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings("ignore", category=UserWarning)

def load_weather_model():
    """Load the weather classification model."""
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    model_path = 'C:/Users/kenny/source/repos/VehicleDetectionAndCounting/Weather_Classification.pth'
    weather_model = WeatherClassificationModel(model_path, device=device)
    return weather_model

def enhance_image(image, weather_condition):
    """Enhance the image based on the weather condition."""
    if weather_condition == 'sunny':
        gamma = 0.75
```



```

    look_up_table = np.array([((i / 255.0) ** gamma) * 255 for i in np.arange(0,
256)]).astype("uint8")
    image = cv2.LUT(image, look_up_table)
    elif weather_condition == 'rainy':
        for c in range(0, 3):
            image[:, :, c] = cv2.equalizeHist(image[:, :, c])
    return image

def preprocess_video(video_path, weather_model):
    cap = cv2.VideoCapture(video_path)
    if not cap.isOpened():
        print("Error: Could not open video.")
        return

    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    fps = cap.get(cv2.CAP_PROP_FPS)
    frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

    out = cv2.VideoWriter(
'C:/Users/kenny/source/repos/VehicleDetectionAndCounting/Video/Sunny_Video_Process
ed.mp4',
    cv2.VideoWriter_fourcc(*'mp4v'),
    fps,
    (frame_width, frame_height)
    )

    frame_number = 0
    while True:
        ret, frame = cap.read()
        if not ret:
            break

```

```

weather_condition = weather_model.predict(frame)
processed_frame = enhance_image(frame.copy(), weather_condition)

out.write(processed_frame)
frame_number += 1
progress = (frame_number / total_frames) * 100
print(f'Processing video: {progress:.2f}% complete', end='\r')

if cv2.waitKey(1) != -1:
    break

cap.release()
out.release()
print("\nVideo processing complete.")
print(f'Weather Detected: {weather_condition}')

if __name__ == "__main__":
    weather_model = load_weather_model()
    video_path = 'C:/Users/kenny/source/repos/VehicleDetectionAndCounting/Video/Sunny_Video.mp4'
    preprocess_video(video_path, weather_model)

```

The video processing first begins with loading the input video from the directory stated. After that load the trained weather classification model in the 'load\_weather\_model' function. Then the 'enhance\_image' function will apply gamma correction to 0.75 to increase the brightness if the weather of the video frame detected is sunny else histogram equalization will be applied to the frame for rainy weather. The main function 'preprocess\_video' is to open the video file and count the total frame. The function also reads the frame sequentially and apply the enhancement on each of the frame based on the weather condition and then output into stated directory using the 'VideoWriter' from the OpenCV library. The output video will have an extra 'processed' in the video file name. For the main execution the path of the original video file was set.

### 5.3.4 Vehicle Detection and Counting

VehicleDetectionAndCounting.py

```
import cv2
import numpy as np
import torch
from yolov5 import YOLOv5
from deep_sort_pytorch.deep_sort import DeepSort
import warnings

warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings("ignore", category=UserWarning)

def load_models():
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    vehicle_model_path = 'C:/Users/kenny/source/repos/VehicleDetectionAndCounting/Vehicle_Model_Training/Vehicle_Model/weights/best.pt'
    vehicle_model = YOLOv5(vehicle_model_path, device=device)
    deep_sort_model_path = 'C:/Users/kenny/source/repos/VehicleDetectionAndCounting/deep_sort_pytorch/deep_sort/deep/checkpoint/ckpt.t7'
    deep_sort = DeepSort(deep_sort_model_path)
    return vehicle_model, deep_sort

def detect_vehicles(frame, model):
    results = model.predict(frame)
    detections = results.xyxy[0].numpy()
    bbox_xywh = np.array([(x1 + x2) / 2, (y1 + y2) / 2, x2 - x1, y2 - y1] for x1, y1, x2, y2, conf, cls_id in detections if conf > 0.3 and int(cls_id) == 0])
    confidences = np.array([conf for x1, y1, x2, y2, conf, cls_id in detections if conf > 0.3 and int(cls_id) == 0])
    return bbox_xywh, confidences
```

```

def process_video(video_path, vehicle_model, deep_sort):
    cap = cv2.VideoCapture(video_path)
    if not cap.isOpened():
        print("Error: Could not open video.")
        return

    frame_count = 0
    frame_rate = cap.get(cv2.CAP_PROP_FPS)
    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    uniqueID_tracks = set()
    print(f"Total frames: {total_frames}")

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        frame_count += 1
        bbox_xywh, confidences = detect_vehicles(frame, vehicle_model)

        if len(bbox_xywh) > 0:
            outputs = deep_sort.update(bbox_xywh, confidences, frame)
            detection_count = len(outputs)

            for output in outputs:
                x1, y1, x2, y2, track_id = output
                uniqueID_tracks.add(track_id)
                cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
                cv2.putText(frame, f'ID: {track_id}', (x1, y1 - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 0), 2)
            else:
                detection_count = 0

```

```

text = f'Detected: {detection_count}, Counted: {len(uniqueID_tracks)}"
cv2.putText(frame, text, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 2)

print(f"Processing frame {frame_count}/{total_frames}... Detections:
{detection_count}")

cv2.imshow('Vehicle Detection', frame)
if cv2.waitKey(1) != -1:
    break

cap.release()
cv2.destroyAllWindows()
vehicle_count = len(uniqueID_tracks)
print(f"Total unique vehicles counted in the video: {vehicle_count}")

if __name__ == "__main__":
    vehicle_model, deep_sort = load_models()
    video_path =
'C:/Users/kenny/source/repos/VehicleDetectionAndCounting/Video/Sunny_Video2_Proces
sed.mp4'
    process_video(video_path, vehicle_model, deep_sort)

```

Initially, in the 'load\_models' function, the trained model detection weight 'best.pt' is loaded together with the 'ckpt.t7', which is the pre-trained object tracking checkpoint from the deep sort algorithm. Then in the 'detect\_vehicles' function first each frame from the video is fed into the YOLOv5 detection model. The bounding box coordinates format of x1, y1 and x2, y2 are the coordinates of the top left and bottom right corners of the bounding box. For tracking purpose, the formula center  $x = (x1+x2) / 2$ , center  $y = (y1+y2) / 2$ , width =  $x2 - x1$  and height =  $y2 - y1$  are applied to fit the tracking purpose. The confidence score is set to be more than 0.3 for better detection purposes. After that, 'process\_video' function will control the video processing loop of each video frame. The detected vehicle with bounding boxes together with their confidence score are used as inputs of the deep sort tracking. Deep sort will track the vehicles detected across all the frames. For counting, each of the detected and tracked vehicle

will be assign a unique identifier. The tracker updates these unique identifiers according to the movement and the appearance of the vehicles. The 'len()' function is used to retrieve all of the unique identifiers being assigned to the tracked vehicles and output as total number of vehicles counted. In the output window, vehicle detected states in the current frame, how many moving vehicles are being detected, whereas counted indicates how many unique vehicles have been counted so far since frame 1.

## 5.4 System Operation

The initial step of this system development is to prepare the datasets for vehicle detection model training and weather classification training.

### 5.4.1 Vehicle Detection Model Training Dataset

The screenshot shows a file explorer interface with four panels, each displaying a table of files and folders. The panels represent different levels of the directory structure:

- Panel 1:** Shows the root directory 'Model dataset' containing folders 'images' and 'labels'.
- Panel 2:** Shows the 'images' folder containing sub-folders 'train' and 'valid'.
- Panel 3:** Shows the 'labels' folder containing sub-folders 'train' and 'valid', along with cache files 'train.cache' and 'valid.cache'.
- Panel 4:** Shows the 'train' sub-folder within 'labels', containing four text documents (MP4 files).
- Panel 5:** Shows the 'valid' sub-folder within 'labels', containing four text documents (MP4 files).

Name	Date modified	Type	Size
images	29/8/2024 10:55 PM	File folder	
labels	30/8/2024 11:38 PM	File folder	
train	30/8/2024 9:29 PM	File folder	
valid	30/8/2024 9:29 PM	File folder	
train	30/8/2024 9:28 PM	File folder	
valid	30/8/2024 9:27 PM	File folder	
train.cache	30/8/2024 11:37 PM	CACHE File	159 KB
valid.cache	30/8/2024 11:38 PM	CACHE File	27 KB
3_mp4-0_jpg.rf.3572a3c47999df82306623...	10/12/2023 8:14 AM	Text Document	1 KB
3_mp4-0_jpg.rf.486762d73755e270c07dd8...	10/12/2023 8:14 AM	Text Document	1 KB
3_mp4-1_jpg.rf.e44963f8da0b658a9c42e5...	10/12/2023 8:14 AM	Text Document	1 KB
3_mp4-3_jpg.rf.2b61a5c3545c950fd08969...	10/12/2023 8:14 AM	Text Document	1 KB
3_mp4-2_jpg.rf.fb3ec30812baf2bbaca871...	10/12/2023 8:14 AM	Text Document	1 KB
3_mp4-8_jpg.rf.4a784fedfdd38503261437...	10/12/2023 8:14 AM	Text Document	1 KB
3_mp4-14_jpg.rf.6597c5732e0d554f1f07e...	10/12/2023 8:14 AM	Text Document	1 KB
3_mp4-17_jpg.rf.fdfd6a4ddf7c5f516a940b...	10/12/2023 8:14 AM	Text Document	1 KB

Figure 5.4.1.1 Vehicle detection model training folder structure

Above Figure 5.4.1.1 is the overview folder structure for the vehicle detection model training. The images folder contains the images used for training and validation. While the labels folder contains annotation text file for the bounding box coordinates and class ID and placed in two subfolders train and valid respectively.

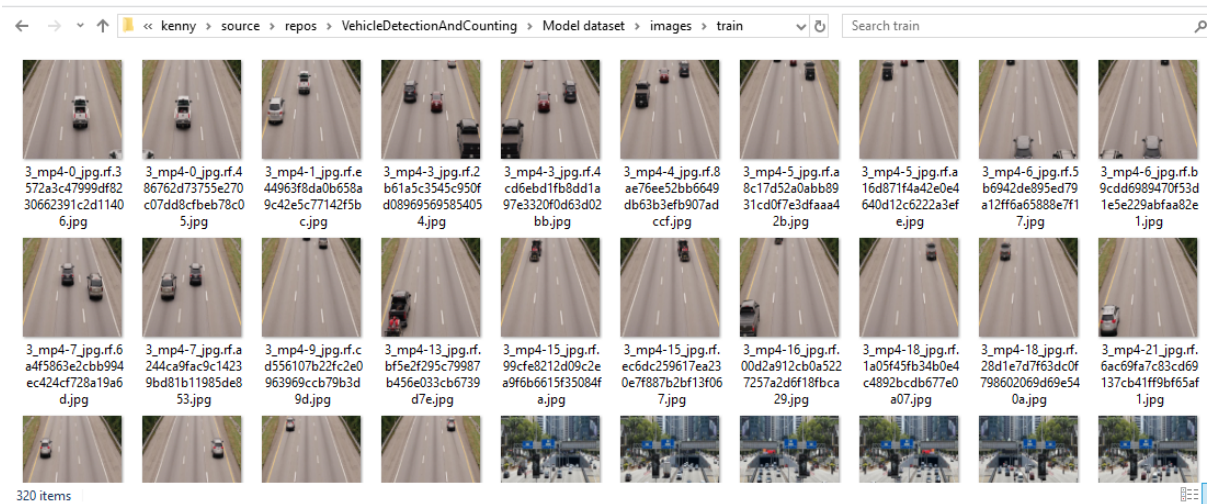


Figure 5.4.1.2 Vehicle model training set images

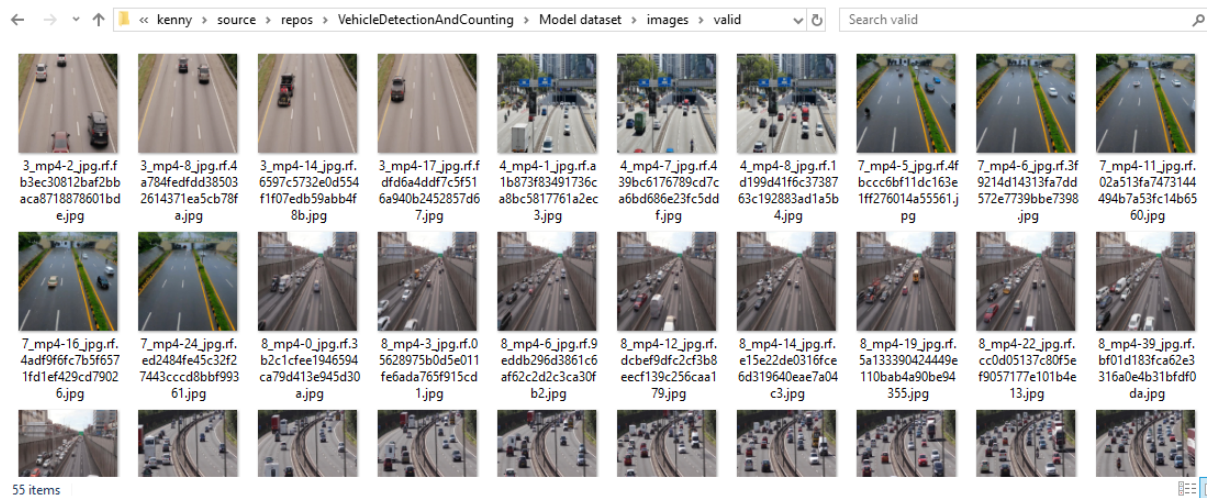


Figure 5.4.1.3 Vehicle model validation set images

Figure 5.4.1.2 contain 320 images used for training the vehicle detection model, and Figure 5.4.1.3 contain 55 images that are used for validating the model during the training phase.

## 5.4.2 Weather Classification Model Training Dataset

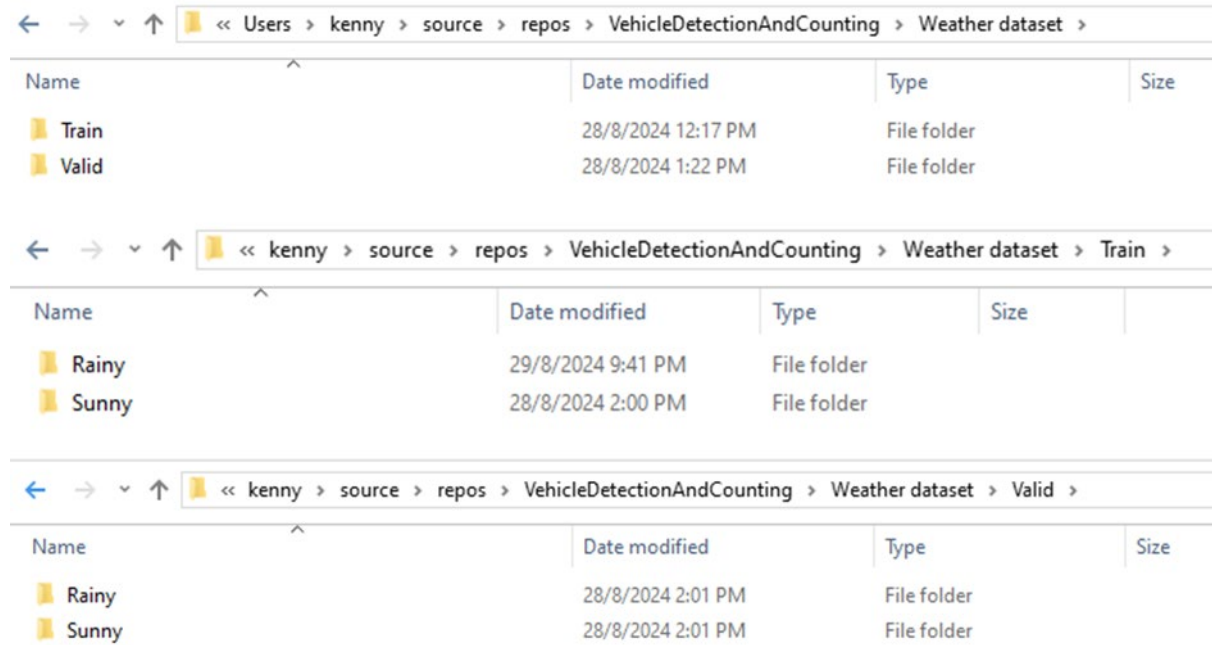


Figure 5.4.2.1 Weather classification model training folder structure

Above Figure 5.4.2.1 is the overview folder structure for the weather classification model training. The train and valid folder each contain the rainy and sunny subfolders respectively.

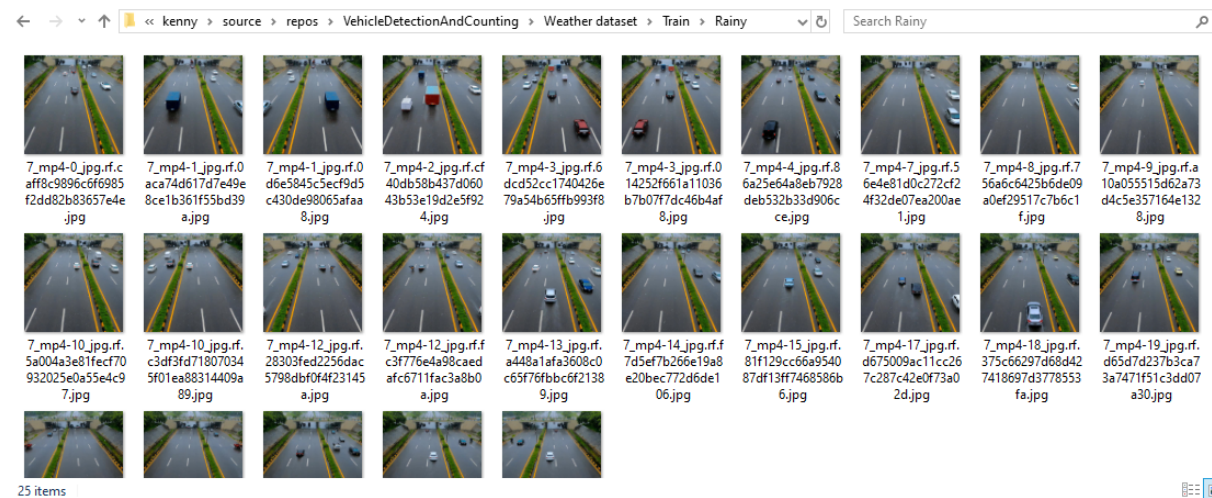


Figure 5.4.2.2 Rainy training set images



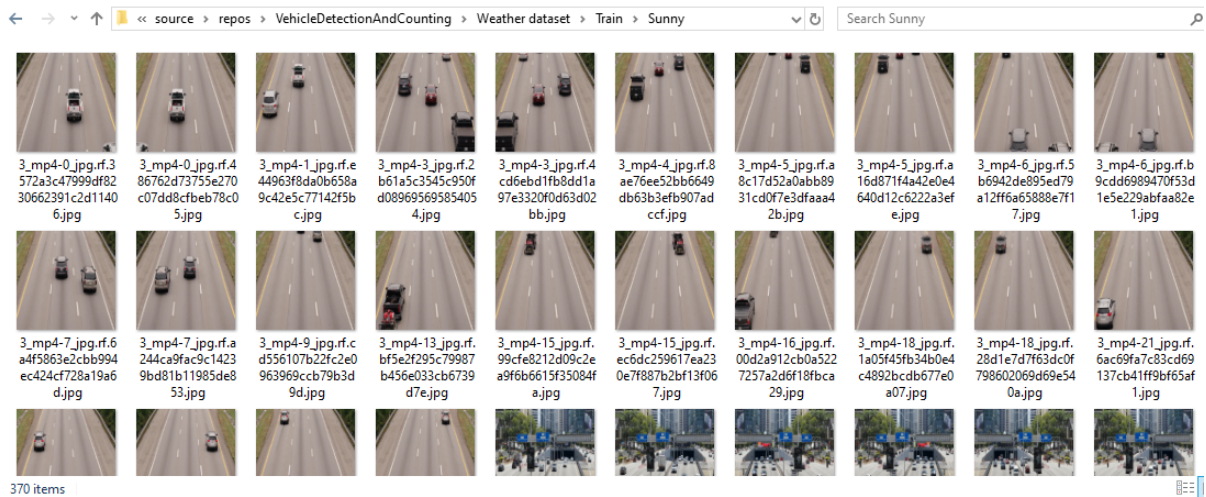


Figure 5.4.2.3 Sunny training set images

Figure 5.4.2.2 contains 25 images that are rainy, and Figure 5.4.2.3 contains 370 images that are sunny. Both will be used for training weather classification.

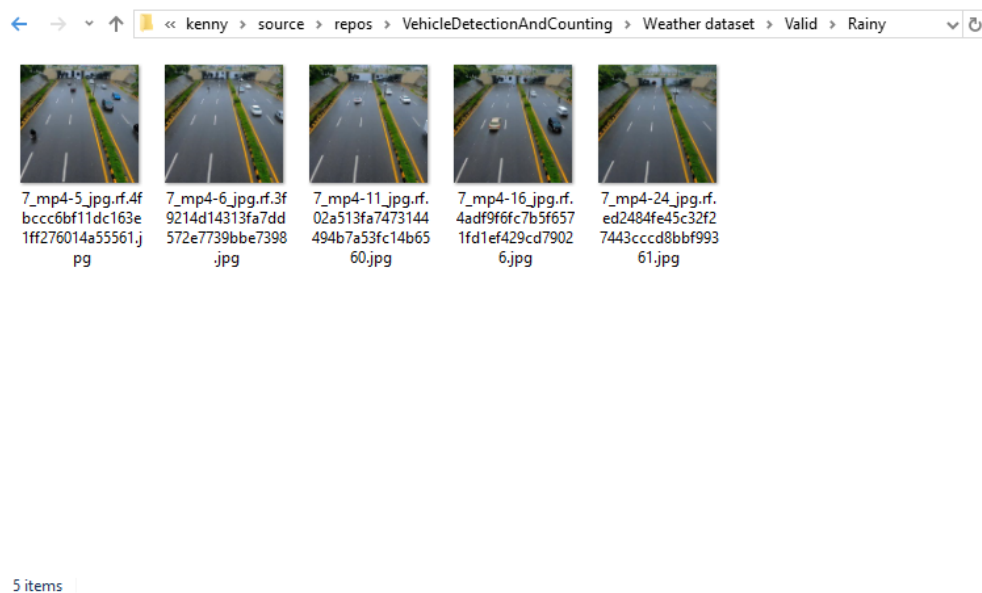


Figure 5.4.2.4 Rainy validation set images

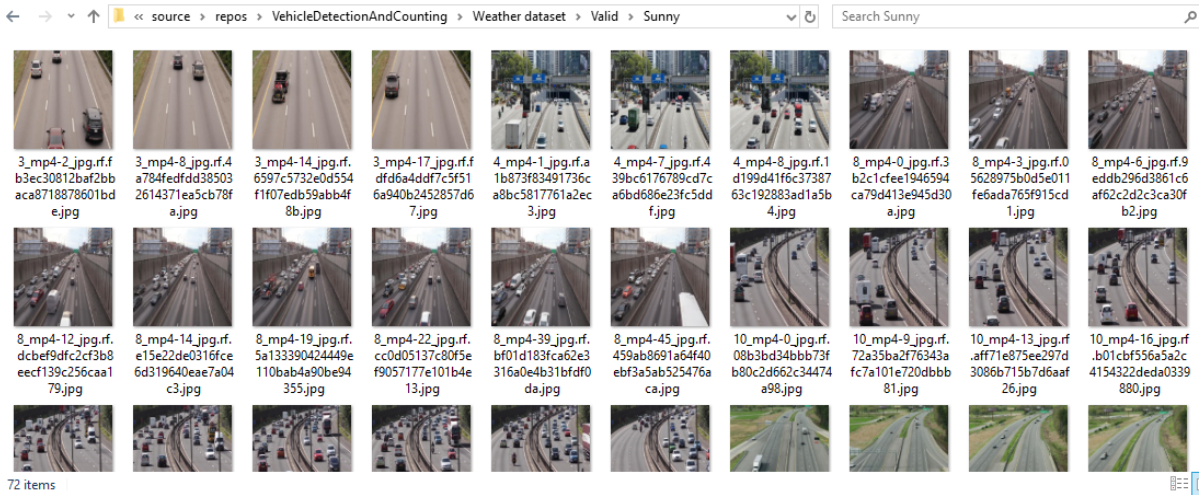


Figure 5.4.2.5 Sunny validation set images

Figure 5.4.2.4 contains 5 images that are rainy, and Figure 5.4.2.5 contains 72 images that are sunny. Both will be used for validating the model during the weather classification training.

### 5.4.3 Vehicle Model Detection Training

Below is the result of training process of the vehicle model detection on the train and valid image set using the pre-trained YOLOv5m weight with modified parameters.

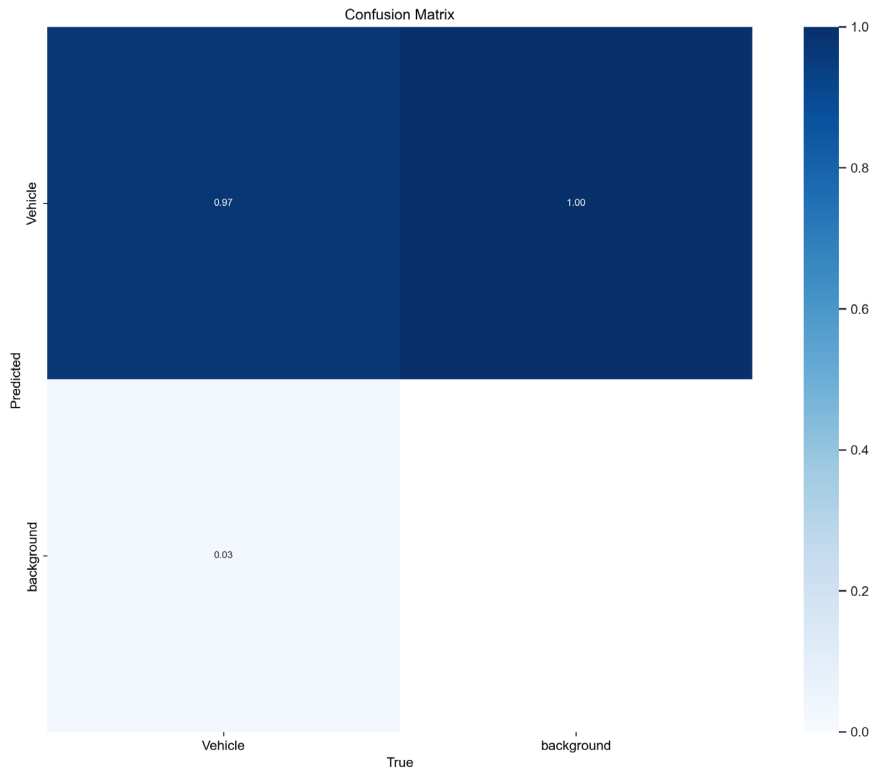


Figure 5.4.3.1 Confusion matrix of the training model

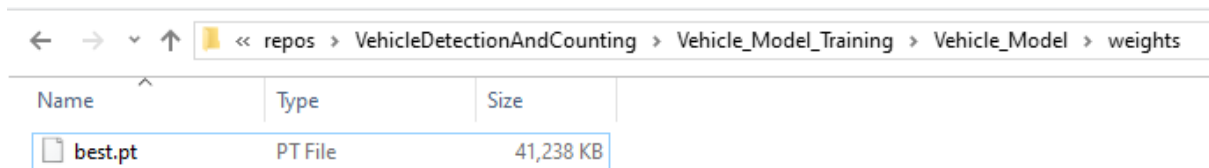
Figure 5.4.3.1 shows the confusion matrix, which shows the performance of the vehicle detection model by categorizing predictions into 4 categories. The vehicle is the positive class, while the background is the negative class. Based on the value 0.97 for the TP and 1.00 for the TN the model is highly effective in achieving high accuracy while distinguishing between the vehicle and the background.

True positive (TP) – The model correctly predicts the vehicle.

True negative (TN) – The model correctly predicts the background.

False positive (FP) – The model wrongly predicts a background as a vehicle.

False negative (FN) – The model is unable to detect a vehicle.



Name	Type	Size
best.pt	PT File	41,238 KB

Figure 5.4.3.2 Output weights of the vehicle model detection training process

Figure 5.4.3.2 above shows the output weights from the vehicle model detection training process. ‘best.pt’ represents is the model weights that has the highest validation accuracy during the training process. This file contains the learned features and optimizations which will then be used in later detection stage.

#### 5.4.4 Weather Classification Model Training

```
Epoch: 2, Batch: 10, Loss: 0.0096
Epoch: 2, Validation Loss: 0.0045, Validation Accuracy: 100.00%
Epoch: 3, Batch: 10, Loss: 0.0021
Epoch: 3, Validation Loss: 0.0006, Validation Accuracy: 100.00%
Epoch: 4, Batch: 10, Loss: 0.0003
Epoch: 4, Validation Loss: 0.0002, Validation Accuracy: 100.00%
Epoch: 5, Batch: 10, Loss: 0.0002
Epoch: 5, Validation Loss: 0.0001, Validation Accuracy: 100.00%
Epoch: 6, Batch: 10, Loss: 0.0001
Epoch: 6, Validation Loss: 0.0002, Validation Accuracy: 100.00%
Epoch: 7, Batch: 10, Loss: 0.0001
Epoch: 7, Validation Loss: 0.0001, Validation Accuracy: 100.00%
Epoch: 8, Batch: 10, Loss: 0.0001
Epoch: 8, Validation Loss: 0.0001, Validation Accuracy: 100.00%
Epoch: 9, Batch: 10, Loss: 0.0001
Epoch: 9, Validation Loss: 0.0001, Validation Accuracy: 100.00%
Epoch: 10, Batch: 10, Loss: 0.0001
Epoch: 10, Validation Loss: 0.0001, Validation Accuracy: 100.00%
Finished model training
Press any key to continue . . . █
```

Figure 5.4.4.1 Weather classification training process

Figure 5.4.4.1 above shows the screenshot of the progression of the weather classification training process throughout the entire 10 epochs. The loss decreases upon each epoch, indicating that the model is improving its ability to classify between 2 of the weather conditions more and more accurately. Once the model is finished training, the model with the lowest validation loss will be saved as ‘weather\_classification.pth’ and be used for the video processing stage.

#### 5.4.5 Video Processing



Figure 5.4.5.1 Original sunny video





Figure 5.4.5.2 Enhanced sunny video

Figure 5.4.5.1 shows the original video of sunny weather before processing, while Figure 5.4.5.2 shows the enhanced sunny video after processing, in which gamma correction is applied to each frame of the video and increases the overall brightness. In Figure 5.4.5.1, the original video brightness was comparatively low due to the excessive sunlight; by applying gamma correction, the video visibility slightly increased, which is able to increase overall detection accuracy in another sense.



Figure 5.4.5.3 Original rainy video

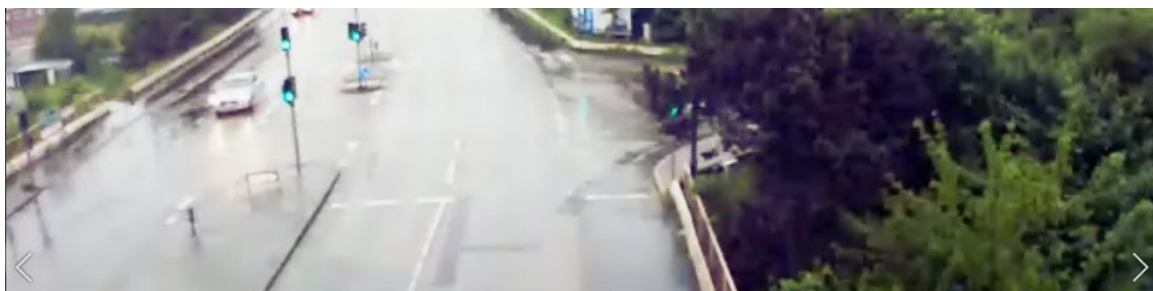
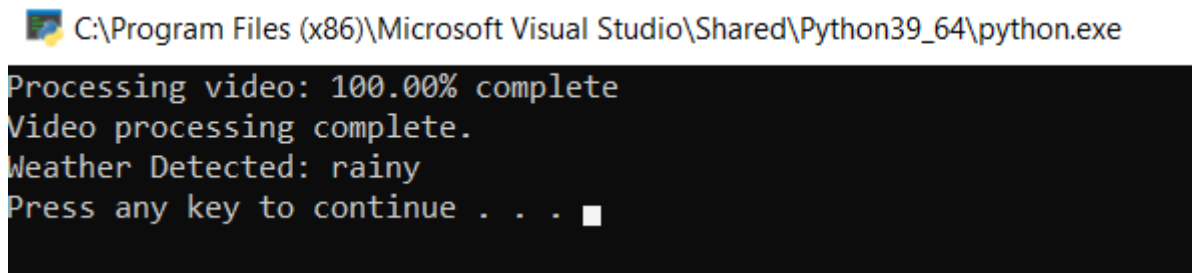


Figure 5.4.5.4 Enhanced rainy video

Whereas Figure 5.4.5.3 shows the original rainy video before going through the processing process, and Figure 5.4.5.4 shows the enhanced rainy video after processing, where histogram equalization was applied to the video frames to improve the visibility of the video. The frame contrast is adjusted by redistributing the pixels of the frame to optimize the available range, therefore enhancing the features that might have been affected by the low light situations which are commonly seen during rainy weather.



```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python39_64\python.exe
Processing video: 100.00% complete
Video processing complete.
Weather Detected: rainy
Press any key to continue . . . █
```

Figure 5.4.5.5 Output of videoprocessing.py

Figure 5.4.5.5 is the output of processing the rainy video, and from the output, the weather detected is rainy, indicating that the system is able to predict the weather conditions correctly from the input video frames.

### 5.4.6 Vehicle Detection

```
Processing frame 258/320... Detections: 4  
Processing frame 259/320... Detections: 6  
Processing frame 260/320... Detections: 6  
Processing frame 261/320... Detections: 6  
Processing frame 262/320... Detections: 6
```

Figure 5.4.6.1 Current processing frame of video

The enhanced sunny video is taken as the input into the YOLOv5 detection model, and the video has a total of 320 frames. Figure above shows the current processing frame of the input video.

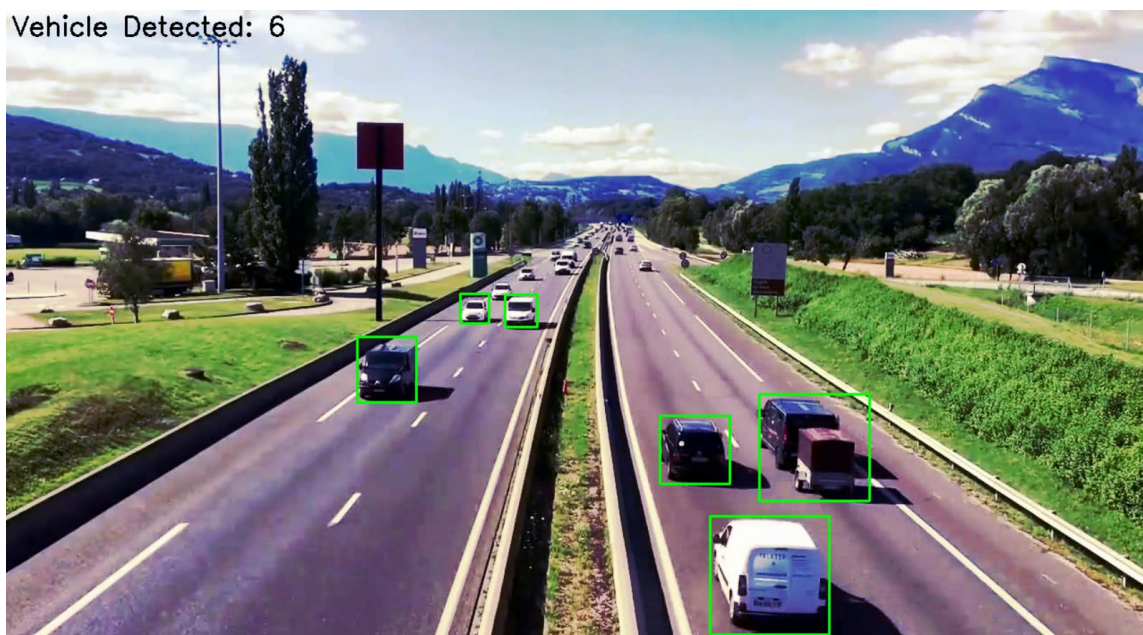


Figure 5.4.6.2 Vehicle detected in the video frame with bounding box

The above illustration is the execution of the YOLOv5 detection model being applied to the processed sunny video without a tracking algorithm and counting logic. This is to showcase that with the 'best.pt' weight the detection model able to perform recognition of the moving vehicles from the background and a green bounding box is drawn on each of the successful detected vehicles. Figure 5.4.6.1 shows the current frame of the video being processed, which is the 262th frame out of the total 320 frames, and the vehicle detected is 6. Whereas Figure 5.4.6.2 shows that each vehicle is marked with a green bounding box.



### 5.4.7 Vehicle Detection with Tracking

Detected: 6

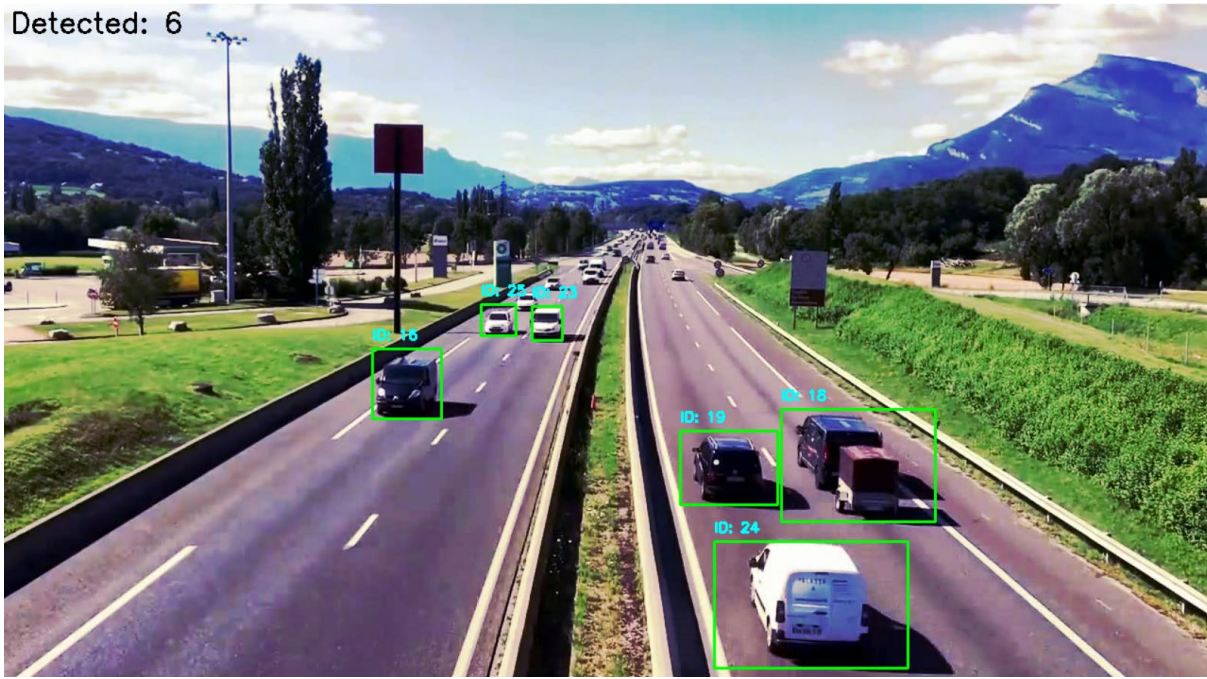


Figure 5.4.7.1 Vehicle detected and tracked in the video frame

Deep sort is applied to keep track of all the detected vehicles within the frame. In order to show that each of the vehicles was being tracked, a unique identifier will be generated and placed on the top left of the bounding box as a unique identifier. Figure 5.4.7.1 shows that 6 vehicles in the video frame have been successfully detected, tracked and assigned a cyan label unique identifier.



### 5.4.8 Vehicle Detection, Tracking and Counting

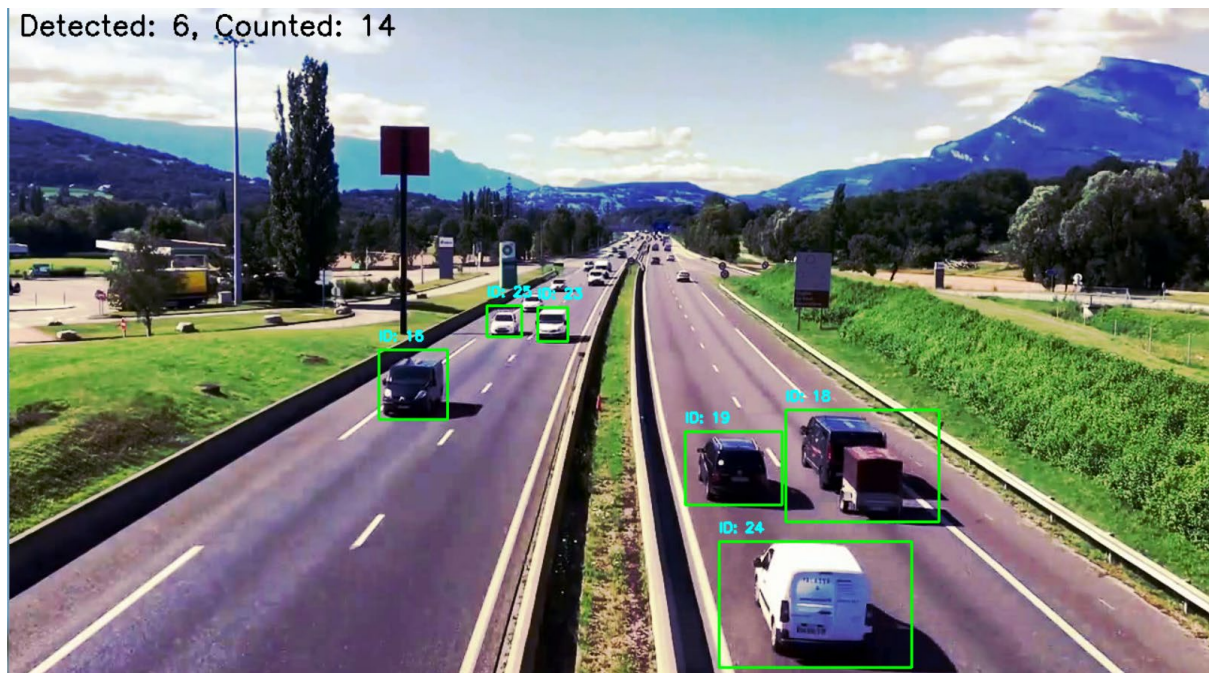


Figure 5.4.8.1 Vehicle detected, tracked and counted in the video frame

A counting logic is applied to the system but accumulates all the unique identifiers that have been assigned to each of the vehicles from the initial frame. Figure 5.4.8.1 shows that in the current frame, the total number of vehicles detected is 6, and from the first frame, the total number of unique vehicles counted is 14. This shows that the counting logic is working correctly and is able to increase the count whenever there is a new vehicle being detected and tracked with a unique identifier.

```
Processing frame 317/320... Detections: 6
Processing frame 318/320... Detections: 6
Processing frame 319/320... Detections: 7
Processing frame 320/320... Detections: 6
Total unique vehicles counted in the video: 17
```

Figure 5.4.8.2 Console output of vehicle detection and counting

Upon all the video frames being processed, the total number of unique vehicles is displayed, which is 17. This shows that for the entire input video, there are a total of 17 vehicles in the moving traffic, and each of them is being detected and counted correctly.

## **5.5 Implementation Issues and Challenges**

Throughout the implementation of the vehicle detection and counting system, the main issues that arise are the performance and computational challenges. Due to the limited device computational power, the process of model training and detection was slow. The model training for the datasets was very time-consuming. The entire session lasted around 10 hours due to the low processing power of the device used. On top of that, the deep learning model detection applied to process the video frame and perform detection was comparatively slower also, where the input video being only 10 seconds with a total of 320 frames, the entire detection, tracking and counting process required more than 2 minutes to finish. This shows a huge bottleneck due to the intensive computational demands of the deep learning model, YOLOv5 detection model and the deep sort tracking algorithm. The challenge also arises with the resolution of the video frame, the higher the quality of the video frame, the more processing power is needed as the number of pixels that need to be processed also increases.

# Chapter 6

## System Evaluation and Discussion

### 6.1 System Testing and Performance Metrics

Several factors are considered when testing the developed vehicle detection and counting system, such as the system's accuracy, reliability, and efficiency. The performance metrics used as the measure are mainly based on the accuracy metric used to compare the model's performance. High accuracy indicates that the model is able to detect moving vehicles from the background and count all of them, whereas low accuracy represents that the model is unable to distinguish between moving vehicles correctly from the background or the system miscount one vehicle as two and detect the vehicle as two unique objects. The counting accuracy is mainly based on the detection model as the tracking algorithm works in the way of taking the detection output as the input and taking count into the counting logic.

#### 6.1.1 Accuracy

The method of testing the accuracy metrics is by first manually counting moving vehicles of the original video and recording it down as the total number of ground truth instance, and the total number of vehicles counted after the execution of detection and counting system is recorded also and applying the formula to compute the accuracy. The following formula is used to count the accuracy:

$$Accuracy = (D / N) \times 100\%$$

where

$D$  is the number of vehicles counted by the system

$N$  is the total number of ground truth instances

If the system overcounts, the accuracy calculation of the ground truth instance can be adjusted by swapping the number of vehicles counted by the system with the total number of ground truth instances. The formula is denoted as below:

$$Accuracy = (N / D) \times 100\%$$

*where*

$D$  is the number of vehicles counted by the system

$N$  is the total number of ground truth instances

### 6.1.2 Reliability and efficiency

Other than that, to measure the system's overall performance, the total execution time for the entire processing of the vehicle detection and counting system is also recorded and compared. Each of the different video resolutions, frames and weather conditions are taken into consideration when comparing the system's efficiency. This comparison can provide an insight of the entire system efficiency in the sense of what are the duration needed to fully process the input video frames and perform detection and counting mechanism on each of frame and output the results. If the video resolution is high and the frames are in an acceptable range regardless of the weather conditions, the system execution time should be expected to be around a few minutes as the deep learning algorithm needs to be loaded with the trained weight into the system before performing the detection task. Furthermore, to test the system reliability, several run tests of the same input video frames will be used, and down each iteration will be tested to see whether the results generated are consistent. If the results of each execution of the system are consistent, meaning that the system is reliable, while if the same input video is used and the output results vary, this indicates that the system is not stable.

## 6.2 Testing Setup and Result

### 6.2.1 Detection Model Comparison with Alternative Methods (Faster-RCNN)

During the implementation phase of this project, beside YOLOv5 model, another model Faster-RCNN also has been setup paired with the deep sort to determine which model suits the project better. Faster-RCNN also known as region based convolutional neural networks that use a region proposal network to generate bounding boxes and distinguishing object within the image. Although Faster-RCNN might be effective in certain scenarios but in terms of processing speed it is much slower than YOLOv5. This is because the Faster-RCNN algorithm employs a two-stage approach for object detection. Initially, it creates region ideas for possible item detection then these proposals are refined by identifying the objects and adjusting the bounding box accordingly. On top of that, Faster-RCNN accuracy in vehicle detection was extremely low compare with YOLOv5 as YOLOv5 provides a more integrated approach in forming bounding box and class probability which is determining the vehicle from the background. Below is comparison of accuracy between Faster-RCNN and YOLOv5 using the same input video which is the sunny enhanced video.



Figure 6.2.1.1 Faster-RCNN with Deep Sort

From Figure 6.2.1.1 above, while processing frame 150 out of 320, the vehicles detected by the Faster-RCNN were 12 vehicles, and each of the vehicles was marked with a unique identifier. This shows that the Faster-RCNN model is unable to differentiate the vehicle model



correctly from the frame, as supposedly, there are only 4 vehicles that should be detected in this particular frame. For counting, once the detection is not correct meaning that the counting logic will also be wrong, as the counting logic works by counting each of the unique identifiers assigned to the detected vehicles.



Figure 6.2.1.2 YOLOv5 with Deep Sort

Whereas from Figure 6.2.1.2, the YOLOv5 with deep sort able to detect correctly in the frame 150. This showcase that the accuracy of YOLOv5 is higher than Faster-RCNN. In terms of processing speed, below is the result upon completion of execution the system using Faster-RCNN and YOLOv5.

```
Execution Time: 28 minutes, 11 seconds  
Press any key to continue . . .
```

Figure 6.2.1.3 Execution time of Faster-RCNN

```
Execution Time: 2 minutes, 37 seconds  
Press any key to continue . . .
```

Figure 6.2.1.4 Execution time of YOLOv5

From both figures above, it is obvious that the processing time of YOLOv5 is genuinely fast compared to Faster-RCNN. Both of the video input frames are the same which is 320 frames

for Faster-RCNN to complete processing all the frames the time needed is 28 minutes and 11 seconds while YOLOv5 only require 2 minutes and 37 seconds.

Performance Metric	Faster R-CNN	YOLOv5
Execution Time (seconds)	$(28 \times 60) + 11 = 1691$	$(2 \times 60) + 37 = 157$
Accuracy (%)	$(4 / 12) \times 100\% = 33.33$	$(4 / 4) \times 100\% = 100.00$

Table 6.2.1 Performance comparison of Faster-RCNN and YOLOv5

### 6.2.2 Accuracy Assessment of Video Enhancement Effects

First the testing setup as setup as below where 4 videos will be used as input fed into the detection and counting system. Figures below shows the output results for the rainy video detection and counting.

```
Processing frame 308/309... Detections: 3
Processing frame 309/309... Detections: 3
Total unique vehicles counted in the video: 7
Execution Time: 2 minutes, 55 seconds
Press any key to continue . . . █
```

Figure 6.2.2.1 Original rainy video output

```
Processing frame 307/309... Detections: 4
Processing frame 308/309... Detections: 3
Processing frame 309/309... Detections: 3
Total unique vehicles counted in the video: 8
Execution Time: 2 minutes, 49 seconds
Press any key to continue . . . █
```

Figure 6.2.2.2 Enhanced rainy video output

Rainy video:

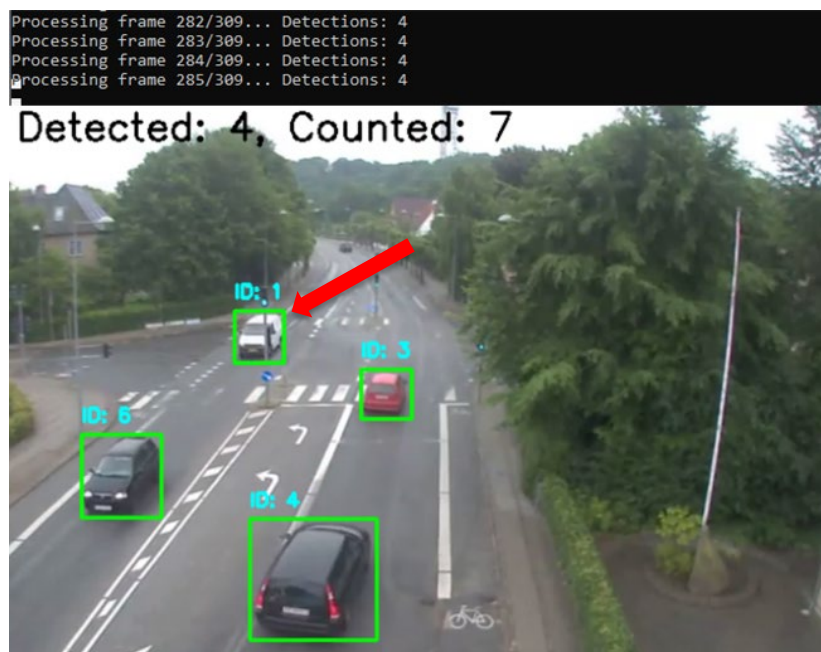


Figure 6.2.2.3 Example of misidentification

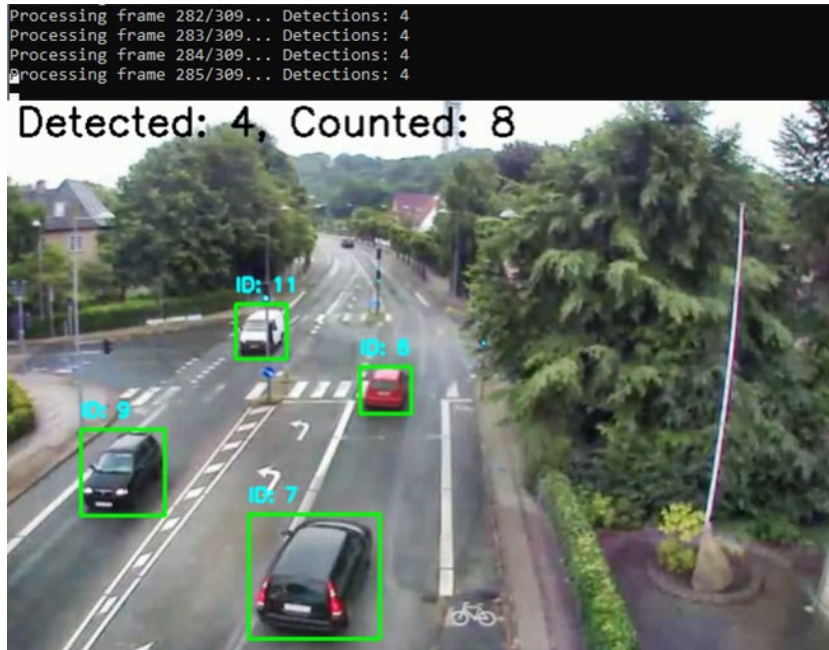


Figure 6.2.2.4 Example of correct identification

In Figure 6.2.2.3, the vehicle that is pointed by the red arrow illustrates a case of misidentification from the original rainy video. The id = 1 has been assigned once previously in the previous frame. The system falsely identifies and tracks the white vehicle, which causes an inaccurate count in frame 285, supposing it should have a total number of 8 vehicles counted instead of 7. Whereas in Figure 6.2.2.4 for the enhanced rainy video, the system was able to correctly identify the white vehicle and assign it a new identifier, demonstrating improved accuracy of tracking and counting after video enhancements.

Sunny video:

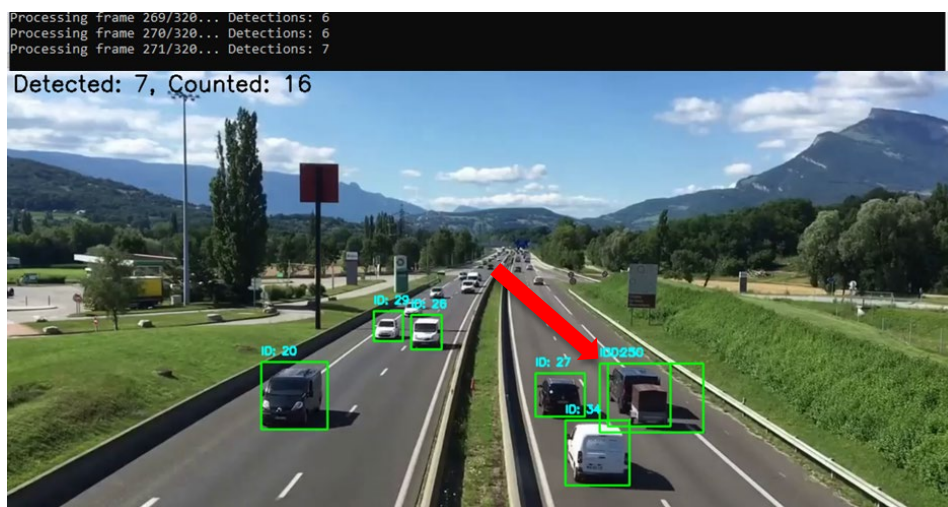


Figure 6.2.2.5 Example of false detection





Figure 6.2.2.6 Example of correct detection

In Figure 6.2.2.5, the vehicle that is pointed by the red arrow is an example of false detection from the original sunny video. The vehicle was detected as 2 individual vehicles, and 2 bounding boxes were drawn on it instead of 1. Whereas in Figure 6.2.2.6 for the enhanced sunny video, the system was able to correctly detect the vehicle as a single vehicle, and only 1 bounding box was drawn. This showcases that from the first example, the vehicle counted in frame 271 was 16, which is an overcount where the system should only be counted as 14.

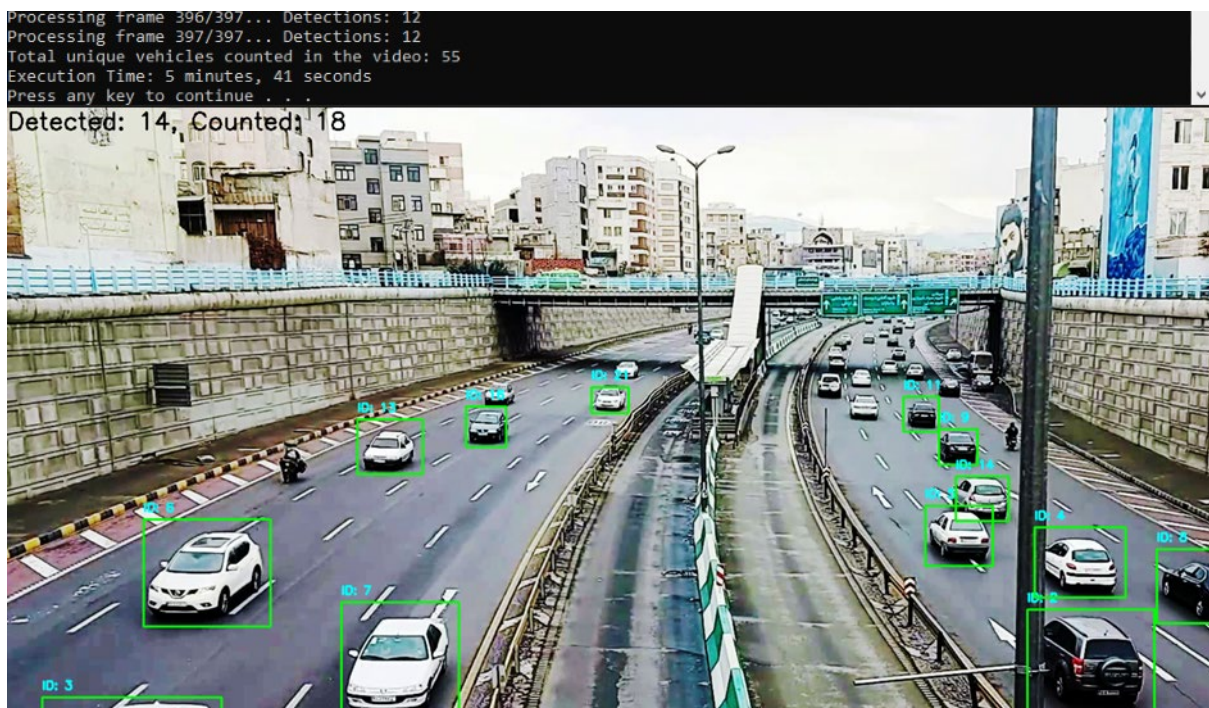


Figure 6.2.2.7 Example of heavy traffic conditions

Other than that, an input video containing heavy traffic was also tested to determine the accuracy of the system. The aim of using heavy traffic is to showcase that the developed system also works in heavy traffic where there are a lot of vehicles moving on the road. From the above Figure 6.2.2.7, the detection and counting system work perfectly by detecting each of the vehicles in the traffic and able to count them accordingly.

For better quantitative visualization of the differences between the original and enhanced video conditions below Table 6.1.1 shows

Video Type	Ground Truth Instance	Vehicle Counted	Accuracy (%)
Original Sunny Video	17	24	$(17 / 24) \times 100 = 70.83\%$
Enhanced Sunny Video	17	17	$(17 / 17) \times 100 = 100.00\%$
Original Rainy Video	8	7	$(7 / 8) \times 100 = 87.50\%$
Enhanced Rainy Video	8	8	$(8 / 8) \times 100 = 100.00\%$
Enhanced Heavy Traffic Video	60	55	$(55 / 60) \times 100 = 91.67\%$

Table 6.2.2.1 Vehicle detection accuracy across video conditions

### 6.2.3 Reliability Assessment

```
Processing frame 319/320... Detections: 7
Processing frame 320/320... Detections: 6
Total unique vehicles counted in the video: 17
Execution Time: 2 minutes, 37 seconds
Press any key to continue . . .
```

Figure 6.2.3.1 1<sup>st</sup> iteration of vehicle detection process

```
Processing frame 319/320... Detections: 7
Processing frame 320/320... Detections: 6
Total unique vehicles counted in the video: 17
Execution Time: 2 minutes, 38 seconds
Press any key to continue . . .
```

Figure 6.2.3.2 2<sup>nd</sup> iteration of vehicle detection process

```
Processing frame 319/320... Detections: 7
Processing frame 320/320... Detections: 6
Total unique vehicles counted in the video: 17
Execution Time: 2 minutes, 37 seconds
Press any key to continue . . .
```

Figure 6.2.3.3 3<sup>rd</sup> iteration of vehicle detection process

For testing the system's reliability, several iterations of tests are conducted with the same input video, and each iteration's result is recorded in the table below. The input video used for the test is the enhanced sunny video.

<b>Iteration</b>	<b>Vehicle Counted</b>	<b>Execution Time (seconds)</b>
1	17	$(2 \times 60) + 37 = 157$
2	17	$(2 \times 60) + 38 = 158$
3	17	$(2 \times 60) + 37 = 157$

Table 6.2.3.1 Vehicle counted and execution time across iterations

From the table above, the execution time of 3 of the iteration tests are similar and the difference between each execution is not more than 1 second which shows that the system's reliability is high and each of the unique vehicles counted by the system are the same which is 17.

### 6.3 Objective Evaluation

Generally, the project objectives that wanted to be achieved at the start of the project has been met in the developed system. This weather conditions are overcome by applying the trained weather classification and used as input to enhance the video. For this project, the weather conditions were mainly focused on only two rainy and sunny as the most common weather conditions in this country. On top of that, this system also able to detect not only in particular road structures as the previous provided screenshot and testing conducted the sunny video was the footage from a highway while the rainy video was video capture from a normal road intersection. Lastly the developed system has implemented a deep sort tracking algorithm to keep track of the detected vehicles in the moving traffic and count them accurately, this fulfils the objective of achieving an integrated vehicle detection and counting in a single system.

### 6.4 Project Challenges

Throughout the entire project the main issue that encountered was the hardware limitations that affected the effectiveness of the vehicle detection and counting system. The task of detecting vehicles within a short 10-second period of video input required more than 2 minutes to finish, a delay mostly caused by the constraints of CPU processing capacity. This considerable processing time clearly demonstrates the system's significant dependence on hardware

capabilities, namely the GPU, which was unavailable in this instance. The lack of a robust GPU severely limited the processing, assessment, and output speed of video data. This problem not only illustrates the difficulties of creating high-performance computing systems on constrained hardware but also emphasizes the crucial requirement for secure hardware to efficiently handle real-time data processing requirements.

## Chapter 7

### Conclusion and Recommendations

The vehicle detection model was successfully trained using a YOLOv5 architecture by leveraging a pre-trained model weight with the customized dataset. The trained result weight indicates that the model is able to distinguish precisely the moving vehicle from the background. Other than that, the binary weather classification training was trained mainly for processing the input video frame by enhancing the features by classifying the weather conditions into rainy and sunny and performing enhancements accordingly. In the evaluation part, the enhanced video shows a better accuracy compared to the original video. Applying both trained models, the system accuracy was able to achieve more than 80% overall, which proved the system's robustness and precision towards dynamic conditions. In conclusion, the video-based vehicle detection and counting system developed in this project has achieved all the initial goals set. From the input video, most of the vehicles in the moving traffic are able to be detected correctly and labeled with a unique identifier, and finally counted. With the generated output, which is the total vehicle count within the moving traffic, traffic administrators are able to utilize the data, for example, generating a heat map indicating that the intersection of the road is congested according to the traffic volume. Moreover, there are still room for improvement of the developed system especially in the terms of processing speed, by utilizing a better hardware with higher specifications and processing power, the system able to process and perform detection faster.

## REFERENCES

- [1] D. Li, B. Liang, and W. Zhang, "Real-time moving vehicle detection, tracking, and counting system implemented with OpenCV," IEEE Xplore, Apr. 01, 2014. <https://ieeexplore.ieee.org/abstract/document/6920557> (accessed Jun. 21, 2021).
- [2] M. Lei, D. Lefloch, P. Gouton, and K. Madani, "A Video-Based Real-Time Vehicle Counting System Using Adaptive Background Method," IEEE Xplore, Nov. 01, 2008. <https://ieeexplore.ieee.org/abstract/document/4725850> (accessed Sep. 03, 2023).
- [3] G. D. Sullivan, K. D. Baker, A. D. Worrall, C. I. Attwood, and P. M. Remagnino, "Model-based vehicle detection and classification using orthographic approximations," *Image and Vision Computing*, vol. 15, no. 8, pp. 649–654, Aug. 1997, doi: [https://doi.org/10.1016/s0262-8856\(97\)00009-7](https://doi.org/10.1016/s0262-8856(97)00009-7)
- [4] S. Gupte, O. Masoud, R. F. K. Martin, and N. P. Papanikolopoulos, "Detection and classification of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 37–47, Mar. 2002, doi: <https://doi.org/10.1109/6979.994794>
- [5] Sung Chun Lee and Ram Nevatia, "Speed Performance Improvement of Vehicle Blob Tracking System," Springer eBooks, pp. 197–202, Aug. 2008, doi: [https://doi.org/10.1007/978-3-540-68585-2\\_17](https://doi.org/10.1007/978-3-540-68585-2_17)
- [6] N. Ostu, "A threshold selection method from gray-level histograms.," *IEEE Trans SMC*, vol. 9, p. 62, 1979, Available: <https://cir.nii.ac.jp/crid/1370567187454221059>
- [7] P.-S. Liao, T.-S. Chen, and P.-C. Chung, "A Fast Algorithm for Multilevel Thresholding," *JOURNAL OF INFORMATION SCIENCE AND ENGINEERING*, vol. 17, pp. 713–727, 2001, Accessed: Sep. 03, 2023. [Online]. Available: <https://me.csu.edu.tw/wSite/public/Data/paper/f1655974519515.pdf>

- [8] B. Sun and S. Li, "Moving Cast Shadow Detection of Vehicle Using Combined Color Models," IEEE Xplore, Oct. 01, 2010. <https://ieeexplore.ieee.org/abstract/document/5659321> (accessed Sep. 03, 2023).
- [9] A. Sanin, C. Sanderson, and B. C. Lovell, "Shadow detection: A survey and comparative evaluation of recent methods," Pattern Recognition, vol. 45, no. 4, pp. 1684–1695, Apr. 2012, doi: <https://doi.org/10.1016/j.patcog.2011.10.001>.
- [10] N. Seenouvang, U. Watchareeruetai, C. Nuthong, K. Khongsomboon, and N. Ohnishi, "A computer vision-based vehicle detection and counting system," IEEE Xplore, Feb. 01, 2016. <https://ieeexplore.ieee.org/document/7440510> (accessed Jun. 19, 2021).
- [11] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik, "A real-time computer vision system for vehicle tracking and traffic surveillance," Transportation Research Part C: Emerging Technologies, vol. 6, no. 4, pp. 271–288, Aug. 1998, doi: [https://doi.org/10.1016/s0968-090x\(98\)00019-9](https://doi.org/10.1016/s0968-090x(98)00019-9).
- [12] H. Song, H. Liang, H. Li, Z. Dai, and X. Yun, "Vision-based vehicle detection and counting system using deep learning in highway scenes," European Transport Research Review, vol. 11, no. 1, Dec. 2019, doi: <https://doi.org/10.1186/s12544-019-0390-4>.
- [13] X.-Z. Chen, C.-M. Chang, C.-W. Yu, and Y.-L. Chen, "A Real-Time Vehicle Detection System under Various Bad Weather Conditions Based on a Deep Learning Model without Retraining," Sensors, vol. 20, no. 20, p. 5731, Oct. 2020, doi: <https://doi.org/10.3390/s20205731>.
- [14] B. Neupane, T. Horanont, and J. Aryal, "Real-Time Vehicle Classification and Tracking Using a Transfer Learning-Improved Deep Learning Network," Sensors, vol. 22, no. 10, p. 3813, May 2022, doi: <https://doi.org/10.3390/s22103813>.



# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Trimester 3 Year 3</b>	<b>Study week no.: 2</b>
<b>Student Name &amp; ID: Kenny Saw Wei Wen 21ACB06312</b>	
<b>Supervisor: Dr Muhammad Syaiful Amri Bin Suhaimi</b>	
<b>Project Title: Development of vehicle detection and counting system for traffic analysis using computer vision</b>	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Look through more dataset, especially dataset with rainy conditions. Studied more articles and reviewed on the proposed solution.

## 2. WORK TO BE DONE

Find and download more dataset with different entities.

## 3. PROBLEMS ENCOUNTERED

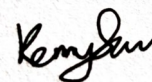
Most of the dataset are low quality due to the traffic camera used. Hard to find a better quality image and video dataset.

## 4. SELF EVALUATION OF THE PROGRESS

So far so good, as progress of current work pace are acceptable.



\_\_\_\_\_  
Supervisor's signature



\_\_\_\_\_  
Student's signature



# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Trimester 3 Year 3</b>	<b>Study week no.: 5</b>
<b>Student Name &amp; ID: Kenny Saw Wei Wen 21ACB06312</b>	
<b>Supervisor: Dr Muhammad Syaiful Amri Bin Suhaimi</b>	
<b>Project Title: Development of vehicle detection and counting system for traffic analysis using computer vision</b>	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Filtered all the downloaded datasets. Labelling and annotation of the dataset are done. Prepared the environment for the project development. All required libraries were installed into the device. Done literature review on the articles found. Studied training model that suits the project.

## 2. WORK TO BE DONE

Complete all the training needed of the datasets.

## 3. PROBLEMS ENCOUNTERED

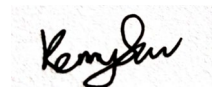
Some image labels are wrong, manual corrections is needed and very time consuming.

## 4. SELF EVALUATION OF THE PROGRESS

Acceptable progress. Keep working hard.



\_\_\_\_\_  
Supervisor's signature



\_\_\_\_\_  
Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Trimester 3 Year 3</b>	<b>Study week no.: 8</b>
<b>Student Name &amp; ID: Kenny Saw Wei Wen 21ACB06312</b>	
<b>Supervisor: Dr Muhammad Syaiful Amri Bin Suhaimi</b>	
<b>Project Title: Development of vehicle detection and counting system for traffic analysis using computer vision</b>	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Training of the vehicle model and weather classification model. Proceed with enhancing the video.

## 2. WORK TO BE DONE

Done the enhancement of video with the trained classification weight.

## 3. PROBLEMS ENCOUNTERED

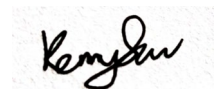
Training process very slow, for each training process more than 10 hours is needed. First training failed due to device auto shut down as battery ran out.

## 4. SELF EVALUATION OF THE PROGRESS

Still manageable with ongoing exams and assignments of other courses.



Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Trimester 3 Year 3</b>	<b>Study week no.: 10</b>
<b>Student Name &amp; ID: Kenny Saw Wei Wen 21ACB06312</b>	
<b>Supervisor: Dr Muhammad Syaiful Amri Bin Suhaimi</b>	
<b>Project Title: Development of vehicle detection and counting system for traffic analysis using computer vision</b>	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Vehicle detection implementation done. Vehicles from the input video are able to be detected correctly with bounding box drawn on it. Studies on the tracking algorithm how to feed the detection input into the tracking algorithm.

## 2. WORK TO BE DONE

Apply the tracking algorithm and perform counting on each of the tracked vehicles. Finalize the report.

## 3. PROBLEMS ENCOUNTERED

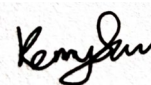
Slow processing power, detection process on each of the video frames are slower than expected.

## 4. SELF EVALUATION OF THE PROGRESS

Hopefully can finalize the development before next week.



Supervisor's signature



Student's signature



# Vehicle Detection and Counting System



## Introduction

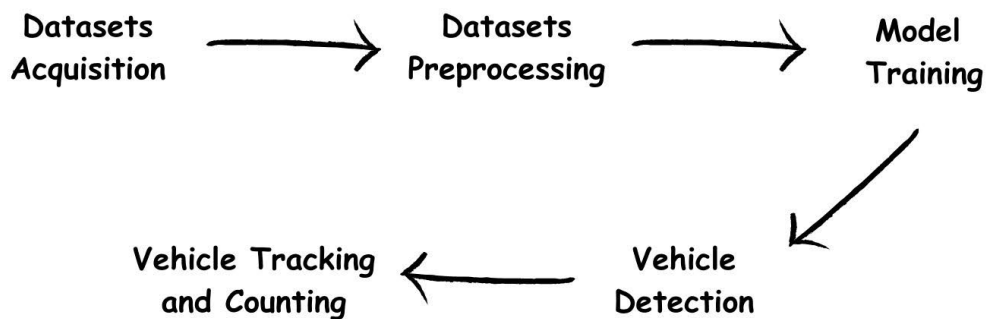
In urban planning and smart city deployment, efficient traffic analysis is one of the key success factors. Most traffic monitoring requires accurate yet reliable data. Hence, this project uses a computer vision approach to detect and count the number of vehicles in moving road traffic to provide a cost-effective yet scalable solution.

## Project Objective



- ✓ Enhance dynamic adaptability of detecting and counting
- ✓ Tackle Diverse Weather Conditions
- ✓ To Develop An Integrated System

## Methodology



## Result & Conclusion

The bounding box drawn around the vehicle indicate vehicle detected and each of the vehicle assigned with unique id will be counted. The detection and counting system able to recognize all vehicles in the moving traffic, showing its robust detection and tracking capabilities.

Student: Kenny Saw Wei Wen  
Supervisor: Dr Muhammad Syaiful Amri bin Suhaimi

## PLAGIARISM CHECK RESULT

21ACB06312

### ORIGINALITY REPORT

<b>13%</b>	<b>10%</b>	<b>7%</b>	<b>7%</b>
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

### PRIMARY SOURCES

<b>1</b>	<b>eprints.utar.edu.my</b> Internet Source	<b>1%</b>
<b>2</b>	<b>www.mdpi.com</b> Internet Source	<b>1%</b>
<b>3</b>	<b>Submitted to Universiti Tunku Abdul Rahman</b> Student Paper	<b>1%</b>
<b>4</b>	<b>Submitted to CSU, Long Beach</b> Student Paper	<b>&lt;1%</b>
<b>5</b>	<b>Chomtip Pornpanomchai. "Vehicle detection and counting from a video frame", 2008 International Conference on Wavelet Analysis and Pattern Recognition, 08/2008</b> Publication	<b>&lt;1%</b>
<b>6</b>	<b>link.springer.com</b> Internet Source	<b>&lt;1%</b>
<b>7</b>	<b>Submitted to University of Northampton</b> Student Paper	<b>&lt;1%</b>
<b>8</b>	<b>Submitted to Hong Kong University of Science and Technology</b> Student Paper	<b>&lt;1%</b>

9	<a href="https://hackernoon.com">hackernoon.com</a> Internet Source	<1 %
10	Submitted to Eastern University Student Paper	<1 %
11	<a href="https://discuss.pytorch.org">discuss.pytorch.org</a> Internet Source	<1 %
12	<a href="http://www.aisi555.com">www.aisi555.com</a> Internet Source	<1 %
13	<a href="http://www.coursehero.com">www.coursehero.com</a> Internet Source	<1 %
14	<a href="https://yolov8.org">yolov8.org</a> Internet Source	<1 %
15	Xiao Zhao, Zhenjia Chen, Qinglong Chen. "Application of Vehicle Recognition Model Based on Yolo Neural Network on Embedded Devices", 2022 6th Asian Conference on Artificial Intelligence Technology (ACAIT), 2022 Publication	<1 %
16	<a href="https://github.com">github.com</a> Internet Source	<1 %
17	Submitted to University of Queensland Student Paper	<1 %
18	Submitted to University of Southern California Student Paper	<1 %



19	pubmed.ncbi.nlm.nih.gov Internet Source	<1 %
20	www.ncbi.nlm.nih.gov Internet Source	<1 %
21	Haojia Lin, Zhilu Yuan, Biao He, Xi Kuai, Xiaoming Li, Renzhong Guo. "A Deep Learning Framework for Video-Based Vehicle Counting", Frontiers in Physics, 2022 Publication	<1 %
22	"IT Convergence and Security 2017", Springer Science and Business Media LLC, 2018 Publication	<1 %
23	Oluwatobi Adeleke, Sina Karimzadeh, Tien-Chien Jen. "Machine Learning-Based Modelling in Atomic Layer Deposition Processes", CRC Press, 2023 Publication	<1 %
24	Submitted to Bournemouth University Student Paper	<1 %
25	Huiting Sun, Jialiang Peng, Zhaogong Zhang. "A survey on face recognition methods with federated learning", Institution of Engineering and Technology (IET), 2023 Publication	<1 %
26	developer.aliyun.com Internet Source	<1 %

27	<a href="http://file.techscience.com">file.techscience.com</a> Internet Source	<1 %
28	Submitted to RMIT University Student Paper	<1 %
29	<a href="http://gitlab.sliit.lk">gitlab.sliit.lk</a> Internet Source	<1 %
30	Submitted to Fakultet elektrotehnike i računarstva / Faculty of Electrical Engineering and Computing Student Paper	<1 %
31	<a href="http://roboflow.com">roboflow.com</a> Internet Source	<1 %
32	L. Kovavisaruch, T. Sanpechuda, J. Chinrungrueng, U. Sununtachaikul, S. Kittipiyakul, S. Samphanyuth. "Accuracy improvement method for vehicle detection using optical sensors", 2009 9th International Conference on Intelligent Transport Systems Telecommunications, (ITST), 2009 Publication	<1 %
33	Submitted to Nanyang Technological University Student Paper	<1 %
34	<a href="http://codesearch.isocpp.org">codesearch.isocpp.org</a> Internet Source	<1 %



35	Bipul Neupane, Teerayut Horanont, Jagannath Aryal. "Real-Time Vehicle Classification and Tracking Using a Transfer Learning-Improved Deep Learning Network", Sensors, 2022 Publication	<1 %
36	Submitted to Brunel University Student Paper	<1 %
37	Submitted to City University Student Paper	<1 %
38	G. Kataev, A. Shabley, S. Vaulin. "Development of an Application for Recognizing Automobile Vehicles", 2020 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon), 2020 Publication	<1 %
39	Zhangu Wang, Jun Zhan, Chunguang Duan, Xin Guan, Kai Yang. "Traffic vehicle cognition in severe weather based on radar and infrared thermal camera fusion", Measurement Science and Technology, 2021 Publication	<1 %
40	<a href="https://huggingface.co">huggingface.co</a> Internet Source	<1 %
41	<a href="https://medium.com">medium.com</a> Internet Source	<1 %
42	<a href="https://gcore.com">gcore.com</a> Internet Source	<1 %

		<1 %
43	journal.50sea.com Internet Source	<1 %
44	learnopencv.com Internet Source	<1 %
45	pdfcoffee.com Internet Source	<1 %
46	code84.com Internet Source	<1 %
47	daily-build.beehiiv.com Internet Source	<1 %
48	openi.pcl.ac.cn Internet Source	<1 %
49	Chunyu Li, Yuhui Zheng, Byeungwoo Jeon. "Pansharpener via Subpixel Convolutional Residual Network", IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2021 Publication	<1 %
50	Submitted to University of Edinburgh Student Paper	<1 %
51	Sadhasivam S, Prakash G, Ranjith Kumar G, Nazeeya Anjum N. "Dynamic Mapping in Confined Spaces: Robotic SLAM with Human	<1 %

Detection and Path Planning", 2023 2nd  
International Conference on Automation,  
Computing and Renewable Systems (ICACRS),  
2023

Publication

---

52	Sunil Kumar. "Chapter 27 Deep Learning", Springer Science and Business Media LLC, 2024 Publication	<1 %
53	Yongpeng Yang, Hong Li, Xiangjun He. "A Financial Accounting Voucher Recognition Strategy Based on Deep Learning", IOS Press, 2024 Publication	<1 %
54	debuggercafe.com Internet Source	<1 %
55	dokumen.pub Internet Source	<1 %
56	doria.fi Internet Source	<1 %
57	pastebin.com Internet Source	<1 %
58	www.diva-portal.org Internet Source	<1 %
59	www.hackster.io Internet Source	<1 %

---

- |    |  |      |
|----|--|------|
| 60 | B. Neupane, T. Horanont, P. Pattarapongsin, A. Thapa. "ROBUST AND SCALABLE REAL-TIME VEHICLE CLASSIFICATION AND TRACKING: A CASE STUDY OF THAILAND", ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2022<br><small>Publication</small> | <1 % |
| 61 | Boqiang Xu, Chao Liu. "Keypoint detection-based and multi-deep learning model integrated method for identifying vehicle axle load spatial-temporal distribution", Advanced Engineering Informatics, 2024<br><small>Publication</small>                                       | <1 % |
| 62 | Chengjing Wei, Guodong Li. "A selective image encryption scheme using LICC hyperchaotic system", IET Image Processing, 2022<br><small>Publication</small>  | <1 % |
| 63 | Jingtao Sun, Jiayin Kou, Wenyan Hou, Yujei Bai. "A multi-agent curiosity reward model for task-oriented dialogue systems", Pattern Recognition, 2025<br><small>Publication</small>   | <1 % |
| 64 | Kai Zhou, Karthik Mahesh Varadarajan, Markus Vincze, Fuqiang Liu. "Hybridization of appearance and symmetry for vehicle-logo localization", 2012 15th International IEEE   | <1 % |

Conference on Intelligent Transportation  
Systems, 2012

Publication

---

**65** Raja Muthalagu, Anudeep Sekhar Bolimera, Dhruv Duseja, Shaun Fernandes. "Object and Lane Detection Technique for Autonomous Car Using Machine Learning Approach", Transport and Telecommunication Journal, 2021

Publication

---

**66** Wei Wei, Danni Zhang, Huichen Wang, Xiaodong Duan, Chen Guo. "Utilizing the Neural Renderer for Accurate 3D Face Reconstruction from a Single Image", Neural Processing Letters, 2023

Publication

---

**67** Xiaochun Wang. "Anomaly Detection in Video Surveillance", Springer Science and Business Media LLC, 2024

Publication

---

**68** [dione.lib.unipi.gr](http://dione.lib.unipi.gr)

Internet Source

---

**69** [dzone.com](http://dzone.com)

Internet Source

---

**70** [ebin.pub](http://ebin.pub)

Internet Source

---

[hdl.handle.net](http://hdl.handle.net)

71	Internet Source	<1 %
72	skannai.medium.com Internet Source	<1 %
73	uwspace.uwaterloo.ca Internet Source	<1 %
74	vulners.com Internet Source	<1 %
75	www.frontiersin.org Internet Source	<1 %
76	www.geeksforgeeks.org Internet Source	<1 %
77	Hashmi, Mohammad Farukh, and Avinash G. Keskar. "Analysis and monitoring of a high density traffic flow at T-intersection using statistical computer vision based approach", 2012 12th International Conference on Intelligent Systems Design and Applications (ISDA), 2012. Publication	<1 %
78	Liangju Fu, Qiang Zhang, Shengli Tian. "Real-time video surveillance on highways using combination of extended Kalman Filter and deep reinforcement learning", Heliyon, 2024 Publication	<1 %

<b>Universiti Tunku Abdul Rahman</b>			
<b>Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)</b>			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**

<b>Full Name(s) of Candidate(s)</b>	Kenny Saw Wei Wen
<b>ID Number(s)</b>	21ACB06312
<b>Programme / Course</b>	CS
<b>Title of Final Year Project</b>	Development of vehicle detection and counting system for traffic analysis using computer vision

<b>Similarity</b>	<b>Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)</b>
<b>Overall similarity index:</b> <u>    13    </u> %  <b>Similarity by source</b> Internet Sources: <u>    10    </u> % Publications: <u>    7    </u> % Student Papers: <u>    7    </u> %	
<b>Number of individual sources listed of more than 3% similarity:</b> <u>    0    </u>	
<b>Parameters of originality required and limits approved by UTAR are as Follows:</b> (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

***Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.***

\_\_\_\_\_  
Signature of Supervisor

\_\_\_\_\_  
Signature of Co-Supervisor

Name: Muhammad Syaiful Amri bin Suhaimi

Name: \_\_\_\_\_

Date: 09/09/2024

Date: \_\_\_\_\_

Bachelor of Computer Science (Honours)  
Faculty of Information and Communication Technology (Kampar Campus), UTAR





**UNIVERSITI TUNKU ABDUL RAHMAN**

**FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY  
(KAMPAR CAMPUS)**

**CHECKLIST FOR FYP2 THESIS SUBMISSION**

Student Id	21ACB06312
Student Name	Kenny Saw Wei Wen
Supervisor Name	Dr Muhammad Syaiful Amri Bin Suhaimi

TICK (√)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
√	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

\*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

(Signature of Student)

Date: 9 September 2024