

**ADAPTIVE TRAFFIC CONTROL SYSTEM  
USING NEXT-GEN AI  
A PROJECT REPORT**

*Submitted by*

**ARUNKUMAR J (211420104026)**

**BHARATHWAJ M (211420104039)**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF ENGINEERING**

*in*

**COMPUTER SCIENCE AND ENGINEERING**



**PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**APRIL 2024**

# **PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

## **BONAFIDE CERTIFICATE**

Certified that this project report **ADAPTIVE TRAFFIC CONTROL SYSTEM USING NEXT-GEN AI** is the bonafide work of “Arunkumar J (211420104026) Bharathwaj M (211420104039)” who carried out the project work under my supervision.

**Signature of the HOD with date**

**DR L.JABASHEELA M.E.,Ph.D.,  
Professor and Head,**

Department of Computer Science and  
Engineering,

Panimalar Engineering College,

Chennai - 123

**Signature of the Supervisor with date**

**Dr. N.Pughazendi M.E.,Ph.D.,  
Professor**

Department of Computer Science and  
Engineering,

Panimalar Engineering College,

Chennai - 123

Submitted for the Project Viva – Voce examination held on \_\_\_\_\_

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## **DECLARATION BY THE STUDENT**

We Arunkumar J[211420104026] Bharathwaj M[211420104039] hereby declare that this project report titled “Adaptive Traffic Control System Using Next-Gen AI”, under the guidance of Dr. N.Pughazendi is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

Arunkumar J

Bharathwaj M

## **ACKNOWLEDGEMENT**

Our profound gratitude is directed towards our esteemed Secretary and Correspondent, **Dr. P. CHINNADURAI, M.A., Ph.D.**, for his benevolent words and fervent encouragement. His inspirational support proved instrumental in galvanizing our efforts, ultimately contributing significantly to the successful completion of this project.

We want to express our deep gratitude to our Directors, **Tmt. C. VIJAYARAJESWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D., and Dr. SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.**, for graciously affording us the essential resources and facilities for undertaking of this project.

Our gratitude is also extended to our Principal, **Dr. K. MANI, M.E., Ph.D.**, whose facilitation proved pivotal in the successful completion of this project.

We express my heartfelt thanks to **Dr. L. JABASHEELA, M.E., Ph.D.**, Head of the Department of Computer Science and Engineering, for granting the necessary facilities that contributed to the timely and successful completion of project.

We would like to express our sincere thanks to Project Coordinator and Project Guide **Dr. N PUGHAZENDI ,M.E., PH.D.**, and all the faculty members of the Department of CSE for their unwavering support for the successful completion of the project.

**ARUNKUMAR J**  
**BHARATHWAJ M**

# **PROJECT COMPLETION CERTIFICATE**

## **ABSTRACT**

In today's bustling urban landscapes, traffic congestion presents a persistent and formidable challenge, necessitating innovative solutions to alleviate gridlock effectively. This paper introduces a pioneering approach that capitalizes on the integration of surveillance cameras and cutting-edge Machine Learning (ML) and Deep Learning (DL) techniques for the autonomous monitoring and regulation of traffic signals. By harnessing a vast dataset acquired from surveillance cameras capturing a diverse array of traffic scenarios, the proposed methodology trains a machine learning model to discern and classify vehicles into various categories. This process is augmented by self-learning mechanisms, further refining precision and efficacy. Notably, the utilization of the YOLOv5 model yields promising results, achieving an impressive 88% accuracy in vehicle recognition. Beyond its immediate applications, this research holds considerable promise for future advancements in two critical domains. Firstly, it lays the groundwork for enhancing road design and monitoring capabilities, thereby fostering safer and more efficient transportation infrastructure. Secondly, it offers a potential avenue for curtailing fuel consumption and optimizing vehicle standby times, thus contributing to environmental sustainability and resource efficiency. Through such proactive measures, substantial reductions in traffic congestion are anticipated, paving the way for smoother and more streamlined urban mobility in the years ahead. In the context of burgeoning urbanization and escalating traffic volumes, the imperative for effective traffic management solutions has never been more pronounced. By leveraging the capabilities of surveillance cameras and advanced ML and DL algorithms, this research endeavors to address the multifaceted challenges posed by modern traffic congestion.

**APPENDIX 3**  
**TABLE OF CONTENTS**

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
	<b>ABSTRACT</b>	<b>iii</b>
	<b>LIST OF TABLES</b>	<b>vii</b>
	<b>LIST OF FIGURES</b>	<b>viii</b>
	<b>LIST OF SYMBOLS</b>	<b>ix</b>
<b>1.</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	Problem Definition	1
<b>2.</b>	<b>LITERATURE REVIEW</b>	<b>5</b>
<b>3.</b>	<b>THEORETICAL BACKGROUND</b>	<b>11</b>
3.1	Implementation Environment	11
3.2	System Architecture	16
3.3	Proposed Methodology	20
	3.3.1 Database Design / Data Set Description	20
	3.3.2 Input Design (UI)	22
	3.3.3 Module Design	26

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
<b>4.</b>	<b>SYSTEM IMPLEMENTATION</b>	<b>18</b>
	4.1 Vehicle Detection	18
	4.2 Data preprocessing	19
	4.3 Model Development and Evaluation	21
	4.4 Density Calculation	22
	4.5 Estimating Signal Time	23
<b>5.</b>	<b>RESULTS &amp; DISCUSSION</b>	<b>25</b>
	5.1 Performance Parameters / Testing	25
	5.2 Results & Discussion	26
<b>6.</b>	<b>CONCLUSION AND FUTURE WORK</b>	<b>28</b>
	6.1 Conclusion	28
	6.2 Future Enhancements	28
	<b>REFERENCES</b>	<b>30</b>
	<b>APPENDICES</b>	<b>31</b>
	A.1 SDG Goals	31
	A.2 Source Code	32
	A.3 Screen Shots	44
	A.4 Plagiarism Report	
	A.5 Paper Publication	



## **LIST OF TABLES**

<b>TABLE NO</b>	<b>TABLE DESCRIPTION</b>	<b>PAGE NO</b>
3.3.3.1	List Of Classes IN ML Model	17
5.1	Test Cases	25

## LIST OF FIGURES

FIG NO	FIGURE DESCRIPTION	PAGE NO
3.2.1	Architecture diagram for Traffic control system	6
3.3.1	Flowchart for AI system	8
3.3.2	Input Design	9
3.3.3	Module Design	9
3.3.3.1	Use Case Diagram for TRAFFIC CONTROL SYSTEM	10
3.3.3.2	Sequence Diagram for TRAFFIC CONTROL SYSTEM	11
3.3.3.3	Activity Diagram for TRAFFIC CONTROL SYSTEM	12
3.3.3.4	State Chart Diagram for TRAFFIC CONTROL SYSTEM	13
3.3.3.5	Class Diagram for TRAFFIC CONTROL SYSTEM	14
3.3.3.6	Component Diagram for TRAFFIC CONTROL SYSTEM	15
3.3.3.7	Package Diagram for TRAFFIC CONTROL SYSTEM	16
4.2.1	Unannotated image for training	19
4.2.2	Annotated image for training	20
5.2.1	CONFUSION MATRIX	26
5.2.2	F1 Confidence Curve	27
5.2.3	METRICS Accuracy Diagram	27
8.1	Medium traffic density	44
8.2	Low traffic density	44
8.3	High traffic density	45
8.4	Traffic Stimulation	45

# **CHAPTER 1**

## **INTRODUCTION**

Road congestion, characterized by spikes in traffic and prolonged commute times, poses significant challenges to urban mobility and transportation efficiency. Imperfect signaling exacerbates congestion, leading to further delays and frustration for commuters. The severity of road congestion intensifies when vehicle density surpasses the road's capacity, causing gridlock and impeding the smooth flow of traffic. To address this issue, the proposed system employs real-time monitoring and prediction of traffic intensity. When a deviation in predicted traffic intensity occurs, a real-time average is computed and updated in the database. This updated information is then utilized to train the model for learning, ensuring that the system adapts dynamically to changing traffic conditions. Central to the system's approach is the utilization of YOLOv5, a state-of-the-art object detection model, to accurately separate and identify different vehicle types. Additionally, Deep Learning techniques are employed to identify the ideal and optimal traffic density levels for efficient traffic management. By integrating these advanced technologies, the system aims to enhance traffic flow, minimize congestion, and improve the overall transportation experience for commuters. Through proactive monitoring and dynamic adjustment of traffic density, the system endeavors to mitigate congestion-related challenges and promote smoother, more efficient.

## **1.1 PROBLEM DEFINITION**

The problem of traffic congestion is a pressing issue in urban areas, characterized by the inefficient movement of vehicles, long wait times, and increased travel durations. Traditional traffic light systems often operate on fixed schedules or simple timers, which may not adapt well to fluctuating traffic conditions throughout the day. As a result, congestion can worsen during peak hours, leading to frustration among commuters and negative impacts on economic productivity and environmental sustainability. The dynamic nature of traffic congestion necessitates innovative solutions that can adapt in real-time to changing traffic patterns. The traditional approach of estimating signal timing based on fixed schedules or manual observation may not suffice in addressing the complexities of modern traffic flows. Therefore, there is a growing need for automated traffic light systems that can intelligently adjust signal timings based on real-time data. Overall, the traffic light automation project seeks to enhance the efficiency and effectiveness of traffic management systems by making them more responsive to real-time traffic conditions. By leveraging advanced technologies and dynamic algorithms, the system aims to mitigate congestion, improve traffic flow, and enhance the overall mobility experience for commuters in urban areas.

## **CHAPTER 2**

### **LITERATURE**

### **REVIEW**

In recent years, engineers have proposed a number of innovative solutions to solve real-time problems related to traffic regulations.

Javaid et al [1] introduced a system integrating mobile and web applications that allows officials to monitor traffic using radio frequency identification (RFID). RFID, combined with data from cameras and sensors, enables real-time monitoring of traffic density, facilitating rapid response to emergencies.

Lanke and Koul [2] proposed a system that relies heavily on RFID hardware for real-time traffic regulation. Their use of inductive loop detection and video analytics, using cameras and infrared sensors, demonstrates a comprehensive approach to traffic management.

Jacob et al [3] presented a system for classifying traffic into high, medium, and low categories using RASPBERRY AI with ultrasonic sensors and cameras. This multi-level classification system provides in-depth understanding of traffic patterns.

Shubo et al [4] implemented a computer vision-based solution using YOLOv4 and OpenCV to identify citizens violating traffic rules. Their deep neural network (DNN) algorithm detects violations such as driving without a helmet, driving the wrong way, reckless driving, traffic light violations, and illegal parking.

Yang et al [5] developed a proximal policy optimization algorithm (modified PPO) for traffic control. The system is based on deep reinforcement learning (DRL) technology, which uses software-defined Internet of Things (SD-IoT) to adjust traffic lights and control vehicles on a global scale, improving urban traffic control performance.

Jin et al [6] introduced an adaptive signal control system that simulates the behavior of professional signal control engineers. This human-in-the-loop parallel learning framework uses machine learning, data analytics, and cloud computing to process and analyze real-time traffic data.

Rao et al [7] proposed a model to improve mobility using Internet of Things (IoT) based traffic messaging devices. The system uses machine learning algorithms, including neural networks and decision trees, to process traffic data in real time, demonstrating the transformative potential of technology in urban life.

Bali et al [8] introduced IoT-based technology that creates “green corridors” for emergency vehicles. Using RFID readers to scan emergency vehicles' RFID tags, this system effectively manages traffic, allowing emergency vehicles to quickly reach their destinations. Together these studies highlight ongoing efforts to integrate technology and data analytics for effective traffic management. The proposed method builds on these foundations, using YOLOv5 to separate vehicles and DL for accurate traffic density identification.

## **CHAPTER 3**

### **THEORETICAL BACKGROUND**

#### **3.1 IMPLEMENTATION ENVIROMENT**

##### **SOFTWARE REQUIREMENT**

Windows 7

GOOGLE COLAB OR JYPUTER NOTEBOOKS

##### **HARDWARE REQUIREMENT**

GPU ENABLED Central Processor

Memory (RAM): 16 GB

Hard Drive: 32 GB

Stable Internet Connection

LIDAR CCTV Camera

Wi-Fi Module IEEE 802.11

### 3.1 SYSTEM ARCHITECTURE

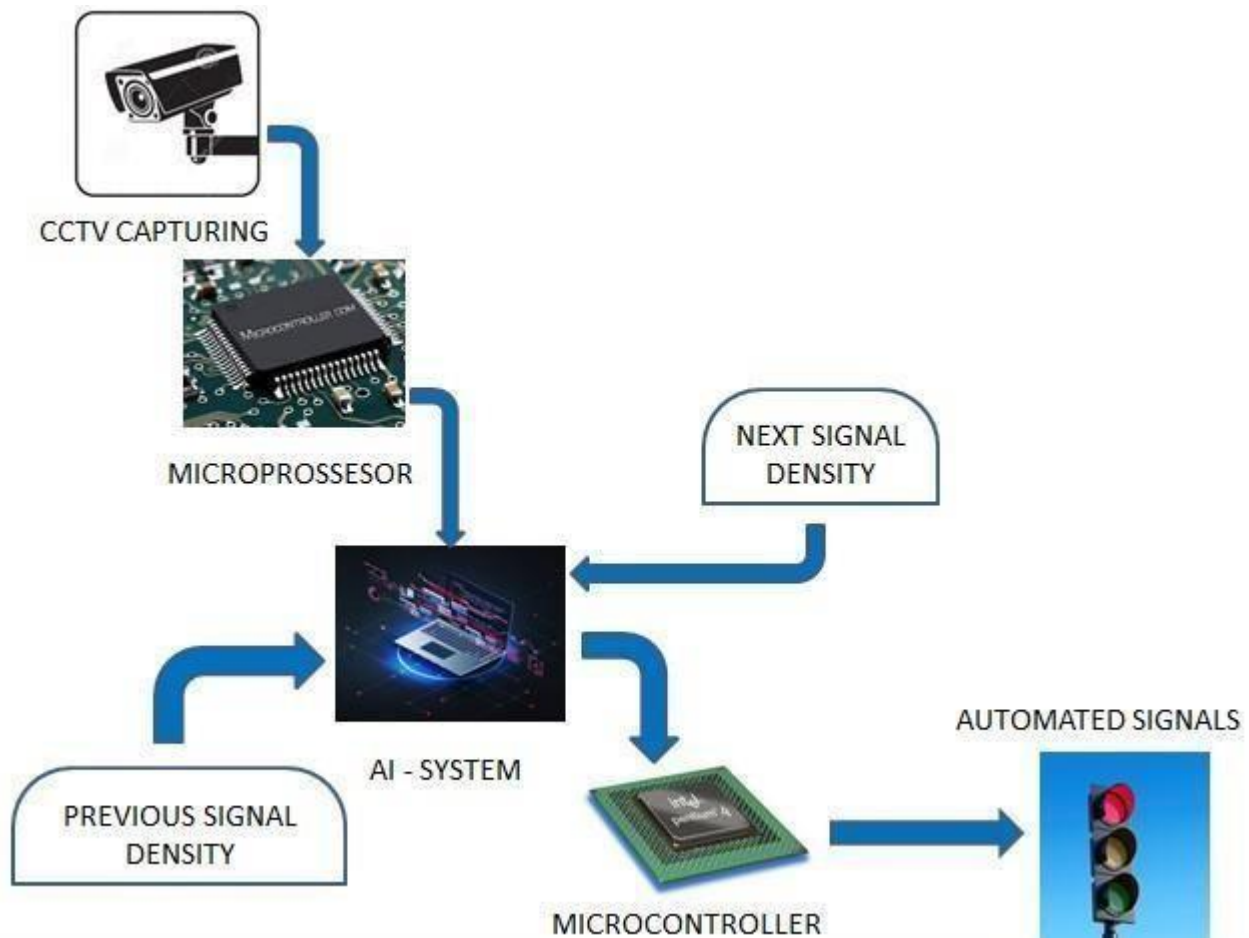


Fig 3.2.1 Architecture diagram for Traffic control system

The figure 3.2.1 shows the entire architecture of AI system and calculating the density which makes the signals to dynamically change as per the estimated time. This is done from camera capturing images of the traffic and detecting the vehicles as per the classes and therefore finding the density and green signal timing.



### **3.2 PROPOSED METHODOLOGY**

This system aims to regulate the traffic congestion dynamically i.e. based on traffic density and traffic flow rate. It aims to improve traffic flow and reduce congestion on busy roads.

This system computes length of time that a traffic signal should remain green based on the value of traffic density in the lane. It is computed by detecting and counting the number of vehicles by the image processing and object detection algorithms such as neural network architecture (YOLOv5) on the feeds from the camera of real-time traffic data.

Based on the density values, system also make decisions about lane closures and traffic rerouting. The main objective of the project is to provide a safer and more efficient travel experience for drivers on the road.

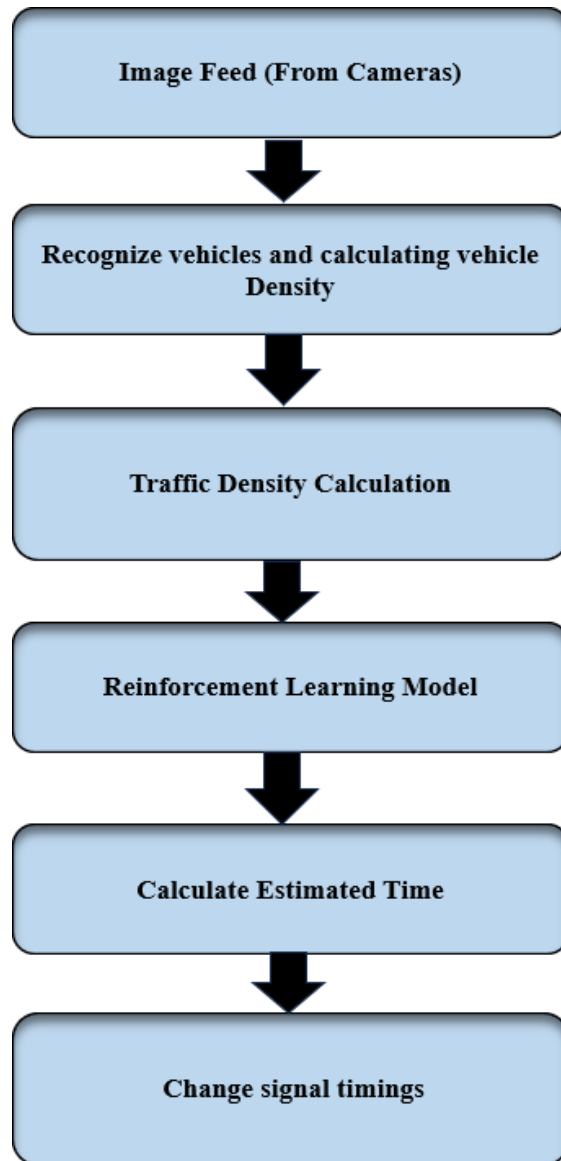


Fig 3.3.1 Flowchart

In Fig 3.3.1 depicts the work flow of the AI system which gets image as the feed and processes to get the vehicle count in that particular lane. This is further used to detect the traffic density which in turn makes the decision of the traffic regulations.

```

Direction-wise Vehicle Counts
Direction 1 : 37
Direction 2 : 41
Direction 3 : 49
Direction 4 : 53
Total vehicles passed: 180
Total time: 300

```

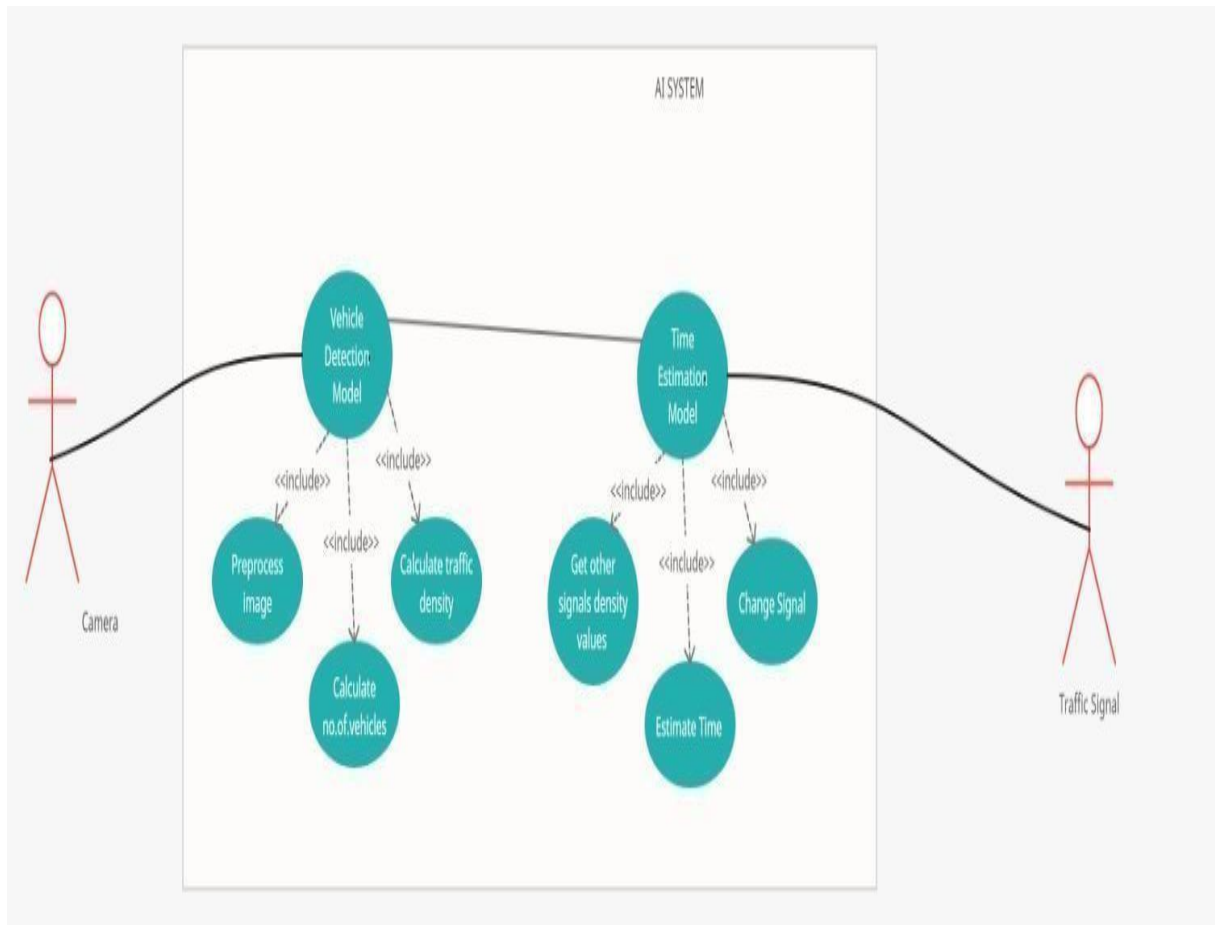
### 3.3.2 Input Design

Fig 3.3.2 shows the Input Design involves determining how users interact with the system to input data or commands, ensuring user-friendly interfaces and efficient data entry methods.



### 3.3.3 Module Design

Fig 3.3.3 shows the module Design focuses on dividing the system into smaller, manageable units or modules, each responsible for specific functionalities, promoting modularization and ease of maintenance in software development.



3.3.3.1 Use case diagram for TRAFFIC CONTROL SYSTEM

The use case diagram in Fig 3.3.3.1 illustrates the interactions between the Traffic Control System and various hardware components, including CCTV cameras, traffic lights, and sensors. Each component is depicted as a separate actor, representing its role and functionality within the system. The diagram delineates how these hardware components interact with the Traffic Control System to perform specific actions, such as capturing real-time video feeds, detecting vehicles and pedestrians, and controlling traffic signals based on the detected.

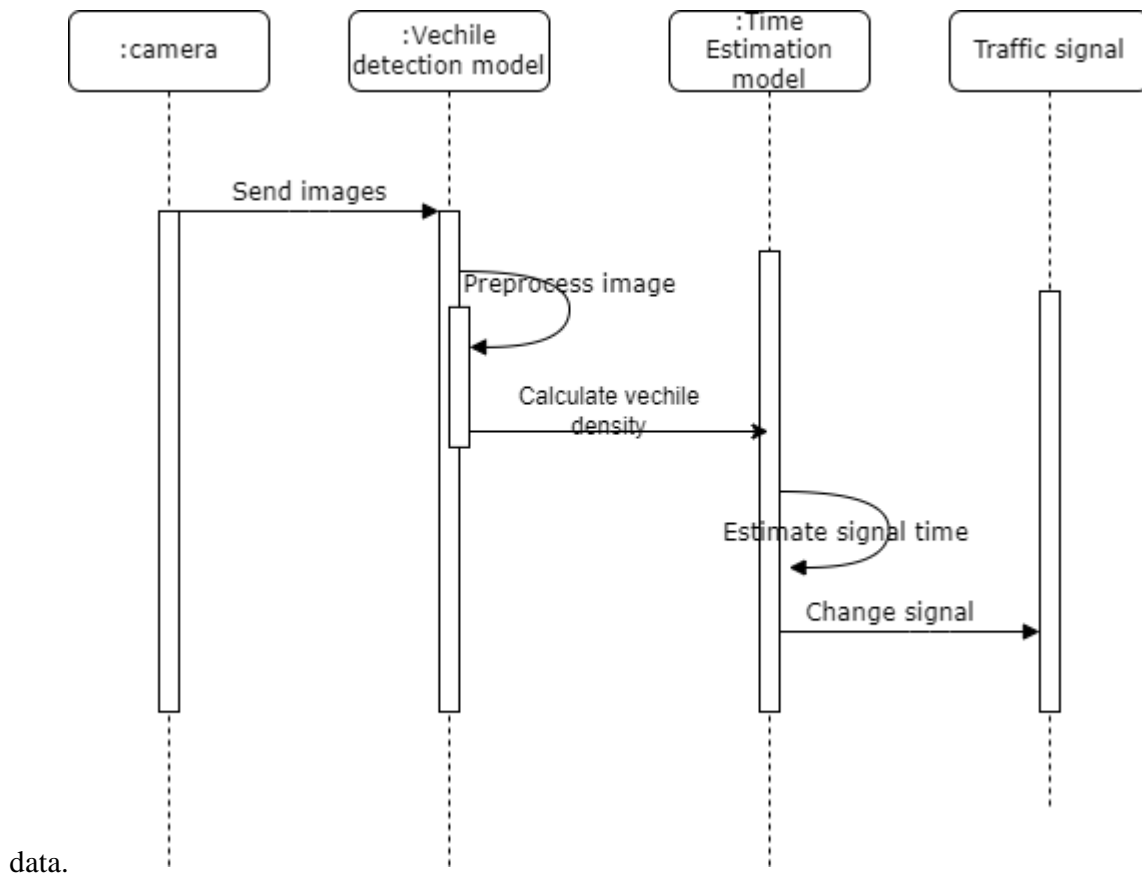


Fig 3.3.3.2 Sequence diagram for TRAFFIC CONTROL SYSTEM

The Sequence diagram in Fig 3.3.3.2 illustrates the chronological sequence of activities within the Traffic Control System for regulating traffic. It details the interactions between various components of the system, such as sensors, traffic lights, and the central control unit, depicting the order in which these components communicate and perform actions. By visually representing the flow of messages and actions between different elements of the system, the diagram provides a clear understanding.

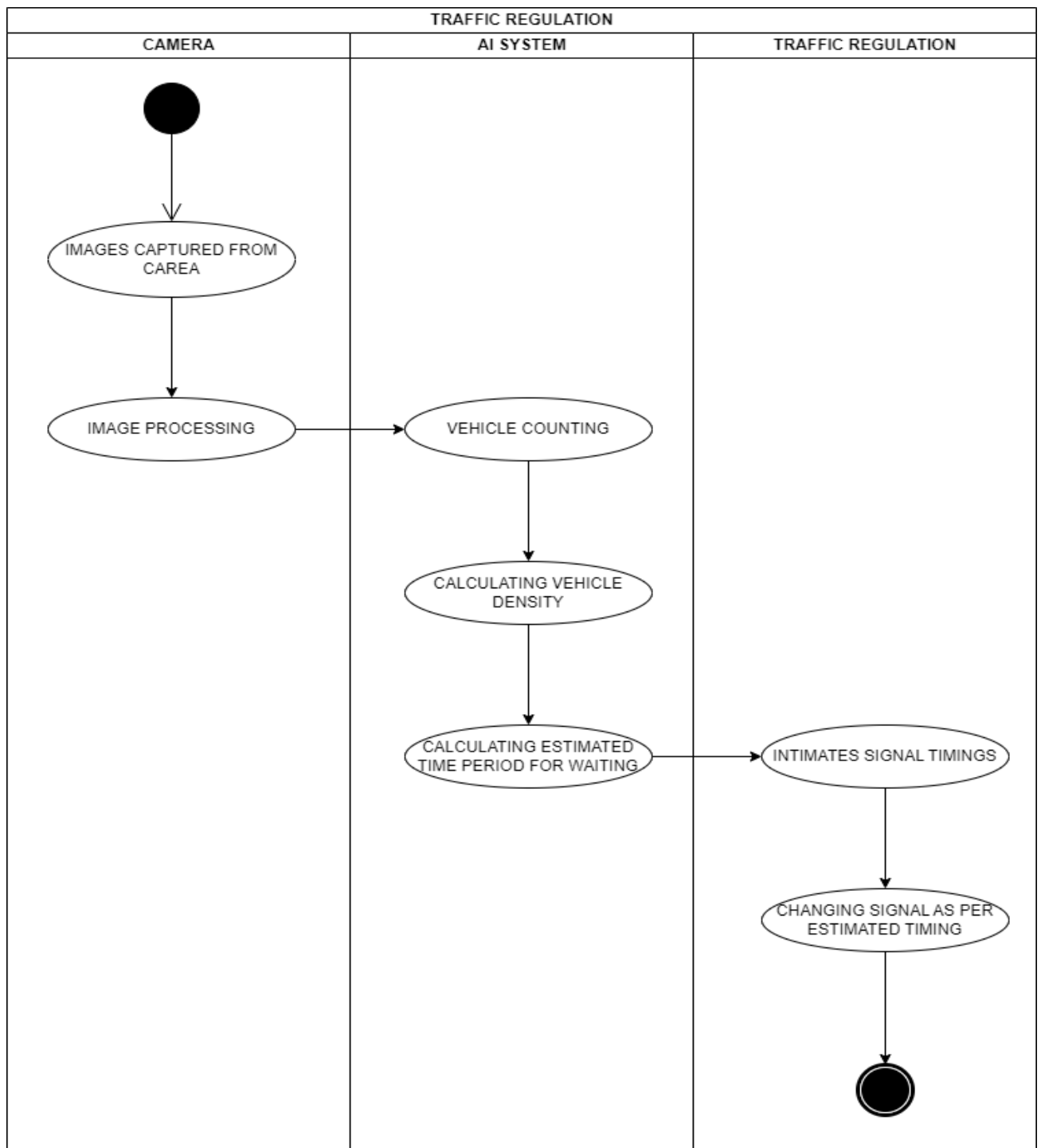


Fig 3.3.3.3 Activity diagram for TRAFFIC CONTROL SYSTEM

In Fig 3.3.3.3 the Activity diagram of TRAFFIC CONTROL SYSTEM shows the flow of states involved in the process of regulating traffic.

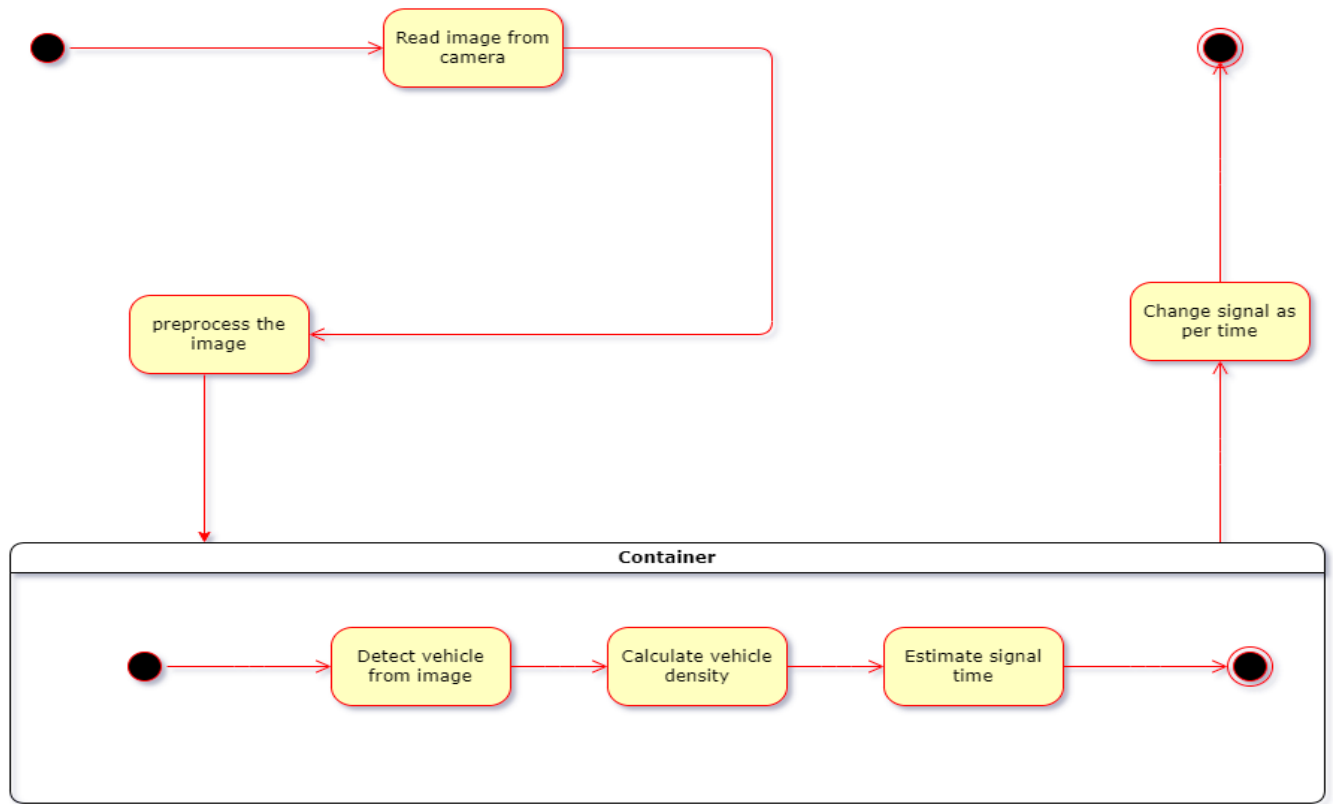


Fig 3.3.3.4 State Chart diagram for TRAFFIC CONTROL SYSTEM

In Fig 3.3.3.4 the State Chart diagram of the Traffic Control System delineates the states integral to traffic regulation, encompassing. Through this diagram, the system's behavior during various traffic scenarios, including normal operation, transition periods, and emergency situations, is visualized.

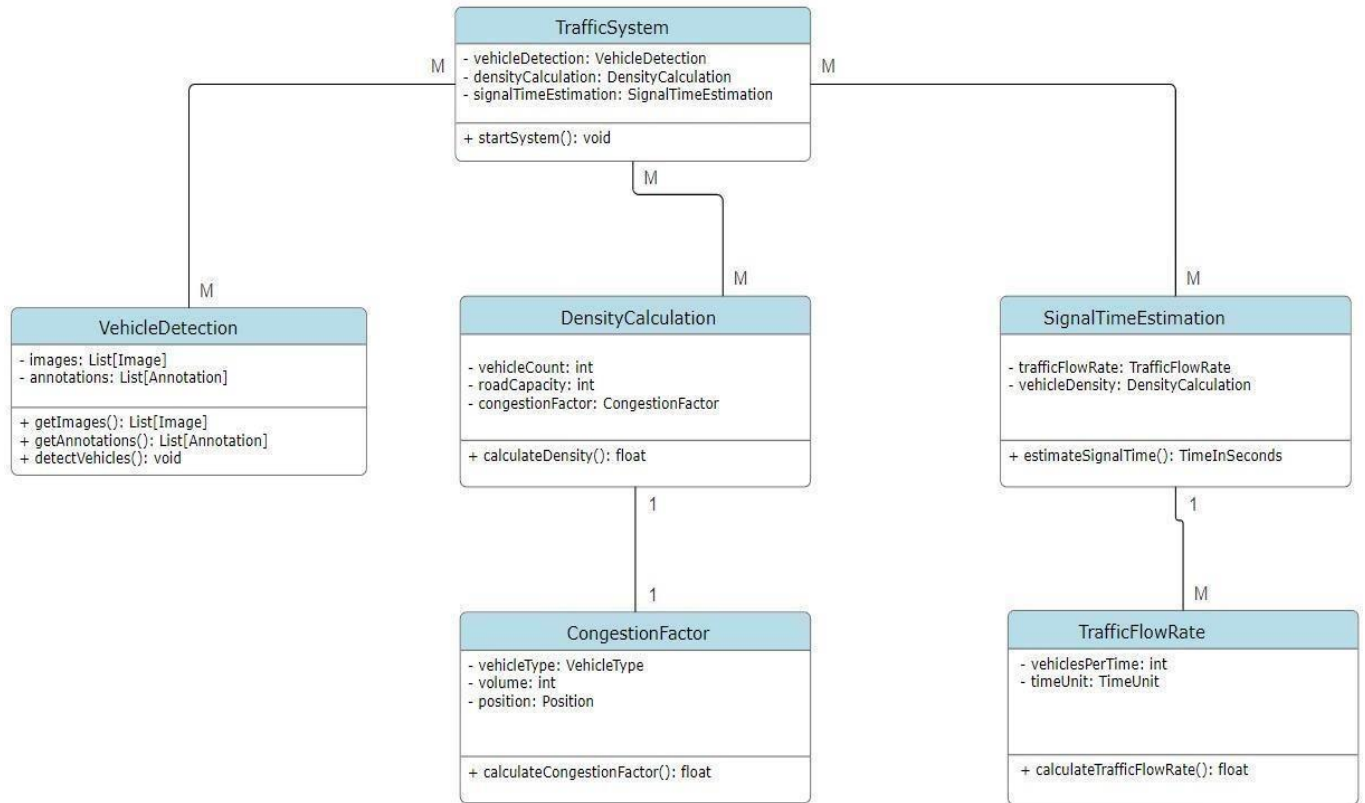


Fig 3.3.3.5 Class diagram for TRAFFIC CONTROL SYSTEM

In Fig 3.3.3.5 the class diagram outlines a traffic regulation system comprising components for vehicle detection, data preprocessing, model development, density calculation, and signal time estimation. Key classes include Traffic System orchestrating the overall system, and specialized components such as Vehicle Detection, Density Calculation and Signal Time Estimation.



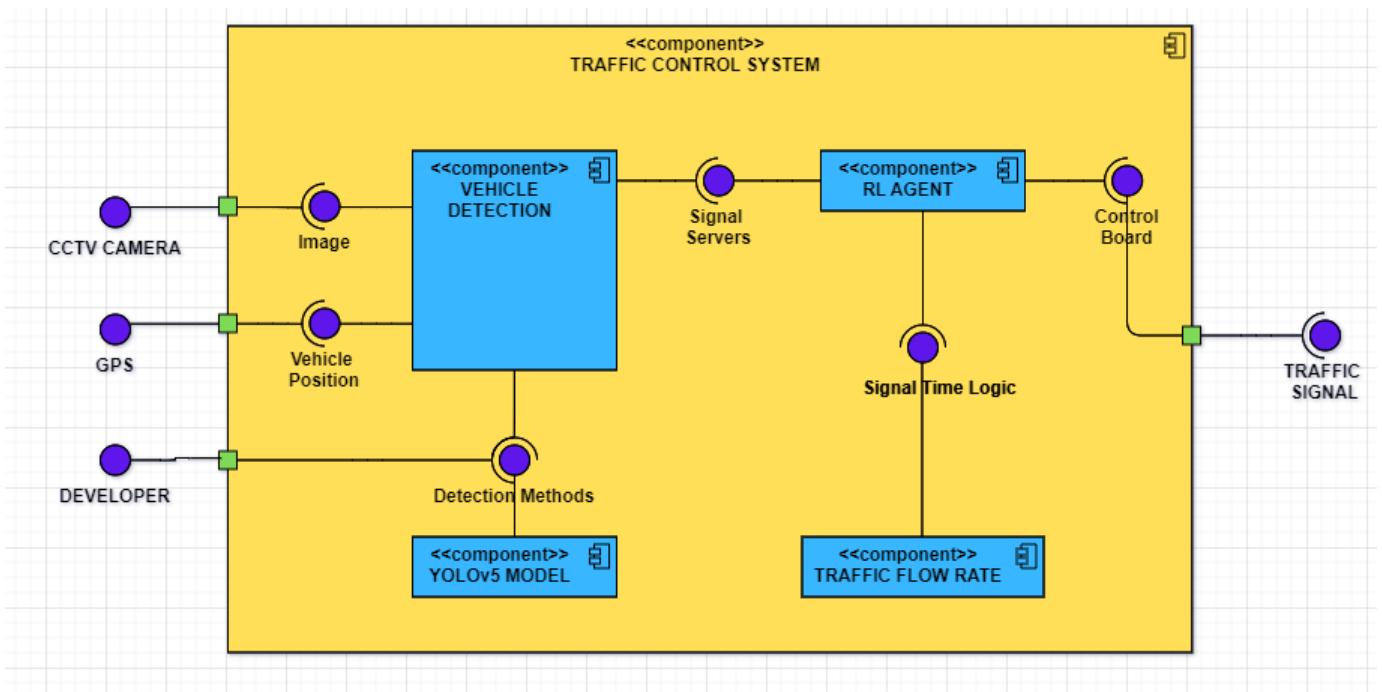


Fig 3.3.3.6 Component diagram for TRAFFIC CONTROL SYSTEM

In Fig 3.3.3.6 the component diagram for a traffic control system illustrates the modular organization of its software components, depicting their interactions and dependencies. This diagram serves as a blueprint for system development, aiding in design, implementation, and maintenance processes.

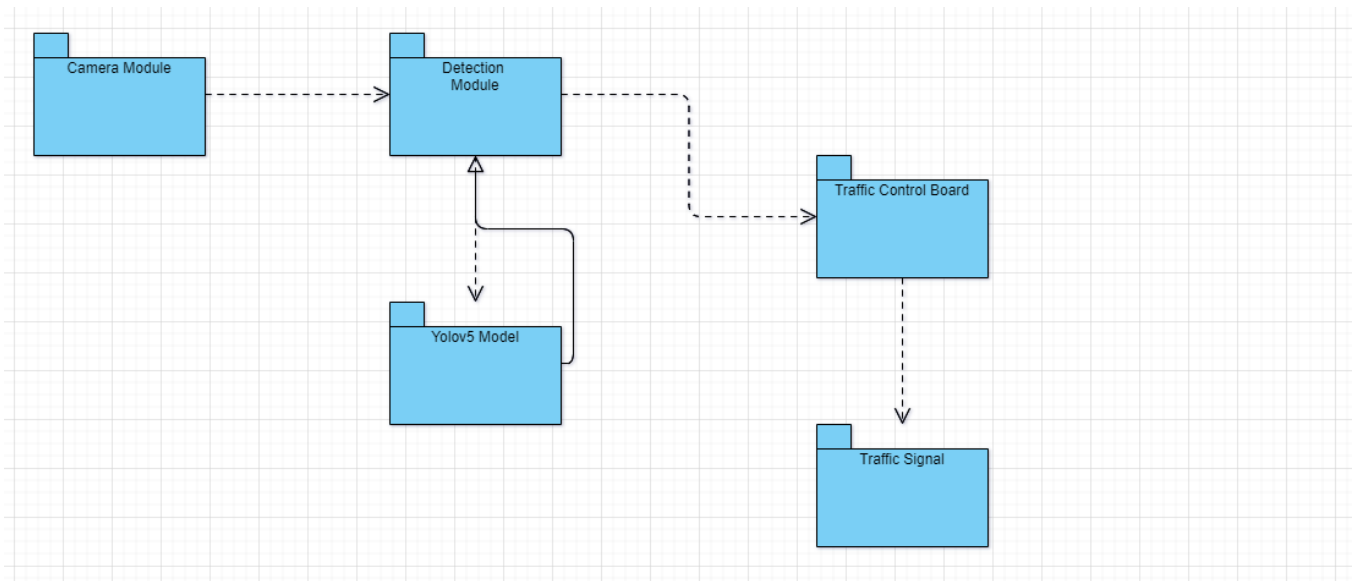


Fig 3.3.3.7 Package diagram for TRAFFIC CONTROL SYSTEM

In Fig 3.3.3.7 the package diagram for a traffic control system provides a hierarchical overview of the system's organization, presenting the grouping of related software components into cohesive units. This diagram aids in visualizing the system's structure, dependencies, and relationships between different parts, facilitating better understanding and management of software development efforts.

<b>S.No</b>	<b>Classes</b>	<b>Average Vehicle Speed</b>	<b>Congestion Factor</b>
1	Ambulance	50 Km/H	37.5 M <sup>3</sup>
2	Army Vehicle	40 Km/H	24.2 M <sup>3</sup>
3	Auto Rickshaw	30 Km/H	6.75 M <sup>3</sup>
4	Bicycle	15 Km/H	1.188 M <sup>3</sup>
5	Bus	40 Km/H	105 M <sup>3</sup>
6	Car	50 Km/H	12.15 M <sup>3</sup>
7	Garbage Van	30 Km/H	27.5 M <sup>3</sup>
8	Human Hauler	25 Km/H	10.8 M <sup>3</sup>
9	Minibus	40 Km/H	25 M <sup>3</sup>
10	Minivan	45 Km/H	17.28 M <sup>3</sup>
11	Motorbike	40 Km/H	1.92 M <sup>3</sup>
12	Pickup	45 Km/H	18 M <sup>3</sup>
13	Police Car	50 Km/H	12.96 M <sup>3</sup>
14	Rickshaw	20 Km/H	5.4 M <sup>3</sup>
15	Scooter	35 Km/H	1.512 M <sup>3</sup>
16	Suv	55 Km/H	15.552 M <sup>3</sup>
17	Taxi	45 Km/H	12.15 M <sup>3</sup>
18	Three Wheelers (Cng)	30 Km/H	6.75 M <sup>3</sup>
19	Truck	40 Km/H	45 M <sup>3</sup>
20	Van	45 Km/H	25 M <sup>3</sup>
21	Wheelbarrow	5 Km/H	0.576 M <sup>3</sup>

### 3.3.3.1 LIST OF CLASSES IN ML MODEL

In table 3.3.3.1 all the classes which are determined in our deep learning model is used. Only with the help of the classes and labeling, our model can understand, train and validate the same.

## CHAPTER 4

### SYSTEM IMPLEMENTATION

Image classification consists of 5 main modules.

They are:

- Vehicle Detection
- Data preprocessing
- Model Development and Evaluation
- Density Calculation
- Estimating Signal Time

#### 4.1 Vehicle Detection

The initial phase in the development of the vehicle detection system involves the preparation of a comprehensive dataset containing images or video sequences capturing various vehicle types and diverse traffic scenarios. The dataset is crucial for training an effective vehicle detection model. For this purpose, we utilize data sourced from the Dhaka-AI dataset available on Kaggle. To enhance the dataset and improve the robustness of the vehicle detection model, the following strategies are used:

Images or video frames depicting diverse traffic scenarios such as highways, urban streets, intersections, and parking lots are included. Images during different weather conditions, including sunny, rainy, and foggy conditions are captured.

A wide range of vehicle types, including cars, trucks, motorcycles, bicycles, and pedestrians. Images featuring different vehicle sizes, colors, and shapes are collected to improve model generalization.

Introduce images or frames with occlusions, where vehicles partially obstruct each other, challenging the model to handle complex scenes. Include scenarios with varying lighting conditions, shadows, and reflections to simulate real-world challenges.

## 4.2 Data preprocessing

The dataset must be preprocessed to remove any noise or unwanted artifacts related to tasks such as image resizing, normalization, and filtering. After preprocessing, the data needs to be labeled by manually marking the position of each vehicle in each frame or frames of the video sequence. After labeling, divide the labeled dataset into training, validation, and test sets to develop the model. The total data is broken down for two reasons. One is for training and the other for validation.



Fig 4.2.1 Unannotated image for training

In Fig 4.2.1 Without Annotation Data preprocessing is a critical step in preparing raw data for machine learning models. It involves various techniques to clean, transform, and organize the data to improve its quality and usability for analysis. In this stage, raw data from diverse sources is collected and formatted into a structured dataset suitable for training and validation.

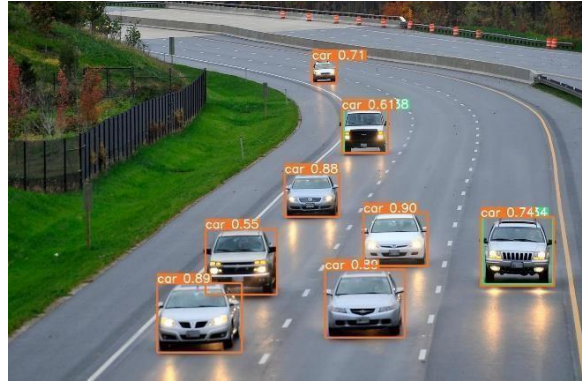


FIG 4.2.2 Annotated image for training

Data preprocessing is a crucial step in the machine learning pipeline, aimed at preparing raw data for analysis and model training. This stage involves various techniques and procedures to clean, transform, and organize the data, ensuring its quality and suitability for downstream tasks.

**Handling Missing Values:** One common preprocessing task involves dealing with missing data points. Techniques such as imputation or removal of missing values are applied to ensure completeness and consistency in the dataset.

**Removing Duplicates:** Duplicated entries within the dataset can skew analysis results and model training. Hence, identifying and removing duplicate records is an essential preprocessing step to maintain data integrity.

**Scaling Features:** Features often have different scales, which can affect the performance of certain machine learning algorithms. Scaling techniques such as standardization or normalization are applied to bring all features to a similar scale, ensuring fair comparison and accurate model training.

**Encoding Categorical Variables:** Machine learning models typically require numerical inputs, necessitating the encoding of categorical variables into numeric representations. Techniques like one-hot encoding or label encoding are commonly used for this purpose.

**Splitting Data:** The dataset is divided into training and testing sets to evaluate model performance accurately. Typically, a portion of the data is reserved for training the model, while the remainder is used for testing its performance on unseen data.

**Noise Reduction:** In some cases, the raw data may contain noise or irrelevant information that can adversely affect model performance. Noise reduction techniques such as smoothing or filtering are applied to enhance the signal-to-noise ratio and improve model accuracy.

Data preprocessing plays a critical role in ensuring the quality and reliability of machine learning models. By addressing data inconsistencies, missing values, and irrelevant information, preprocessing sets the stage for effective model training and analysis.

### **4.3 Model Development and Evaluation**

In the process of developing and evaluating the YOLOv5 model for vehicle detection, several key steps are undertaken to ensure accurate and robust performance. Here's an expanded explanation of the mentioned steps:

The training process involves iterating through epochs, where each epoch represents a full pass through the entire dataset.

Training parameters such as learning rate, optimizer choice, and loss function are carefully tuned to achieve optimal model performance.

Callbacks, such as early stopping and model checkpointing, are implemented to monitor the model's progress during training.

Validation data is used to assess the model's performance on unseen data and prevent overfitting.

During training, various performance metrics are monitored, including:

- **Seconds per step:** Measures the time taken for each training step.
- **Loss:** Indicates the overall model error during training.
- **Precision loss:** Focuses on precision-related errors in object detection.
- **Validation loss:** Evaluates the model's performance on validation data.
- **Epochs Adjustment:** The number of epochs is a hyperparameter that determines how many times the entire dataset is used to train the model. Increasing epochs is considered to improve model accuracy, but careful monitoring is required to prevent overfitting.

At the completion of the training process, the model's final accuracy is predicted, and the effectiveness of the vehicle detection is assessed. The annotated dataset is visualized using AI graph for labeling, a tool commonly used for labeling images with bounding boxes. The labeling information is typically stored in XML format.

The XML-format labeling is converted to text format, facilitating compatibility with the YOLOv5 model's requirements. IN each text file, specific labeling limits are specified to accurately define the boundaries and characteristics of detected vehicles.

#### **4.4 Density Calculation**

After the YOLOv5 model has detected the vehicles in an image or video frame, the next step is to calculate the vehicle density. This is typically done by using formulas that take into account the number of vehicles detected in the particular lane. It is calculated by dividing the number of vehicles and the width of the road .The density value varies with various factors such as the average vehicle length, and the spacing between vehicles. Vehicles are broadly categorized into 4 classes based on the size, weights, gross vehicle weight rating (GVWR) and its contribution to the traffic density.

The vehicles in these classes are as follows:

##### **Light-duty vehicles**

Bicycles and motorcycles generally fall under the category of light-duty vehicles, as they have a relatively low gross vehicle weight compared to other vehicles.

##### **Medium-duty vehicles**

The vehicles having GVWR less than or equal to 6000 pounds (2722 kg) which includes passenger cars, SUVs, pickup trucks, and vans. Light-duty vehicles typically have a smaller engine and are designed for personal use.

##### **Heavy-duty vehicles**

Vehicles having a GVWR of more than 6,000 pounds (2,722 kg) which includes trucks, buses, and other commercial vehicles. These vehicles typically have a larger engine and are designed for commercial use. Very heavy-duty trucks:



$$\text{Vehicle density} = \frac{\sum (\text{weight of the vehicle} * \text{no of vehicles/Average Vehicle Road})}{\text{width of the road}}$$

Once the vehicle density is calculated, the average density is computed by taking the mean of the densities with multiple frames of the video feed for precise density value.

#### 4.5 Estimating Signal Time

The last phase of the traffic regulation system involves estimating the signal time, a critical factor in ensuring efficient traffic flow through intersections. This estimation is based on the analysis of vehicle density, which is a key parameter influencing signal duration. The following steps explain the process in detail:

The traffic flow rate is determined by counting the number of vehicles passing through the intersection within a specific time frame. This metric provides insight into the overall traffic dynamics.

The signal time, or the duration for which the traffic signal remains green, is estimated based on the analyzed vehicle density.

The estimation process may adhere to established traffic engineering guidelines that consider factors such as:

- Intersection geometry and layout.
- Pedestrian crossing requirements.
- Time-of-day variations.
- Special events or circumstances

In advanced systems, adaptive signal control mechanisms may be employed. These systems dynamically adjust signal timings based on real-time traffic conditions, optimizing signal time for current demand.

The estimation of signal time can be integrated with Intelligent Transportation Systems that leverage data from various sensors, cameras, and traffic monitoring devices. This integration enhances the responsiveness and adaptability of signal control.

For optimal traffic management, the estimated signal time may be coordinated with neighboring intersections to create a synchronized traffic signal network.

The signal time estimation process considers variations in traffic density throughout the day, accommodating peak hours and off-peak periods. The system continually monitors traffic conditions, allowing for dynamic adjustments to signal timings in response to changing traffic patterns. Establishing a feedback loop ensures that the estimated signal time aligns with the actual traffic flow, enabling the system to learn and adapt over time.

## CHAPTER 5

### Results & Discussion

#### 5.1 Performance Parameters / Testing

TEST CASE ID	TESTCASE/ ACTION TO BE PERFORMED	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1.	Feeding image with “Motorbike” in it	Detecting Motorbike and annotate it	Detecting the Motorbike and annotate it	Pass
2.	Feeding image with “Taxi” in it	Detecting the Taxi and annotate it	Detecting the Taxi and annotate it	Pass
3.	Feeding image with “Car” in it	Detecting the Car and annotate it	Detecting the Car and annotate it	Pass
4.	Feeding image with “Ambulance” in it	Detecting the Ambulance and annotate it	Detecting the Ambulance and annotate it	Pass
5.	Feed image with low traffic density	Displays low traffic density value	Displays low traffic density value	Pass
6.	Feed image with high traffic density	Displays high traffic density value	Displays high traffic density value	Pass

Table No 5.1 Test Results



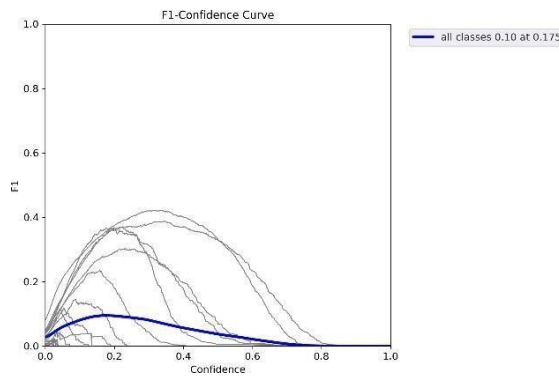


FIG 5.2.2 F1 Confidence Curve

Fig 4.2 is marked along the F1 and confidence for the data. This F1 is marked with taking into consideration Positive predictions and negative predictions which are in both classifier predictions and labelled training data. Our confidence curve currently being conducted with 3000 images, which are broken into only 16 batches and with epochs has given the curve below 0.50. This will be increased when the no of images, epochs, and batches are increased.

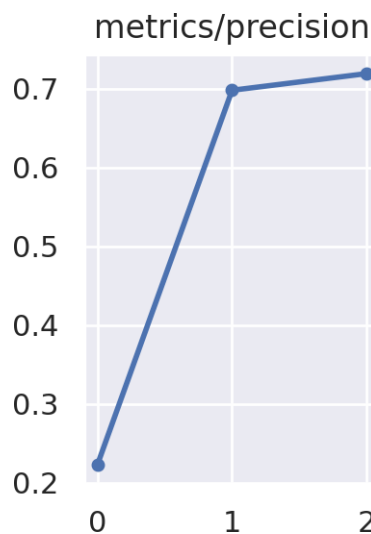


FIG 5.2.3 METRICS Accuracy Diagram

FIG 4.4 shows the accuracy of the metric which has raised from 0.2 to 0.7. This metric's accuracy shows the no of correct positive predictions made. Even in FIG 4.4, the accuracy can be increased to higher rates when prediction epochs are increased. It is calculated with the ratio of correctly predicted positive instances divided by the total number of positive instances that were used to predict.

# CHAPTER 6

## CONCLUSION AND FUTURE WORK

### 6.1 CONCLUSION

This model reduces the traffic drastically which can be replaced for the conventional way of traffic movement. The main objective of developing this system is to minimize the waiting time on the road by optimizing traffic flow when there is no traffic on the other side. Moreover the system also reduces assignment volume. In addition, the system addresses various traffic violations such as unauthorized parking, motorbike riders without helmets, rerouting, and providing a clear passageway for emergency vehicles by utilizing the technique of image classification.

### 6.2 FUTURE ENHANCEMENTS

#### **Active emergency vehicle :**

Our system can identify emergency vehicles by tracking the GPS signals they send to the system. Once GPS is started, the system will begin tracking the vehicle's location in real time. The route the emergency vehicle takes is clear, allowing the vehicle to move freely to its destination.

#### **Elite Artery:**

When VIPs visit our state, traffic flow must be adjusted to accommodate their movements. Our system plays an important role in traffic management during VIP visits. As soon as the VIP convoy begins its journey, the traffic lights along the route will be adjusted to ensure a sufficient level of safety for the VIPs.

#### **Driving against the flow:**

Our system can detect vehicles going in the wrong direction on the road by analyzing their movements. The system treats all lanes as one-way and marks all vehicles traveling in the opposite direction as wrong-way drivers. The driving direction on the road is predetermined in our system, although it can be adapted to the specific road being monitored.

#### **Reroute:**

In the event of traffic congestion, rerouting can be an effective strategy to minimize delays and minimize impact on people in traffic. Our system is equipped to analyze real-time traffic patterns using image classification with DEEP LEARNING and recommend alternative routes to drivers.

**Motorcyclists not wearing helmets:**

Our system uses a four-step process to detect motorcyclists not wearing helmets. First, all motorcycles and riders are detected and classified in the video stream. The motorbike storage area is then separated into individual images, which are fed into the YOLOv5 microscopic model in the third step. This model detects the presence of a helmet on the driver in each image. If no helmet is detected, the driver will be marked in the video image as an unhelmeted driver.

**Identify suspicious vehicles:**

Detect vehicles that are not authorized to be on the road at this exact time. This is done when the image is sent to the YOLO V5 model and is designed for image processing. The image shown detects unauthorized vehicles using our model. Vehicles caught in traffic jams recorded by surveillance cameras will be framed and processed.

**Ignore the signal:**

Traffic light is integrated into our system, and it scans intersections on the road to detect people who do not obey traffic laws whether intentionally or unintentionally when the light red traffic. Visual markers at intersections serve as reference points for our system. If a vehicle passes the marker when the light is red, it will be recorded as a traffic light violation.

## REFERENCES

- [1] Sabeen Javaid, Ali Sufian, Saima Pervaiz, and Mehak Tanveer , “Smart Traffic Management System Using Internet of Things” , International Conference on Advanced Communications Technology(ICACTION), 2018
- [2] Ninad Lanke and Sheetal Koul, “Smart Traffic Management System”, International Journal of Computer Applications , 2013
- [3] Sheena Mariam Jacob, Shobha Rekh, Manoj G and J John Paul, “Smart Traffic Management System with Real Time Analysis”, 2018
- [4] Fahimul Hoque Shubho, Fahim Iftekhar, Ekhfa Hossain and Shahnewaz Siddique, “Real-time traffic monitoring and traffic offense detection using YOLOv4 and OpenCV DNN”, 2021 IEEE Region 10 Conference (TENCON).
- [5] Jiachen Yang, Jipeng Zhang and Huihui Wang , “Urban Traffic Control in Software Defined Internet of Things via a Multi-Agent Deep Reinforcement Learning Approach” , IEEE Transactions on Intelligent Transportation Systems
- [6] Junchen Jin, Haifeng Guo, Jia Xu, Xiao Wang and Fei-Yue Wang ,”An End-to-End Recommendation System for Urban Traffic Controls and Management Parallel Learning Framework”,IEEE Transactions on Intelligent Transportation Systems
- [7] T. V. Janardhana Rao, R. Arun Sekar, B. Drakshyani, P Kalyan Chakravarthi, N. Kumaresan and S. P. Karthi , “IoT based Smart solution for Traffic congestion at Metro Cities” , 2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems.
- [8] Dr. Vikram Bali, Ms. Sonali Mathur, Dr. Vishnu Sharma and Dev Gaur ,”Smart Traffic Management System using IoT Enabled Technology”, 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN).
- [9] [www.geeksforgeeks.org](http://www.geeksforgeeks.org)
- [10] [www.stackoverflow.com](http://www.stackoverflow.com)
- [11] <https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data>



## **APPENDICES**

### **A.1 SDG GOALS**

Goal 3: Good Health and Well-being - The system prioritizes emergency scenarios by ensuring the smooth flow of vital services like ambulances, thus promoting better health outcomes in urban areas.

Goal 9: Industry, Innovation, and Infrastructure - The proposed method leverages machine learning and deep learning techniques to innovate traffic management systems, contributing to advancements in urban infrastructure.

Goal 11: Sustainable Cities and Communities - By reducing traffic congestion through real-time monitoring and regulation of traffic signals, the system aims to create more sustainable and livable urban environments.

Goal 13: Climate Action - By reducing traffic congestion and associated greenhouse gas emissions, the system aligns with efforts to mitigate climate change and promote sustainable transportation practices.

## A.2 Source Code

### Agent.ipynb

```
# code to install dependency
!pip install stable_baselines3[extra]
# Import GYM stuff
import gym
from gym.spaces import Box

# Import helpers
import numpy as np
import random
import os

# Import Stable baselines stuff
from stable_baselines3 import PPO
from stable_baselines3.common.vec_env import DummyVecEnv
from stable_baselines3.common.env_checker import check_env
from stable_baselines3.common.evaluation import evaluate_policy
!git clone https://github.com/ultralytics/yolov5 # clone
%cd yolov5
%pip install -qr requirements.txt # install

# import torch
import utils
display = utils.notebook_init() # checks
import torch
import os
import pandas as pd
```

```

model = torch.hub.load('.', 'custom', path='/content/drive/MyDrive/last.pt', source='local')
path="/content/drive/MyDrive/dataset/images/train/"
files=os.listdir(path)[:20]
def den_calc(res,capacity):
    pred = res.pandas().xyxy[0].value_counts("name")
    vech = list(pred.index)
    if len(vech) != 0:
        sum=0
        for i in vech:
            sum += (pred[i] * Vech_weights[i][1])/Vech_weights[i][0]
        # print("Density:",sum/capacity)
        return sum/capacity
    else:
        return 0
env = DummyVecEnv([lambda: traffic_env()])
env=traffic_env()
model = PPO('MlpPolicy', env, verbose=1, tensorboard_log="/content/log path")
model.learn(total_timesteps=100000)
model.save("/content/drive/MyDrive/PPO_model/ppo_model_100k_new")

```

### **Stimulation.py**

```

import random
import time
import threading
import pygame
import sys
import os
import torch

from io import BytesIO
import numpy as np
from tkinter import Tk, filedialog
from fns import calc_density

```

```

from env import traffic_env
from stable_baselines3 import PPO
pygame.init()
simulation = pygame.sprite.Group()
print("The Model is Loading. .... ")
path_hubconfig = "yolov5"
path_weightfile = "best.pt" # or any custom trained model
model = torch.hub.load(path_hubconfig, "custom", path=path_weightfile, source="local")
rl_model=PPO.load("./ppo_model_100k_new")
# Screensize
screenWidth = 1400
screenHeight = 800
screenSize = (screenWidth, screenHeight)

image = None

screen = pygame.display.set_mode(screenSize)
pygame.display.set_caption("TRAFFIC CONTROL SYSTEM")

imgcords=[(150,30),(929,30),(929,550),(140,550)]
black = (0, 0, 0)
white = (255, 255, 255)
RED = (255, 0, 0)

font = pygame.font.Font(None, 30)

# Default values of signal timers
defaultGreen = {0:20, 1:30, 2:40, 3:10}
defaultRed = 150
defaultYellow = 5

signals = []

```

```

noOfSignals = 4
currentGreen = 0 # Indicates which signal is green currently
nextGreen = (currentGreen+1)%noOfSignals # Indicates which signal will turn green next
currentYellow = 0 # Indicates whether yellow signal is on or off

speeds = {'car':2.25, 'bus':1.8, 'truck':1.8, 'bike':2.5} # average speeds of vehicles

# Coordinates of vehicles' start
x = {'right':[0,0,0], 'down':[755,727,697], 'left':[1400,1400,1400], 'up':[602,627,657]}
y = {'right':[348,370,398], 'down':[0,0,0], 'left':[498,466,436], 'up':[800,800,800]}

paused=True
timeElapsed = 0
simulationTime = 300
rounds=1
timeElapsedCoods = (1100,50)
# vehicleCountTexts = ["0", "0", "0", "0"]

# vechicle_count=[0,0,0,0]
vehicleCountCoods = [(530,180),(810,180),(810,530),(530,530)]
vehicleCountCoods_cnt = []
for i in range(4):
    vehicleCountCoods_cnt.append((vehicleCountCoods[i][0]+20,vehicleCountCoods[i][1]))
class TrafficSignal:
    def __init__(self, red, yellow, green):
        self.red = red
        self.yellow = yellow
        self.green = green
        self.signalText = ""
class Button:
    def __init__(self, x, y, width, height, color, text):
        self.rect = pygame.Rect(x, y, width, height)
        self.color = color
        self.text = text

```

```

def draw(self, screen):
    pygame.draw.rect(screen, self.color, self.rect)
    pygame.draw.rect(screen, white, self.rect, 2)
    font_surface = font.render(self.text, True, white)
    font_rect = font_surface.get_rect(center=self.rect.center)
    screen.blit(font_surface, font_rect)
def update(self,new_color,new_text):
    self.color = new_color
    self.text = new_text
class Vehicle(pygame.sprite.Sprite):
    def __init__(self, lane, vehicleClass, direction_number, direction, will_turn):
        pygame.sprite.Sprite.__init__(self)
        self.lane = lane
        self.vehicleClass = vehicleClass
        self.speed = speeds[vehicleClass]
        self.direction_number = direction_number
        self.direction = direction
        self.x = x[direction][lane]
        self.y = y[direction][lane]
        self.crossed = 0
        self.willTurn = will_turn
        self.turned = 0
        self.rotateAngle = 0
        vehicles[direction][lane].append(self)
        self.index = len(vehicles[direction][lane]) - 1
        self.crossedIndex = 0
        path = "images/" + direction + "/" + vehicleClass + ".png"
        self.originalImage = pygame.image.load(path)
        self.image = pygame.image.load(path)

    if(len(vehicles[direction][lane])>1 and vehicles[direction][lane][self.index-1].crossed==0):
        if(direction=='right'):
            self.stop = vehicles[direction][lane][self.index-1].stop
            - vehicles[direction][lane][self.index-1].image.get_rect().width

```

```

        - stoppingGap
elif(direction=='left'):
    self.stop = vehicles[direction][lane][self.index-1].stop
    + vehicles[direction][lane][self.index-1].image.get_rect().width
    + stoppingGap
elif(direction=='down'):
    self.stop = vehicles[direction][lane][self.index-1].stop
    - vehicles[direction][lane][self.index-1].image.get_rect().height
    - stoppingGap
elif(direction=='up'):
    self.stop = vehicles[direction][lane][self.index-1].stop
    + vehicles[direction][lane][self.index-1].image.get_rect().height
    + stoppingGap
else:
    self.stop = defaultStop[direction]

# Set new starting and stopping coordinate
if(direction=='right'):
    temp = self.image.get_rect().width + stoppingGap
    x[direction][lane] -= temp
elif(direction=='left'):
    temp = self.image.get_rect().width + stoppingGap
    x[direction][lane] += temp
elif(direction=='down'):
    temp = self.image.get_rect().height + stoppingGap
    y[direction][lane] -= temp
elif(direction=='up'):
    temp = self.image.get_rect().height + stoppingGap
    y[direction][lane] += temp
# vehicle_count[direction_number]+=1
simulation.add(self)
def render(self, screen):
    screen.blit(self.image, (self.x, self.y))
def move(self):

```

```

if(self.direction=='right'):
    if(self.crossed==0 and self.x+self.image.get_rect().width>stopLines[self.direction]):
        self.crossed = 1
        vehicles[self.direction]['crossed'] += 1
        # vehicle_count[0]-=1
        if(self.willTurn==0):
            vehiclesNotTurned[self.direction][self.lane].append(self)
            self.crossedIndex = len(vehiclesNotTurned[self.direction][self.lane]) - 1
        if(self.willTurn==1):
            if(self.lane == 1):
                if(self.crossed==0 or self.x+self.image.get_rect().width<stopLines[self.direction]+40):
                    if((self.x+self.image.get_rect().width<=self.stop or (currentGreen==0 and
currentYellow==0) or self.crossed==1) and (self.index==0 or
self.x+self.image.get_rect().width<(vehicles[self.direction][self.lane][self.index-1].x - movingGap) or
vehicles[self.direction][self.lane][self.index-1].turned==1)):
                        self.x += self.speed
                else:
                    if(self.turned==0):
                        self.rotateAngle += rotationAngle
                        self.image = pygame.transform.rotate(self.originalImage, self.rotateAngle)
                        self.x += 2.4
                        self.y -= 2.8
                        if(self.rotateAngle==90):
                            self.turned = 1
                            vehiclesTurned[self.direction][self.lane].append(self)
                            self.crossedIndex = len(vehiclesTurned[self.direction][self.lane]) - 1
                        else:
                            if(self.crossedIndex==0 or
(self.y>(vehiclesTurned[self.direction][self.lane][self.crossedIndex-1].y +
vehiclesTurned[self.direction][self.lane][self.crossedIndex-1].image.get_rect().height + movingGap))):
                                self.y -= self.speed
                    elif(self.lane == 2):
                        if(self.crossed==0 or self.x+self.image.get_rect().width<mid[self.direction]['x']):
                            if((self.x+self.image.get_rect().width<=self.stop or (currentGreen==0 and

```



```

currentYellow==0) or self.crossed==1) and (self.index==0 or
self.x+self.image.get_rect().width<(vehicles[self.direction][self.lane][self.index-1].x - movingGap) or
vehicles[self.direction][self.lane][self.index-1].turned==1)):
    self.x += self.speed
else:
    if(self.turned==0):
        self.rotateAngle += rotationAngle
        self.image = pygame.transform.rotate(self.originalImage, -self.rotateAngle)
        self.x += 2
        self.y += 1.8
        if(self.rotateAngle==90):
            self.turned = 1
            vehiclesTurned[self.direction][self.lane].append(self)
            self.crossedIndex = len(vehiclesTurned[self.direction][self.lane]) - 1
        else:
            if(self.crossedIndex==0 or
((self.y+self.image.get_rect().height)<(vehiclesTurned[self.direction][self.lane][self.crossedIndex-1].y -
movingGap)))):
                self.y += self.speed
            else:
                if(self.crossed == 0):
                    if((self.x+self.image.get_rect().width<=self.stop or (currentGreen==0 and
currentYellow==0)) and (self.index==0 or
self.x+self.image.get_rect().width<(vehicles[self.direction][self.lane][self.index-1].x - movingGap))):
                        self.x += self.speed
                    else:
                        if((self.crossedIndex==0) or
(self.x+self.image.get_rect().width<(vehiclesNotTurned[self.direction][self.lane][self.crossedIndex-1].x -
movingGap)))):
                            self.x += self.speed
                elif(self.direction=='down'):
                    if(self.crossed==0 and self.y+self.image.get_rect().height>stopLines[self.direction]):
                        self.crossed = 1
                        vehicles[self.direction]['crossed'] += 1

```

```

# vehicle_count[1]-=1
if(self.willTurn==0):
    vehiclesNotTurned[self.direction][self.lane].append(self)
    self.crossedIndex = len(vehiclesNotTurned[self.direction][self.lane]) - 1
if(self.willTurn==1):
    if(self.lane == 1):
        if(self.crossed==0 or self.y+self.image.get_rect().height<stopLines[self.direction]+50):
            if((self.y+self.image.get_rect().height<=self.stop or (currentGreen==1 and
currentYellow==0) or self.crossed==1) and (self.index==0 or
self.y+self.image.get_rect().height<(vehicles[self.direction][self.lane][self.index-1].y - movingGap) or
vehicles[self.direction][self.lane][self.index-1].turned==1)):
                self.y += self.speed
        else:
            if(self.turned==0):
                self.rotateAngle += rotationAngle
                self.image = pygame.transform.rotate(self.originalImage, self.rotateAngle)
                self.x += 1.2
                self.y += 1.8
                if(self.rotateAngle==90):
                    self.turned = 1
                    vehiclesTurned[self.direction][self.lane].append(self)
                    self.crossedIndex = len(vehiclesTurned[self.direction][self.lane]) - 1
            else:
                if(self.crossedIndex==0 or ((self.x + self.image.get_rect().width) <
(vehiclesTurned[self.direction][self.lane][self.crossedIndex-1].x - movingGap))):
                    self.x += self.speed
    elif(self.lane == 2):
        if(self.crossed==0 or self.y+self.image.get_rect().height<mid[self.direction]['y']):
            if((self.y+self.image.get_rect().height<=self.stop or (currentGreen==1 and
currentYellow==0) or self.crossed==1) and (self.index==0 or
self.y+self.image.get_rect().height<(vehicles[self.direction][self.lane][self.index-1].y - movingGap) or
vehicles[self.direction][self.lane][self.index-1].turned==1)):
                self.y += self.speed
        else:

```

```

if(self.lane == 1):
    if(self.crossed==0 or self.x>stopLines[self.direction]-70):
        if((self.x>=self.stop or (currentGreen==2 and currentYellow==0) or self.crossed==1) and
(self.index==0 or self.x>(vehicles[self.direction][self.lane][self.index-1].x +
vehicles[self.direction][self.lane][self.index-1].image.get_rect().width + movingGap) or
vehicles[self.direction][self.lane][self.index-1].turned==1)):
            self.x -= self.speed
    else:
        if(self.turned==0):
            self.rotateAngle += rotationAngle
            self.image = pygame.transform.rotate(self.originalImage, self.rotateAngle)
            self.x -= 1
            self.y += 1.2
            if(self.rotateAngle==90):
                self.turned = 1
                vehiclesTurned[self.direction][self.lane].append(self)
                self.crossedIndex = len(vehiclesTurned[self.direction][self.lane]) - 1
            else:
                if(self.crossedIndex==0 or ((self.y + self.image.get_rect().height)
<(vehiclesTurned[self.direction][self.lane][self.crossedIndex-1].y - movingGap))):
                    self.y += self.speed
        elif(self.lane == 2):
            if(self.crossed==0 or self.x>mid[self.direction]['x']):
                if((self.x>=self.stop or (currentGreen==2 and currentYellow==0) or self.crossed==1) and
(self.index==0 or self.x>(vehicles[self.direction][self.lane][self.index-1].x +
vehicles[self.direction][self.lane][self.index-1].image.get_rect().width + movingGap) or
vehicles[self.direction][self.lane][self.index-1].turned==1)):
                    self.x -= self.speed
            else:
                if(self.turned==0):
                    self.rotateAngle += rotationAngle
                    self.image = pygame.transform.rotate(self.originalImage, -self.rotateAngle)
                    self.x -= 1.8
                    self.y -= 2.5

```

```

        if(self.rotateAngle==90):
            self.turned = 1
            vehiclesTurned[self.direction][self.lane].append(self)
            self.crossedIndex = len(vehiclesTurned[self.direction][self.lane]) - 1
        else:
            if(self.crossedIndex==0 or
(self.y>(vehiclesTurned[self.direction][self.lane][self.crossedIndex-1].y +
vehiclesTurned[self.direction][self.lane][self.crossedIndex-1].image.get_rect().height + movingGap)))):
                self.y -= self.speed
            else:
                if(self.crossed == 0):
# Initialization of signals with default values
def initialize():
    global image_data,density,pause

    minTime = randomGreenSignalTimerRange[0]
    maxTime = randomGreenSignalTimerRange[1]

    image_data[0],density[0]=calc_density(1,model)
    obs=(random.uniform(0,1),density[0],random.uniform(0,1))
    # print(obs)

    green_time= int(rl_model.predict(obs)[0][0])
    # print("nextGreen",green_time)
    if(randomGreenSignalTimer):
        ts1 = TrafficSignal(0, defaultYellow, green_time)
        signals.append(ts1)
        ts2 = TrafficSignal(ts1.red+ts1.yellow+ts1.green, defaultYellow,
random.randint(minTime,maxTime))
        signals.append(ts2)
        ts3 = TrafficSignal(defaultRed, defaultYellow, random.randint(minTime,maxTime))
        signals.append(ts3)
        ts4 = TrafficSignal(defaultRed, defaultYellow, random.randint(minTime,maxTime))
        signals.append(ts4)

```

```

    paused=False
else:
    ts1 = TrafficSignal(0, defaultYellow, green_time)
    signals.append(ts1)
    ts2 = TrafficSignal(ts1.yellow+ts1.green, defaultYellow, defaultGreen[1])
    signals.append(ts2)
    ts3 = TrafficSignal(defaultRed, defaultYellow, defaultGreen[2])
    signals.append(ts3)
    ts4 = TrafficSignal(defaultRed, defaultYellow, defaultGreen[3])
    signals.append(ts4)
    paused=False
repeat()

# Print the signal timers on cmd
def printStatus():
    for i in range(0, 4):
        if(signals[i] != None):
            if(i==currentGreen):
                if(currentYellow==0):
                    print(" GREEN TS",i+1,"-> r:",signals[i].red," y:",signals[i].yellow," g:",signals[i].green)
                else:
                    print("YELLOW TS",i+1,"-> r:",signals[i].red," y:",signals[i].yellow," g:",signals[i].green)
            else:
                print(" RED TS",i+1,"-> r:",signals[i].red," y:",signals[i].yellow," g:",signals[i].green)
    print()

def repeat():
    global currentGreen, currentYellow, nextGreen, rounds, image_data
    while(signals[currentGreen].green>0): # while the timer of current green signal is not zero
        # printStatus()
        updateValues()
        time.sleep(1)
    currentYellow = 1

```

### A.3 Screenshots

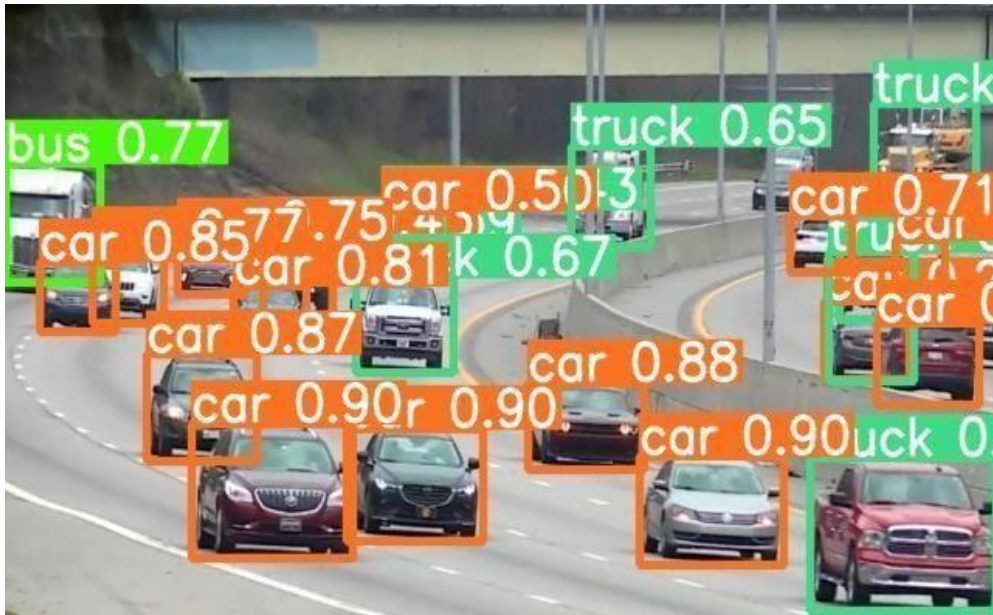


Fig 8.1 Medium traffic density

Figure 8.1 depicts a scenario of medium traffic density, where vehicles are detected and the total traffic density is classified as medium. . The density of vehicles is neither too sparse nor too congested, suggesting a balanced distribution of traffic on the road.

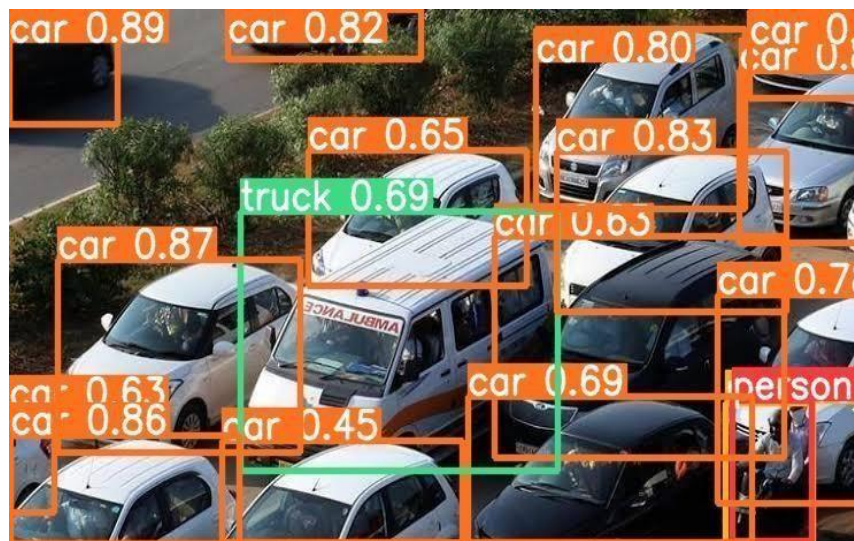


Fig 8.2 Low traffic density

Figure 8.2 depicts a scenario of low traffic density, where vehicles are detected, and the total traffic density is classified as low. In the image, there are fewer vehicles observed on the road compared to Figure 8.1.





# ADAPTIVE TRAFFIC CONTROL SYSTEM USING NEXT-GEN AI

*by* BHARATHWAJ M

---

**Submission date:** 21-Mar-2024 03:05PM (UTC+0530)

**Submission ID:** 2325672236


**File name:** ADAPTIVE\_TRAFFIC\_CONTROL\_SYSTEM\_Revised.pdf (702.64K)

**Word count:** 4530

**Character count:** 26366



# ADAPTIVE TRAFFIC CONTROL SYSTEM USING NEXT-GEN AI

 Dr. N PUGHAZENDI  
*Computer Science Engineering*  
*Panimalar Engineering College*  
*Chennai, India*  
[pughazendi@gmail.com](mailto:pughazendi@gmail.com)

BHARATHWAJ.M  
*Computer Science Engineering*  
*Panimalar Engineering College*  
*Chennai, India*  
[bharath190802@gmail.com](mailto:bharath190802@gmail.com)

ARUNKUMAR.J  
*Computer Science Engineering*  
*Panimalar Engineering College*  
*Chennai, India*  
[arunkj203@gmail.com](mailto:arunkj203@gmail.com)

**Abstract**— In modern urban areas, traffic congestion is a major problem. To effectively decrease gridlock, innovative technical solutions are required for autonomous traffic signal monitoring and regulation. This paper presents a method that utilizes surveillance cameras and employs Machine Learning (ML) for vehicle detection and a Reinforcement Learning (RL) model for traffic management. The suggested method entails using a significant set of data that is derived from a surveillance camera which produce a huge number of training images to train a machine learning model. The process begins with the identification and grouping of vehicles into several categories. The next steps include calculating traffic density and estimating traffic light signals. Using the YOLOv5 model, the accuracy achieved for vehicle recognition is 88%. This research contributes to future advancements in two key areas. Firstly, it lays the groundwork for improving road design and monitoring. Secondly, it offers a potential avenue for controlling fuel consumption and optimizing standby times. The ultimate goal is to synchronize a significant portion of traffic signals through a flexible traffic management system, leading to a substantial reduction in traffic congestion.

**Keywords**—Machine learning, Deep learning, synchronization, YOLOv5, CNN

## I. INTRODUCTION

Traffic congestion remains a significant issue in urban road systems, characterized by high traffic volumes, prolonged travel times, and inefficient signaling systems. The problem intensifies during emergencies, affecting crucial services such as ambulances, fire trucks, and police vehicles. In this context, we propose a solution that leverages surveillance cameras, employing ML and DL to regulate and optimize traffic flow.

Urban environments in India, facing high traffic congestion, witness adverse effects on various aspects of the country's functioning. Traffic snarls result from various factors, including inadequate road space, uncontrolled vehicle behaviors, and ineffective traffic management systems. Prolonged red signals, lack of awareness about roadblocks, and re-routing issues also contribute to road congestion. In this paper, we address the pressing need for innovative ideas to combat these challenges.

Surveillance cameras have become ubiquitous tools for traffic monitoring. The inefficiency of signal lights and limited analysis of prevailing situations contribute to lengthy red signals, impeding traffic flow. Our approach involves identifying vehicle types based on mass and measurements, allowing the calculation of

traffic intensity. Larger vehicles, such as lorries and buses, occupy more road space, impacting traffic density. Various mechanisms, including sensors and video cameras, are employed to collect data on passing vehicles and classify them based on size and shape.

## II. Literature Review:

In recent years, engineers have proposed several innovative solutions to address real-time problems related to traffic regulation.

Javaid et al. [1] introduced a system incorporating mobile and web applications for officials to monitor traffic using Radio Frequency Identification (RFID). RFID, combined with camera and sensor data, enables real-time tracking of traffic density, facilitating quick responses to emergencies.

Lanke and Koul [2] proposed a system heavily dependent on RFID hardware for real-time traffic regulation. Their use of inductive loop detection and video analysis, employing cameras and infrared sensors, demonstrates a comprehensive approach to traffic management.

Jacob et al. [3] introduced a framework ordering traffic into high, medium, and low classes utilizing RASPBERRY man-made intelligence with ultrasonic sensors and cameras. This multi-layered order framework offers a nuanced comprehension of traffic designs.

Shubo et al. [4] implemented a computer vision-based solution using YOLOv4 and OpenCV to detect pedestrians violating traffic regulations. Their deep neural network (DNN) algorithm identifies offenses such as riding without a helmet, driving in the wrong direction, reckless driving, traffic light violations, and unauthorized parking.

Yang et al. [5] devised a Modified Proximal Policy Optimization (Modified PPO) algorithm tailored for traffic management. Leveraging deep reinforcement learning (DRL), this system employs software-defined Internet of Things (SD-IoT) to dynamically regulate traffic lights and coordinate vehicles on a global scale, thereby improving urban traffic control efficiency.

Jin et al. [6] introduced novel adaptive signal control system has been implemented, mimicking the decision-making processes of expert signal control engineers. This system operates within a human-in-the-loop parallel learning framework, leveraging machine learning, data analysis, and cloud computing to process real-time traffic data.

Rao et al. [7] developed groundbreaking paradigm for enhancing mobility via roadside communication devices rooted in the Internet of Things (IoT) has been put forth. This system harnesses machine learning algorithms, such as neural networks and decision trees, to analyze real-time traffic data, showcasing the transformative capabilities of technology in urban environments.

Bali et al. [8] introduced Internet of Things (IoT) enabled technology has been implemented to establish "Green Corridors" dedicated to emergency vehicles. This innovative system effectively regulates traffic flow, facilitating the swift arrival of emergency vehicles at their destinations. It operates through RFID readers that detect radio frequency identification tags, ensuring seamless passage for emergency responders.

Fan Qi et al. [9] developed a deep learning model using Deep Q-Learning Network (DQN) algorithm to implement a dynamic signal timing based on the traffic intensity. The study focused on dynamically adapting to real-time intersection conditions by extracting scenario models and optimizing signal timing based on various traffic parameters. The proposed method was validated on a virtual simulation platform based on a typical intersection in Xi'an, and the results showed that the traffic signal control method based on the enhanced DQN algorithm outperformed traditional methods by reducing vehicle waiting times by 26.7% and queue lengths, thus improving intersection efficiency.

Li Wang et al. [10] proposes a simulation system that addresses the evolving needs of urban traffic control simulation technology by introducing a scene-driven approach. The system integrates real-time traffic control with simulation, providing a platform for constructing and simulating various traffic scenarios, optimizing control strategies, and implementing effective traffic management solutions. By separating data and control strategies, researchers can focus on algorithm development without the need to handle detection data and basic information manually. The system's automatic calibration of simulation parameters based on detection and GIS data ensures simulation accuracy. However, the researchers also noted some challenges, including the need for universal protocol support for signal controllers in China and detailed parameter calibration work in existing traffic simulation software.

These studies collectively underscore the ongoing efforts to integrate technology and data analytics for effective traffic management. Our proposed approach builds on these foundations, employing YOLOv5 for

vehicle segregation and DL for precise traffic density recognition.

### III. Proposed Framework:

With the use of real-time data on flow rate and traffic density, this system is intended to efficiently manage traffic congestion. Improving traffic flow and reducing congestion are its main objectives, especially on busy roadways. The way the technology works is that it determines how long traffic lights should be green by looking at the amount of traffic in each lane. Algorithms for object detection and image processing, such as YOLOv5, make this assessment easier by detecting and counting cars from real traffic camera feeds. The technology can also decide whether to close lanes or possibly reroute traffic based on these density estimations. In the end, the project seeks to improve motorist efficiency and safety by offering optimized traffic laws.

The artificial intelligence system's workflow is shown in Figure 3.1. The system uses traffic camera photos as input and processes them to calculate the number of vehicles in a given lane. Subsequent traffic management decisions are subsequently informed by the information used to determine the present traffic density.

#### A. Data preprocessing

In the initial phase of our framework, we gather a dataset from the Dhaka-AI dataset on Kaggle, comprising various vehicle types and traffic scenarios. This dataset, categorized into 21 classes, serves as the foundation for training our vehicle identification model. The inclusion of diverse classes enhances accuracy and ensures the desired outcome. We employ deep learning to address issues like traffic congestion.

Our ATSDR system prioritizes data processing techniques, following a structured approach inspired by the referenced paper. Our method aims to advance autonomous vehicle technology, enabling precise and reliable sign detection for enhanced road safety [10].

We standardize image types to 'jpeg', 'jpg', 'BMP', and 'png', eliminating additional formats by using the cv2 library. Preprocessing includes operations including noise reduction, normalization, and scaling. The positions of every vehicle in pictures or video frames are marked in the dataset through hand

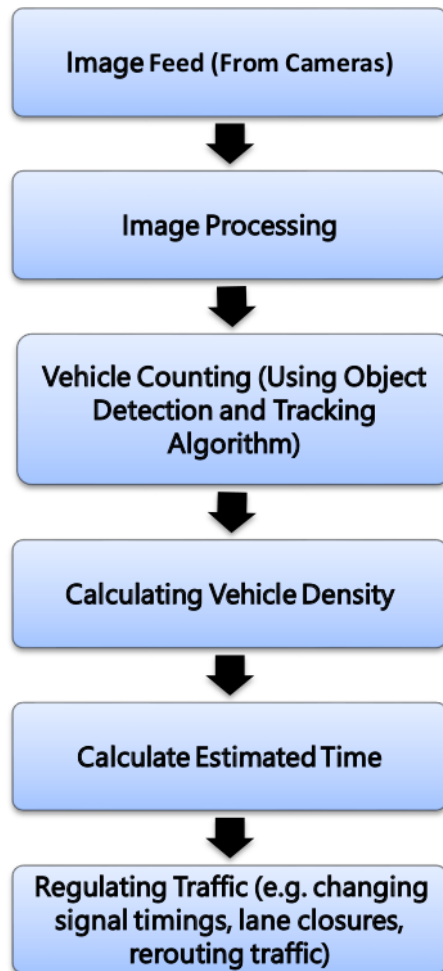


FIG 3.1 Flowchart

labeling. The dataset is divided into training (70%) and validation (30%) sets after labeling.

This method improves accuracy and training precision. Next, a random partitioning process divides the labeled dataset into three subsets: training (20%), validation (20%), and testing (20%). A representative sampling of vehicle types and traffic conditions is ensured by this distribution. Random web photos that were not viewed by the model during training are included for independent model testing.

The model is kept from picking up biases based on data order by rearranging images within subsets. The

testing set examines the model's practicality, the validation set analyses performance during training, and the training set optimizes the model's parameters. Robust model training and evaluation are ensured by this exhaustive procedure.

## B. Model Design and Validation

Using processed data, a YOLOv5 architecture model is trained to recognise and count distinct types of vehicles. For improved accuracy, the dataset has been split into 16 batches, each having an equal number of images. These batches are then fed into the model iteratively through multiple epochs. The model gets optimised with parameters, including callbacks, validation data, epochs, and train data, during training. To assess how well it is performing, it computes measures such as steps per second, loss, validation loss, and accuracy loss. At the end of training, the maximum accuracy is predicted. Artificial intelligence was used to label the dataset, which was first saved in XML format. Each text file had threshold marks for labeling.

It's critical to evaluate the model's performance on distinct validation and testing datasets after training. This is accomplished by using the validation dataset that was extracted during preprocessing. The "detect" feature of YOLOv5 helps with validation. Evaluation is then conducted using measures like accuracy, recall, and F1 score to measure precision and accuracy.

Subsequently, the model undergoes testing on a completely new dataset to verify its generalization capability. Depending on validation results, adjustments or fine-tuning may be necessary to enhance performance. Furthermore, it's essential to do a thorough analysis of the YOLOv5 model's performance following training and assessment. This entails testing its capacity to precisely identify and count different kinds of cars in a range of environmental factors, including changing traffic density, weather, and lighting. Moreover, carrying out a comprehensive error analysis might offer insightful information about potential weak points or biased areas in the model.

## C. Density Calculation:

The Density Calculation module in our proposal is crucial in providing a reliable means of assessing and quantifying the traffic volume on individual road segments. The module aims to develop an innovative and coherent strategy that takes into account several essential factors such as the number of vehicles, road capacity, driving geometry, averaging speed of vehicles, congestion, and bottlenecks.



The first step of the module is to gather the necessary data, which includes average vehicle speed, road capacity, the number of vehicles on the road segment, and variables associated with congestion. A significant improvement we have made in our module is the inclusion of the dynamic congestion factor. This factor is computed using the number of cars and their geographic distribution along the route. It shows the real-time effects of congestion on traffic density.

S.NO	CLASSES	Average Vehicle Speed	Congestion factor
1	Ambulance	50 km/h	37.5 m <sup>3</sup>
2	Army Vehicle	40 km/h	24.2 m <sup>3</sup>
3	Auto Rickshaw	30 km/h	6.75 m <sup>3</sup>
4	Bicycle	15 km/h	1.188 m <sup>3</sup>
5	Bus	40 km/h	105 m <sup>3</sup>
6	Car	50 km/h	12.15 m <sup>3</sup>
7	Garbage van	30 km/h	27.5 m <sup>3</sup>
8	Human Hauler	25 km/h	10.8 m <sup>3</sup>
9	Minibus	40 km/h	25 m <sup>3</sup>
10	Minivan	45 km/h	17.28 m <sup>3</sup>
11	Motorbike	40 km/h	1.92 m <sup>3</sup>
12	Pickup	45 km/h	18 m <sup>3</sup>
13	police car	50 km/h	12.96 m <sup>3</sup>
14	Rickshaw	20 km/h	5.4 m <sup>3</sup>
15	Scooter	35 km/h	1.512 m <sup>3</sup>
16	SUV	55 km/h	15.552 m <sup>3</sup>
17	Taxi	45 km/h	12.15 m <sup>3</sup>
18	Three Wheelers (Cng)	30 km/h	6.75 m <sup>3</sup>
19	Truck	40 km/h	45 m <sup>3</sup>
20	Van	45 km/h	25 m <sup>3</sup>

The module calculates traffic density using a refined formula by considering the sum of each vehicle's congestion-adjusted contribution and also the factors suggested by [15]. The formula integrates the number of vehicles, congestion factor, average vehicle speed, and road capacity.

$$\text{Traffic Density (D)} = \frac{\sum (\text{Number of Vehicles} * \text{Congestion factor} / \text{Average Vehicle Speed})}{\text{Road Capacity}}$$

Intersection traffic signal control methods include fixed-time, actuated, and adaptive strategies, with fixed-time relying on historical data and actuated adjusting based on sensor feedback. Adaptive strategy

predicts real-time conditions for cycle adjustments. Microscopic simulation, using software like SUMO, AIMSUN, and VISSIM, enables dynamic modeling and interaction through APIs, commonly used for intersection research.[11]. The module generates a numerical result that depicts the calculated traffic density. This result provides a valuable indicator of how congested the road stretch is by accounting for both the volume and dispersion of vehicles.

#### D. Reinforcement Learning-Based Traffic Light Estimation:

The Reinforcement Learning-Based Traffic Light Estimation module aims to optimize traffic signal control dynamically by leveraging the power of reinforcement learning. The module takes into account several influencing elements such as traffic density, the accessibility of public transportation, weather, time of day, land use, and special events to improve the effectiveness of traffic signal management.

Together, the input parameters create a state space, which offers a thorough depiction of the traffic situation as it exists right now.

traffic light detection and recognition using YOLOv5 and YOLOv5+DeepSort models, aiming to provide vital information to drivers and reduce intersection accidents, particularly in regions with common illegal crossing behaviors [12].

The traffic light controller can learn optimal policies through interactions with the environment thanks to the module's adoption of a reinforcement learning framework. Important elements of the framework include:

- State Space: Describes the existing circumstances in light of the input parameters.
- Action Space: Defines the range of possible actions the RL agent can perform, each of which represents a distinct traffic light configuration.
- Reward System: Creates a system of rewards to encourage traffic signal management measures that result in less traffic, better traffic flow, and faster travel times.
- Transition Model: Explains how, in response to the agent's activities, the system changes over time.

The module trains the model using Proximal Policy Optimisation (PPO). The selected algorithm learns from the results of its activities over time and adjusts its approach accordingly. The main objective of the algorithm is to find a policy that maximizes the expected cumulative reward in a given environment, while also minimizing the average waiting time. The model is designed in such a way that it can handle the

randomness of the traffic and sense the problem of gridlock[16].To achieve this, PPO combines two key components: the policy's expected return and a penalty term that discourages large policy updates. PPO often incorporates a value function to estimate the expected return for each state, which is then updated to reduce the bias in the advantages and improve the accuracy of the policy updates. Either simulated scenarios or historical traffic data are used to create training data. This data consists of observations of the state space, selected actions, awarded actions, and following states.

To ensure the trained model is flexible and efficient in a variety of traffic scenarios, it undergoes a thorough testing process using either unseen data or a simulated environment[10].

After a successful validation process, the module is included in actual traffic management systems to offer an adaptable and dynamic traffic light control system. The module continuously learns from and adjusts to shifting traffic patterns[18] and outside influences to maintain its responsiveness to changing circumstances.

#### 1) Emergency vehicles on the job :

By tracking the GPS signals that emergency vehicles broadcast, our system is able to detect them. Upon activation of the GPS, the system commences real-time tracking of the vehicle's location. The technique frees up traffic in the emergency vehicle's path so it can travel freely and without interruption to its destination. Although this could occasionally cause difficulties for our system, which normally tracks every vehicle on the road, we have implemented methods to handle these interruptions and ensure the unimpeded passage of emergency vehicles.

#### 2) Dignitary Convoy's Journey:

During visits by dignitaries to our state, traffic adjustments are necessary to make accommodations for their movements. Our system plays a vital role in managing traffic during these VIP visits. Upon the commencement of the dignitary convoy's journey, traffic lights along their designated route are modified to provide enhanced protection for the dignitaries. This proactive measure enables the public to anticipate road closures and ensures smooth traffic flow during VIP visits, prioritizing the safety of citizens.

#### 3) Motorcyclists riders without helmet:

Our system employs a four-step process to detect motorbike riders without helmets. First, all motorbikes and their riders are detected and classified in the video stream. Next, the area containing the motorbikes is isolated into individual images, which are fed into a

YOLOv5-tiny model in step three. This model detects the presence of helmets on the riders in each image. If a helmet is not detected, the rider is marked in the video frame as a motorbike rider without a helmet. While this approach increases the time complexity of our system, we have found that using the YOLOv5-tiny model for helmet detection strikes a balance between speed and accuracy.

The first stage of our system is critical for accurately detecting traffic offenses, as it affects the precision with which vehicle positions and classes are logged and analyzed. To this end, we have trained and tested two models for vehicle detection and classification: a YOLOv5-tiny model, which achieved a mean average precision of 72.02% and a frame rate of 24-30 FPS on our setup; and a YOLOv5 model, which processed the video feed at 15-20 FPS.

#### 4) Signal violation:

Several research works have extended the YOLOv3 framework's applicability across domains, focusing on feature extraction for traffic violation detection. YOLOv3 utilizes a feature extractor called Darknet-53, with 53 layers incorporating skip connections to prevent gradient diminishment. The detector module processes feature vectors, employing a series of convolutional layers to generate final outputs, including bounding box coordinates and objectiveness scores. The system makes assumptions about visibility, video source stability, and RGB frame structure, utilizing the MSCOCO dataset for training and validation, known for its large-scale object detection and segmentation capabilities [13].

Our system incorporates a traffic light equipped with scanning capabilities at intersections to detect individuals who intentionally or inadvertently disregard traffic regulations while the signal is red. Our system uses a visual marker placed at the intersection as a reference point. Should a vehicle pass this marker during a red signal, it is flagged as violating the traffic light. Once a vehicle commits such a violation, it is logged and remains marked, regardless of subsequent signal changes.

## IV. Results And Discussion

This approach significantly reduces the volume of traffic that can be absorbed by an ordinary method of road development. The primary target of fostering this framework is to limit the holding up time out and about by advancing traffic stream when there is no traffic on the contrary side. In addition, responsibility is reduced by the framework.

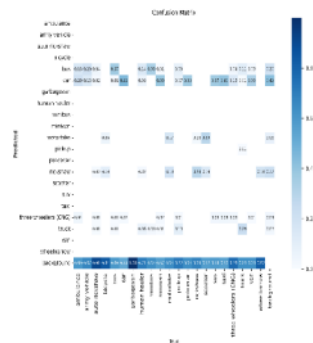


FIG 4.1 CONFUSION MATRIX

Moreover, the framework addresses various traffic violations such as unauthorized parking, motorcyclists without helmets, illegal rerouting, and failure to yield to emergency vehicles using image processing techniques. The "Disarray Network" in Figure 4.1 illustrates the summary of prediction outcomes for the Traffic Automation Issue, including the total count of correct and incorrect predictions, broken down by each class.

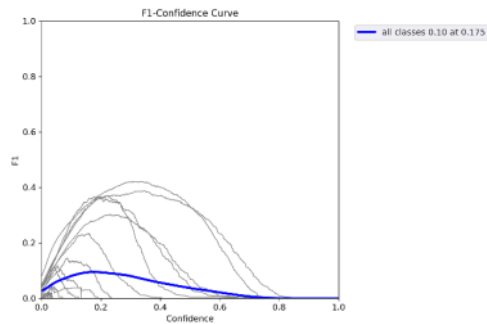


FIG 4.2 F1 Confidence Curve

Figure 4.2 is delineated based on the F1 score and confidence levels for the data. The F1 score considers both true positive and false positive predictions, signifying them with the classifier predictions and labeled training data. Our confidence curve is currently derived from 3000 images, segmented into 16 clusters, yielding a curve below the 0.50 threshold. This value is expected to rise with an increase in the number of images, segments, and clusters utilized.

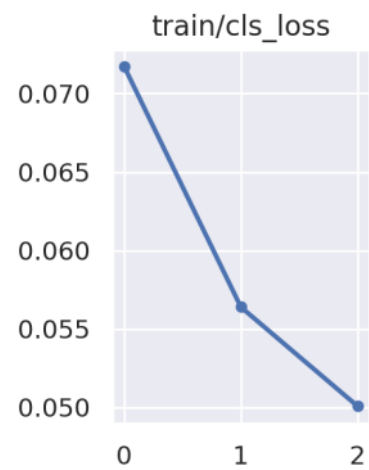


FIG 4.3 Classification Loss

In Figure 4.3, the loss value for order degradation has declined from above 0.70 to approximately 0.50 in just three epochs. This decrease in loss is attributed to the limited number of epochs. However, due to the extensive runtime of the deep learning model, the final precise loss cannot be displayed.

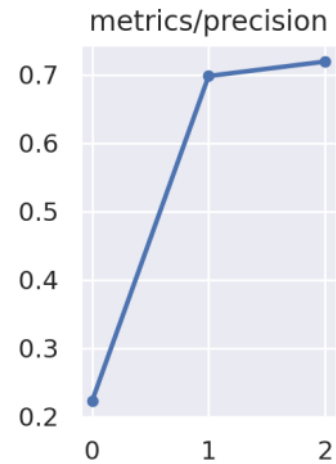


FIG 4.4 METRICS Accuracy Diagram

Figure 4.4 depicts the precision metric, which has increased from 0.2 to 0.7. This metric's precision presents the number of correct positive predictions made. Even in Figure 4.4, precision can be further increased to higher percentages when prediction

epochs are increased. It is calculated by the ratio of accurately predicted positive events divided by the total number of positive examples used for prediction.

## V. Conclusion And Future Enhancement

The proposed flexible traffic signal system, leveraging state-of-the-art PC-based intelligence, addresses the significant change of congestion in modern urban environments. By harnessing the power of artificial intelligence and deep learning models, along with real-time surveillance camera data, the system offers a unique approach to managing traffic regulation and optimization. Through vehicle detection, traffic density calculation, and reinforcement learning-based traffic light assessment, the system aims to alleviate congestion, enhance traffic flow, and reduce travel times. The results obtained from model training and evaluation demonstrate promising accuracy and performance, paving the way for future enhancements and widespread implementation. With further refinement and integration into existing traffic management systems, this innovative solution holds the potential to transform urban mobility and mitigate the adverse effects of congestion on society as a whole.

1) Proceeding in the forbidden direction: Our system can detect vehicles deviating from the designated direction on the road by analyzing their movements. All lanes in a single direction are treated by the system, marking any vehicles moving in the opposite direction as wrong-way drivers. The road's driving direction is predefined in our system, although it can be adjusted to accommodate the specific road being monitored. Our system tracks the movement of each vehicle along the x and y axes of the video frame to determine their change in position. This enables us to identify the direction in which vehicles are turning (x-axis) and moving forward (y-axis). By analyzing the changes observed along the two axes, our system can detect when a vehicle is veering off course.

2) Advising drivers on detours: When congestion occurs, rerouting can be an effective strategy to reduce delays and minimize the impact on road users. Our system is trained to analyze traffic patterns in real-time using image processing with Deep Learning and provide suggested alternative routes to drivers. When our system detects congestion on a particular road, it can automatically suggest alternative routes to drivers. This can help divert traffic from congested areas and reduce delays for all road users. In some cases, rerouting can also be used to alleviate traffic in a specific area by encouraging drivers to use less busy roads. Our system can be a valuable tool for managing

traffic in real-time, making travel more efficient and less stressful for drivers.

3) Rogue vehicle detection: Vehicles that are not authorized to be on the road at that time are identified. This is done when the image is sent to the YOLOv5 model and processed for image processing. The processed image identifies the unauthorized vehicle with the assistance of our model. The unauthorized vehicles captured by CCTV cameras are turned into frames and processed.

## REFERENCES:

- [1] Sabeen Javaid, Ali Sufian, Saima Pervaiz, and Mehak Tanveer, "Smart Traffic Management System Using Internet of Things", *International Conference on Advanced Communications Technology (ICACT)*, 2018
- [2] Ninad Lanke and Sheetal Koul, "Smart Traffic Management System", *International Journal of Computer Applications*, 2013
- [3] Sheena Mariam Jacob, Shobha Rekh, Manoj G and J John Paul, "Smart Traffic Management System with Real Time Analysis", 2018
- [4] Fahimul Hoque Shubho, Fahim Iftekhar, Ekha Hossain and Shahnewaz Siddique, "Real-time traffic monitoring and traffic offense detection using YOLOv4 and OpenCV DNN", *2021 IEEE Region 10 Conference (TENCON)*.
- [5] Jiachen Yang, Jipeng Zhang and Huihui Wang, "Urban Traffic Control in Software Defined Internet of Things via a Multi-Agent Deep Reinforcement Learning Approach", *IEEE Transactions on Intelligent Transportation Systems*
- [6] Junchen Jin, Haifeng Guo, Jia Xu, Xiao Wang and Fei-Yue Wang, "An End-to-End Recommendation System for Urban Traffic Controls and Management Parallel Learning Framework", *IEEE Transactions on Intelligent Transportation Systems*
- [7] T. V. Janardhana Rao, R. Arun Sekar, B. Drakshyani, P Kalyan Chakravarthi, N. Kumareshan and S. P. Karthi, "IOT based Smart solution for Traffic congestion at Metro Cities", *2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems*.
- [8] Dr. Vikram Bali, Ms. Sonali Mathur, Dr. Vishnu Sharma and Dev Gaur, "Smart Traffic Management System using IoT Enabled Technology", *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*
- [9] Fan Qi, Ruolei, Longhao Yan, Junfeng Yao, Ping Wang, Xiangmo Zhao, "Traffic Signal Control with Deep Q-Learning Network (DQN) Algorithm at Isolated Intersection", *34th Chinese Control and Decision Conference (CCDC)*, 2022



- [10] Li Wang,Lili Zhang,Lingyu Zhang, Min Li, Haibo Zhang, Kailong Li, Weijie Xiu,"On-line Simulation System of Urban Road Traffic Signal Control Based on Scene Driven",2019 IEEE 8th Data Driven Control and Learning Systems Conference (DDCLS)
- [11] Amit Juyal1, Sachin Sharma and Priya Matta ,"Traffic Sign Detection using Deep Learning Techniques in Autonomous Vehicles",2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICES)
- [12] Wei Miao; Long Li; Zhiwen Wang,"A Survey on Deep Reinforcement Learning for Traffic Signal Control", 2021 33rd Chinese Control and Decision Conference (CCDC)
- [13] Meng DeRong,Teng ZhongMei,"Remote Traffic Light Detection and Recognition Based on Deep Learning", 2023 6th World Conference on Computing and Communication Technologies
- [14] Ruben J Franklin, Mohana,"Traffic Signal Violation Detection using Artificial Intelligence and Deep Learning",Proceedings of the Fifth International Conference on Communication and Electronics Systems (ICES 2020)
- [15] Tahere. Royani, Javad. Haddadnia,MohammadReza Pooshideh,"A simple method for calculating vehicle density in traffic images",6th Iranian Conference on Machine Vision and Image Processing,2010
- [16] Nan Li, Guangzhou Zhao,"Adaptive signal control for urban traffic network gridlock", UKACC 11th International Conference on Control (CONTROL),2016
- [17] Zhao-Xia Yang, Ming-Hua Zhu,"A Dynamic Prediction Model of Real-Time Link Travel Time Based on Traffic Big Data", 2019 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS)
- [18] Yang Wang, Yong Zhang, Lixun Wang, Yongli Hu, Baocai Yin,"Urban Traffic Pattern Analysis and Applications Based on Spatio-Temporal Non-Negative Matrix Factorization",IEEE Transactions on Intelligent Transportation Systems ,2022

# ADAPTIVE TRAFFIC CONTROL SYSTEM USING NEXT-GEN AI

## ORIGINALITY REPORT

23%

SIMILARITY INDEX

21%

INTERNET SOURCES

7%

PUBLICATIONS

5%

STUDENT PAPERS

## PRIMARY SOURCES

1

[ijritcc.org](http://ijritcc.org)

Internet Source

18%

2

[www.ijritcc.org](http://www.ijritcc.org)

Internet Source

1%

3

Submitted to University of Wollongong

Student Paper

1%

4

Submitted to Mepco Schlenk Engineering college

Student Paper

<1%

5

Submitted to York University

Student Paper

<1%

6

[www.proceedings.com](http://www.proceedings.com)

Internet Source

<1%

7

[pdffox.com](http://pdffox.com)

Internet Source

<1%

8

[www.ijraset.com](http://www.ijraset.com)

Internet Source

<1%

9

"Emerging Cutting-Edge Developments in Intelligent Traffic and Transportation

<1%

10

[www.ncbi.nlm.nih.gov](http://www.ncbi.nlm.nih.gov)

Internet Source

<1 %

11

[www.researchgate.net](http://www.researchgate.net)

Internet Source

<1 %

12

Jiachen Yang, Jipeng Zhang, Huihui Wang.  
"Urban Traffic Control in Software Defined  
Internet of Things via a Multi-Agent Deep  
Reinforcement Learning Approach", IEEE  
Transactions on Intelligent Transportation  
Systems, 2020

Publication

<1 %

13

Amit Juyal, Sachin Sharma, Priya Matta.  
"Multiclass Objects Localization Using Deep  
Learning Technique in Autonomous Vehicle",  
2022 6th International Conference on  
Computation System and Information  
Technology for Sustainable Solutions  
(CSITSS), 2022

Publication

<1 %

14

Roopa Ravish, Shanta Rangaswamy,  
Kausthub Char. "Intelligent Traffic Violation  
Detection", 2021 2nd Global Conference for  
Advancement in Technology (GCAT), 2021

Publication

<1 %

---

Exclude quotes      On

Exclude matches      Off

Exclude bibliography      On