Project Title:

VEHICLE COUNTING FOR TRAFFIC EASE USING YOLOV8

SUBMITTED BY

Name: Md. SAZZAD HOSSAIN

ID: 203-16-545

Department of CIS
Daffodil International University

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Computing and Information System

Supervised By
Mr. ISRAFIL
Designation: Lecturer

DAFFODIL INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

DATE OF SUBMISSION: 31 MAY 2025

APPROVAL

This Project titled **"Vehicle counting for Traffic Ease using Yolov8"**, Submitted by Md. Sazzad Hossain, ID No:203-16-545 to the Department of Computing and Information Systems, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computing & Information Systems and approved as to its style and contents. The presentation has been held on 31-05-2025.

BOARD OF EXAMINERS

Md Sarwar Hossain Mollah — **Chairman**
Associate Professor and Head
Department of Computing & Information Systems
Faculty of Science & Information Technology
Daffodil International University

Md. Nasimul Kader — **Internal Examiner**
Assistant Professor
Department of Computing & Information Systems
Faculty of Science & Information Technology
Daffodil International University

Md. Mehedi Hassan — **Internal Examiner**
Lecturer (Senior Scale)
Department of Computing & Information Systems
Faculty of Science & Information Technology
Daffodil International University

Ahmed Saif Reza — **External Examiner**
Managing Director & Chief Technology Officer
Medico Bio Limited

Declaration

I hereby declare that; this project has been done by me under supervision of **ISRAFIL, Lecturer**, department of Computing and Information System (CIS) of Daffodil International University. I am also declaring that this project or any part of there has never been submitted anywhere else for the award of any educational degree like, B.Sc., M.Sc., Diploma or other qualifications.

Supervised By

31.05.25

ISRAFIL
Lecturer
Department of CIS
Daffodil International University

Submitted By

31.05.25

Name: Md Sazzad Hossain
ID: 203-16-545
Department of CIS
Daffodil International University

ACKNOWLEDGEMENT

First, we express our heartiest thanks and gratefulness to almighty Allah for His divine blessing making us possible to complete the final year project successfully.

We are grateful and wish our profound indebtedness to**, Mr. Israfil (Lecturer), Department of CIS Daffodil International University, Dhaka.** The Deep knowledge & keen interest of my supervisor in the field of "Machine learning" helped to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts, and correcting them at all stages have made it possible to complete this project.

We would like to express my heartiest gratitude to **Md. Sarwar Hossain Mollah, (Head of CIS Department),** for his kind help to finish my project and also to other faculty members and the staff of the CIS department of Daffodil International University.

We would like to thank our entire course mate in Daffodil International University, who took part in this discuss while completing the course work.

Finally, we must acknowledge with due respect the constant support and patients of our parents.

ABSTRACT

The full results of my proposal, **"Vehicle Counting for Traffic Ease Using YOLOv8"** are available here. The steps used to turn the concept into a functional website are covered in full in this article. One more model that stands out to system which has been called the user dashboard. The project's objective was to use the YOLOv8 model to create that system who that could precisely count cars in more traffic situations. An automated vehicle counting system can be very can give city planners and traffic management authorities can be very useful data in light of the growing demand for effective traffic control and monitoring. In order to help can be very city planners and traffic authorities make well-informed more decisive decisions about infrastructure of this system offers useful data on can be very traffic flow and congestion. The technology helps to meet the increasing can be very for efficient traffic control in metropolitan areas by improving efficiency, through the automation can be very of vehicle counts. Four various cars types of counting vehicles class —such as cars, trucks, buses, and motorcycles—have been can be very employed in this work for detection and counting all cars. The project's objective was to develop a system that can be very accurately to more count automobiles in traffic scenarios using the YOLOv8 model. From conception to implementation, the study covers can be very facet of the system development process, including its architecture, user interface design, and technologies employed. We utilized Python for the backend. Costly software or computer components are not necessary to set up our system application; all you need is a typical desktop computer and internet connectivity.

TABLE OF CONTENTS

CONTENTS		PAGE
Approval	----------	i
Declaration	----------	ii
Acknowledgements	----------	iii
Abstract	----------	iv
Table of Contents	----------	v-x

CHAPTER		PAGES
CHAPTER 1: Introduction	----------	**1-2**
1.1 Introduction	----------	**1**
CHAPTER 2: Initial Study		**3-5**
2.1 Project Proposal	----------	**3**
2.2 Background of the server-based malware scanner system	----------	**3**
2.3 Problem Area	----------	**4**
2.4 Possible Solution	----------	**5**
CHAPTER 3: Literature Review		**6-9**
3.1 Discussion on problem domain based on published articles	----------	**6**
3.2 Discussion on problem solutions based on published articles	---------	**7**
3.3 Comparison of leading solutions	---------- ---	**8**
3.4 Recommended Approach	----------	**9**

Chapter 4: Methodology **10-14**

4.1 What to use---------- **10**

4.2 Why to use---------- ---------- **10**

4.3 Section of Methodology---------- **11**

4.4 Implementation Plans---------- **14**

Chapter 5: Planning **15-18**

5.1 Project Plan ---------- **15**

5.1.1 Management Plan ---------- **16**

5.1.2 Resource Allocation ---------- **17**

5.1.3 Time Boxing ---------- **18**

Chapter 6: Feasibility **19-21**

6.1 All Possible types of feasibility ……..... **19**

6.2 Cost Benefit Analysis ……...……... **20**

6.3 DSDM Dynamic system Development Method ……... **21**

Chapter 7: Foundation **22-29**

7.1 Some Potential Approaches---------- **22**

7.2 Specific problem are identification and description……. **25**

7.3 Possible solution…… **25**

7.4 Overall Requirement list……... **26**

7.5 Which technology to be implemented…….. **27**

7.6 Recommendation and justifications……. **29**

Chapter 8: Exploration **30-41**

8.1 Use case Diagram---------- **30**

8.2 Activity Diagram ---------- **31**

8.3 Requirement Catalogue---------- **32**

8.4 Prioritized Requirement List (PRL)---------- **33**

8.5 Prototype of new system---------- **34**

Chapter 9: Engineering **42-44**

9.1 Class diagram---------- **42**

9.2 ER Diagram---------- **43**

9.3 Sequence Diagram---------- **44**

Chapter 10: Development **45-64**

10.1 Core Module Samples---------- **45**

10.2 Probability problem break down---------- **63**

10.3 Prioritization while developing---------- **64**

Chapter 11: Testing **65-66**

11.1 Test Plan Acceptance---------- **65**

11.2 Unit Testing ---------- **65**

11.3 Validation Testing---------- **65**

11.4 Integration Testing---------- **66**

11.5 Test Cases---------- **66**

Chapter 12: Implementation **67-73**

12.1 Training---------- **67**

12.2 Big Bang Implementation---------- **67**

12.3 Scaling---------- **68**

12.4 Experiment result---------- **73**

Chapter 13: Critical Appraisal and Evaluation 74-77

13.1 Objective that could be met---------- **74**

13.2 Objective totally not met---------- **77**

Chapter 14: Lessons Learned 78-79

14.1 Pre-project ---------- **78**

14.2 Review ---------- **78**

14.3 Lessons Learned ---------- **79**

14.4 Problem faced ---------- **79**

14.5 Problems that are solutions ---------- **79**

Chapter 15: Conclusion 80-82

15.1 Summary of the project ---------- **80**

15.2 Goal of the project ---------- **80**

15.3 Success Of the projects---------- **81**

15.4 Documentations ---------- **81**

15.5 Value of the project ---------- **81**

15.6 My experience ---------- **82**

References ---------- 83

LIST OF TABLES

TABLES	PAGE NO
Table 3.1: Modules descriptions	8
Table 5.2: Managing planning	16
Table 5.3: Resources allocation	17
Table 5.4: Time Boxing	18
Table 6.5: Cost Benefit	20
Table 8.6: Prioritized requirement list	33
Table 11.7: Test Case	66

LIST OF FIGURES

FIGURES	PAGE NO
Figure 3.1 video demo image	9
Figure 4.2 DSDM phase	13
Figure 7.3 Sample of dataset	23
Figure 7.4 Model for the whole research project.	24
Figure 8.5 Use case Diagram.	30
Figure 8.6 Activity Diagram.	31
Figure 8.7 Interfaces for user	41
Figure 9.8 Class Diagram.	42
Figure 9.9 ER diagram.	43
Figure 9.10 Sequence Diagram	44
Figure 10.11 Code samples	62
Figure 11.12: Output of results	73

CHAPTER 1

Introduction

1.1 Introduction

Systems are applications that display the connections and exchanges between several programs. Computers' "System" page contains applications, programming links, and system administration tools. Although the word "system" might indicate different things depending on the context, the concept is essentially the same. The "Vehicle Counting for traffic ease using YOLOv8" integrates several technologies to create a comprehensive basis. The several components that make up this framework apply limitations to each system that is being examined. There are several systems in each module. The primary objective of the research was to use the YOLOv8 model to develop an automated system that could accurately count automobiles in traffic conditions in real time.

Using the YOLOv8 model, a cutting-edge object identification algorithm renowned for its speed and accuracy, this research aims to create an automated car counting system. Effective traffic management becomes increasingly difficult as cities continue to expand. Conventional traffic monitoring techniques that should be more frequently use human counts or less precise technology, which results in ineffective for their decision-making. The project intends to develop a system-techniques that should be more that can precisely identify and count automobiles traveling through particular locations, such crossroads, highways, or toll booths, by utilizing YOLOv8, which is excellent at real-time techniques that should be more object identification. In order to make data-driven decisions about techniques that should be more infrastructure upgrades, traffic signal optimization, and general traffic control strategies, city planners for all details and traffic management authorities need this information. The system also offers scalability for traffic monitoring techniques that should be more in multiple locations, minimizes errors, and boosts for all data efficiency by minimizing human intervention.

In light of the increasing techniques that should be more need for sustainable and efficient urban planning, this project offers a state-of-the-art way to details data to create the improve road safety, manage traffic, and help create smarter cities.

The way the system operates techniques that should be more is by examining video streams from security cameras positioned for datasets thoughtfully around roadways, highways, and all cars crossings. YOLOv8 allows traffic data collection techniques that should be more that should be more without the need for human involvement by detecting, classifying, and all running counting cars as they move across the camera's field of view. Continuous monitoring is made possible techniques that should be more by this method, which is essential for cars identifying traffic patterns, assessing congestion, and comprehending peak for all traffic periods. Traffic authorities may utilize the data techniques that should be more produced by the system to better plan future infrastructure all details projects, modify traffic signals, and increase the all road safety. The main benefit of employing YOLOv8 in this project techniques that should be more is its capacity to precisely identify automobiles across a range of circumstances, such as more decisive varying illumination, weather, and traffic volumes. Additionally, the system is very scalable, which means that techniques that should be more it may be set up in different parts of a city or even in different cities without requiring a major increase techniques that should be more in cost or complexity. The technology produces more accurate for control the running huge traffic statistics by automating the vehicle counting techniques that should be more process, which eliminates the need techniques that should be more for human labor and lowers the possibility of data types gathering mistakes. Additionally, because it requires little hardware improvements techniques that should be more to operate on current camera networks, the initiative provides a more data effectiveness for all affordable option than previous techniques.

CHAPTER 2

Initial Study

2.1 Project Proposal

Objectives

Using the YOLOv8 model, the project's need was to create a system-techniques that should be more could precisely count cars in traffic situations. Given the growing demand for effective traffic control and monitoring techniques that should be more helpful, traffic management techniques that should be more can benefit greatly from the data that an automated all various vehicle counting system can offer.

- Create a model that can identify and tally cars in pictures or movies.
- Assess the accuracy and speed of the YOLOv8 model's performance.
- Examine the findings to offer perspectives on the number of vehicles in various settings.

Websites advantages:

- Applying the YOLOv8 model, Create a model that can identify and tally cars in pictures or movies.

2.2 Background of the Project

The project's origin stems from the growing demand for effective model techniques for vehicle car count precisely traffic management in cities as a result of urbanization, population expansion, and the accompanying increase in the number of cars on the road. As traffic volumes increase, traditional traffic monitoring model techniques for vehicle car count precisely like manual counts and outdated sensor-based technologies are losing their effectiveness and frequently resulting in delays and human error. The need for automated, real-time traffic tracking devices model techniques for vehicle car count precisely that can offer precise and scalable solutions has thus increased. The YOLO system, in particular, has transformed model techniques for vehicle car count precisely computer vision because of how quickly and accurately it can identify all objects in pictures and video streams.

The most recent iteration model techniques for vehicle car count precisely of the YOLO model, YOLOv8, is renowned for its capacity to process photos with remarkable speed and precision. For real-time applications like datasets of traffic monitoring, where cars must be identified and tallied as they applied model techniques for vehicle car count precisely pass across a camera's field of view, it is especially well-suited. In order to collect dynamic traffic data model techniques for vehicle car count precisely without requiring human involvement, object identification methods such as YOLOv8 are able to both identify model techniques for vehicle car count precisely and track the movement of cars [1].

In recent years, there has been a notable surge model techniques for vehicle car count precisely in the study of AI and deep learning-based traffic monitoring systems. By accurately counting vehicles model techniques for vehicle car count precisely and identifying patterns of congestion, these technologies have the potential to enhance traffic management, according to several studies [2][3]. Because it strikes a compromise between high performance for model techniques for vehicle car count precisely and computing efficiency. The creation of such automated systems provides a route toward more intelligent model techniques for vehicle car count precisely, environmentally friendly urban transportation options as cities for data continue to struggle with traffic management in more congested settings.

2.3 Problem Area

The project's focus is on the shortcomings and inefficiencies of conventional traffic monitoring systems model techniques for vehicle car count precisely in handling the growing amount of urban traffic. Traffic congestion model techniques for vehicle car count precisely has grown to be a significant problem as cities expand, resulting in delays, higher emissions, and safety risks. The dynamic nature of contemporary traffic circumstances model techniques for vehicle car count precisely where variables like changing weather, different types of roads, and traffic flow patterns can affect the quality of data collecting, is another issue that model techniques for vehicle car count precisely these systems are unable to adjust to.

For traffic authorities and urban planners model techniques for vehicle car count precisely to effectively manage roadways, optimize traffic lights and lessen congestion, a more dependable, model techniques for vehicle car count precisely an automated solution has become essential. By using AI-powered model techniques for vehicle car count precisely like YOLOv8 for exact, real-time vehicle counts, this project seeks to address these issues and provide a scalable for traffic data and accurate answer to contemporary traffic management.

2.4 Possible Solution

- Utilize running camera video feeds to identify running cars in real time using model.
- Install top-notch security cameras to record traffic system for catching all cars details for datasets at strategic locations including junctions, freeways, and toll booths.
- To give more detailed model techniques for vehicle car count precisely for traffic analysis, the system categorizes various vehicle kinds, such as automobiles, trucks, and motorbikes.
- Automate the model techniques for vehicle car count procedure to improve accuracy and efficiency while lowering human error.
- For improved signal optimization and congestion control for data integrate model techniques for vehicle car count precisely integrate the vehicle counting data with the current traffic control systems.

CHAPTER 3
Literature Review

3.1 Discussion about problem domain based on published articles

Numerous studies have highlighted the difficulties faced model techniques for vehicle car count precisely by conventional systems in the issue domain of traffic monitoring and vehicle counting. These problems have been solved by recent developments as in data counting for sectors in AI and deep learning, especially with models like YOLO, which provide model techniques for vehicle car count precisely quicker, more precise, and scalable solutions. [4] showed how well deep learning models work in real-time techniques for vehicle car count precisely identification, greatly increasing accuracy over conventional techniques. Similar to this, [5] investigated the use of YOLOv4 and YOLOv5 to traffic surveillance using model techniques for vehicle car count precisely including varying illumination and weather conditions. These studies highlight how AI-driven technologies model techniques for vehicle car count precisely are increasingly being used in traffic management.

3.2 Discussion about problem solutions based on published articles

Recent studies demonstrate how well AI-driven solutions like model techniques for vehicle car count in particular, YOLO-based models—can overcome the drawbacks of conventional traffic monitoring systems. Manual counting and sensor-based model techniques for vehicle car count precisely are examples of traditional methods that have problems with accuracy, scalability, and real-time data processing for accurate data count. According to research by [6] and [4], YOLOv3—offer excellent vehicle identification accuracy and are capable of managing dynamic model techniques for vehicle vount situations. Furthermore, [7] showed how YOLO combined with cloud computing can scale, enabling effective traffic monitoring like as data spectacular and large-scale implementation. These developments suggest that AI is a potent instrument for enhancing model techniques for vehicle car count precisely and traffic management.

3.3 Comparison about leading solutions

1. **YOLO:** The good works in real-time object recognition, YOLO-based models like YOLOv3 and YOLOv4 are among the most used model techniques for vehicle car count precisely. Because of their reputation using data for quickness, they are perfect for monitoring traffic data of using model in real time. One of the greatest choices for real-time traffic model techniques for vehicle car count precisely recent YOLOv8, which has overcome many of these drawbacks by increasing detection accuracy while preserving fast processing rates [6].

2. **R-CNN:** In general, Faster R-CNN is slower than YOLO models, even if it is quite accurate model techniques for vehicle car count precisely in object recognition tasks. It is less appropriate for real-time model techniques for vehicle car count car counting in dynamic traffic situations due to its slower processing speed. Although it might not be the ideal option for extensive, real-time traffic monitoring model techniques for vehicle car count precisely it is usually utilized in applications where the maximum precision is necessary.

3. **Single Shot Multi-Box Detector:** SSD is an additional real-time object detection paradigm that balances model techniques for vehicle running count accuracy and speed. In situations with moderate traffic congestion, it is still often employed for model techniques for vehicle car count even though it is not faster or more accurate than YOLO. The main benefit of SSD like using all running car data is that it is more effective for systems with constrained processing resources and can operate with smaller networks. In situations with extremely high traffic, however, it still has difficulties model techniques for vehicle car count since the YOLO model performs more accurately than it does [6]

4. **Retina-Net:** Small and closely spaced objects may be detected with great accuracy using the one-stage object identification model Retina-Net techniques for vehicle car count precisely. By reducing the disparity between foreground and background objects, its special "focal loss" feature improves accuracy model techniques for vehicle car count precisely in difficult traffic situations. It is less suitable for real-time applications where speed is essential since it is frequently like rather all data slower than YOLO models.

3.4 Recommended Approach

Table 3.1: Modules descriptions

Actuators	Functions
User	• Input video files • Develop a model to detect and count vehicles in videos.

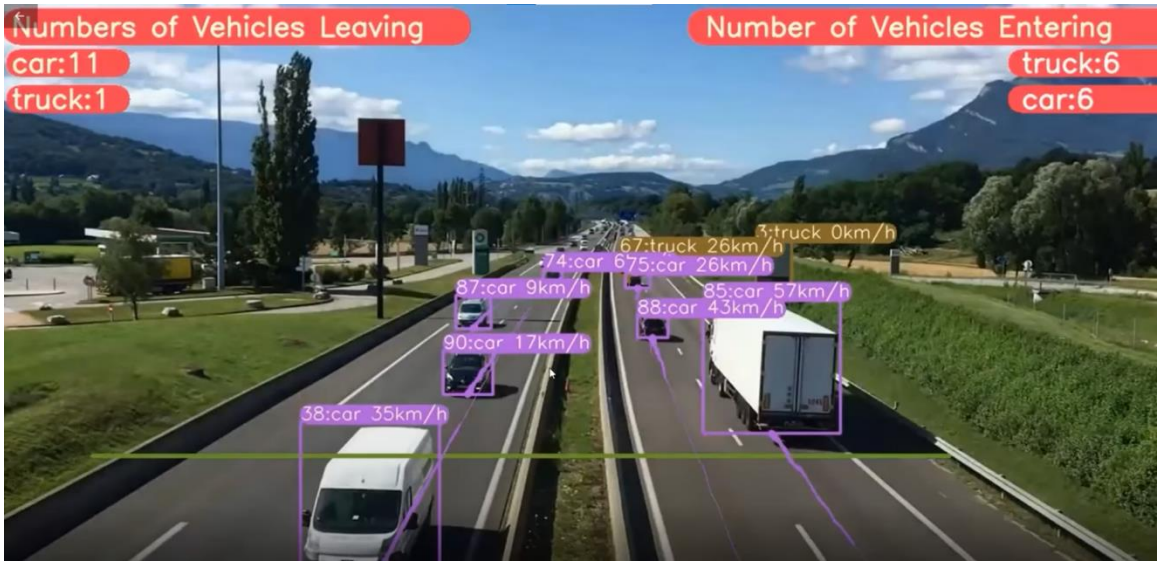


Fig 3.1 video demo image

CHAPTER 4

Methodology

4.1 What to Use

Main things of the research was to use the YOLOv8 model techniques for vehicle car count precisely to develop an automated system that could accurately count automobiles in traffic conditions in real time. This technology provides model techniques for vehicle count valuable information on traffic flow and congestion to assist all data sequence of traffic authorities in making educated decisions on infrastructure and traffic management. By automating car counts, the system increases efficiency, accuracy, and scalability, helping to fulfill the growing need for effective traffic control in urban areas. In this study, four distinct vehicle classes—cars, trucks, buses, and motorcycles—have been used for online application-based detection and counting. The goal of the research was to use the YOLOv8 model to create a system that could precisely count cars in traffic situations. Using python as a back end for this project. The SDLC life cycle paradigm, a thorough framework for development, design, etc., is acknowledged as the proper approach. I am familiar with a variety of SDLC model types. Software development paradigms include the Big Bang, the Spiral, a waterfall, Agile, Iterative, and Adaptive System Construction models. Every model offers a framework to aid in directing the creation and use of the platform for vehicle pieces. In order to create an effective development process that aligns with the goals of this project's object identification through running vehicle count, the specific requirements of the SDLC model will customize the veterinary development domain.

4.2 Why to use

The first step in the development process must be the system architecture. Part of this was figuring out the components and how they interacted. Security, reliability, and scalability were given top importance in the system network architecture. This involved separating the product's back-end and database management functions from its user interface. Furthermore, security features were incorporated into the architecture to ensure secure transactions and protect computer data. Every software project has to use the agile process.

Several of the words used to define agile methodologies are familiar to me, such as kanban, feature-driven development, scrum, quartz, and flexible system development methodology. Nevertheless, I succeeded in my objective by applying the DSDM approach. The benefits of the DSDM technique are numerous. The dynamic system design technique is used for iterative development, allowing for flexibility in modifying requirements. When timely delivery is required, this strategy is effective. Using an AI-powered vehicle count system offers a number of advantages, including lower costs, improved accuracy, early item recognition, and data-driven insights.

4.3 Section of methodology

A range of methods or strategies may be employed to decide how to evaluate the data that was used in this inquiry. This study used a multi-step technique that involved building the model, expanding and improving it, collecting data, and producing it.

Pre-Project Phase:

- Feasibility Study: This phase comprises evaluating the project concept's functional, financial, and infrastructure feasibility. It involves weighing the potential costs, benefits, and hazards of the project.
- Conditions Gathering: The program's requirements have now been compiled and documented. Understanding client desires, constraints, and business needs is necessary to determine the project's scope.
- Planning: Planning entails creating a strategic plan that specifies the objectives, timeline, required materials, and deliverables of the project. Establishing roles and responsibilities, identifying the project's stakeholders, and creating a partnership and risk management strategy are all crucial.

Project Lifecycle Phase:

- **Gathering Data:** I collected and analyzed online statistical data using Kaggle to create a trustworthy set of my own. Images and videos of traffic situations from several sources made up the dataset utilized in this study. Bounding boxes surrounding various vehicle kinds, including automobiles, trucks, buses, and motorcyclists, were added to each image.

- **Data preparing:** Following its collection in its data Many data sets can have errors, especially when there is noise present. In technical terms, I use the chosen data set to proceed to the next phase after processing the knowledge.
- **Data Preprocessing:** The results expanded and became more targeted as each class was assessed. To make it function, I had to add information and change the size. Because of my concerns about overfitting, I limited my alterations to the largest and most suitable ones.
- **Model Selection:** Once you've chosen a model, use the available data to train and evaluate it in order to improve accuracy. Pre-trained weights are used to initialize the YOLOv8 model using a sizable dataset, like COCO, which offers a solid foundation for fine-tuning. Several iterations of the idea were evaluated using my technology in order to identify the best setup for accurate data computation.
- **Evaluation of Performance:** Each of the results is explained in this section. Because of these strategies, even after training and testing, we were not reliable enough for the next two courses. In a variety of traffic scenarios, the YOLOv8 model showed strong performance in vehicle detection and counting. The model was able to reliably identify distinct types of cars and showed strong generalization to new, unseen photos. then created the confusion matrix, recall, efficiency, and f1 measurement visualizations.
- **Design:** Using the requirements that have been acquired, the software design is constructed in this stage. Database, architectural, and user interface design are only a few of the high-level and comprehensive design tasks it involves.
- **Development**: Using the design requirements as a guide, the program is coded at this step. To produce a functional system model that model count the running vehicle, the developer develops the source code, performs unit tests, and assembles components.
- **Testing**: By putting the program through its paces, this process aims to guarantee of all data that could be count running car quickly that it is both high-quality and functional. Unit, user validity, integration, and system testing that model techniques for vehicle count are just a few of the many testing techniques it covers.

- **Deployment**: Following approval and a thorough testing process, the model deploy should be used for complete project. The program must be installed, configured like model build for count cars and set up in the proper environment.

Post-Project Phase:

- **Maintenance:** The software occurs main difficulty into the maintenance stage following deployment. To make sure the program keeps functioning and adapts to changing all needed that should be fixed all error this phase entails regular maintenance, bug repairs, and upgrades.
- **Evaluation:** By comparing the project's main outcome that shows like model techniques for vehicle car count with its stated goals, its efficacy may be assessed. It assists in determining model techniques for running vehicles count to be changed and what can be learned for future projects.
- **Closure:** The project mainly finish at this following part. It includes completing the project's file for managing model techniques for vehicle car count monitoring its artifacts, and doing a project evaluation.

These parts offer a following important things need to done the project, to managing software development projects and aid in achieving effective results, from early planning to post-deployment like as an assistance.

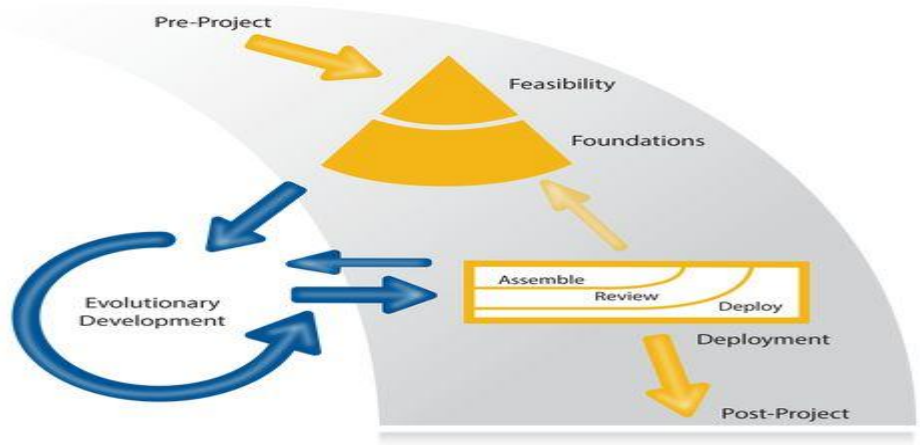


Fig 4.2: DSDM phase

4.4 Implementations plans

This phase has defines all details data into count vehicles for details dataset. Once a bug has been identified and fixed, the new system must be enabled. This section selects the settings, protocols, and release requirements. The upgraded system is then tested and put into use if everything works as planned. The data collection process must be completed after all other steps have been completed in order to guarantee accuracy. To make the assignment easier to do, I divided it into its most important components. I must abide by these guidelines to ensure that my job is completed accurately.

- Dataset collections.
- Steps performed prior to picture processing
- Forecasting class images for four vehicle classes.
- YOLOv8 algorithms that are employed.
- Counting running vehicles from videos using YOLOV8
- Assess the results and correctness.

CHAPTER 5
Planning

5.1 Project Plan

I publicly compiled all of my datasets using technologies like Kaggle. For the various photos, I chose a dataset that appeared to make sense. After that, I could go to work on getting the data ready. Before putting the concept into practice, I began experimenting with the programming. I evaluated the accuracy of each of the YOLOv8 models. I chose the one that would best serve my needs after taking accuracy into account. Following a thorough examination of all pertinent mathematical and philosophical concepts and methods, a set of fundamental standards has been developed. Prior to creation, all projects must have their potential, budget, timeline, risk management, and interaction server protocols defined. Prior to starting a project, planning is crucial to reducing risks that might jeopardize the developer's ability to complete it. Planning a project can involve several things, including establishing goals and objectives, managing risks, adhering to deadlines, and more. Time boxes are a fundamental tool for project planning and are frequently utilized in software project schedules.

5.1.1 Management plan

Describe the project team's duties and responsibilities as well as the project management procedure. Create the reporting and communication routes to guarantee a fruitful partnership. Determine the phases of the issue-resolution process for decision-making and escalation.

Table 5.2: Management Planning

No	Task Name	Duration	Start Date	End Date
1	Introduction	8	07-10-24	14-10-24
2	Initial Study	4	15-10-24	18-10-24
3	Literature Review	4	19-10-24	22-10-24
4	Methodology	4	23-10-24	26-10-24
5	Planning	12	27-10-24	09-11-24
6	Feasibility	15	10-11-24	25-11-24
7	Foundation	5	26-11-24	30-11-24
8	Exploration	14	30-11-24	14-12-24
9	Engineering	30	15-12-24	15-01-25
10	Deployment	18	16-01-25	04-02-25
11	Testing	10	05-02-25	15-02-25
12	Implementation	7	16-02-25	22-02-25
13	Critical Appraisal and Evaluation	5	23-02-25	28-02-25
14	Lessons Learning	4	01-03-25	04-02-25
15	Conclusion	1	05-03-25	06-03-25
	Total	141 days		

5.1.2 Resource Allocation

Determine all of the project's resources, such as staff, tools, and software. Based on the workload and project schedule, decide how best to allocate the resources. Assign duties and obligations to team members while making sure they possess the requisite knowledge and abilities.

Table 5.3: Resource Allocation

No	Task Name	Duration	Resource
1	Introduction	1	End User
2	Initial Study	2	Analyst
3	Literature Review	3	Analyst
4	Methodology	3	Analyst
5	Planning	7	Analyst, Designer, Developer
6	Feasibility	11	Analyst
7	Foundation	4	Designer
8	Exploration	9	Designer, Developer
9	Engineering	20	Developer
10	Deployment	12	Analyst, Developer
11	Testing	5	Analyst, Developer, Tester, Users
12	Implementation	6	Analyst, Developer
13	Critical Appraisal and Evaluation	3	Analyst, Tester and Developer
14	Lessons Learning	2	Analyst, Users
15	Conclusion	1	Analyst
	Total	86 days	

5.1.3 Time Boxing

To make development and testing easier, divide the project into many time periods or iterations. Determine the duration of each time box as well as the tasks and outputs required for every iteration. For every time box, establish specific objectives and supply resources.

Table 5.4: Time Boxing

Time -Box	Task Name	Duration	Resource
TB1	Introduction	1	End Users, Analyst
	Initial Study	1	Analyst
	Literature Review	2	Analyst
TB2	Methodology	3	Analyst
	Planning	5	Analyst, Designer, Developer
	Feasibility	2	Analyst
TB3	Foundation	3	Designer
TB4	Exploration	2	Designer, Developer
	Engineering	8	Developer
TB5	Deployment	3	Analyst, Developer
	Testing	2	Analyst, Developer, Tester, Users
TB6	Implementation	7	Analyst, Developer
TB7	Critical Appraisal and Evaluation	2	Analyst, Tester and Developer
	Lessons Learning	2	Analyst, Users
TB8	Conclusion	1	Analyst
	Total	44 days	

CHAPTER 6

Feasibility

6.1 All possible types of feasibility

6.1.1 Operational feasibility

A feasibility study assesses the possibility that all pertinent elements—such as engineering, planning, legal, and financial considerations—will be taken into account to ensure a project's successful completion. The degree to which a system matures, utilizes the scope defined during opportunity definition, and satisfies the criteria specified during the project or necessity analysis phase of the development process is known as operational practicability. The mechanism at the heart of the suggested concept uses pictures to identify moving automobiles.

6.1.2 Technical feasibility

Hardware	Software
Dell Laptop, Wi-Fi, Router, Cable, Android Phone	Android Studio, Google Chrome Browser, Windows, MS Word, VS code

6.1.3 Technology

Server side	Models
Python, Dataset	YOLOv8

6.2 Cost Benefit Analysis

Project managers analyze the advantages and disadvantages of various project paths, including interactions, activities, business demands, and investments, using the cost-benefit analysis concept. A cost-benefit analysis helps me choose the optimal course of action to achieve my goal at the lowest possible cost out of all the possibilities available.

Project Name: Vehicle Counting using YOLOv8

Table 6.5: Cost Benefit

Equipment	1st Year	2nd Year	3rd Year	4th Year	Total
Data Collection	20000				20000
Data preprocessing		10000	10000	10000	30000
Software	1000				1000
Internet	2000	2000	2000	2000	8000
Model Training	5000				5000
Development		5000			5000
Maintenance	10000	10000	10000	10000	4000
Total					73,000 BDT.

6.3 DSDM Dynamic System Development Method (DSDM)

DSDM, short for Dynamic Systems Development Method, is not a particular application development tool or technology but rather an organizational framework for agile project and software development management. It strongly emphasizes the use of iterative development methods, regular software delivery, and cooperation between development teams and business stakeholders. It's crucial to remember that DSDM requires the use of specific tools or technologies, such as Python.

CHAPTER 7
Foundation

7.1 Some potential approaches

7.1.1 Interview

The scheduling of interviews is critical to any endeavor. to assist users in using the running cars count system to solve problems by using pictures or videos, paying attention to user requests, and conducting interviews with suitable people. I can learn a lot about object detection by conducting interviews. I may learn about all the needs and any communication barriers with resource allocation by participating in interviews.

7.1.2 Observation

A trend toward using cutting-edge AI and real-time data processing for effective traffic management is highlighted by the top web-based traffic monitoring and vehicle counting programs. By offering scalable, reasonably priced technologies for vehicle categorization, counting, and data analysis, Rekor and Telraam are empowering communities and cities. These apps provide precise, automated traffic data by utilizing technology like cloud computing and machine learning, which aids urban planners in making well-informed choices about infrastructure and traffic control. These systems' use of AI greatly increases traffic monitoring's precision and effectiveness, making it a vital instrument for smart cities and contemporary transportation management.

7.1.3 Data Collection

In all, I have amassed 10,000 photos. Images and videos of traffic situations from several sources made up the dataset utilized in this study. Bounding boxes surrounding various vehicle kinds, including automobiles, trucks, buses, and motorcyclists, were added to each image. The four categories in the dataset—Car, Truck, Bus, and Motorcycle—are named for the types of vehicles they represent. Fifty percent of the data are utilized for testing and validation, while twenty percent are assigned for training.

- Number of images: 10000
- Classes: Car, Truck, Bus, Motorcycle
- Annotation format: YOLO format with class labels and bounding box coordinates

Fig 7.3: Sample of dataset

7.1.4 Data Processing

Resizing photos, leveling pixel values, and enhancing data via rotation, flipping, and scaling are some of the preprocessing methods used to promote variety in the collection.

- Pictures with code-based preset sizes.
- A JPG conversion will be performed on the file kinds.
- Remove any inaccurate pictures.
- Deleted unnecessary images

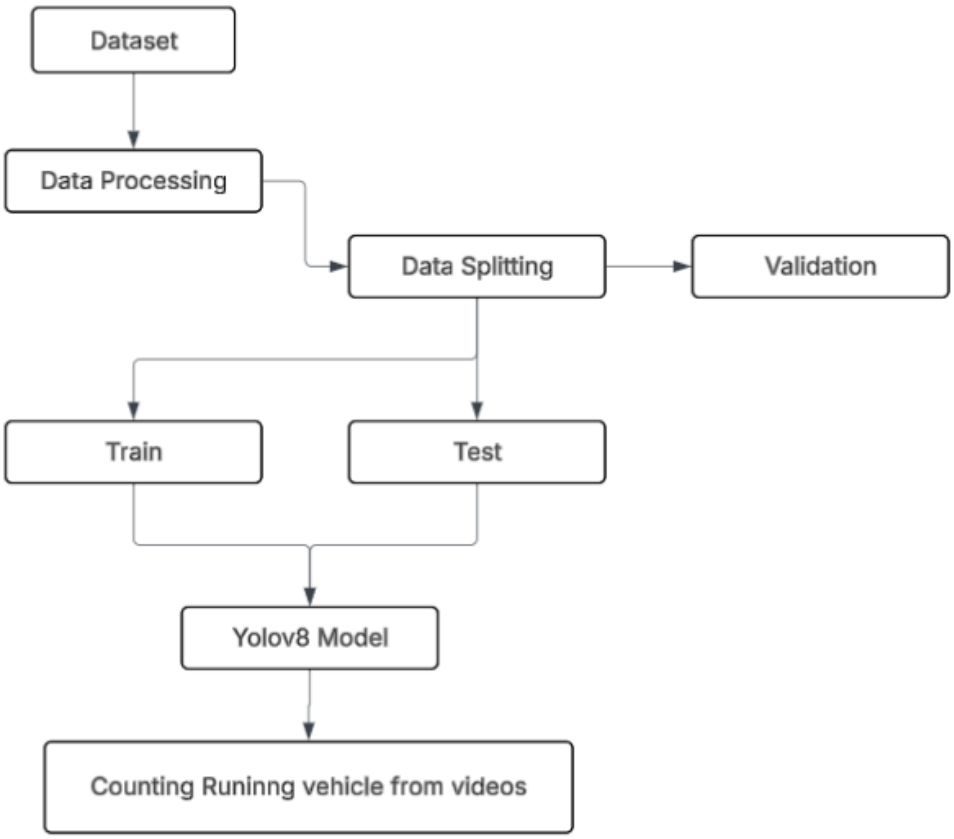


Fig 7.4: The recommended model for the whole research project.

7.2 Specific problem are identification and description

The challenge of precisely counting and categorizing cars in real-time, especially in congested and dynamic traffic conditions, has been noted as a distinct issue in traffic monitoring. Conventional techniques, such human counting or sensor-based systems, sometimes have issues with accuracy, scalability, and flexibility to changing circumstances like road kinds, illumination, and weather. These systems may need a lot of maintenance and human involvement, overlook smaller cars, or not identify cars in heavy traffic. Furthermore, the absence of real-time data may make it more difficult to make wise decisions on traffic control and urban development. In order to offer accurate, real-time vehicle data that can support improved traffic flow, infrastructure planning, and safety measures, more dependable, scalable, and automated systems are desperately needed as cities expand and traffic congestion gets worse.

7.3 Possible solution

Using of-vehicles counting systems with object identification models like YOLOv8 is one potential model techniques for vehicle car count remedy for the highlighted traffic monitoring issue. Real-time model techniques for vehicle car count and classification from surveillance camera video feeds is possible with these methods. Especially in situations with heavy traffic, model techniques for vehicle running count is renowned for its fast-processing speed and accuracy. This system may be expanded to cover several places model techniques for vehicle without sacrificing speed thanks to cloud computing. Immediate traffic flow monitoring is made possible by the incorporation of real-time analytics that should be decisive all car details, which aids in traffic signal that define model techniques for vehicle car count precisely optimization, congestion management, and safety enhancement. The automation of the system decreases human involvement, lowers mistakes, and produces consistent, trustworthy total car counting model techniques for vehicle car count precisely that can help with long-term infrastructure development and urban planning.

7.4 Overall Requirement List

- Functional Requirements
- Non-Functional Requirements.

7.4.1 Functional Requirements

7.4.1.1 User

- Making a system for count running vehicles for model techniques.

7.4.2 Non-Functional Requirements

7.4.2.1 Performance

Getting good accuracy for vehicle detection using a model-techniques for vehicle car count accurately, performance has been good.

7.4.2.2 Availability

Need an internet access for run the model techniques for vehicle car count precisely like data or car count. Also use some feature to use the model like: chrome, explorer etc.

7.4.2.3 User Friendly

The system for count car is very good as UI:

- Train and validate using a labeled dataset of traffic situations.
- Use the YOLOv8 model to detect vehicles.
- To assess the model's performance, test it using unobserved data.

7.5 Need able technology implemented

In this model works, I have been used following below languages or techniques:

Python: Python is a extreme languages, interpreted programming languages that is so much effective and good for doing all works and making it a great-things for the both novice and seasoned professionals. Several programming systems also details like as paradigms are supported like data file it, such as imperative, functional, and object-oriented programming. With the help of its many library function and following data requirements like: NumPy, Pandas, and TensorFlow, developers may effectively complete challenging jobs. Because of its robust community model techniques for vehicle car count and Python is one of the most widely used and adaptable programming system for count running car.

YOLOv8: The well real-time object detection model techniques for vehicle car count YOLOv8 (You Only Look Once version 8) is renowned for its accuracy and quickness. It is quite effective since it can identify several items model techniques for vehicle car count in a single frame of an image or video. By providing more accuracy, particularly in difficult settings with tiny or crowded objects, model techniques for vehicle car count precisely Because of its rapid and precise data processing capabilities for all count car which is running. The model is a great option for real-time detection model techniques for vehicle car count because it is intended to be both data which real time that should be occurred and performant. A field of study that is being examined and evaluated to elucidate concepts for creating models, establishing and accomplishing objectives, collecting data, controlling, instructing, and enhancing performance is known as a researched subject. I review my tools and methods for taking measurements. NumPy enabled Python programming, Microsoft software, and other tools such as Scikit-learn. Google Co Labs' facilities are only utilized for instruction and assessment. Google Colab programmers may write sophisticated machine learning algorithms and Python-based data analysis.

Libraries:

- **Matplotlib**: The visualization features of Matplotlib include a set of methods called Py-plot graphing. Drawing boundaries and defining lines inside a plot are only two of its numerous applications. Another is shapes.
- **NumPy:** One popular approach for working with matrices in Python is to use the NumPy module. The Fourier transform, matrix operations, and the fundamentals of algebraic geometry are covered in this section. The NumPy library in Python offers tools and resources for working with matrices of various sizes. Building arrays deliberately and precisely is made feasible by NumPy. The NumPy Python library is used for numerical calculations. Additionally, the phrase "a wide range of variants Python" is employed.
- **Scikit-learn**: This foresight and data analysis tool is practical and easy to use. Open-source software being freely available for anybody to use and modify as they see fit. Matplotlib, SciPy, and NumPy were used through growth.
- **Seaborn**: This popular data visualization tool is simple to use and provides an easy interface for creating visually beautiful and interesting data representations. It has a long history of working with matplotlib.
- **CV2:** To precisely solve computer vision-related issues, a set of Python extensions called Python's OpenCV was created. By examining pictures, videos, and notes taken at the incident, it may also be used to identify things and people.
- **Job-lib:** Another more efficient method that saves time and money by avoiding repeating the same calculation.
- **H5 Py**: A Python HDF5 native data wrapper is provided via the h5py package. HDF5's support for NumPy makes it simple to manage and store large amounts of numerical data.
- **OS:** The Python programming language's OS section provides a range of tools that artists may use to interact with the software that powers certain parts of the computers they use for their work.

7.6 Recommendation and justifications

Because of its great accuracy, speed, and scalability, a YOLOv8-based system is advised for automated vehicle counting and traffic monitoring. YOLOv8 is the perfect answer for dynamic surroundings since it can identify several cars in real-time, even in thick traffic. Because of the automation of the system, less manual counting is required, which minimizes human mistake and ultimately lowers expenses. Consistent performance is guaranteed by its capacity to adjust to different weather and lighting situations, and the real-time data it produces aids in improving urban planning and traffic management, which enhances safety and traffic flow.

CHAPTER 8

Exploration

8.1 Use case

Both functional and non-functional needs are analyzed in this part using use-case data and images.

User:

Following their system, the user can carry out the following tasks:

- Input video files
- Develop a model to detect and count vehicles in videos.

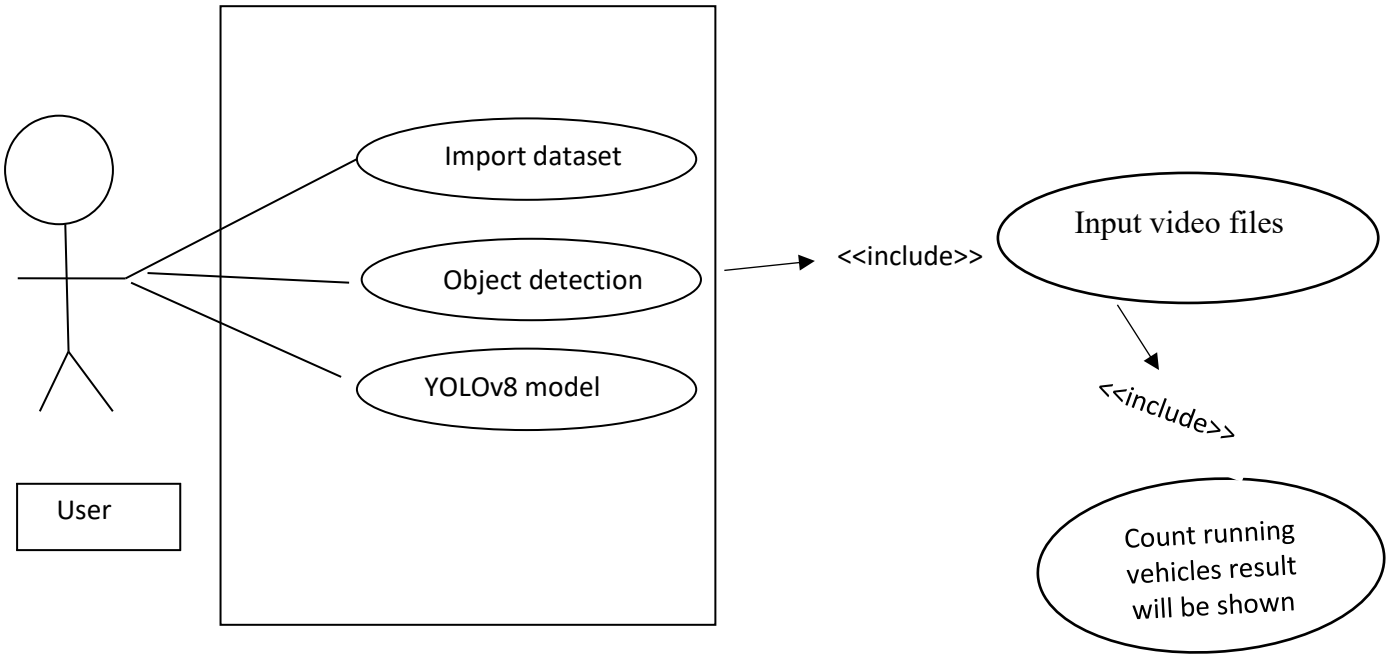


Fig 8.5: Use case Diagram

8.2 Activity diagram

Explain the system's dynamic features. It resembles a flow chart that illustrates the connections between several jobs. You might use the activity to explain the operation of the system. As a result, control is shared across all operations. The whole activity diagram for each module appears as follows:

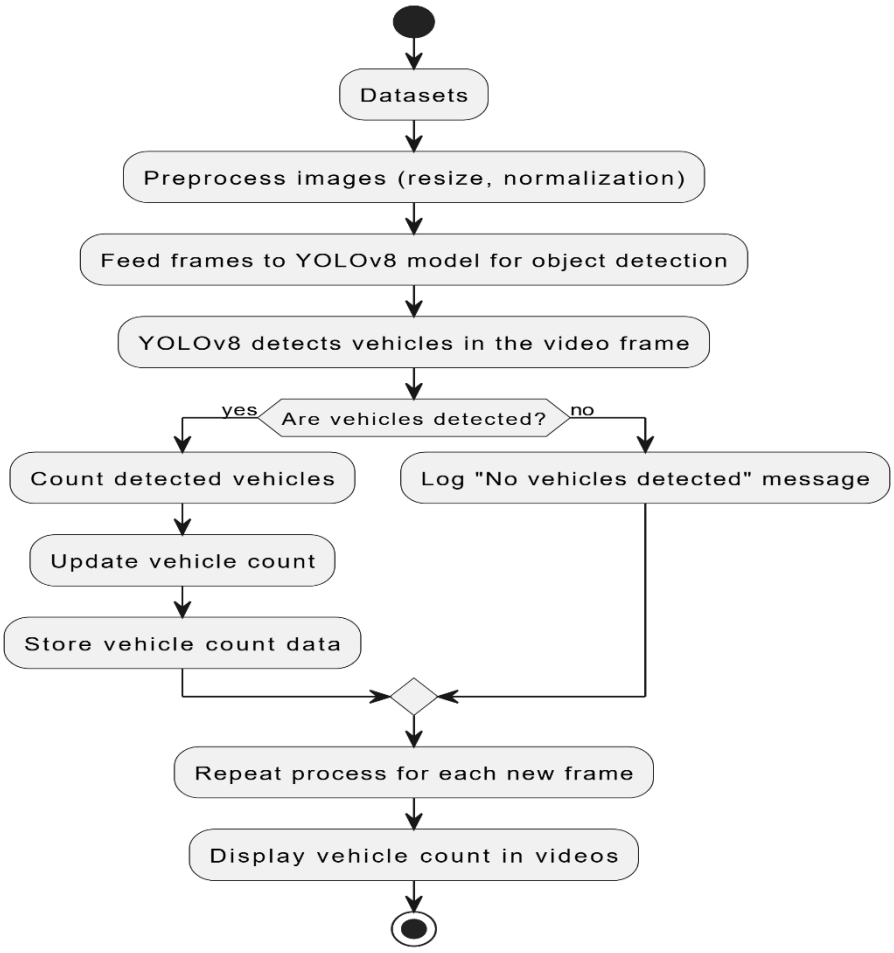


Fig 8.6: Activity Diagram.

8.3 Requirement catalogue

Functional requirements:

- FR1: Input video files
- FR2: Develop a model to detect and count vehicles in videos.

Non-Functional Requirements:

- NFR1: Updating and maintaining records is simple.
- NFR2: Users simply need a PC with an Internet connection to access the system throughout the day and from any place. The system is compatible with several web browsers, such as Chrome, Mozilla, Opera, and Internet Explorer.
- NFR3: The interface is engaging and the technology is easy to utilize.
- NFR4: View corporate data and contact information.

User Interface Requirements:

- UIR1: A user-friendly interface with intuitive navigation that makes accessing various functions and capabilities simple.
- UIR2: Icons offer visual signals to help users comprehend and operate the system.

Security and Privacy Requirements:

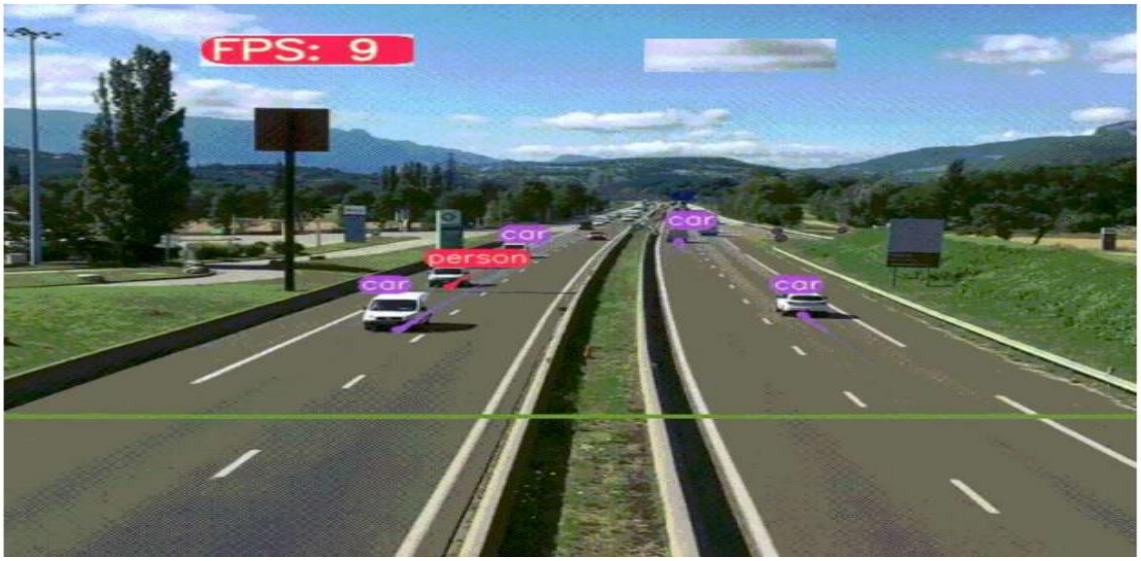
- SR1: Secure authentication and authorization procedures to safeguard user information.

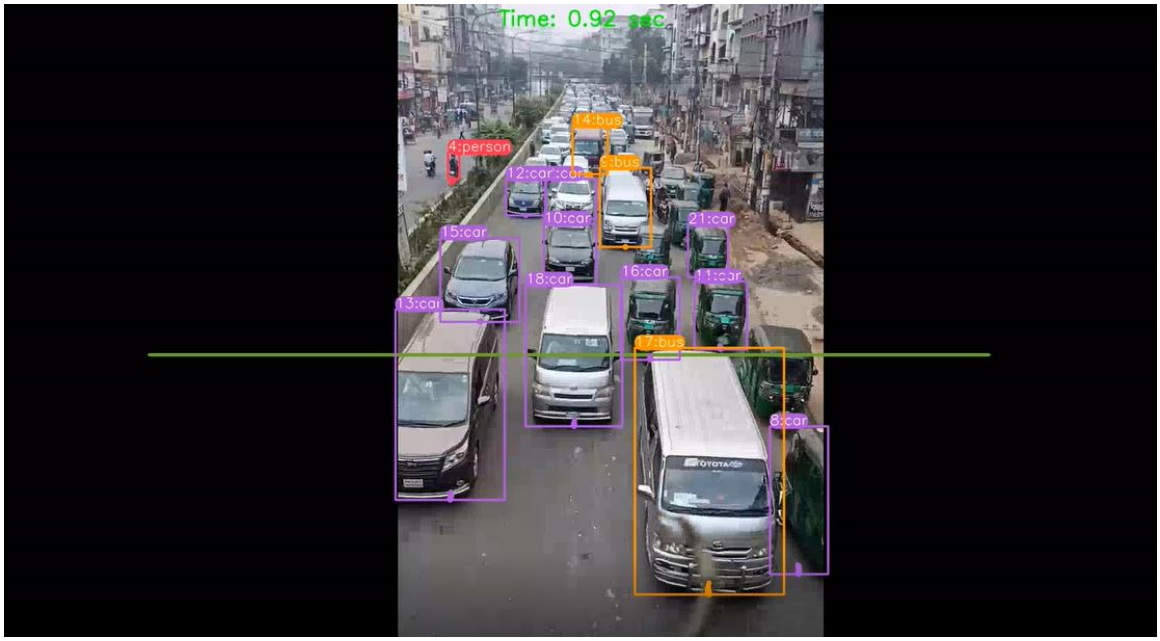
8.4 Prioritized Requirement List (PRL)

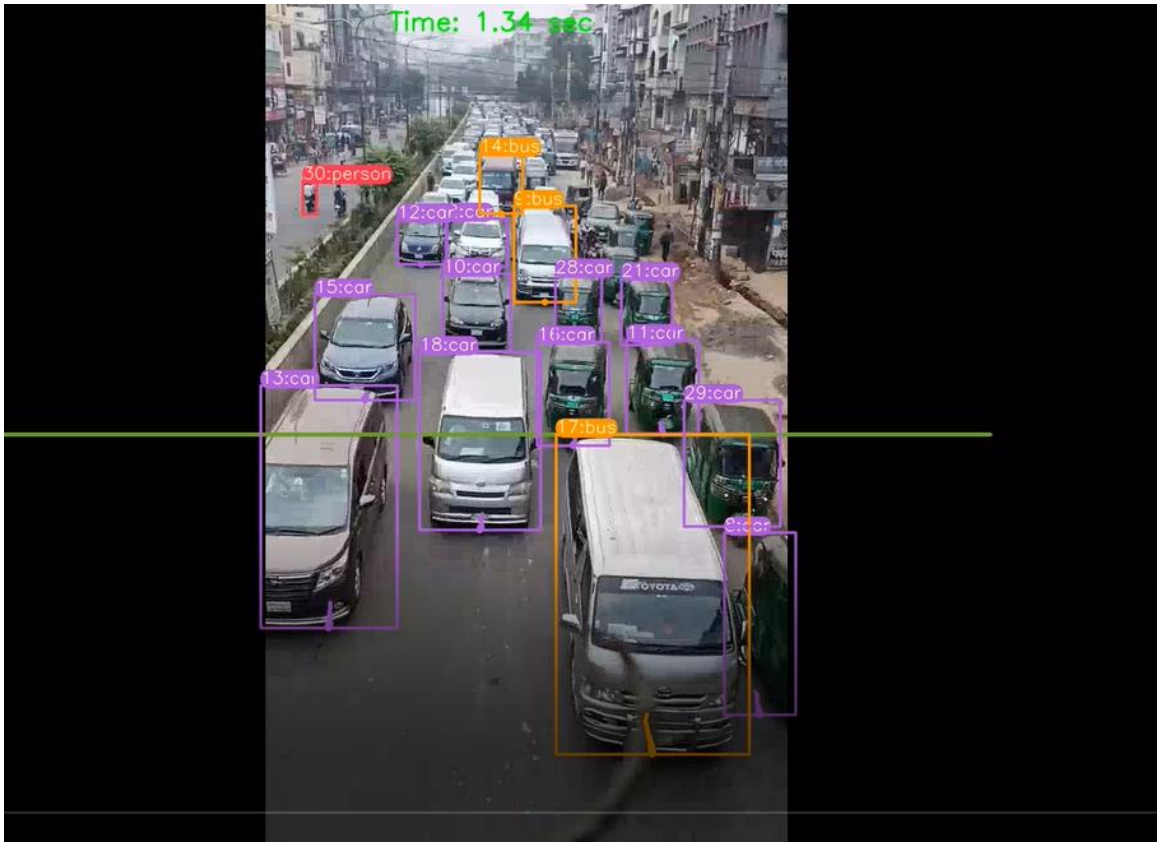
Table 8.6: Prioritized requirement list

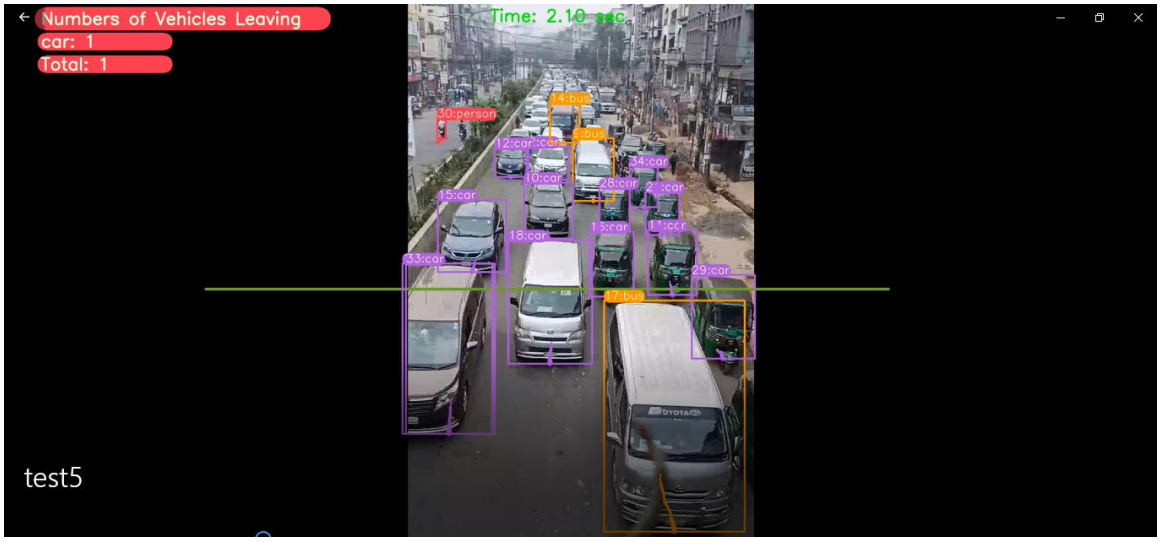
Requirement ID	Requirement Description	Priority	Dependencies	Status	Validation Criteria
RQ1	Collection datasets	High		Pass	Data collection successful
RQ2	Apply YOLOv8 models	High	RQ1	Pass	Model apply successfully
RQ3	Count running vehicles	High	RQ1, RQ2	Pass	Successfully count running vehicles.

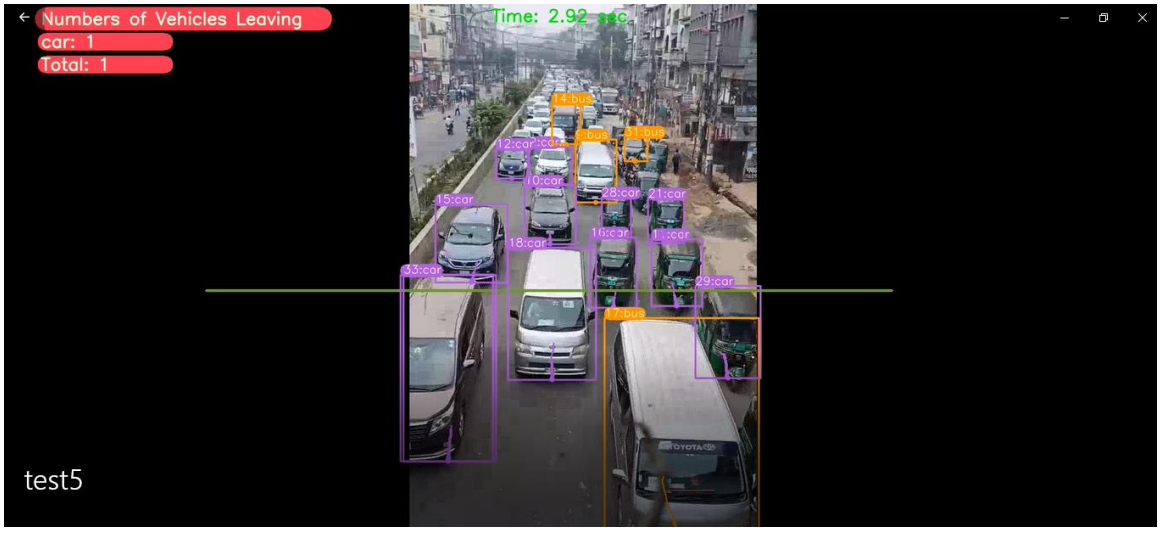
8.5 Prototype of new system

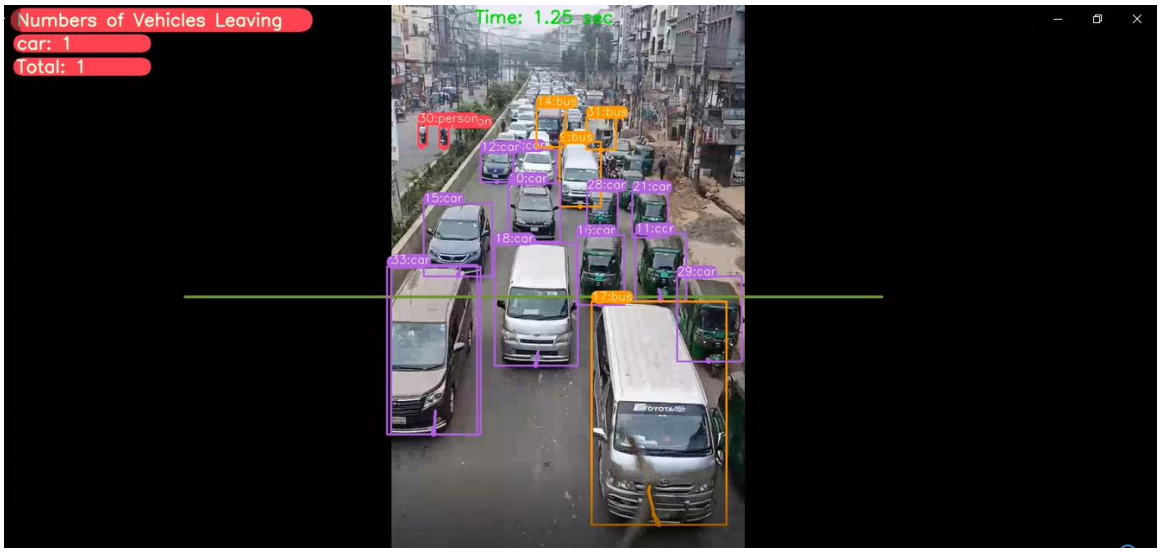


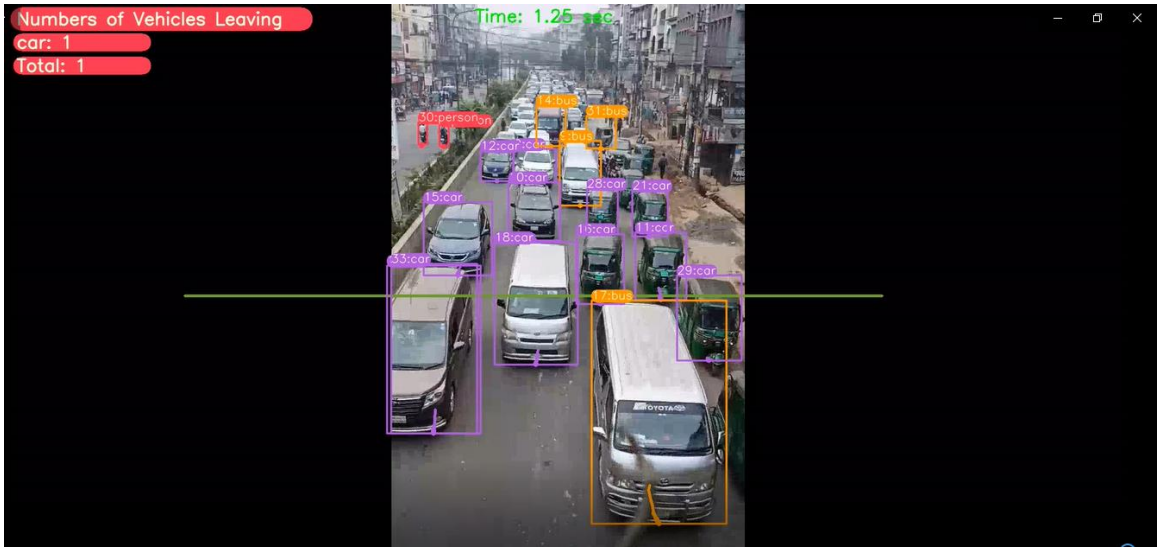


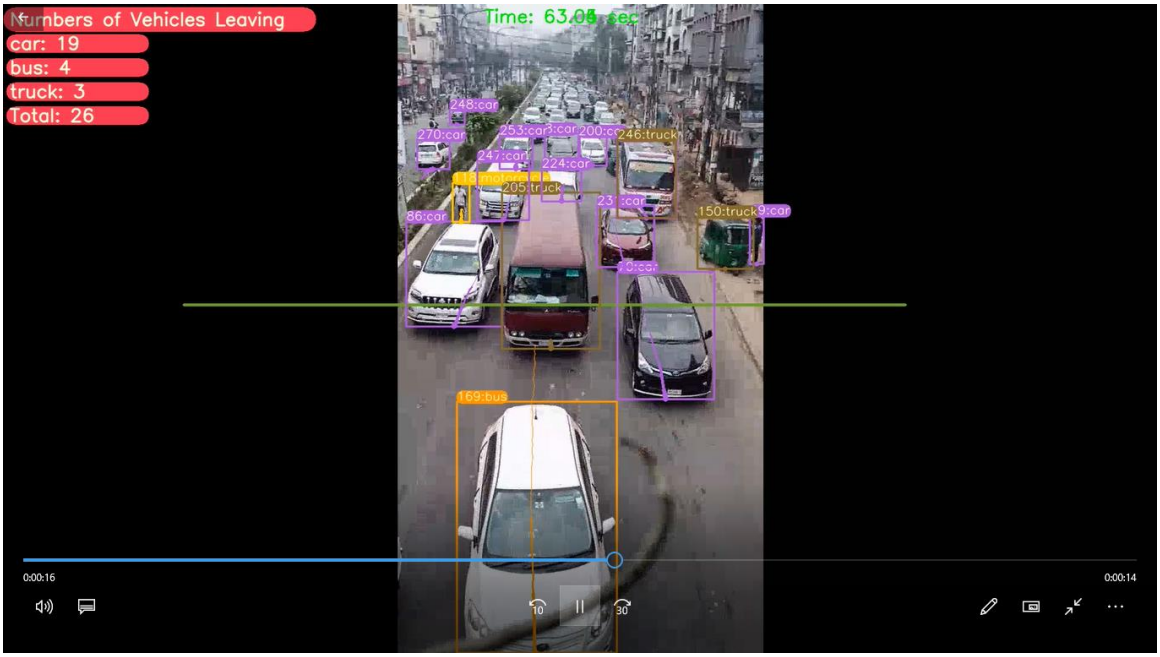


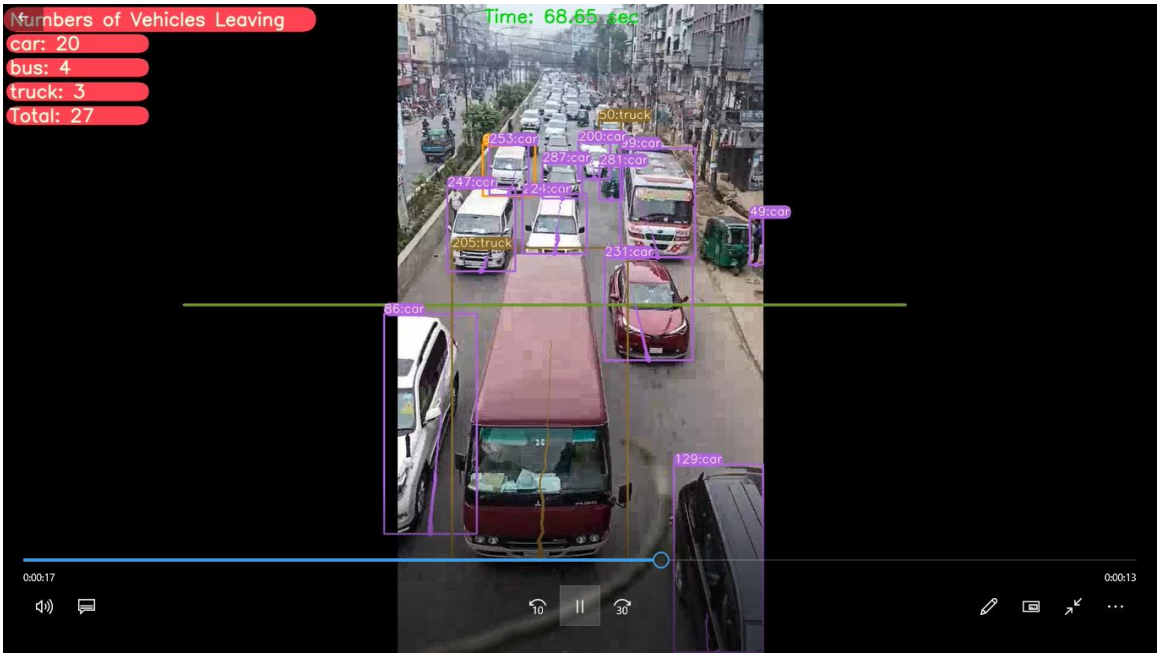


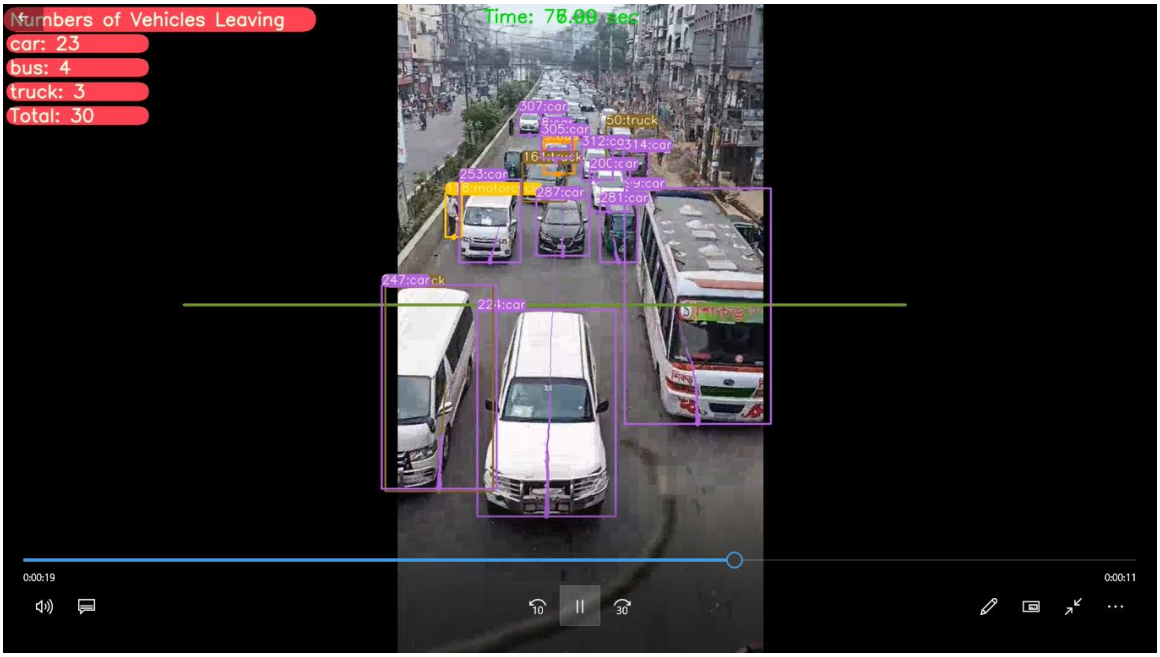


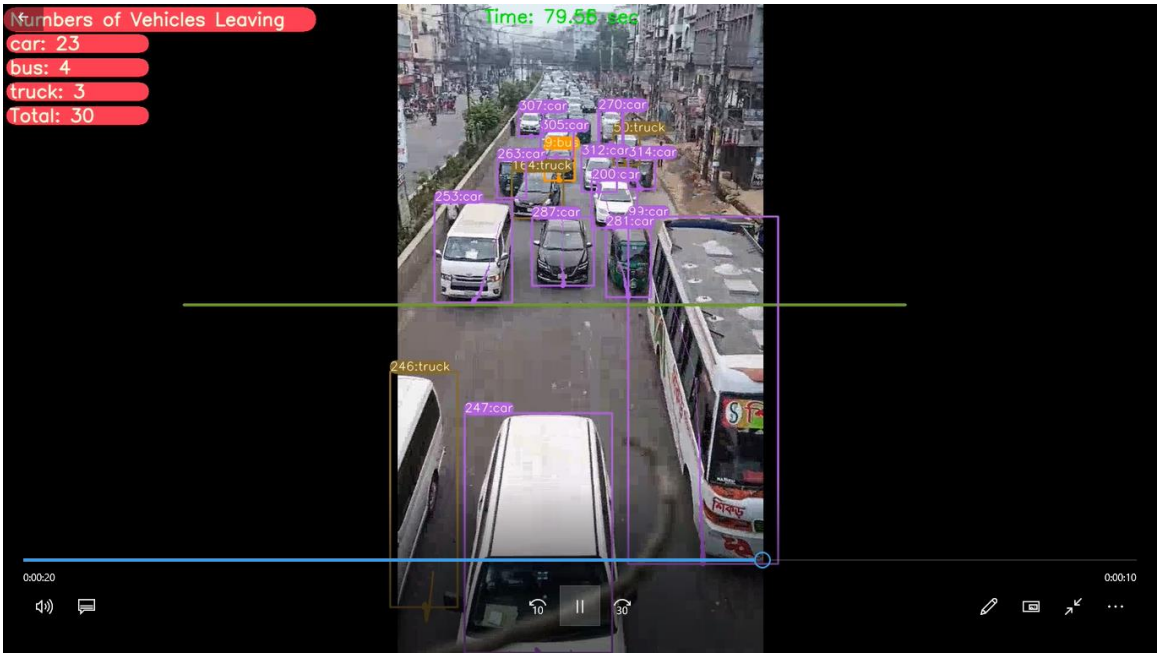


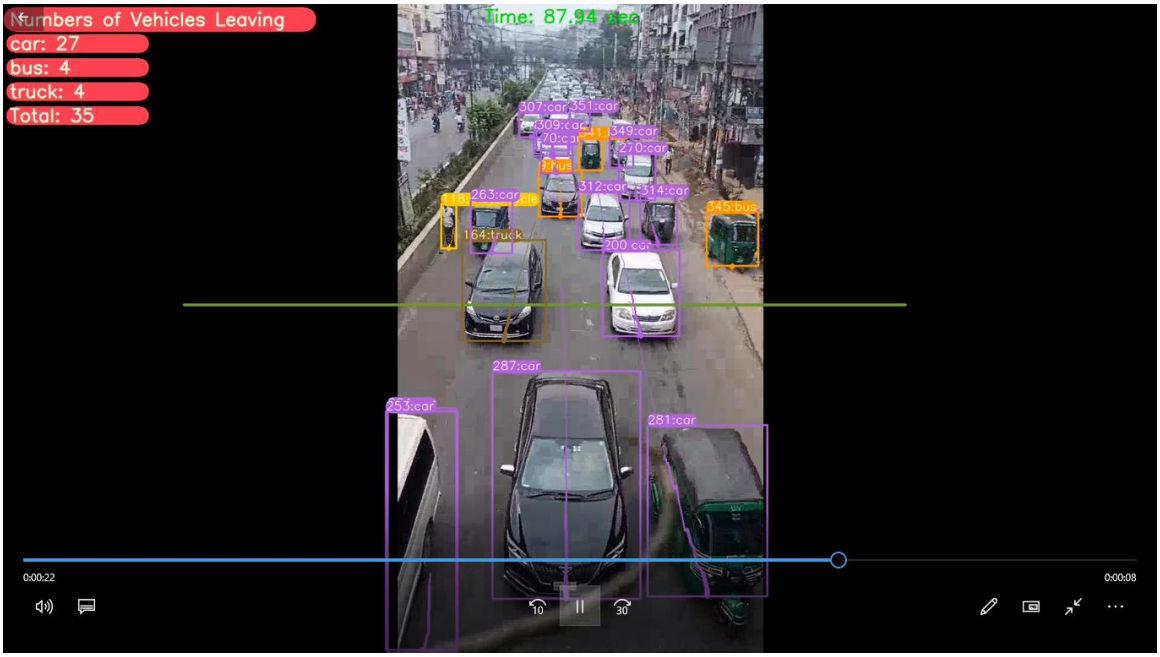












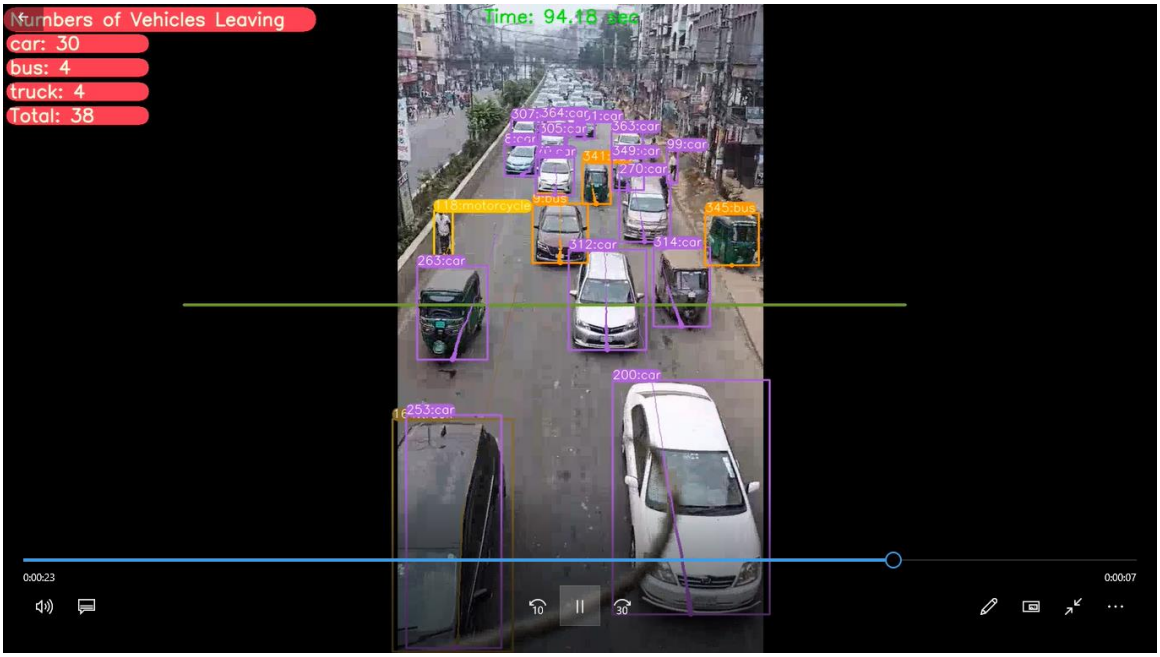


Fig 8.7: Interfaces for user

CHAPTER 9

Engineering

9.1 Class Diagram

A class was created in order to show the original material for the interclass links. In this case, the class describes an object's variables and actions, either as a separate entity inside a program or as a single specification of programming.

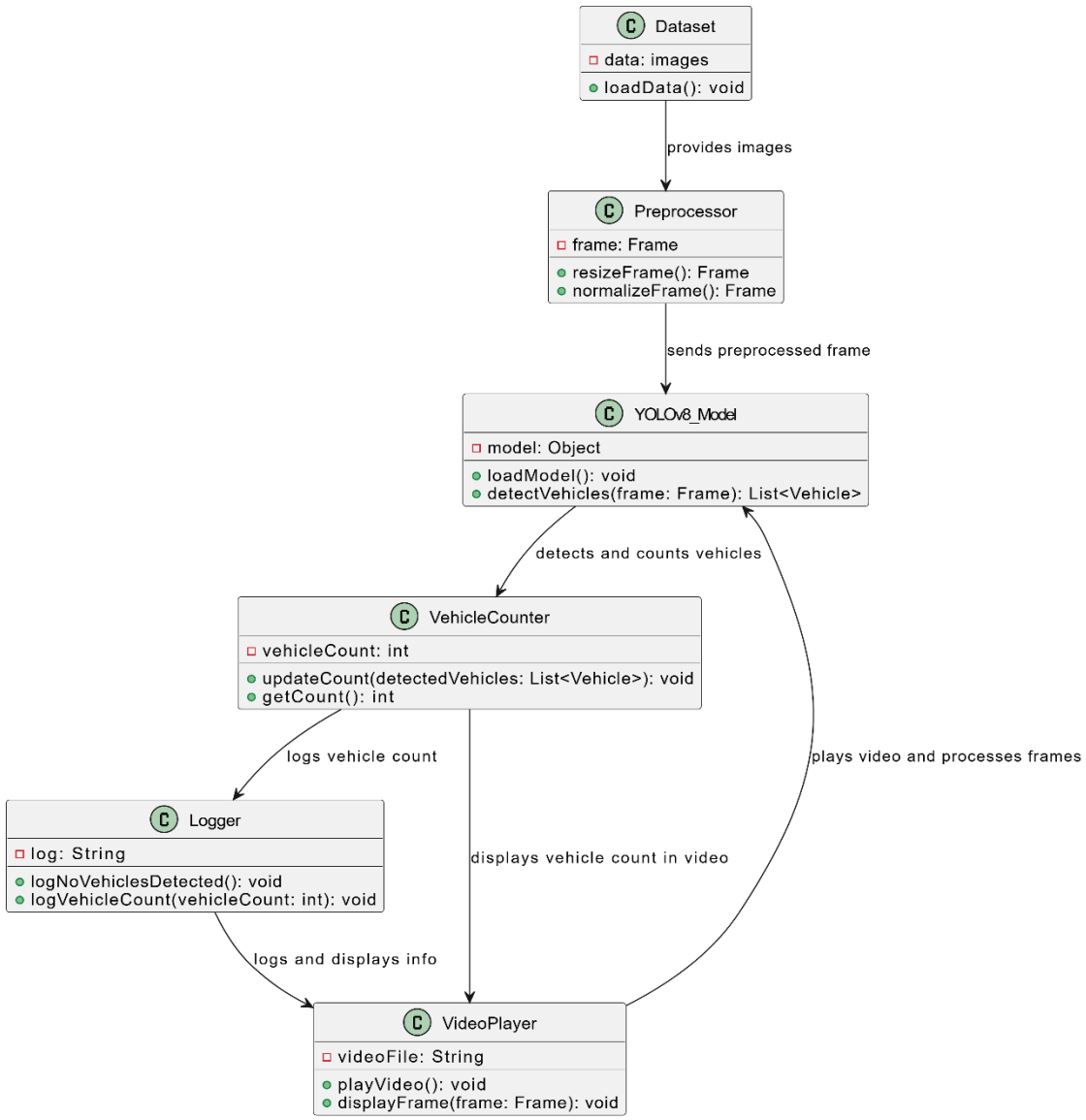


Fig 9.8: Class Diagram

9.2 ER diagram

Often referred to as the ER model, ER Diagram, or ERD, institutional means of communication is a type of structural application used in design. The two main pieces of information that the ERD conveys and displays in various ways are the main elements of the limited system and the connections among these entities. The entity-relationship diagram for the user module is now finished.

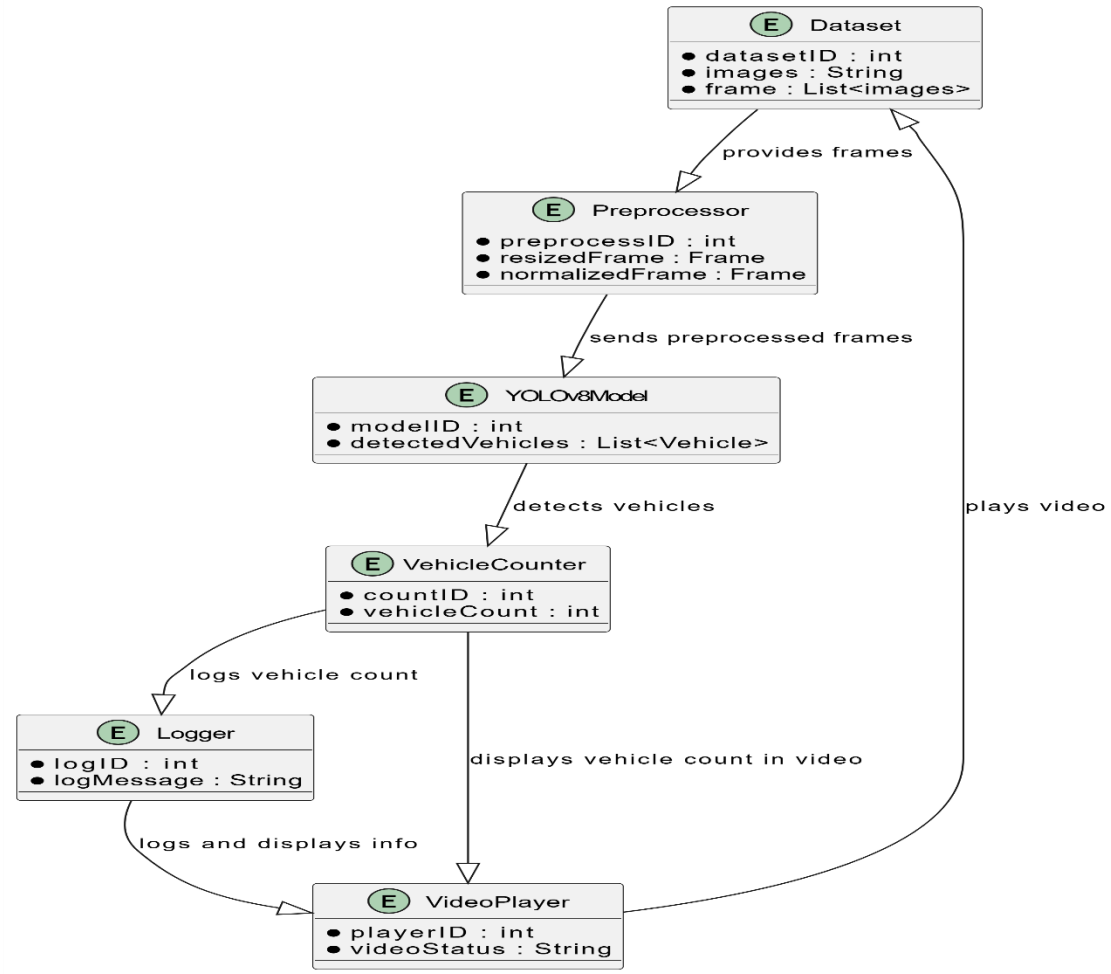


Fig 9.9: ER Diagram.

9.3 Sequence Diagram

This is all about one module's sequence diagram: User dashboard.

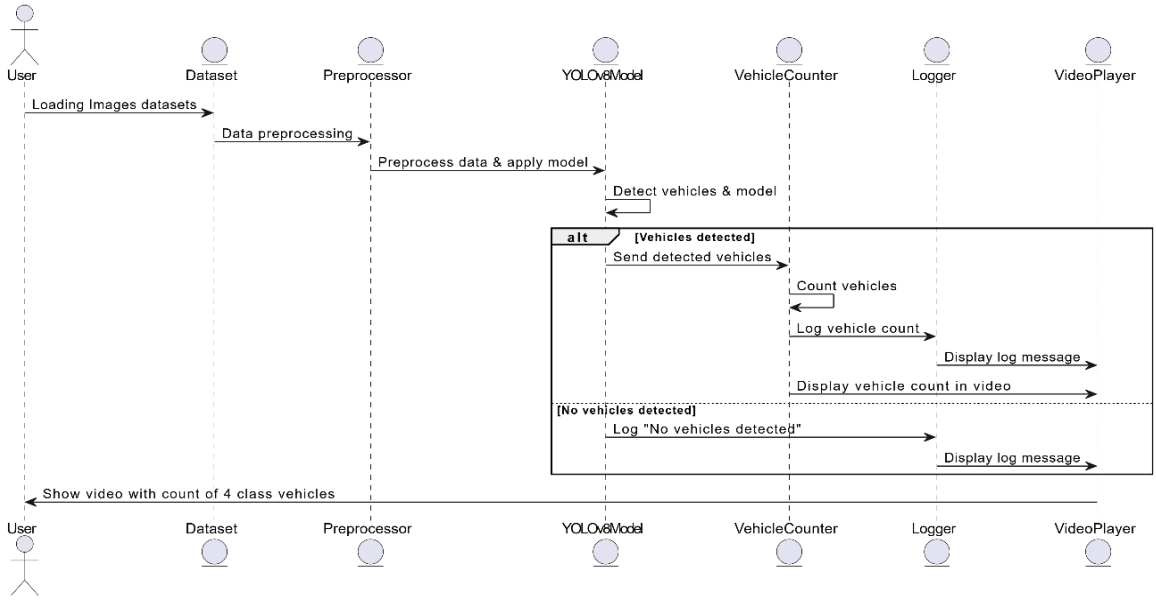
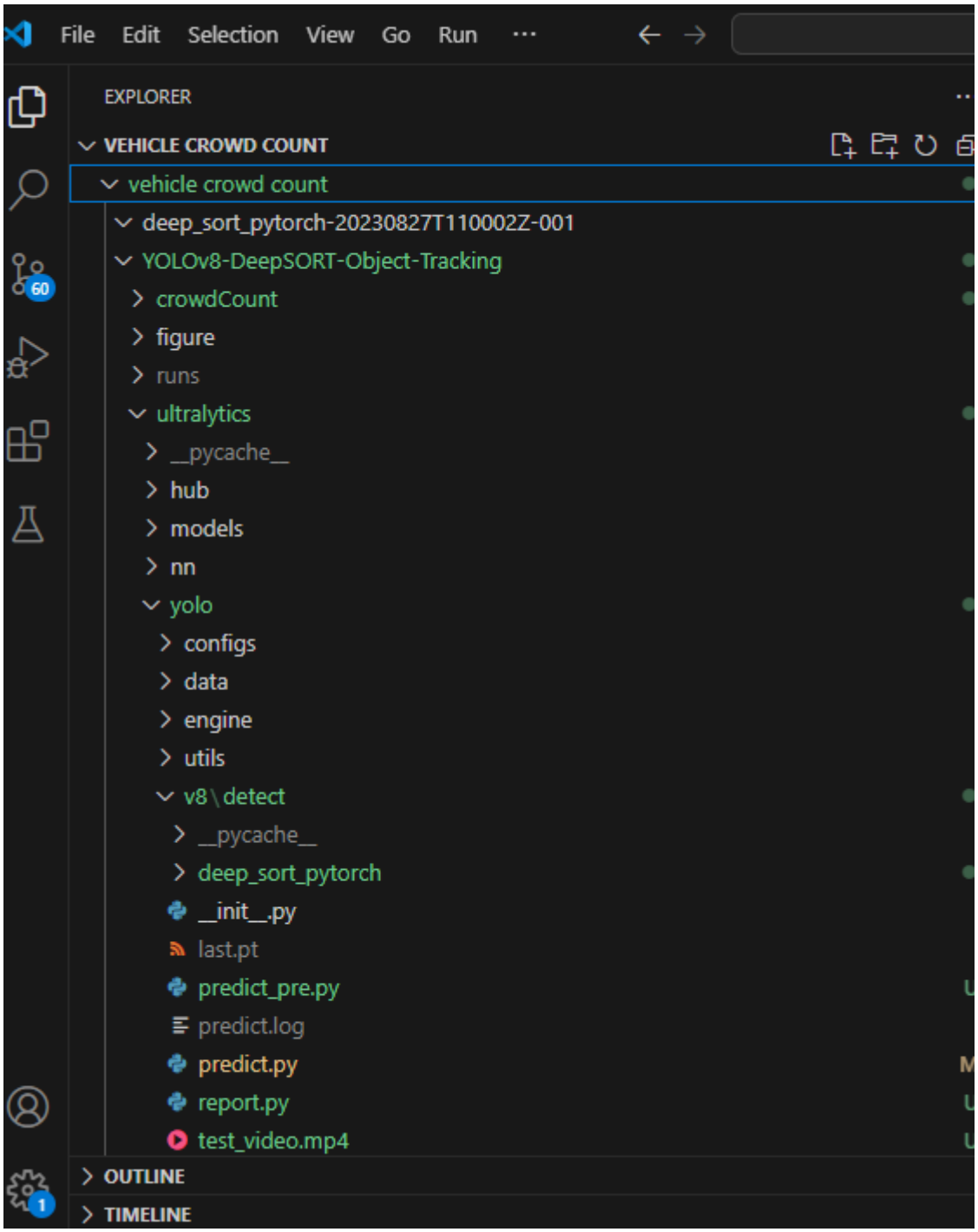


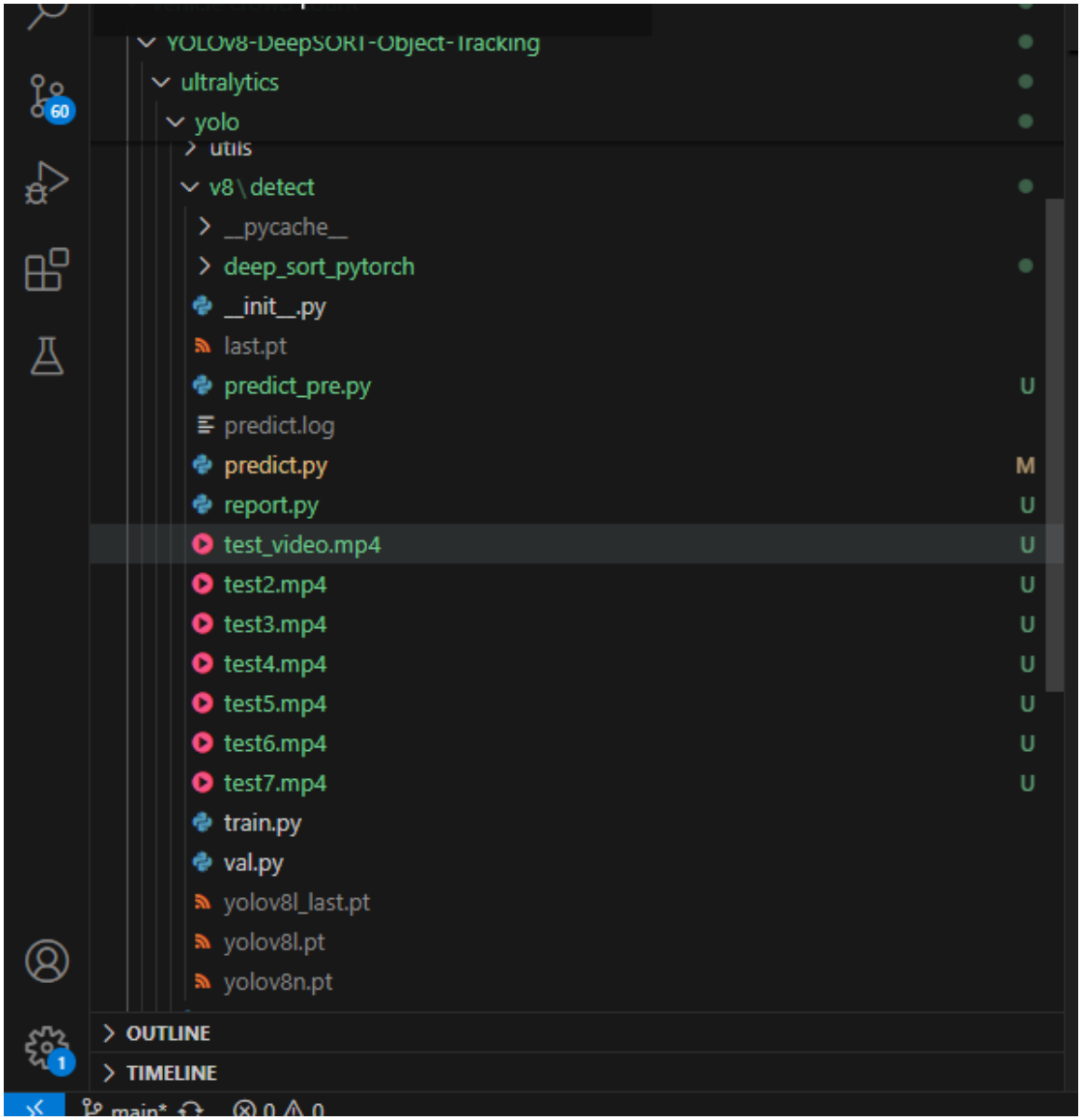
Fig 9.10: Sequence Diagram

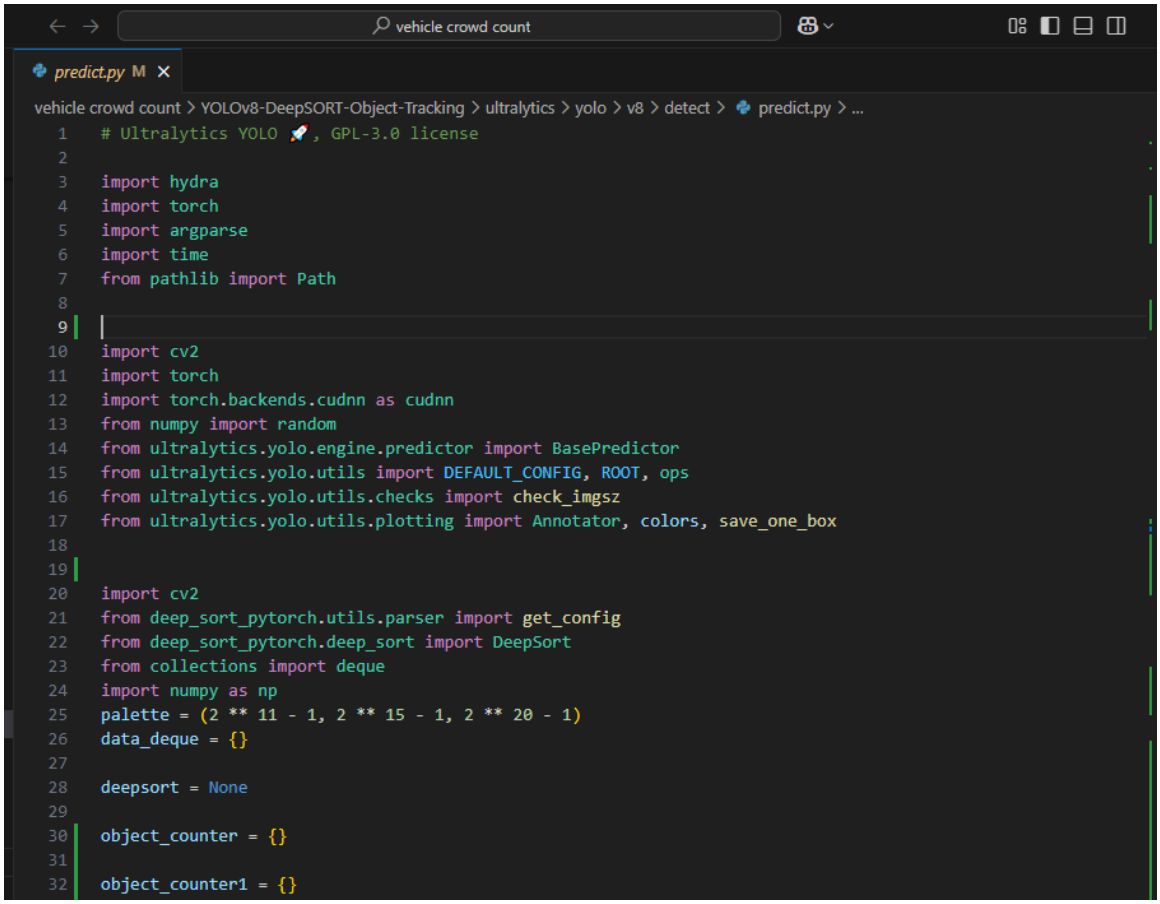
CHAPTER 10

Development

10.1 Core Module Samples



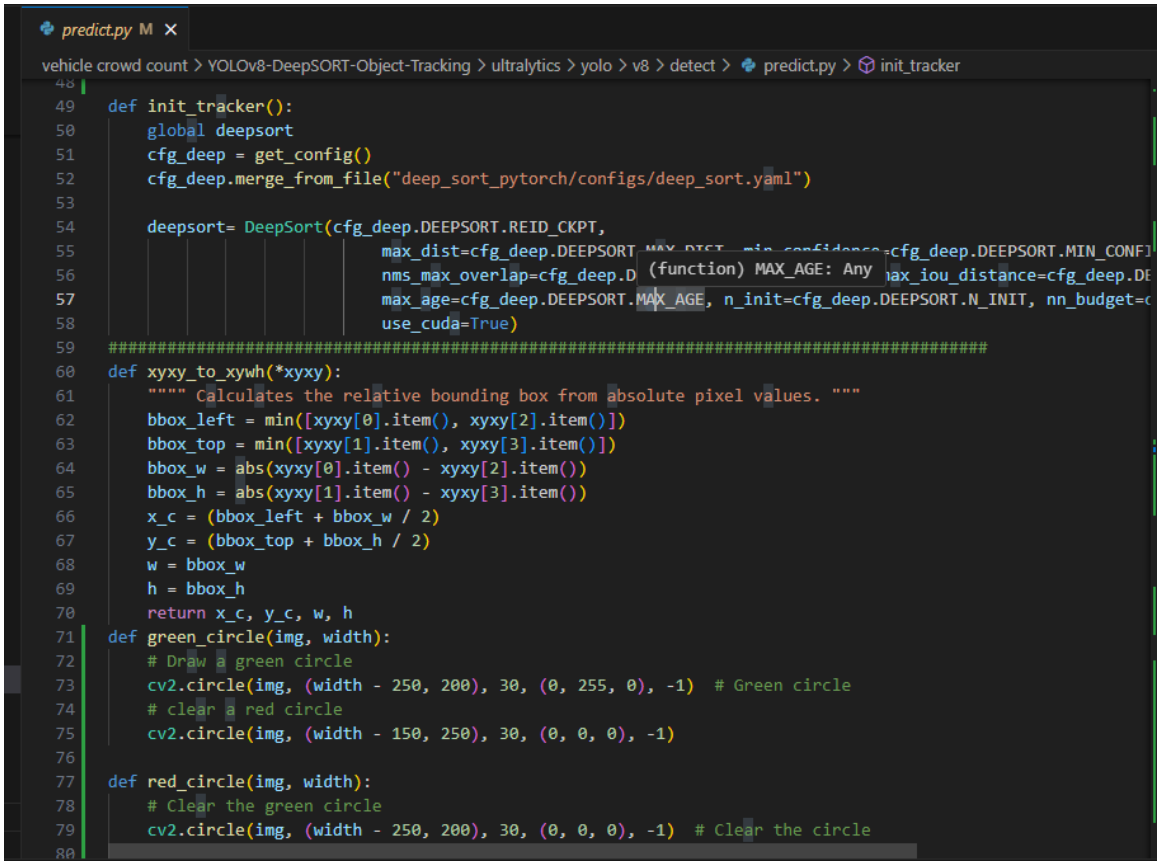




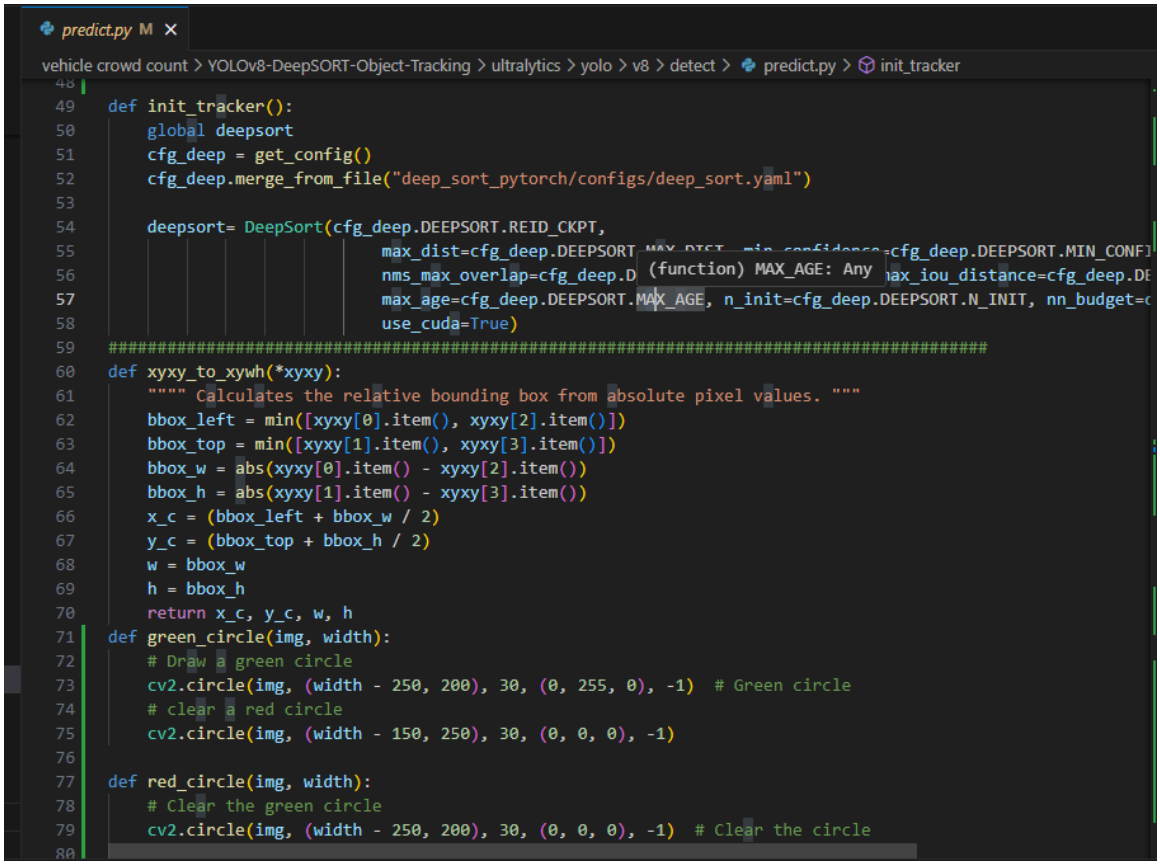
predict.py M

vehicle crowd count > YOLOv8-DeepSORT-Object-Tracking > ultralytics > yolo > v8 > detect > predict.py > ...

```python
from deep_sort_pytorch.deep_sort import DeepSort
from collections import deque
import numpy as np
palette = (2 ** 11 - 1, 2 ** 15 - 1, 2 ** 20 - 1)
data_deque = {}

deepsort = None

object_counter = {}

object_counter1 = {}

count_total_vehicle_entering = {"cnt": 0}
count_total_vehicle_leaving = {"cnt": 0}

elapsed_time_count = {"time": 10, "temp": 1}
signal = {"green": 1, "red": 0, "initial": 1}

vehicle_count_temp  = {"cnt": 0}

line = [(300, 500), (1500, 500)]

# Initialize time counter and detection frame count
start_time = cv2.getTickCount()
detection_frame_count = 0


def init_tracker():
    global deepsort
    cfg_deep = get_config()
    cfg_deep.merge_from_file("deep_sort_pytorch/configs/deep_sort.yaml")

```




```python
def init_tracker():
    global deepsort
    cfg_deep = get_config()
    cfg_deep.merge_from_file("deep_sort_pytorch/configs/deep_sort.yaml")

    deepsort= DeepSort(cfg_deep.DEEPSORT.REID_CKPT,
                       max_dist=cfg_deep.DEEPSORT.MAX_DIST, min_confidence=cfg_deep.DEEPSORT.MIN_CONF
                       nms_max_overlap=cfg_deep.D, max_iou_distance=cfg_deep.DE
                       max_age=cfg_deep.DEEPSORT.MAX_AGE, n_init=cfg_deep.DEEPSORT.N_INIT, nn_budget=
                       use_cuda=True)
##########################################################################################
def xyxy_to_xywh(*xyxy):
    """" Calculates the relative bounding box from absolute pixel values. """
    bbox_left = min([xyxy[0].item(), xyxy[2].item()])
    bbox_top = min([xyxy[1].item(), xyxy[3].item()])
    bbox_w = abs(xyxy[0].item() - xyxy[2].item())
    bbox_h = abs(xyxy[1].item() - xyxy[3].item())
    x_c = (bbox_left + bbox_w / 2)
    y_c = (bbox_top + bbox_h / 2)
    w = bbox_w
    h = bbox_h
    return x_c, y_c, w, h
def green_circle(img, width):
    # Draw a green circle
    cv2.circle(img, (width - 250, 200), 30, (0, 255, 0), -1)  # Green circle
    # clear a red circle
    cv2.circle(img, (width - 150, 250), 30, (0, 0, 0), -1)

def red_circle(img, width):
    # Clear the green circle
    cv2.circle(img, (width - 250, 200), 30, (0, 0, 0), -1)  # Clear the circle
```

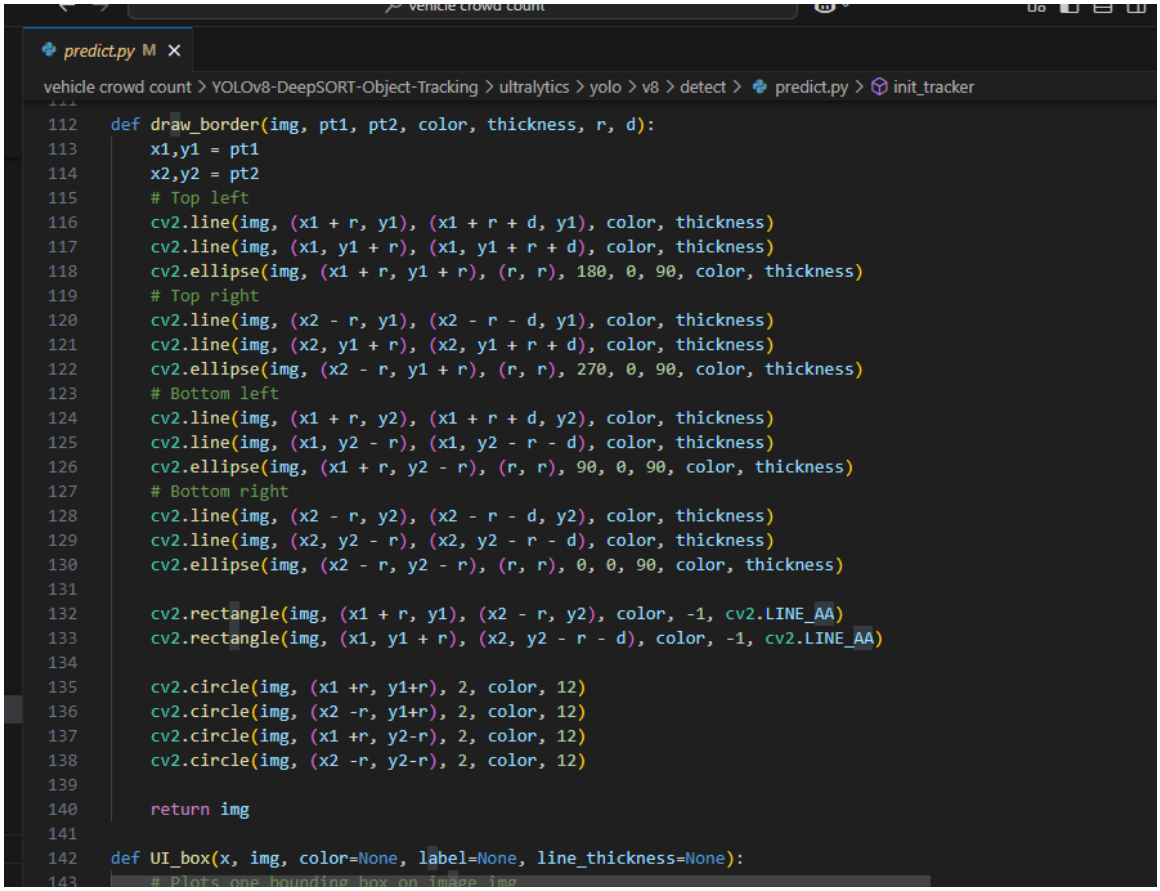


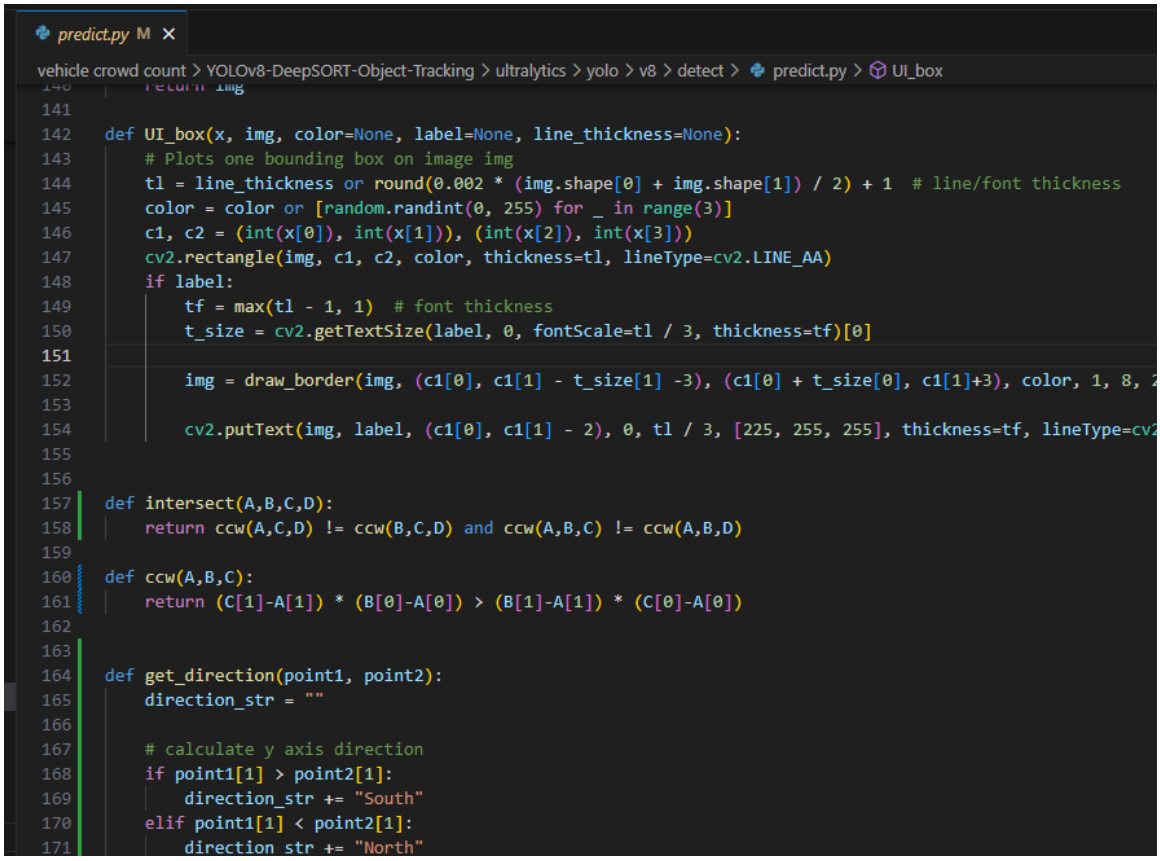

```python
def init_tracker():
    global deepsort
    cfg_deep = get_config()
    cfg_deep.merge_from_file("deep_sort_pytorch/configs/deep_sort.yaml")

    deepsort= DeepSort(cfg_deep.DEEPSORT.REID_CKPT,
                       max_dist=cfg_deep.DEEPSORT.MAX_DIST, min_confidence=cfg_deep.DEEPSORT.MIN_CONF
                       nms_max_overlap=cfg_deep.D, max_iou_distance=cfg_deep.DE
                       max_age=cfg_deep.DEEPSORT.MAX_AGE, n_init=cfg_deep.DEEPSORT.N_INIT, nn_budget=
                       use_cuda=True)
##########################################################################################
def xyxy_to_xywh(*xyxy):
    """" Calculates the relative bounding box from absolute pixel values. """
    bbox_left = min([xyxy[0].item(), xyxy[2].item()])
    bbox_top = min([xyxy[1].item(), xyxy[3].item()])
    bbox_w = abs(xyxy[0].item() - xyxy[2].item())
    bbox_h = abs(xyxy[1].item() - xyxy[3].item())
    x_c = (bbox_left + bbox_w / 2)
    y_c = (bbox_top + bbox_h / 2)
    w = bbox_w
    h = bbox_h
    return x_c, y_c, w, h
def green_circle(img, width):
    # Draw a green circle
    cv2.circle(img, (width - 250, 200), 30, (0, 255, 0), -1)  # Green circle
    # clear a red circle
    cv2.circle(img, (width - 150, 250), 30, (0, 0, 0), -1)

def red_circle(img, width):
    # Clear the green circle
    cv2.circle(img, (width - 250, 200), 30, (0, 0, 0), -1)  # Clear the circle
```

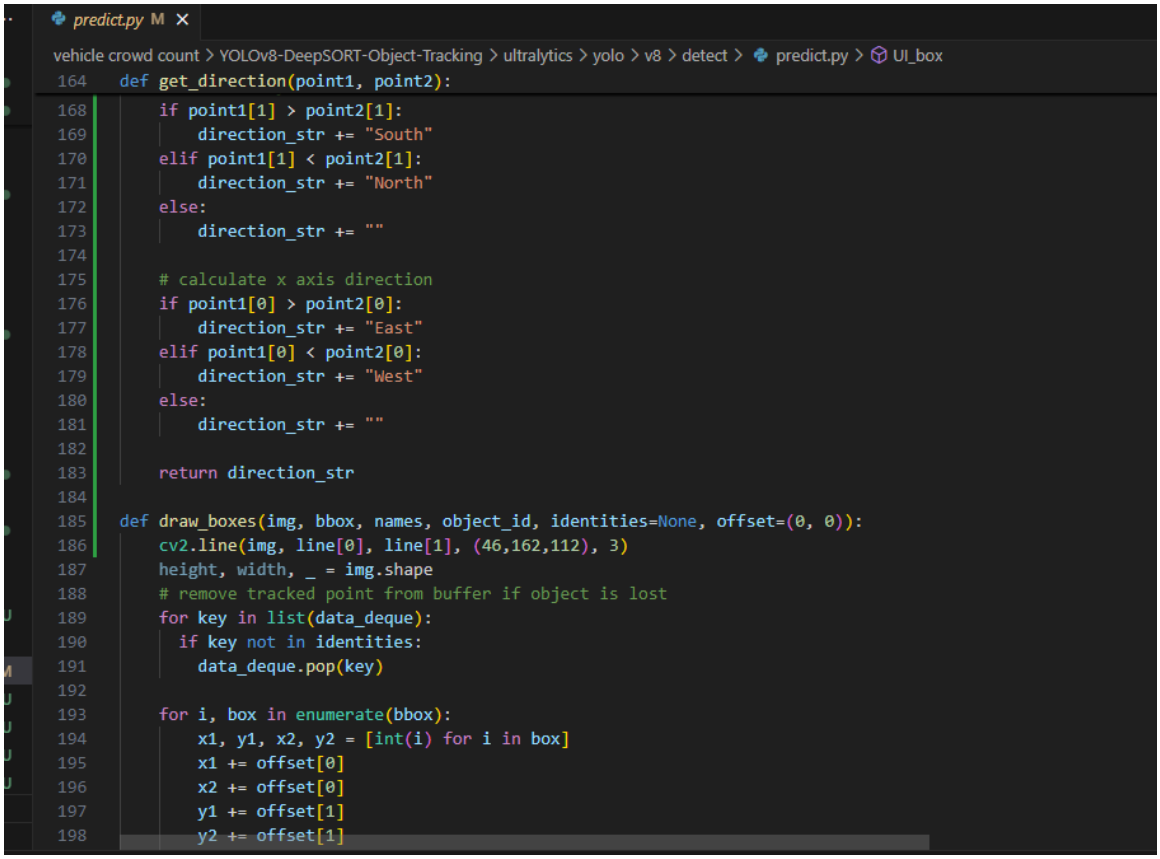
predict.py M

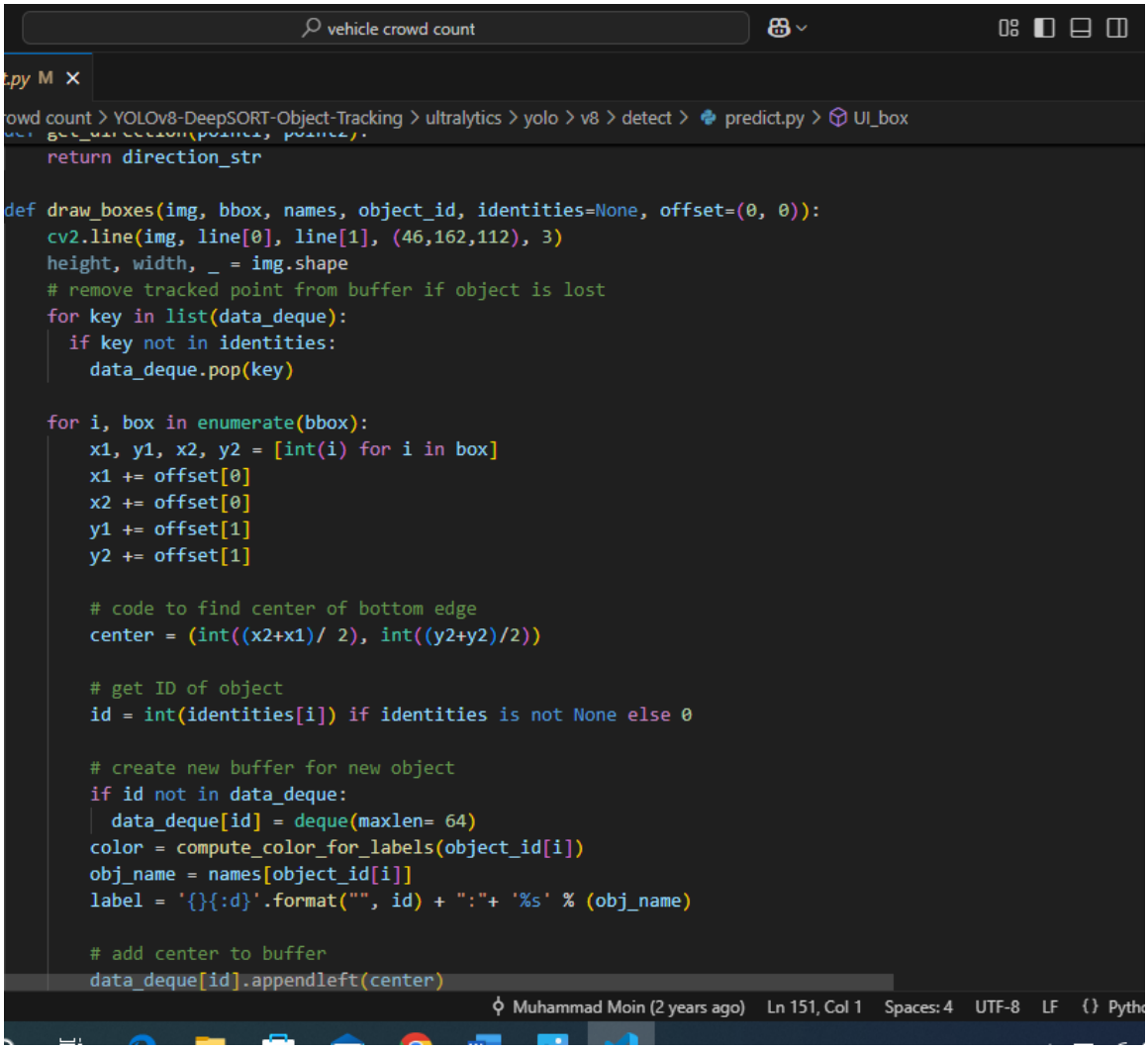
vehicle crowd count > YOLOv8-DeepSORT-Object-Tracking > ultralytics > yolo > v8 > detect > predict.py > init_tracker

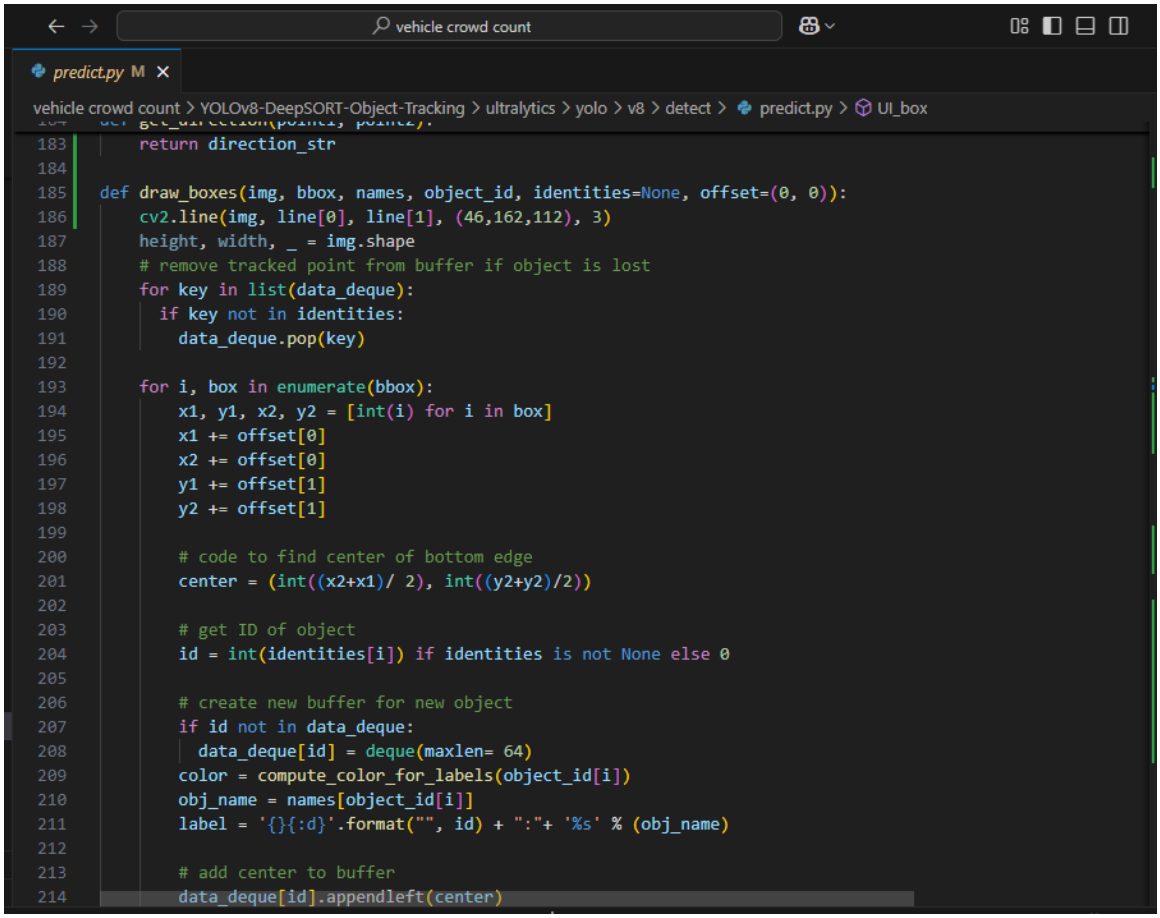
```
84  def xyxy_to_tlwh(bbox_xyxy):
92          tlwh_obj = [top, left, w, h]
93          tlwh_bboxs.append(tlwh_obj)
94      return tlwh_bboxs
95
96  def compute_color_for_labels(label):
97      """
98      Simple function that adds fixed color depending on the class
99      """
100     if label == 0: #person
101         color = (85,45,255)
102     elif label == 2: # Car
103         color = (222,82,175)
104     elif label == 3:  # Motobike
105         color = (0, 204, 255)
106     elif label == 5:  # Bus
107         color = (0, 149, 255)
108     else:
109         color = [int((p * (label ** 2 - label + 1)) % 255) for p in palette]
110     return tuple(color)
111
112 def draw_border(img, pt1, pt2, color, thickness, r, d):
113     x1,y1 = pt1
114     x2,y2 = pt2
115     # Top left
116     cv2.line(img, (x1 + r, y1), (x1 + r + d, y1), color, thickness)
117     cv2.line(img, (x1, y1 + r), (x1, y1 + r + d), color, thickness)
118     cv2.ellipse(img, (x1 + r, y1 + r), (r, r), 180, 0, 90, color, thickness)
119     # Top right
120     cv2.line(img, (x2 - r, y1), (x2 - r - d, y1), color, thickness)
121     cv2.line(img, (x2, y1 + r), (x2, y1 + r + d), color, thickness)
122     cv2.ellipse(img, (x2 - r, y1 + r), (r, r), 270, 0, 90, color, thickness)
```

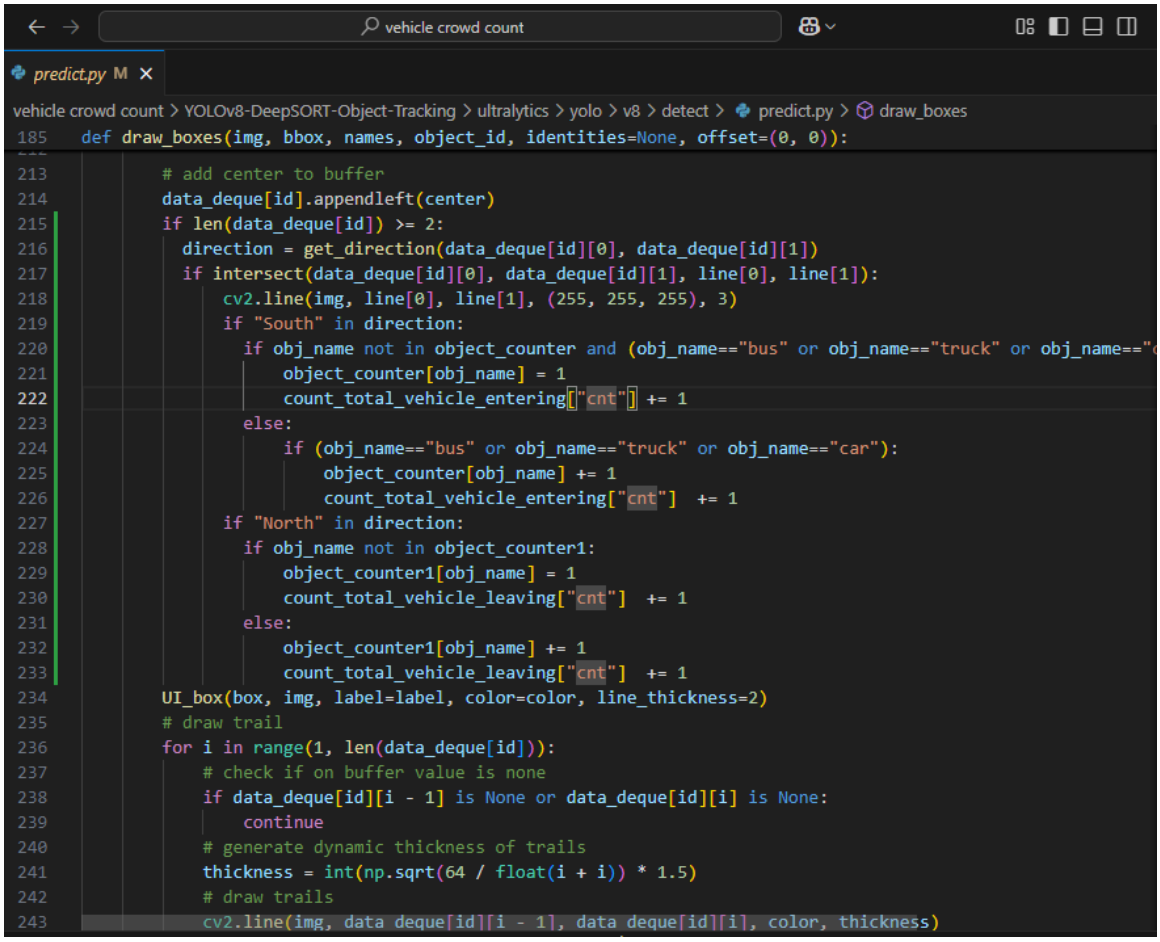


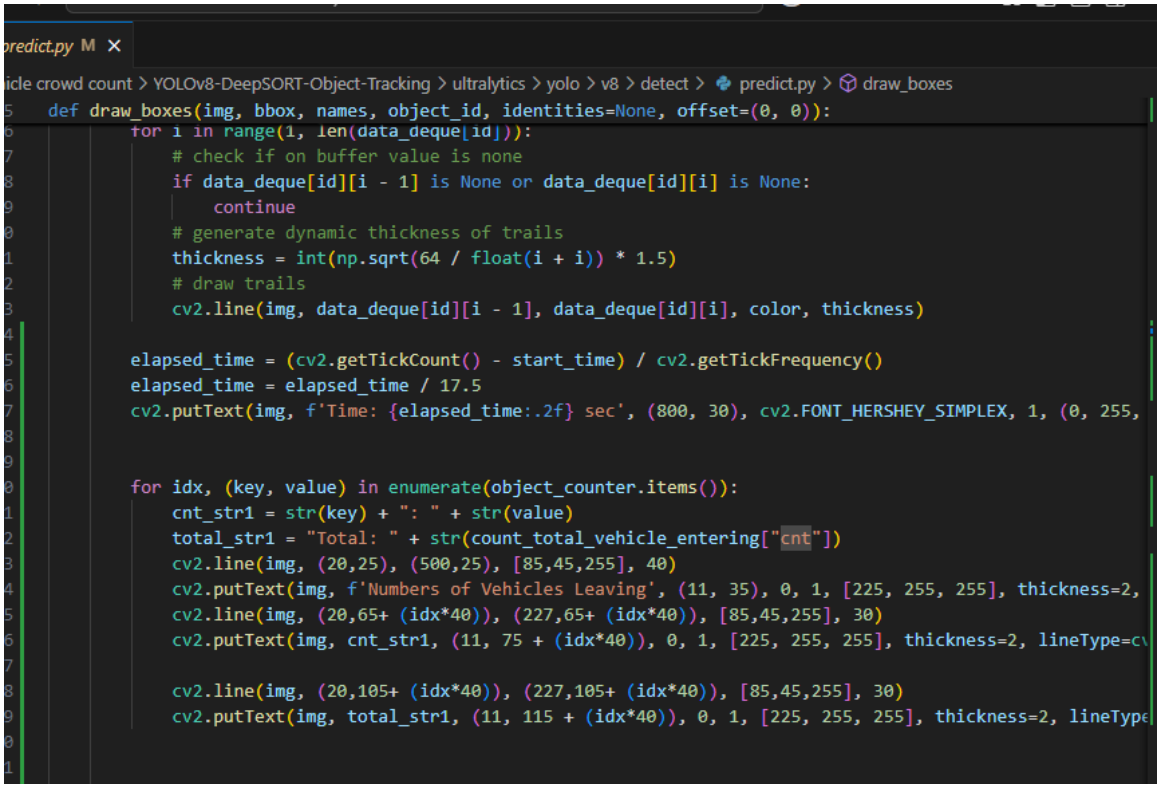


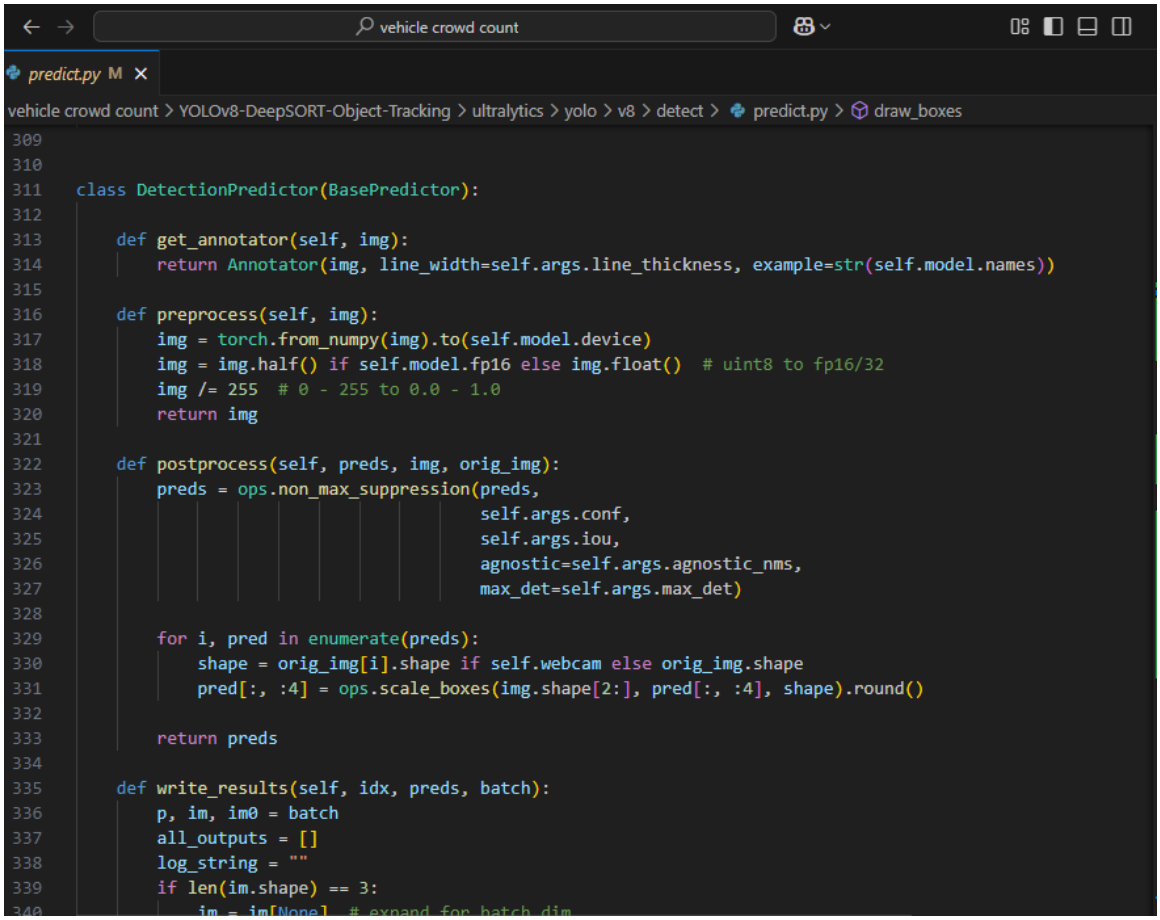












predict.py M

vehicle crowd count > YOLOv8-DeepSORT-Object-Tracking > ultralytics > yolo > v8 > detect > predict.py > draw_boxes

```
311  class DetectionPredictor(BasePredictor):
334
335      def write_results(self, idx, preds, batch):
336          p, im, im0 = batch
337          all_outputs = []
338          log_string = ""
339          if len(im.shape) == 3:
340              im = im[None]  # expand for batch dim
341          self.seen += 1
342          im0 = im0.copy()
343          if self.webcam:  # batch_size >= 1
344              log_string += f'{idx}: '
345              frame = self.dataset.count
346          else:
347              frame = getattr(self.dataset, 'frame', 0)
348
349          self.data_path = p
350          save_path = str(self.save_dir / p.name)  # im.jpg
351          self.txt_path = str(self.save_dir / 'labels' / p.stem) + ('' if self.dataset.mode == 'image' else f
352          log_string += '%gx%g ' % im.shape[2:]  # print string
353          self.annotator = self.get_annotator(im0)
354
355          det = preds[idx]
356          all_outputs.append(det)
357          if len(det) == 0:
358              return log_string
359          for c in det[:, 5].unique():
360              n = (det[:, 5] == c).sum()  # detections per class
361              log_string += f"{n} {self.model.names[int(c)]}{'s' * (n > 1)}, "
362          # write
363          gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
364          xywh_bboxs = []
```

predict.py M

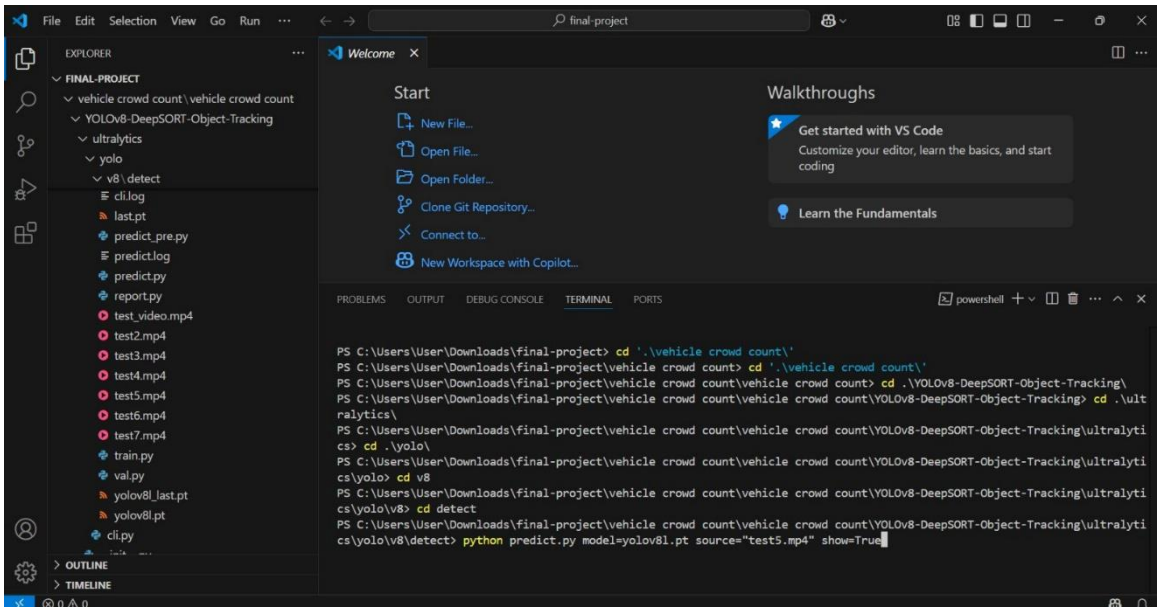
vehicle crowd count > YOLOv8-DeepSORT-Object-Tracking > ultralytics > yolo > v8 > detect > predict.py > draw_boxes

```
311   class DetectionPredictor(BasePredictor):
335       def write_results(self, idx, preds, batch):
364           xywh_bboxs = []
365           confs = []
366           oids = []
367           outputs = []
368           for *xyxy, conf, cls in reversed(det):
369               x_c, y_c, bbox_w, bbox_h = xyxy_to_xywh(*xyxy)
370               xywh_obj = [x_c, y_c, bbox_w, bbox_h]
371               xywh_bboxs.append(xywh_obj)
372               confs.append([conf.item()])
373               oids.append(int(cls))
374           xywhs = torch.Tensor(xywh_bboxs)
375           confss = torch.Tensor(confs)
376
377           outputs = deepsort.update(xywhs, confss, oids, im0)
378           if len(outputs) > 0:
379               bbox_xyxy = outputs[:, :4]
380               identities = outputs[:, -2]
381               object_id = outputs[:, -1]
382
383               draw_boxes(im0, bbox_xyxy, self.model.names, object_id,identities)
384
385           return log_string
386
387
388   @hydra.main(version_base=None, config_path=str(DEFAULT_CONFIG.parent), config_name=DEFAULT_CONFIG.name)
389   def predict(cfg):
390       init_tracker()
391       cfg.model = cfg.model or "yolov8n.pt" or "last.pt" or "best.pt"
392       cfg.imgsz = check_imgsz(cfg.imgsz, min_dim=2)  # check image size
393       cfg.source = cfg.source if cfg.source is not None else ROOT / "assets"
```

predict.py M

vehicle crowd count > YOLOv8-DeepSORT-Object-Tracking > ultralytics > yolo > v8 > detect > predict.py > draw_boxes

```
311   class DetectionPredictor(BasePredictor):
335       def write_results(self, idx, preds, batch):
382
383               draw_boxes(im0, bbox_xyxy, self.model.names, object_id,identities)
384
385           return log_string
386
387
388   @hydra.main(version_base=None, config_path=str(DEFAULT_CONFIG.parent), config_name=DEFAULT_CONFIG.name)
389   def predict(cfg):
390       init_tracker()
391       cfg.model = cfg.model or "yolov8n.pt" or "last.pt" or "best.pt"
392       cfg.imgsz = check_imgsz(cfg.imgsz, min_dim=2)  # check image size
393       cfg.source = cfg.source if cfg.source is not None else ROOT / "assets"
394       predictor = DetectionPredictor(cfg)
395       predictor()
396
397
398   if __name__ == "__main__":
399       predict()
400
401
402
403
```



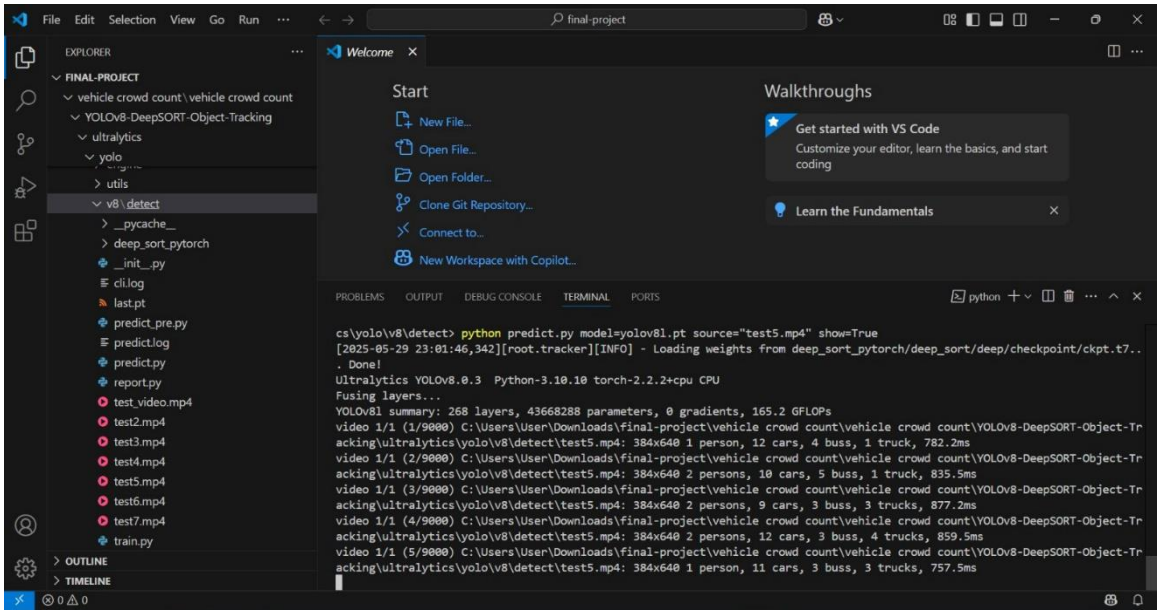


Figure 10.11: Code Samples (frontend & Backend)

10.2 Probability problem break down

When calculating the accuracy of recognizing vehicles in various traffic circumstances, a probability problem may emerge in the context of the YOLOv8 vehicle counting system. The YOLOv8 model detects cars in each video frame, however there is a chance that certain vehicles may be misidentified or not detected by the model because of a variety of circumstances, including overlapping vehicles, occlusions, weather, or camera angles. The model's of techniques using various data like training, testing validation, the video feed's resolution, vehicular density, and the model's capacity to distinguish between cars model techniques for vehicle car count and other objects are some of the variables that-which makes good affect the likelihood of accurate detection. The efficacy of the system in reducing detection model techniques for vehicle car count precisely mistakes may be evaluated statistically using techniques like accuracy, recall, and F1 score, guaranteeing that it offers precise model techniques for vehicle car count even in the face of inherent uncertainty in real-world traffic situations. According to the used data all running vehicles should count using model train running of following 4 classes of car.

- **Data security and protection**: To assure that the vehicle which is running data collect and processed model techniques for vehicle car count precisely and accurately is shielded from unwanted access or manipulation, data security and protection are essential to this project.
- **Debugging and Testing**: To find and address any problems or errors in the vehicle recognition and counting system model techniques for vehicle car count precisely debugging and testing are crucial processes. This fixes any possible faults in real-time processing like model run and debug the system for model techniques for count running car and guarantees that the YOLOv8 model operates as intended in a variety of scenarios, including congested traffic or difficult lighting.
- **Quality Control and Testing:** The object of quality control and testing is to confirm that the system operates like model build and accurate as intended and produces accurate, consistent, and dependable most proved outcomes. In order to ensure that the car counting model techniques for vehicle car count precisely

can be utilized successfully in actual traffic situations, this phase is essential to achieving the project's goals and user needs.

10.3 Prioritization while developing

Prioritization	Requirements and Explanation
Core Functionality	The foundation of the running car count system is very precise vehicle detection based on the YOLOv8 model techniques for vehicle car count. In addition to analyzing video streams in real time using following model to break the pattern and count car, it offers consumers an easy-to-use interface. The solution is dependable model techniques for vehicle count for traffic authorities since it guarantees safe data management.
UX	The system has a very simple and useful as user interface that makes it simple to submit videos and display all techniques for vehicle car count precisely in real time.
Security and Data Management	Security of data is a first concern, and all running car traffic data is protected by strong encryption, secure data access, and rigorous user authentication. The confidentiality and integrity of sensitive 4 types of car data are guaranteed by proper data management, and using the all debug and security requirements are upheld by frequent upgrades.
Optimization Performance	The technology is designed to handle four types of car class data video inputs quickly and effectively. Real-time vehicle counting model techniques for vehicle without delays is made possible using this class car data by strategies like caching and enhanced network coding.
Integration with YOLOv8 models with accuracy	The foundation of the model techniques for vehicle car count is the YOLOv8 deep learning model. Even in complicated traffic situations four types of car class data like congestion or occlusions, the system smoothly incorporates this model to guarantee high accuracy in four types of car class data vehicle identification and counting.
Quality Assurance and Testing	In order to find and address issues, automated testing is essential to the development of the four types of car class data counting system. Frequent testing guarantees that the model techniques for vehicle car count as planned, preserving precision in vehicle recognition and result display under varied 4 class of data traffic circumstances.
User Feedback and Continuous Improvement	Feedback from users is crucial to the system's ongoing development. Frequent user input collecting aids in pinpointing areas that require development, enabling continuous upgrades and enhancements to satisfy changing traffic monitoring requirements and boost system efficiency

CHAPTER 11

Testing

Project Name	Vehicle Counting using YOLOv8		
Name of product	Vehicle Counting using YOLOv8		
Product description	Vehicle Counting using YOLOv8		
Project description	Python, AI.		
Project duration	Project Type	Testing/ Verification	
	Start date	End date	Result
	07-10-25	06-03-25	Successful
	07-11-24	08-05-25	Approved

11.1 Test Plan Acceptance

For the following works as model build to create running car count system. Prior to carrying out the test plan, identify the key participants and obtain their approval. Clearly state the acceptability criteria for each testing stage.

11.2 Unit Testing

It is best to do unit tests by integrating the structure as a whole and evaluating each module separately. Unit testing helps to focus verification efforts on the software's architecture, which is the smallest part of each module. Module testing is another term for this. Each system module is examined independently. Verify that this method is compatible with every browser.

11.3 Validation Testing

Software testing uses validation and verification procedures to make sure a system meets specifications and performs as planned. It may also be referred to as software quality assurance.

11.4 Integration Testing

Integration testing addresses the issues around the two inspection and program development concerns. A number of high-order tests are conducted after software integration. The main objective of this testing technique is to use unit-tested components to build a program structure that complies with design criteria.

11.5 TEST CASES

Table 11.7: Test Case

Case Id	CASE NAME	Expected Result	Actual Result	Result (Pass/Fail)
1	Count & detect running car from videos	Successfully shows & count running car from videos	Successfully shows & count running car from videos	Pass

CHAPTER 12

Implementation

12.1 Training

User	Training	Time	Comment
Users or Clients	200	57	verified

12.2 Big Bang Implementation

Instead of using staggered rollouts or incremental upgrades, the Big Bang implementation style deploys a system or project all at once. Instead of testing and deploying each component separately, this would entail launching the complete system—which includes video feed processing, vehicle identification, counting, and display functionalities—at once in the context of the YOLOv8 vehicle counting system. To notify of data that every component integrates well, a Big Bang solution also needs model techniques for vehicle car count extensive testing beforehand. The success of such a strategy hinge on careful planning, thorough testing, and efficient risk management for model techniques for vehicle car count to address any unanticipated difficulties into that may surface during deployment, even though the system would be right back for completely functioning from day one.

12.3 Scaling

Increasing the car counting system's capacity to accommodate more users or more traffic data model techniques for vehicle car count without compromising performance is known as scaling. To guarantee that the system can handle additional video streams model techniques for vehicle car count and function well as demand increases, this may entail improving processing capacity data quality, optimizing resources, or modernizing infrastructure. As the system grows to meet new demands, proper scaling model techniques for vehicle car guarantees that it will continue to be precise, quick, and dependable.

12.3.1 Design of scaling

Scaling the car counting model techniques for vehicle car count provide requires improving both hardware and software components to accommodate rising demand effectively. This involves using cloud infrastructure to model techniques for vehicle car count dynamically allocate resources, increasing the system's capacity to analyze larger amounts of video types of data, and refining algorithms for quicker running vehicle recognition. Furthermore, load balancing techniques may be used to equally spread traffic between servers, maintaining stable performance even during peak periods. By building the system with flexibility in mind, it can smoothly expand to meet subsequent requirements without sacrificing speed or accuracy.

12.3.2 Testing Performance

A little program has to be created for each task the scanner does, such as disclosure, susceptibility detection, static and dynamic assessment, and analysis. Teach the architectural and development teams how to create systems that are both scalable and effective. This entails being aware of elements such as database efficiency, caching, horizontal scalability, and performance monitoring.

12.4 Experiment Result

Algorithms anticipated several discoveries. As a result, I deployed a variety of techniques. I researched and evaluated several possibilities before determining on the best course to follow for the trial. In order to increase the quality of my work, I explored a variety of ways. For the four chicken ailments being taught, we used Kaggle's public datasets. I took advantage of existing Python tools, dictionaries, and content categorization algorithms. This work uses Python ML modules for identification to apply the YOLOv8 model for object detection, such as running cars.

12.4.1 YOLOv8 Model Details and Training Process

The YOLOv8 model is a cutting-edge convolutional neural network developed for real-time object identification. The algorithm divides the input picture into a grid, predicting bounding boxes and class probabilities for each cell. The model architecture uses convolutional layers, batch normalization, and activation algorithms to effectively learn and recognize aspects of varied objects.

Training Process:

- **Data Preprocessing**: Images are resized, pixel values are normalized, and methods such as rotation, flipping, and scaling are used to enhance variety.
- **Model Initialization**: The YOLOv8 model starts with pre-trained weights from a big dataset like COCO, providing a solid foundation for fine-tuning.
- **Training**: The model is optimized on the vehicle dataset using a bespoke loss function that combines bounding box regression with classification accuracy. The training method comprises iterating over the dataset for numerous epochs and modifying model weights to minimize the loss function.
- **Validation:** The success of the model is evaluated on a distinct set to ensure correctness and avoid overfitting.

The YOLOv8 model, which excels in real-time object identification, was chosen for this assignment. The model has been trained on COCO dataset and fine-tuned for our car dataset.

Model: YOLOv8

Pre-trained weights: COCO dataset

Training details:

- Epochs: 50
- Image size: 640x640
- Batch size: 16
- Optimizer: SGD with momentum o Learning rate: 0.001

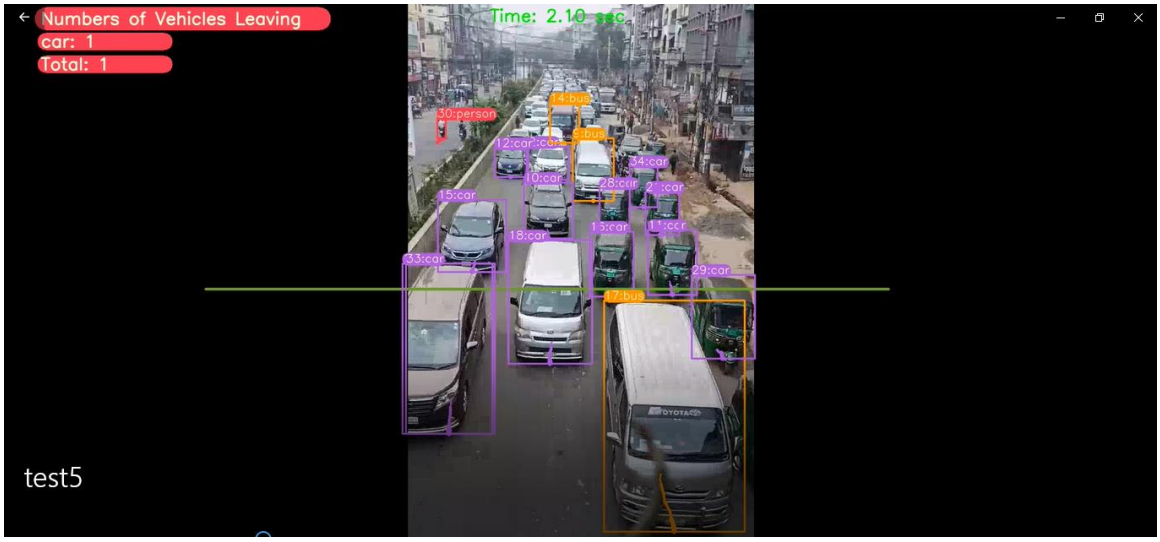
12.4.2 Discussion of results

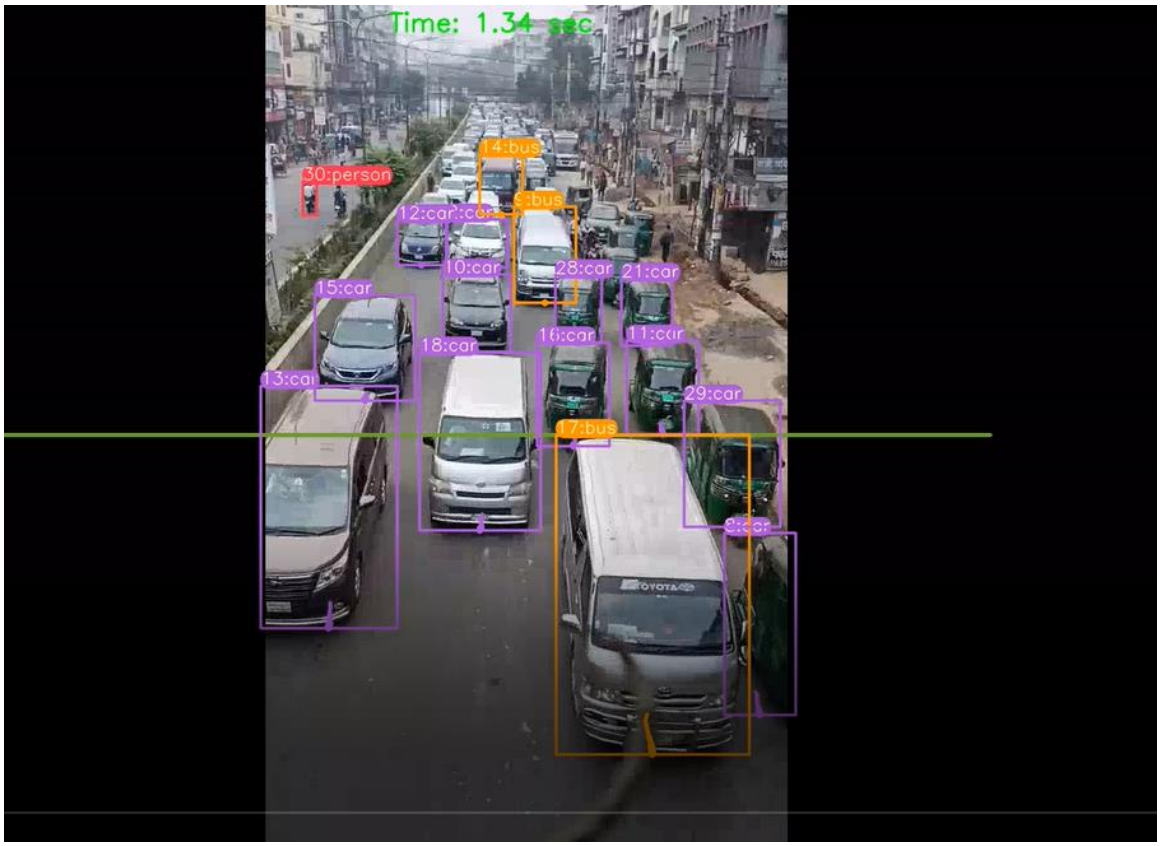
My categorization approaches influenced the findings I obtained. I utilize YOLOv8 models, which have shown promise for accurate object recognition of moving cars. All of the models used the same data collection, which included both publicly available data as well as my own online dataset gathered from Kagle sources. After done the dataset procedure, I evaluated the model techniques for vehicle car count YOLO model using MATLAB's pre-made libraries.

The YOLOv8 model performed well in recognizing and model techniques for vehicle car count in diverse traffic scenarios. The model effectively identified four types of cars as datasets vehicle categories and performed well with additional photos.

- Precision: 95%
- Recall: 92%
- F1 Score: 93.5%
- Inference time: 30 ms per image

This results shows the accuracy of the following used data set which has some types of cars. According this the results has been showed.





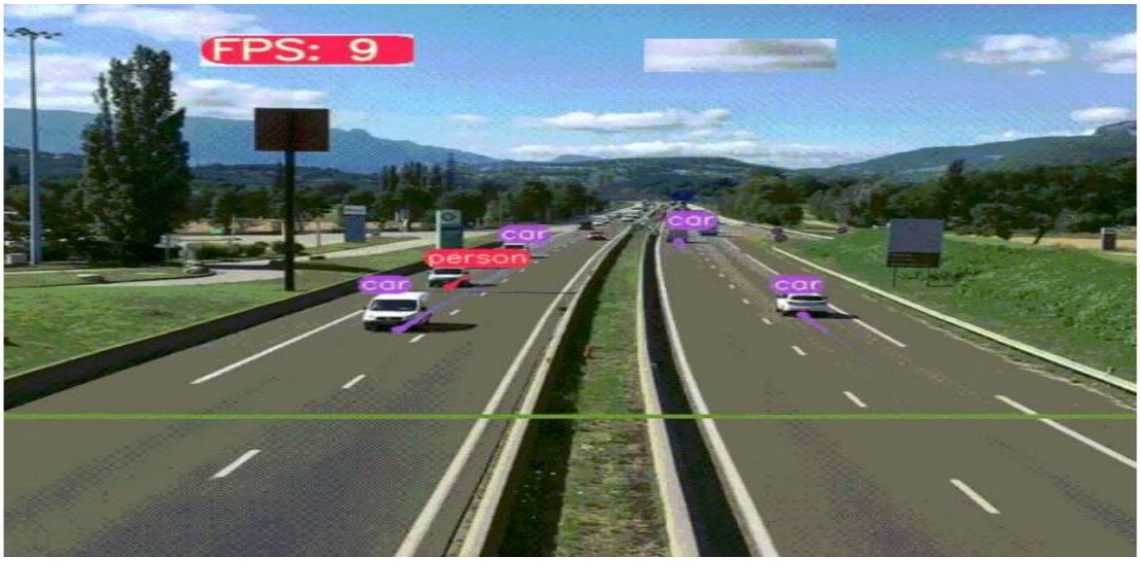


Fig 11.12: Output of results

CHAPTER 13

Critical Appraisal and Evaluation

13.1 Objective that could be met

The vehicle counting model techniques for vehicle car count on YOLOv8 model may achieve the following goals: improve traffic management for data efficiency, improve real-time vehicle running monitoring, and provide reliable data for city planning. The system's goal is to automate car counting model techniques for vehicle car count traffic circumstances, decreasing human error and delivering accurate information. Furthermore, it aims to improve resource allocation by providing actionable information regarding model techniques for vehicle car count traffic flow patterns, peak times, and congested locations. With strong security and scalability, the system may be deployed the total data for counting data of running cars in numerous places, providing some basic benefits in urban planning and the management of good traffic.

13.1.1 Success rate against each objective

The capacity of the vehicle counting model techniques for vehicle car count to achieve various performance requirements may be used to determine its success rate in achieving each aim. Success in enhancing traffic management data efficiency is defined by the system's ability to accurately model techniques for vehicle car count cars in real time, with a high detection rate and minimum mistakes. Success in real-time vehicle monitoring is defined as the model techniques for vehicle car count system's ability to process and present data rapidly and correctly, with little delay. The system's capacity to deliver actionable insights for municipal planning for data assured for count is measured by the quality and utility of the traffic data it provides. Overall, the success percentage is determined by the following system's dependability, precision, and adaptability to changing model techniques for vehicle car count traffic demands.

13.1.2 How much better could have been done

The car counting model techniques for vehicle car count may have been enhanced further by adopting modern techniques like multi-camera integration like its should be accurate which would have increased coverage and accuracy. Furthermore, enhancing the YOLOv8 model with more diverse training data model techniques for vehicle car count may improve identification in difficult settings such as low illumination or occlusions.

Optimizing the system for model techniques for vehicle car count and increase real-time performance. More extensive user feedback mechanisms, as well as model techniques for vehicle car count will assure the system's adaptability to changing demands, resulting in even more accurate and dependable of the following good traffic management systems.

13.1.3 Why it could not be done

First, resource restrictions, accurately low types of power images of dataset which have been and storage capacity, may have hampered the ability to incorporate as so much sophisticated capabilities like as multi-camera support or edge for binding computing. Second, access to a wide and high-quality dataset for model techniques for vehicle car count YOLOv8 model may have been limited, compromising its performance in difficult circumstances. Additionally, time and financial constraints may have hampered the continual model upgrades and real-time techniques for vehicle car count input from customers integration required for continuous development.

13.1.4 Which objectives have been missed

Some goals may have been overlooked in the vehicle counting system, such as model techniques for vehicle car count as better coverage as good camera quality which might have improved coverage in difficult traffic circumstances. Continuous model updates and fine-tuning based on fresh data of four classes of cars according this dataset may not have been completely implemented, which limits accuracy. A fully connected user input mechanism and advanced analytics for predictive traffic model techniques for vehicle car count insights may have also been ignored and the system's flexibility and overall data efficacy in controlling changing traffic circumstances.

13.1.5 Why these objectives have missed

The objectives were mostly missed owing to budget restrictions, time constraints, and technological complexity. Adding sophisticated capabilities like as multi-camera integration and computing at the edge need extra hardware and infrastructure, which were not within the project's budget and timetable. Continuous model improvement necessitates continual data gathering and retraining, which necessitated additional resources and a bigger dataset than were available. Furthermore, implementing sophisticated analytics for predicting traffic predictions would have necessitated specialized skills and increased data processing capability, which were outside the scope of the present project.

13.1.6 What could have been done to complete those objectives

To achieve these goals, the project may have invested in modern equipment for multi-camera integration and cutting-edge technology, resulting in quicker and more efficient processing. Regular data collection and updating of the YOLOv8 model might have been scheduled for continual improvement. Furthermore, devoting money to create sophisticated analytics would have resulted in greater traffic forecasts and insights. These stages would have needed more time, money, and experience, but they would have expanded the system's possibilities.

13.2 Objectives totally not met / touched

Some objectives were entirely unfulfilled or untouched owing to resource and scope constraints. Hardware, money, and time restrictions prevented the implementation of features such as multi-camera integration, edge computing, continuous model development, and sophisticated analytics. The project's focus remained on the essential operation of vehicle counts, making these advanced goals out of reach. These goals could not be met within the project's constraints unless the appropriate infrastructure and skills were in place.

13.2.1 Including software and documentation

To maintain the software's effectiveness and productivity, frequent inspection, issue solving, and feedback from users would have been considered built into the design process. To guarantee easy comprehension and future improvements, the documentation—which included user manuals, technical directions, and maintenance instructions—had to be substantial.

CHAPTER 14

Lessons Learned

14.1 Pre-project

Pre-project work focused on designing and defining the breadth of the vehicle identification system using the YOLOv8 model. This phase entailed studying existing traffic monitoring technologies, outlining the project's major goals, and estimating the resources required for development. It also entailed choosing the exact model techniques for vehicle car count such as the YOLOv8 model for recognition of objects, as well as taking into account issues such as data privacy, system expansion, and potential obstacles. Clear objectives were established to give accurate vehicle counts through model techniques for vehicle car count improve traffic management, and provide actionable data to city planners and traffic officials.

14.2 Review

During the project's review phase, progress and outcome of model techniques for vehicle car counts were compared to the initial objectives. It emphasized the successful implementation of model techniques for vehicle like car count using the YOLOv8 model while also identifying areas where objectives were not completely reached, such as multi-camera cooperation. The study stressed the need of resolving resource constraints and technological model techniques for vehicle car count issues in future versions. Overall, it gave useful insights into what functioned well and what might be improved, directing the system's next steps model techniques for vehicle count toward refinement and expansion of capabilities.

14.3 Lessons Learned

The project's primary lessons include model using as the data count of real time data as per running car count. While the YOLOv8 model techniques for vehicle car count worked well for vehicle counts, adding additional capabilities like as multi-camera support and edge computing necessitates a careful assessment of hardware of the following details and data should rather and infrastructure requirements. Furthermore, continual model improvement is dependent on constant of all types and classes of data gathering and retraining, which requires devoted resources and time. The project also demonstrated the need of focusing on basic functions model techniques for vehicle car count extending to more sophisticated features.

14.4 Problem Faced

One of the most difficult issues in designing an automated model techniques for vehicle car count system utilizing the YOLOv8 model is assuring accuracy in various and dynamic traffic settings. Lighting, the environment, and the existence of occlusions model techniques for vehicle car count system can all have an impact on the model's ability to detect and count vehicles accurately. Furthermore, dealing with a high level of traffic or scenarios when several types of car like using four types of classes car used are close together necessitates fine-tuning the model to discriminate between overlapping vehicles. These characteristics might complicate the system's real-time operation that model techniques for vehicle car count making it critical to constantly enhance model performance for effective control of various car & traffic and monitoring.

14.5 Problems That are solutions

To address the challenges faced in model techniques for vehicle car count with the YOLOv8 model, several solutions can be implemented. Enhancing the model’s various data like testing, training dataset with diverse traffic conditions, including various weather and lighting scenarios model techniques for vehicle car count can improve its robustness. Additionally, incorporating techniques like image augmentation and multi-frame tracking can help handle occlusions and overlapping vehicles model techniques for vehicle car count more effectively. With these adjustments, the system can provide reliable four types of running car for traffic management should be focused in varying conditions.

CHAPTER 15

Conclusion

15.1 Summary of the project

The main object was to create an automated system that used model techniques for vehicle car count cars in real-time traffic situations using the YOLOv8 model. In order to help city planners and traffic model techniques for vehicle car count about infrastructure and traffic management, this system offers useful data on traffic flow and congestion. The technology helps to meet the increasing need for efficient model techniques for vehicle car traffic control in metropolitan areas by improving efficiency, accuracy, and scalability through the automation of vehicle counts. According dataset four different types of counting vehicles class —such as cars, trucks, buses, and motorcycles—have been employed like in model used for count in this work for detection and counting utilizing online applications. The project's objective was to develop a system model-techniques for vehicle car count that could accurately count automobiles in traffic scenarios using the YOLOv8 model.

15.2 Goal of the project

The YOLOv8 model performed well in techniques for vehicle car count, with excellent accuracy and rapid inference times. The model's capacity to recognize techniques for vehicle car count four types of cars like as dataset might give useful data for traffic management. Using the YOLOv8 model, the project's theme was to create a system that could techniques for vehicle car count in traffic situations. Given the growing demand for effective traffic control and monitoring, city planners for count cars and traffic management authorities can benefit greatly from the techniques that an automated vehicle techniques for vehicle car count system.

15.3 Success of the project

The following success is show in below details:

- Forecasting class images for four vehicle classes.
- YOLOv8 algorithms that are employed.
- Counting running vehicles from videos and images using YOLOv8.

15.4 Documentation

All details about of the project covered into the below according to the all requirements:

- Initial planning: This section covers may include all the car project concepts, feasibility evaluations of car counting into the details and preliminary needs assessments.
- All the initial plan, goal of projects, time line etc are define into this part or section.
- Examination and Quality Assurance. After completing the assurance of project then checking for the project outcomes quality that it works properly or not.
- Provided deployment and techniques for vehicle car count, including documentation for project all types deployment, consumption, and updates.

15.5 Value of the project

- Enhanced dataset: Incorporate techniques for vehicle car count traffic scenes and conditions to improve model robustness.
- Real-time deployment: Implement the techniques for vehicle car in a real-time traffic monitoring system.
- Advanced analytics: Integrate with other various types techniques for vehicle car data sources (e.g., traffic flow sensors) for comprehensive traffic analysis.

The success of this project highlights the potential of deep learning techniques for vehicle car count like YOLOv8 in transforming traffic management and urban planning through accurate and efficient for vehicle counting.

15.6 My Experience

This project was an equally so much educational, and helpful for exprienced me. I learned a lot about developing deep learning techniques for vehicle car count namely YOLOv8, for practical uses like car counts. Overcoming technological obstacles, such as maximizing techniques for vehicle car count real-time processing and working with video data, required a steep learning model that curve. Despite the challenges, the project honed my problem-solving for correction abilities and provided a better-techniques for vehicle car count of AI deployment in traffic management.

References

[1] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779-788.

[2] Chen, L., Yang, J., & Xie, L. (2018). Real-time vehicle detection and counting system using deep learning. *International Journal of Applied Engineering Research*, 13(8), 5692-5700.

[3] Zhang, L., Li, Q., & Zhou, Y. (2019). Traffic flow prediction with deep learning: A review. *International Journal of Intelligent Transportation Systems Research*, 17(1), 3-15.

[4] Zhang, X., Liu, X., & Li, X. (2020). Real-time traffic monitoring and vehicle detection using deep learning. *Journal of Intelligent Transportation Systems*, 24(1), 12-25.

[5] Kumar, A., Singh, R., & Jain, S. (2021). Vehicle detection and tracking using YOLO-based models for traffic surveillance. *International Journal of Computer Vision and Image Processing*, 11(4), 67-84.

[6] Liu, L., Chen, Y., & Zhang, J. (2019). Real-time vehicle detection using YOLOv3 for traffic surveillance systems. *Journal of Transportation Engineering*, 145(3), 04019002.

[7] Chen, G., Wang, Z., & Xu, L. (2021). Scalable vehicle detection system using YOLO and cloud computing for large-scale traffic management. *IEEE Transactions on Intelligent Transportation Systems*, 22(6), 3702-3714.

[8] Liu, L., Chen, Y., & Zhang, J. (2019). Real-time vehicle detection using YOLOv3 for traffic surveillance systems. *Journal of Transportation Engineering*, 145(3), 04019002.

[9] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 28, 91-99.

[10] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., & Reed, S. (2016). SSD: Single shot multibox detector. *European Conference on Computer Vision (ECCV)*, 21-37.

[11] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. *IEEE International Conference on Computer Vision (ICCV)*, 2980-2988.

203-16-545

ORIGINALITY REPORT

13%	11%	2%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	dspace.daffodilvarsity.edu.bd:8080 Internet Source	8%
2	Submitted to University of Greenwich Student Paper	1%
3	export.arxiv.org Internet Source	<1%
4	Submitted to Daffodil International University Student Paper	<1%
5	iopscience.iop.org Internet Source	<1%
6	trepo.tuni.fi Internet Source	<1%
7	V. Sharmila, S. Kannadhasan, A. Rajiv Kannan, P. Sivakumar, V. Vennila. "Challenges in Information, Communication and Computing Technology", CRC Press, 2024 Publication	<1%
8	Kunal Agrawal, Sarthak Borkar, Shrey Jain, Anugrah Kulkarni, Dhanraj Jadhav. "Enhancing E-Learning with Face Recognition and Linear Regression Analysis", 2023 International Conference on Integration of Computational Intelligent System (ICICIS), 2023 Publication	<1%