

SMART TRAFFIC REGULATION USING MACHINE LEARNING

A Major Project Report Submitted in Partial Fulfillment for the Degree of

Bachelor of Technology

In

Computer Science and Engineering

Submitted to



**Dr. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY,
LUCKNOW**

Submitted by:

Akshat Mishra (2000100100018)

Anjani Kumar Pandey (2000100100029)

Abhishek Singh (2000100100006)

Ajeet Pandey (2000100100013)

UNDER THE SUPERVISION OF

Dr. Vijay Kumar Dwivedi



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNITED COLLEGE OF ENGINEERING AND RESEARCH,

PRAYAGRAJ

MAY 2024

CANDIDATE’S DECLARATION

We, hereby certify that the project entitled “Smart Traffic Regulation using Machine Learning” submitted by us in partial fulfillment of the requirement for the award of degree of the B. Tech. (Computer Science & Engineering) submitted to **Dr.A.P.J. Abdul Kalam Technical University, Lucknow** at **United College of Engineering and Research, Prayagraj** is an authentic record of our own work carried out during a period from June 2023 to May 2024 under the guidance of Dr. Vijay Kumar Dwivedi Head Of Department CSE , Department of Computer Science & Engineering. The matter presented in this project has not formed the basis for the award of any other degree, diploma, fellowship or any other similar titles.

Signature of the Student

Akshat Mishra (2000100100018)

Signature of the Student

Anjani Kumar Pandey (2000100100029)

Signature of the Student

Abhishek Singh (2000100100006)

Signature of the Student

Ajeet Pandey (2000100100013)

Place: PRAYAGRAJ

Date:

CERTIFICATE

This is to certify that the project titled “Smart Traffic Regulation using Machine Learning” is the bona fide work carried out by Akshat Mishra (2000100100018), Anjani Kumar Pandey (2000100100029), Abhishek Singh (2000100100006), and Ajeet Pandey (2000100100013) in partial fulfillment of the requirements for the award of the degree of B.Tech (Computer Science & Engineering). This project has been submitted to Dr. A.P.J Abdul Kalam Technical University, Lucknow, at United College of Engineering and Research, Prayagraj, and is an authentic record of their own work carried out during the period from June 2023 to May 2024 under the guidance of Dr. Vijay Kumar Dwivedi, Head of the Department of CSE.

Signature of the Guide_____

[Dr. Vijay Kumar Dwivedi]

Signature of Project Coordinator_____

[Mr. Shyam Bahadur Verma]

Signature of the Head of Department_____

[Dr. Vijay Kumar Dwivedi]

Place : PRAYAGRAJ

Date:

ABSTRACT

The "Smart Traffic Regulation using Machine Learning" project aims to address the growing issue of traffic congestion in urban areas through the implementation of an intelligent traffic management system. By leveraging real-time data collection and advanced algorithms, this system dynamically adjusts traffic light patterns to optimize traffic flow, reduce wait times, and improve overall transportation efficiency.

Our project integrates cutting-edge technologies such as OpenCV2 for image processing and TensorFlow and Keras for machine learning, enabling the system to accurately analyze traffic conditions through visual data. The smart controller processes real-time video feeds to detect vehicle density at intersections, allowing for dynamic adjustment of traffic light timings.

The data collection process involves the use of sensor networks and cameras to gather comprehensive traffic information. This data is then fed into our machine learning models, which continuously learn and adapt to changing traffic patterns, prioritizing heavily congested routes and minimizing delays.

The Smart Traffic Light Controller offers significant improvements over traditional systems, including reduced fuel consumption and emissions, improved emergency response times, and better accommodation of pedestrian and cyclist traffic. Our solution has been rigorously tested through simulations, showing promising results such as a significant reduction in average wait times and smoother traffic flow.

This project represents a scalable and adaptable framework that can be customized to meet the specific needs of different urban environments. The successful implementation of our system demonstrates a significant step towards smarter, more efficient urban traffic management, contributing to sustainable urban development.

ACKNOWLEDGEMENT

We would like to express our deepest gratitude to Mr. Vijay Kumar Dwivedi, the Head of the Department of Computer Science and Engineering, for his invaluable guidance, encouragement, and support throughout the duration of our project, “Smart Traffic Regulation using Machine Learning”. His expert insights and suggestions played a crucial role in the successful completion of our project.

We also extend our sincere thanks to our seniors and other faculty members of the Computer Science and Engineering department. Their assistance and advice provided us with the necessary knowledge and tools to tackle the challenges we faced during this project.

A special thank you to our friends and batchmates for their unwavering support, collaboration, and encouragement. Their camaraderie and teamwork were instrumental in overcoming obstacles and achieving our project goals.

Furthermore, we are grateful to our families for their continuous support and motivation, which helped us stay focused and dedicated.

The completion of this project marks a significant milestone in our academic journey, and we are truly thankful to everyone who contributed to this endeavor.

Table of Content

Title	
Candidate's Declaration	i
Certificate	ii
Abstract	iii
Acknowledgement	iv
List of Figures	vii

Chapter 1

Introduction	
1.1 Problem Definition	2
1.2 Project Overview/Specification	2
1.2.1 Introduction	2
1.2.2 Objectives	2
1.2.3 Key Features	3
1.3 Hardware Specification	8
1.3.1 Sensors and Cameras	8
1.3.2 Communication Infrastructure	8
1.3.3 Control Centre Hardware	8
1.3.4 Edge Computing Devices	8
1.3.5 Emergency Vehicle Priority System	8
1.3.6 Variable Message Signs (VMS) and Traffic Displays	9
1.3.7 Power Supply and Backup System	9
1.3.8 Data Storage and Analytics Infrastructure	9
1.3.9 Environmental Sensors and Monitoring Devices	9
1.3.10 Network Security and Cybersecurity Equipment	9

Chapter 2

Literature Survey	
-------------------	--

2.1 Existing System	12
2.2 Proposed System	14
2.3 Methodology	17
2.4 Feasibility Study	20
2.4.1 Introduction	20
2.4.2 Technological Feasibility	20
2.4.3 Economical Feasibility	20
2.4.4 Operational Feasibility	21
2.4.5 Social Feasibility	21
2.5 Python in Smart Traffic light Regulator	22

Chapter 3

Software, Libraries and Frameworks

3.1 Extensive Libraries and Frameworks	23
3.1.1 OpenCV2	23
3.1.2 TensorFlow and Keras	23
3.1.3 Support for Machine Learning and Data Analysis	23
3.1.4 Pandas	24
3.1.5 NumPy	24
3.1.6 Matplotlib	24
3.2 Community and Documentation	24
3.3 Implementation in the Project	24
3.4 Data Collection	24
3.5 Image Processing	25
3.6 Machine Learning	25
3.7 Control Logic	25
3.8 Deep Learning in Smart Traffic Light Regulation Systems	25
3.8.1 Traditional Traffic Light Systems vs. Deep Learning	25
3.8.2 Deep Learning Techniques for Traffic Light Control	26
3.9 System Architecture	26

3.10 Challenges and Considerations	27
3.11 Future Directions	27

Chapter 4

Diagrams and Figures

4.1 Data flow Diagram	28
4.1.1 DFD (level-0)	28
4.1.2 DFD (level-1)	29
4.1.3 DFD (level-2)	30
4.2 Entity- relation diagram (ERD)	31
4.3 Result and Outputs	32
4.3.1 Resizing image to size (320,320)	32
4.3.2 Changing the color format to BGR to RGB	32
4.3.3 Generating the position (X, Y) co-ordinates and dimensions of the objects detected inside the image.	33
4.3.4 Name of the objects detected in the image	33
4.3.5 Detected objects in the image	34
4.3.6 Time allocation to lane	34
4.3.7 Testing the model on video input	35
4.3.8 Importing the necessary library	36
4.3.9 Installing OpenCV module and importing various modules like, matplotlib, pandas, NumPy, PIL, Keras, Sklearn, etc.	36
4.3.10 Reading the first image which have to be trained	37
4.3.11 Checking or Verifying the first image which have to be trained	37
4.3.12 Verifying the shape of first image that has to be trained	38
4.3.13 Pre-processing of Image data to fit in the model	38
4.3.14 Verifying the Image from different axis	39
4.3.15 Uploading the Training and Testing data into the variable	39
4.3.16. Classified Image	40
4.3.17 Checking the information of training dataframe	40

4.3.18 Validating the images	41
4.3.19 Constructor Stage	41
4.3.20 Compilation stage and verifying the summary of the model	42
4.3.21 Training stage and fitting the model on the algorithm	42
4.3.22 Plotting the graph of training and loss value for checking the overfitting	43
4.3.23 Testing Database to the model	43
4.3.24 Testing a random image and predicting the output as emergency or non-emergency vehicle	44
 Chapter 5	
Conclusion	
5.1 Key Achievements	45
 Chapter 6	
Future Scope	
6.1 Integration with Real-Time video data	46
6.2 Enhanced Machine Learning Scope	46
6.3 Vehicle-to-Infrastructure (V2I) Communication	46
6.4 Public Transport Signal Priority	46
6.5. Eco-driving and Emissions Reduction	47
6.6. Scalability to Larger Networks	47
6.7. Adaptive Traffic Signal Control (ATSC)	47
 Chapter 7	
References	48
Plagiarism Report	50

List of figures

Sr. No.	Figure Name	Page No.
1	Data flow diagram	28
2	Entity relation diagram	31
3	Resizing image to size (320,320)	32
4	Changing the color format from BGR to RGB	32
5	Generating the position (X,Y) co-ordinates and dimensions of the objects detected inside the image	33
6	Name of the objects detected in the image	33
7	Detected objects in the image	34
8	Time allocation to lane	34
9	Testing the model on video input	35
10	Importing the necessary library	36
11	Installing OpenCV module and various modules like, matplotlib, pandas, NumPy, PIL, Keras, sklearn, etc	36
12	Reading the first image which have to be trained	37
13	Checking or verifying the first image which have to be trained	37
14	Verifying the shape of first image that has to be trained	38
15	Pre-processing of Image data to fit in the model	38
16	Verifying the Image from different axis	39
17	Uploading the Training and Testing data into Variable	39
18	Classified image	40
19	Checking the information of training dataframe	40
20	Validating the images	41
21	Constructor Stage	41
22	Compilation stage and verifying the summary of model	42
23	Training stage and fitting the model on the algorithm	42
24	Plotting the graph of training and loss value for checking the overfitting	43

25	Testing Database to the model	43
26	Testing a random image and prediction the output as emergency or non-emergency vehicle	44

CHAPTER: 1

INTRODUCTION

Traffic congestion has become a prevalent issue in urban areas worldwide, leading to increased time in travelling, fuel consumption, and pollution of environment. Conventional traffic control systems, relying on fixed timing schedules and pre-defined patterns, often fail to adapt to dynamic traffic conditions, resulting in inefficiencies and frustration for commuters. To address these challenges, Smart Traffic Regulation (STR) have emerged as innovative solutions leveraging advanced technologies to optimize traffic flow, enhance safety, and reduce congestion.

Smart Traffic Regulation integrate various cutting-edge technology including computer vision, Internet of Things (IoT), machine learning (ML), and artificial intelligence (AI) to transform conventional traffic management strategies. These systems employ real-time data collection from sensors, cameras to monitor traffic conditions, analyse patterns, and make known decisions for traffic optimization.

ML algorithms, which analyze both historical and current traffic data to forecast traffic flow patterns and identify regions likely to experience congestion, are a crucial part of STR. By utilizing ML models, STR can dynamically adjust traffic signal timings, lane assignments, and route recommendations to optimize traffic flow and reduce delays.

Overall, Smart Traffic Regulation represent a paradigm shift in traffic management, offering intelligent, adaptive, and data-driven solutions to alleviate congestion, enhance safety, and promote sustainable urban mobility. As cities continue to grow and face increasingly complex traffic challenges, the adoption of STR promises to revolutionize the way we manage and navigate urban transportation networks.

In addition to improving traffic flow and safety, STR also contribute to environmental sustainability by reducing vehicle idling times, fuel consumption, and emissions. By optimizing traffic patterns and minimizing stop-and-go traffic, STR help mitigate the environmental impact of urban transportation systems.

1.1 PROBLEM DEFINITION

The problem with traditional traffic control systems lies in their static and reactive nature, which often leads to inefficiencies, congestion, safety hazards, and environmental concerns. These systems rely on fixed timing schedules for traffic signals and lack the ability to adapt to real-time traffic conditions, resulting in suboptimal traffic flow and increased travel times for commuters.

Furthermore, traditional traffic control systems do not effectively prioritize emergency vehicles or support multimodal transportation initiatives, leading to challenges in managing diverse transportation modes and promoting sustainable mobility.

In addition, the limited integration of data analytics and communication technologies hinders the ability to gather and utilize real-time data for making informed traffic management decisions. This lack of data-driven insights and proactive strategies contributes to ongoing issues such as traffic jams, accidents, and pollution in urban areas.

Overall, the main problem with traditional traffic control systems is their inability to keep pace with the dynamic and complex nature of modern urban transportation, leading to negative impacts on efficiency, safety, and environmental sustainability.

1.2 PROJECT OVERVIEW/ SPECIFICATION

1.2.1 Introduction:

In order to overcome the drawbacks and inefficiencies of conventional traffic management techniques, the project intends to design, develop, and implement a smart traffic control system. Utilizing cutting-edge technologies like big data analytics, IoT, and artificial intelligence, the system will enhance traffic flow, improve safety, reduce congestion, and promote sustainable transportation practices in urban areas.

1.2.2 Objectives:

1. Optimize Traffic Flow: Develop algorithms to dynamically modify signal timings in response to traffic data in real time, therefore easing congestion and enhancing traffic flow overall efficiency.

2. Enhance Safety: Implement systems for detecting and responding to potential hazards, such as speeding vehicles, red-light runners, and pedestrian crossings, to prevent accidents and improve road safety.

3. Support Multimodal Integration: Integrate data from various transportation modes (e.g., buses, trains, bicycles) to promote multimodal integration, prioritize public transit, and reduce reliance on single-occupancy vehicles.

4. Improve Environmental Sustainability: Minimize vehicle idling times, optimize fuel consumption, and reduce emissions by promoting smoother traffic flow and reducing stop-and-go patterns.

5. Facilitate Data-Driven Decision Making: Create a thorough data analytics platform to collect, handle, and evaluate real-time traffic data. This will allow traffic management strategies to be made based on data.

1.2.3 Key Features:

1. Real-Time Data Collection: Install a network of sensors and cameras to gather data in real-time on environmental factors, vehicle speed, traffic volume, and congestion levels.

2. Dynamic Signal Control: Develop algorithms to dynamically adjust traffic signal timings based on incoming data, prioritize emergency vehicles, and optimize traffic flow patterns.

3. Intelligent Routing Suggestions: Provide intelligent routing suggestions to drivers based on real-time traffic conditions, alternate routes, and multimodal transportation options.

4. Safety Monitoring Systems: Implement systems for detecting and responding to safety hazards, such as automated speed enforcement, red-light violation detection, and pedestrian safety alerts.

5. Integration with Public Transit: Integrate data from public transit systems to prioritize buses, trains, and other modes of public transportation, reducing overall congestion and promoting sustainable mobility.

6. Data Analytics Platform: Develop a data analytics platform to process and analyse real-time traffic data, generate actionable insights, and support data-driven decision-making for traffic management strategies.

Implementation Plan:

1. Requirements Analysis: Conduct a thorough analysis of stakeholder requirements, traffic patterns, and existing infrastructure to define system requirements and design specifications.

2. System Design: Develop a detailed system architecture, including hardware components (sensors, cameras, traffic signal controllers) and software modules (algorithms, data analytics platform, user interface).

3. Prototype Development: Build and test a prototype of the smart traffic control system in a controlled environment, ensuring functionality, reliability, and performance.

4. Pilot Deployment: Deploy the system in a pilot area to gather real-world data, evaluate system performance, and gather user feedback for refinement and optimization.

5. Full-Scale Deployment: Roll out the smart traffic control system across targeted urban areas, integrating feedback, addressing scalability challenges, and continuously optimizing system performance.

6. Monitoring and Maintenance: Establish monitoring and maintenance protocols to ensure ongoing system functionality, data accuracy, and performance optimization.

Machine Learning: Empowering Intelligent Systems

The technique known as machine learning (ML) has become a game changer, allowing computers to learn from data and make judgments or predictions without explicit programming. It has unleashed the potential of data-driven insights and intelligent automation, revolutionizing a range of sectors from healthcare and banking to transportation and entertainment. This essay examines the fundamentals, uses, difficulties, and potential paths of machine learning.

Principles of Machine Learning

Fundamentally, machine learning is the use of algorithms and methodologies that enable computers to gain knowledge from experience (data) and gradually become more proficient at a certain activity. Among the fundamental ideas of machine learning are:

- 1. Supervised Learning:** The algorithm learns on labelled training data in supervised learning, where each data point is linked to a specific target or output. Learning a mapping function that can correctly forecast the result for fresh, untainted data is the aim.
- 2. Unsupervised Learning:** Unsupervised learning is the process of deriving structures and patterns from unlabelled data. With no explicit help from tagged samples, the program seeks to find hidden patterns, clusters, or correlations within the data.
- 3. Reinforcement Learning:** According to the concept of reinforcement learning, an agent gains decision-making skills through interaction with its surroundings and feedback in the form of incentives or punishments. The agent seeks to discover the best possible policies or methods in order to maximize cumulative rewards.
- 4. Deep Learning:** A subset of machine learning known as "deep learning" makes use of multiple-layered deep neural networks to teach itself hierarchical data representations. In applications including speech recognition, picture identification, and natural language processing, deep learning has shown impressive results.

Applications of Machine Learning

Machine learning finds applications across diverse domains, driving innovation and efficiency in various industries:

- 1. Healthcare:** Drug development, medical image analysis, individualized treatment recommendations, illness diagnosis, and healthcare resource optimization are all done with the use of machine learning algorithms.

2. Finance: Machine learning is used in finance to manage risk, identify fraud, score credit, perform algorithmic trading, and segment customers.

3. Transportation: ML powers autonomous vehicles, traffic prediction and optimization, route planning, predictive maintenance for transportation fleets, and Smart Traffic Regulation.

4. Retail: ML algorithms enable personalized recommendations, demand forecasting, supply chain optimization, inventory management, and customer sentiment analysis.

5. Manufacturing: In the industrial sector, machine learning is utilized for process optimization, anomaly detection, supply chain optimization, quality control, and predictive maintenance.

6. Natural Language Processing (NLP): NLP techniques powered by machine learning are used for sentiment analysis, text summarization, machine translation, chatbots, and voice recognition systems.

Challenges in Machine Learning

Although machine learning has enormous promise, there are a number of issues that need to be resolved, which academics and practitioners are still working to resolve:

1. Data Quality and Quantity: For training, machine learning models strongly depend on high-quality labeled data. The performance of the model might be negatively impacted by biases, imbalances, noise, and lack of data.

2. Model Interpretability: Deep learning models in particular are sometimes regarded as "black boxes," which makes it difficult to decipher their choices and comprehend the fundamental principles guiding forecasts.

3. Overfitting and Generalization: Overfitting is the process by which a model memorizes the training set instead of applying it to new data. A key issue in machine learning is striking a balance between generalization and model complexity.

4. Ethical and Bias Concerns: ML models may unintentionally reinforce biases found in the training set, producing unfair or discriminating results. Fairness, accountability, and openness are important ethical factors to take into account when developing ML.

5. Computational Resources: Large-scale datasets and high-performance computing systems are two of the most important computational resources needed for training complicated machine learning models, particularly deep learning models.

Future Directions in Machine Learning

The future of machine learning is marked by ongoing advancements and innovations in several key areas:

1. Explainable AI: Efforts are underway to develop explainable AI techniques that enhance the interpretability and transparency of machine learning models, enabling humans to understand and trust AI-driven decisions.

2. Robustness and Security: Research focuses on developing robust ML models resilient to adversarial attacks, ensuring model reliability, security, and privacy in real-world applications.

3. Continual Learning: Continual learning aims to enable ML models to learn incrementally from new data while retaining previously acquired knowledge, facilitating lifelong learning and adaptation in dynamic environments.

4. AI Ethics and Governance: There is growing emphasis on ethical AI frameworks, governance mechanisms, and regulatory guidelines to address societal concerns, biases, and ethical implications of AI technologies.

5. Edge Computing and Federated Learning: Edge computing and federated learning paradigms enable distributed ML training and inference on decentralized devices, enhancing scalability, privacy, and efficiency in AI systems.

In conclusion, machine learning stands at the forefront of technological innovation, driving transformative changes across industries and shaping the future of intelligent systems. As we

navigate challenges and embrace advancements, the potential of machine learning to unlock new possibilities and create positive societal impact remains vast and promising.

1.3 Hardware Specification

1.3.1 Sensors and Cameras:

- Traffic Sensors: Deployed at key intersections and road segments to capture real-time data on traffic volume, vehicle speeds, and congestion levels.
- Vehicle Detection Sensors: Including inductive loops, radar sensors, or infrared sensors to detect vehicle presence and count.
- Video Cameras: High-resolution cameras for video surveillance, vehicle recognition, and traffic monitoring.

1.3.2 Communication Infrastructure:

- Fibre Optic Cables: High-speed and reliable communication backbone for data transmission between sensors, control centres, and traffic management systems.
- Wireless Communication: Cellular networks (e.g., 4G/5G), Wi-Fi, or dedicated radio frequency (RF) communication for real-time data exchange and remote-control capabilities.

1.3.3 Control Centre Hardware:

- Traffic Signal Controllers: Advanced traffic signal controllers capable of dynamic signal timing adjustments based on real-time data inputs.
- Centralized Control System: Servers, storage, and networking equipment for processing and analysing traffic data, running optimization algorithms, and managing traffic flow.

1.3.4 Edge Computing Devices:

- Edge Servers: Deployed at intersections or roadside cabinets for local data processing, reducing latency and bandwidth requirements for real-time decision-making.

- Edge AI Processors: Hardware accelerators (e.g., GPUs, TPUs) for running AI-based algorithms locally, such as object detection, vehicle classification, and predictive analytics.

1.3.5 Emergency Vehicle Priority Systems:

- Radio Communication Equipment: Two-way communication systems between emergency vehicles (e.g., fire trucks, ambulances) and traffic signals to request priority clearance.
- Traffic Signal Pre-emption Devices: Installed on emergency vehicles to send pre-emptive signals to traffic signals, enabling safe and efficient passage.

1.3.6 Variable Message Signs (VMS) and Traffic Displays:

- LED Displays: Variable message signs, electronic boards, and dynamic traffic displays for conveying real-time traffic information, alerts, and guidance to drivers.
- Digital Signage Controllers: Hardware controllers for managing content, scheduling messages, and displaying visual cues for traffic management purposes.

1.3.7 Power Supply and Backup Systems:

- Uninterruptible Power Supply (UPS): Backup power systems to ensure continuous operation during power outages or disruptions.
- Power Distribution Units (PDU): Managed power distribution units for distributing power to traffic control equipment, sensors, and communication devices.

1.3.8 Data Storage and Analytics Infrastructure:

- Storage Servers: High-capacity storage servers for storing historical traffic data, video recordings, and analytics results.
- Analytics Platforms: Hardware for running data analytics, machine learning algorithms, and predictive models to derive insights, optimize traffic flow, and generate reports.

1.3.9 Environmental Sensors and Monitoring Devices:

- Air Quality Sensors: Monitoring air pollution levels, particulate matter, and environmental conditions at traffic intersections.

- Weather Stations: Collecting weather data (e.g., temperature, humidity, precipitation) for adaptive traffic management based on weather conditions.

1.3.10 Network Security and Cybersecurity Equipment:

- Firewalls and Intrusion Detection Systems (IDS): Protecting traffic control systems from cyber threats, unauthorized access, and data breaches.

CHAPTER: 2

LITERATURE SURVEY

1. "Smart Traffic Control System Using IoT and Machine Learning" by Singh et al. (2020)

- This paper proposes a smart traffic control system architecture leveraging IoT sensors and machine learning algorithms. The study focuses on real-time data collection, traffic analysis, and adaptive signal control for optimizing traffic flow and reducing congestion.

2. "Intelligent Traffic Signal Control Systems: A Review of State-of-the-Art Techniques" by Chen et al. (2019)

- The review provides an overview of intelligent traffic signal control systems, including traditional methods and emerging techniques such as deep reinforcement learning and predictive modelling. It discusses the challenges, benefits, and future directions of intelligent traffic signal control.

3. "Machine Learning Techniques for Traffic Flow Prediction: A Review" by Li et al. (2021)

- This review explores various machine learning techniques applied to traffic flow prediction, including regression models, neural networks, and ensemble methods. The study

evaluates the performance, accuracy, and scalability of different ML approaches for traffic prediction.

4. "IoT-Based Smart Traffic Management System for Smart Cities" by Kumar et al. (2020)

- The paper presents an IoT-based smart traffic management system designed for smart cities. It discusses the integration of IoT devices, cloud computing, and data analytics for real-time traffic monitoring, congestion detection, and adaptive signal control.

5. "Deep Learning-Based Traffic Congestion Prediction and Management: A Survey" by Wang et al. (2021)

- This survey reviews deep learning-based approaches for traffic congestion prediction and management. It covers topics such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and attention mechanisms for traffic forecasting and optimization.

6. "Intelligent Transportation Systems: A Comprehensive Review" by Zhang et al. (2018)

- The comprehensive review discusses intelligent transportation systems (ITS), including smart traffic control, vehicle-to-infrastructure communication, and autonomous vehicles. It examines the role of IoT, AI, and data analytics in shaping the future of transportation systems.

7. "Smart Traffic Light Control System Using IoT and Machine Learning" by Patel et al. (2022)

- This research paper presents a smart traffic light control system combining IoT sensors and machine learning techniques. It explores the use of real-time traffic data, predictive models, and adaptive signal control to improve traffic efficiency and reduce travel times.

8. "Traffic Flow Prediction and Management in Smart Cities: A Survey" by Liu et al. (2019)

- The survey investigates traffic flow prediction and management strategies in smart cities, including data-driven approaches, predictive modelling, and traffic simulation techniques. It analyses the effectiveness of different methods in addressing urban traffic challenges.

9. "Deep Reinforcement Learning for Traffic Signal Control: A Review" by Wu et al. (2020)

- This review focuses on deep reinforcement learning (DRL) techniques for traffic signal control. It discusses DRL algorithms, simulation-based testing, and real-world applications of adaptive signal control systems using reinforcement learning.

10. "Integration of Machine Learning and IoT for Traffic Management: A Review" by Gupta et al. (2021)

- The review paper explores the integration of machine learning and IoT technologies for traffic management applications. It discusses sensor deployment, data analytics, predictive modelling, and intelligent decision-making in traffic control systems.

These literature sources collectively provide insights into the advancements, challenges, and potential solutions in the field of Smart Traffic Regulation, highlighting the integration of IoT, machine learning, data analytics, and intelligent decision-making for efficient and sustainable urban transportation management.

2.1 Existing System:

The existing system of traffic control relies heavily on conventional methods that are largely static and manual in nature. This system typically employs fixed-time traffic signal schedules,

predetermined signal cycles, and manual adjustments based on historical traffic patterns. While these methods have served as the backbone of traffic management for decades, they exhibit several limitations and inefficiencies in addressing modern-day traffic challenges.

Limitations of the Existing System:

- 1. Static Signal Timings:** Fixed-time traffic signal schedules do not adapt to real-time traffic conditions, leading to suboptimal traffic flow and increased congestion during peak hours or unforeseen events such as accidents or road closures.
- 2. Manual Monitoring and Control:** Traffic management often relies on manual monitoring by traffic personnel who manually adjust signal timings or intervene during traffic incidents. This manual approach is time-consuming, reactive, and prone to human errors.
- 3. Limited Data Utilization:** The existing system lacks comprehensive data utilization and analytics capabilities. Traffic data collection is often limited to basic sensors or cameras, with minimal integration of advanced technologies for data analysis and decision-making.
- 4. Inefficient Resource Allocation:** Without real-time data insights, resources such as traffic signals, road signs, and traffic personnel are not optimally allocated, leading to inefficiencies in traffic management and utilization of infrastructure.
- 5. Safety Concerns:** Inadequate response to emergencies, lack of prioritization for emergency vehicles, and limited traffic enforcement measures contribute to safety hazards and delays on roads.

Challenges in the Existing System:

- 1. Traffic Congestion:** Fixed-time signal schedules and manual interventions contribute to traffic congestion, particularly in urban areas with high traffic volumes and complex road networks.
- 2. Safety and Accidents:** Inefficient traffic management practices can lead to accidents, delays in emergency response, and compromised road safety for both drivers and pedestrians.

3. Environmental Impact: Traffic congestion and suboptimal traffic flow result in increased fuel consumption, emissions, and environmental pollution, contributing to sustainability challenges.

4. Limited Adaptability: The existing system lacks adaptability to changing traffic patterns, weather conditions, and special events, hindering proactive traffic management strategies.

Proposed System: Smart Traffic Regulation using Machine Learning

Overview

The proposed system aims to revolutionize traffic light regulation by leveraging cutting-edge technologies such as computer vision and machine learning. The system is designed to optimize the duration of traffic lights based on real-time traffic conditions, improving traffic flow, reducing congestion, and ensuring prompt passage for emergency vehicles. This section details the proposed system, including its architecture, technologies used, and operational workflow.

System Architecture

The architecture of the Smart Traffic Regulation using Machine Learning system is composed of several integral components:

1. **Camera System:** High-definition cameras are installed at strategic points to monitor traffic on all lanes approaching an intersection. These cameras continuously capture video feeds, which are processed in real-time.

2. **Image Processing Unit:** The video feeds are converted into image snapshots at regular intervals. This conversion is managed using OpenCV (cv2), a robust library for real-time computer vision.
3. **Vehicle Detection and Counting Module:** The snapshots are analyzed using machine learning algorithms to detect and count the number of vehicles in each lane. We employ TensorFlow and Keras, leveraging pre-trained convolutional neural networks (CNNs) for object detection. These models are fine-tuned to recognize various types of vehicles, including cars, trucks, motorcycles, and bicycles.
4. **Emergency Vehicle Detection:** The system is equipped to identify emergency vehicles, such as ambulances, fire trucks, and police cars, using specialized object detection algorithms. The presence of an emergency vehicle prompts the system to prioritize its passage by adjusting the traffic light sequence accordingly.
5. **Traffic Light Control Unit:** Based on the vehicle count and the detection of any emergency vehicles, the system dynamically adjusts the length of the green light for each lane. This decision-making process is handled by an intelligent controller that uses predefined rules and real-time data inputs.
6. **Communication Interface:** A communication module ensures seamless data transmission between the cameras, processing units, and traffic light controllers. This interface supports real-time updates and allows for remote monitoring and control.

Technologies Used

Camera and Image Processing

- **OpenCV (cv2):** This open-source computer vision library is utilized for capturing video frames and converting them into image snapshots. OpenCV provides essential functions for image manipulation, feature detection, and pre-processing, which are crucial for preparing the images for further analysis.

Machine Learning Frameworks

- **TensorFlow:** This end-to-end open-source machine learning framework is used to create and train deep learning models that identify and categorize automobiles. For

our project, its large library and support for neural network designs make it the perfect option.

- **Keras:** NN model construction and training are done using this high-level neural network API, which is written in Python and can run on top of TensorFlow. Keras facilitates quick experimentation and streamlines the process of creating sophisticated neural networks..

Operational Workflow

1. **Video Capture and Snapshot Conversion:** The system continuously captures video from the installed cameras. At predetermined intervals, frames are extracted and converted into image snapshots using OpenCV. These snapshots provide a static view of the traffic at each moment.
2. **Vehicle Detection and Counting:** The image snapshots are fed into the machine learning model built using TensorFlow and Keras. The model processes each image to detect vehicles and count their numbers in each lane. This process involves several steps, including image pre-processing, object detection, and post-processing to refine the count.
3. **Emergency Vehicle Identification:** Concurrently, the system scans for the presence of emergency vehicles. Upon detecting an emergency vehicle, the system flags the lane and adjusts the traffic light timings to expedite the vehicle's passage.
4. **Traffic Light Adjustment:** The vehicle counts, and emergency vehicle alerts are sent to the traffic light control unit. This unit uses the data to determine the optimal duration of the green light for each lane. For instance, lanes with higher vehicle counts or those containing emergency vehicles receive longer green light durations to alleviate congestion and ensure quick passage.
5. **Real-Time Updates and Monitoring:** The communication interface facilitates real-time updates, ensuring that any changes in traffic conditions are promptly addressed. Traffic management authorities can monitor the system remotely, making manual adjustments if necessary.

Conclusion

The proposed Smart Traffic Regulation using Machine Learning system integrates advanced technologies to enhance urban traffic management. By using cameras, computer vision, and machine learning, the system dynamically adjusts traffic light durations based on real-time traffic data. This approach not only improves traffic flow but also prioritizes emergency vehicles, contributing to overall public safety and efficiency. The deployment of such a system represents a significant step forward in smart city infrastructure, promising substantial benefits for urban mobility.

Methodology

Step 1: Image Capturing

- Algorithm Used:
 - Camera Interface Algorithms: Ensuring images are captured at regular intervals.
- Details:
 - High-resolution cameras capture images at intersections every second.
 - Images are stored in a buffer for processing.

Step 2: Image Cleaning/Unblurring/Noise Reduction

- Algorithms Used:
 - Gaussian Blur: Reduces noise by smoothing the image.
 - Wiener Filter: Corrects blurriness.

- Details:
 - Apply Gaussian Blur to reduce noise.
 - Use Wiener Filter to deblur images, especially in low light.

Step 3: Reshaping of Image

- Algorithm Used:
 - cv2 (OpenCV): For resizing images for object detection.
 - TensorFlow Keras ImageDataGenerator: For preprocessing images for emergency and non-emergency classification.
- Details:
 - Use OpenCV to resize images to a standard size (e.g., 416x416 pixels) for object detection.
 - Use TensorFlow's ImageDataGenerator to preprocess images for classification models.

Step 4: Object Detection

- Algorithm Used:
 - cv2.dnn module: Specifically utilizing pre-trained models.
- Details:
 - Use the cv2.dnn module to load a pre-trained YOLO or MobileNet-SSD model.
 - Perform object detection to identify vehicles and their positions in the image.

Step 5: Vehicle Counting

- Algorithm Used:
 - cv2.dnn module: Continuing from the object detection step.
- Details:
 - Track and count vehicles using the detected bounding boxes from the object detection step.

- Ensure unique vehicle identification using centroid tracking to avoid double counting.

Step 6: Emergency/Non-Emergency Classification

- Algorithm Used:
 - Convolutional Neural Networks (CNNs): Trained to recognize emergency vehicles.
- Details:
 - Use TensorFlow/Keras to train CNN models on images of emergency and non-emergency vehicles.
 - Classify each detected vehicle to determine if it is an emergency vehicle.

Step 7: Density Calculation

- Algorithm Used:
 - Density Estimation Algorithms: Based on vehicle count and types.
- Details:
 - Calculate the total number of vehicles per frame.
 - Estimate density considering the road segment area and the number of vehicles.

Step 8: Time Allocation

- Algorithm Used:
 - Adaptive Traffic Signal Control Algorithm: Based on vehicle density and priority rules.
- Details:
 - Adjust traffic signal timing dynamically based on real-time vehicle density.
 - Minimum green light duration: 10 seconds.
 - Maximum green light duration: 40 seconds.
 - Exception for emergency vehicles: Green light can extend up to 65 seconds.

Step 9: Sharing the Output

- Algorithm Used:

- Internet of Things (IoT) Protocols: Such as MQTT for real-time data sharing.
- Details:
 - Share processed data with a central traffic management system using IoT protocols.
 - Log data for further analysis and machine learning model improvements.

Step 10: Learning and Model Improvement

- Algorithm Used:
 - Reinforcement Learning (RL): For improving traffic signal strategies.
 - Supervised Learning: For updating object detection and classification models.
- Details:
 - Use RL algorithms to continuously improve traffic signal timing based on feedback.
 - Periodically retrain supervised learning models with new data to enhance accuracy.

Key Points

- Minimum Green Light Duration: 10 seconds.
- Maximum Green Light Duration: 40 seconds.
- Exception for Emergency Vehicles: Green light can extend up to 65 seconds.
- Round-robin Strategy: To prevent starvation at any traffic signal.

2.3 Feasibility Study

2.3.1. Introduction

A feasibility study is conducted to assess the viability and potential success of implementing a smart traffic control system in urban areas. This study considers technological feasibility, economic feasibility, operational feasibility, and social feasibility to determine whether the project is feasible and beneficial for stakeholders and the community.

2.3.2. Technological Feasibility

- Available Technologies: Assess the availability and maturity of technologies required for a smart traffic control system, machine learning algorithms, and control centre hardware.
- Integration Challenges: Evaluate the complexity of integrating various technologies, ensuring interoperability, data consistency, and real-time communication between devices and systems.
- Scalability: Consider the scalability of the system to handle increasing traffic volumes, diverse transportation modes, and future technological advancements.

2.3.3. Economic Feasibility

- Cost Analysis: Conduct a comprehensive cost-benefit analysis, considering the initial investment in hardware, software, infrastructure, and ongoing maintenance costs.
- Return on Investment (ROI): Estimate the potential ROI by quantifying the expected benefits, such as reduced congestion, improved traffic flow, fuel savings, reduced emissions, and economic productivity gains.
- Funding Sources: Determine possible funding sources, such as grants from the government, PPPs, and revenue-generating schemes like toll collecting and digital display advertising.

2.3.4. Operational Feasibility

- System Reliability: Evaluate the reliability and uptime of the smart traffic control system, ensuring minimal downtime and robust performance during peak traffic hours and adverse weather conditions.
- Training and Skills: Assess the training needs and skill requirements for personnel operating and maintaining the system, including traffic engineers, data analysts, and IT specialists.

- **Regulatory Compliance:** Ensure compliance with regulatory requirements, traffic safety standards, data privacy regulations, and environmental regulations governing traffic management systems.

2.3.5. Social Feasibility

- **Community Impact:** Analyse the potential social impact of the smart traffic control system on residents, commuters, businesses, and emergency services. Consider factors such as travel time savings, improved road safety, reduced noise pollution, and accessibility for vulnerable populations.
- **Stakeholder Engagement:** Involve key stakeholders, including local government authorities, transportation agencies, law enforcement, public transit operators, and community representatives, in the feasibility assessment and decision-making process.
- **Public Perception:** Assess public perception and acceptance of the smart traffic control system, addressing concerns related to privacy, data security, equity in transportation access, and the overall quality of urban life.

2.4 Python in Smart Traffic Regulation

High-level programming language Python, which is renowned for its ease of use and adaptability, is essential to the creation of the Smart Traffic Regulation using Machine Learning. Its large libraries and frameworks make it the perfect option for putting this project's numerous components into practice.

Ease of Use and Readability

Python's syntax is designed to be clear and concise, making it accessible to developers and easy to read. This ease of use accelerates the development process and reduces the likelihood of errors. For a complex project like the Smart Traffic Regulation using Machine Learning,

where clarity and maintainability of code are crucial, Python's readability ensures that the codebase remains organized and understandable.

Chapter 3

Software, Libraries & Frameworks

3.1 Extensive Libraries and Frameworks

Python's rich ecosystem includes numerous libraries that facilitate the integration of advanced functionalities:

3.1.1 OpenCV2:

An open-source software library for computer vision and machine learning is called OpenCV (Open-Source Computer Vision Library). It offers a large selection of real-time image processing capabilities. In this project, OpenCV2 is used to capture and analyze video feeds from traffic cameras. The library allows for the detection and counting of vehicles at intersections, enabling the system to assess traffic density and flow.

3.1.2 TensorFlow and Keras:

Google created the open-source machine learning library TensorFlow. It offers an adaptable platform for creating apps utilizing deep learning. Neural network design and training is made simpler using Keras, a high-level API that runs on top of TensorFlow. These tools are instrumental in analyzing traffic patterns and making intelligent decisions for traffic light control. By training models on historical traffic data and real-time inputs, the system can predict traffic flow and optimize light timings accordingly.

3.1.3 Support for Machine Learning and Data Analysis

Python is renowned for its capabilities in machine learning and data analysis. Several libraries are integral to this project:

3.1.4 Pandas:

Pandas is an effective library for data analysis and manipulation. It offers data structures such as Data Frames, which are necessary for effectively managing and evaluating traffic data.

3.1.5 NumPy:

The core Python library for scientific computing is called NumPy. Large, multidimensional arrays and matrices are supported, as well as a number of mathematical operations that may be performed on the arrays.

3.1.6 Matplotlib:

Matplotlib is a robust Python charting package that can be used to create interactive, animated, and static displays. The main goal of Matplotlib is to give users the capability and tools to visually depict data, which facilitates analysis and understanding. It was created by John D. Hunter in 2003, and a sizable development community currently manages it.

3.2 Community and Documentation

Because Python has a sizable and vibrant developer community, there are a ton of tools, guides, and documentation accessible. This vast support network guarantees that developers can get answers and direction at any stage of the project lifecycle, which is crucial for problem-solving and ongoing education.

3.3 Implementation in the Project

Python is utilized in various aspects of the Smart Traffic Regulation using Machine Learning project, from data collection to decision-making:

3.4 Data Collection:

Python scripts interface with sensors and cameras to collect real-time traffic data. These scripts gather information on vehicle counts, type (Emergency / Non-emergency) and traffic density at different intersections.

3.5 Image Processing:

Using OpenCV2, the system processes video feeds to detect and count vehicles. This real-time image processing capability is essential for assessing current traffic conditions and making informed decisions.

3.6 Machine Learning:

Machine learning models are created and trained using TensorFlow and Keras. In order to forecast traffic patterns, these models examine both real-time inputs and historical traffic

data. The traffic signal timings are dynamically adjusted based on the projections, improving traffic flow and easing congestion. For visualizing images and data, Matplotlib is utilized.

3.7 Control Logic:

Python handles the control logic for adjusting traffic light timings. Based on the processed data and model predictions, the system dynamically changes the traffic light sequences to prioritize congested routes and improve overall traffic efficiency.

3.8 Deep Learning in Smart Traffic Light Regulation Systems:

Around the world, traffic congestion is becoming a bigger issue in metropolitan areas. It causes drivers to lose time, use more fuel, and become frustrated. Smart traffic signal regulation systems, a potential application of deep learning, a potent subset of artificial intelligence, provide a solution. This study investigates the use of deep learning in traffic flow optimization, congestion reduction, and overall traffic management enhancement.

3.8.1 Traditional Traffic Light Systems vs. Deep Learning

Traditional traffic light systems rely on fixed timing cycles or basic sensors like inductive loops buried under the road. These methods struggle to adapt to real-time traffic conditions, leading to inefficient light sequences during rush hour or unexpected events. Deep learning offers a dynamic approach by analyzing real-time data and making intelligent decisions on light timing.

3.8.2 Deep Learning Techniques for Traffic Light Control

- **Object Detection and Tracking:** Deep learning algorithms can be used to detect and track vehicles and pedestrians at intersections in real-time using video cameras. This data provides valuable insights into traffic density, vehicle types, and pedestrian presence.

- **Traffic Flow Prediction:** Recurrent Neural Networks (RNNs) can analyze historical traffic data and real-time information to predict future traffic patterns. This allows the system to anticipate congestion and adjust light timing proactively.
- **Reinforcement Learning:** Deep Reinforcement Learning (DRL) agents can be trained in simulated traffic environments to learn optimal light control strategies. The DRL agent receives rewards for smooth traffic flow and penalties for congestion, leading to the development of adaptive and efficient light sequences.

3.9 System Architecture

A smart traffic light system powered by deep learning typically comprises the following components:

- **Data Acquisition:** High-resolution cameras or LiDAR sensors capture real-time traffic data at intersections.
- **Data Processing Unit (DPU):** The DPU pre-processes the raw data (e.g., image resizing, noise reduction) for efficient deep learning model inference.
- **Deep Learning Model:** The trained deep learning model analyzes the pre-processed data to detect vehicles, predict traffic flow, and recommend optimal light timing.
- **Traffic Light Controller:** The controller receives the recommendations from the model and adjusts the light sequence accordingly.

3.10 Challenges and Considerations

- **Data Availability and Quality:** Both the volume and quality of training data are critical to the performance of deep learning models. It is essential to have representative and diversified datasets.

- **Cost of Computing:** Deep learning model training and operation might demand a large amount of computing power. Real-world deployment requires the use of specialized hardware and algorithm optimization.
- **Security and Privacy Concerns:** Protecting data privacy while collecting and processing traffic data needs careful consideration. Robust security measures must be implemented.

3.11 Future Directions

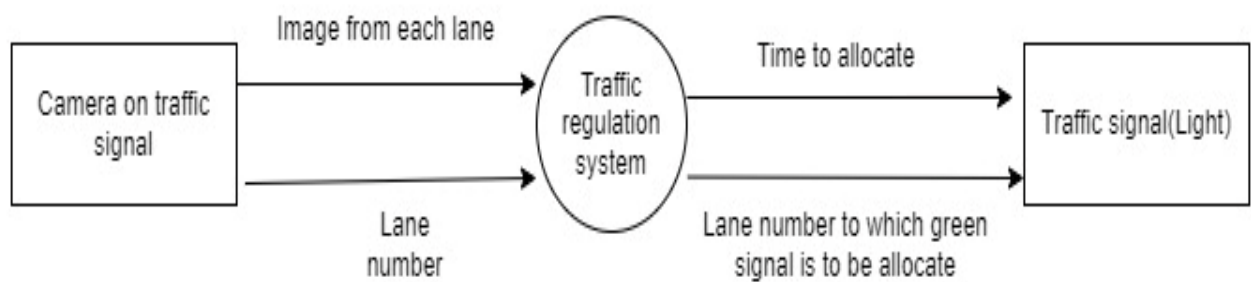
- Integration with other intelligent transportation systems (ITS) for comprehensive traffic management.
- Development of explainable AI (XAI) techniques to provide transparency into deep learning-based traffic control decisions.
- Continuous research on improving the accuracy and efficiency of deep learning models for traffic light control.

Chapter 4

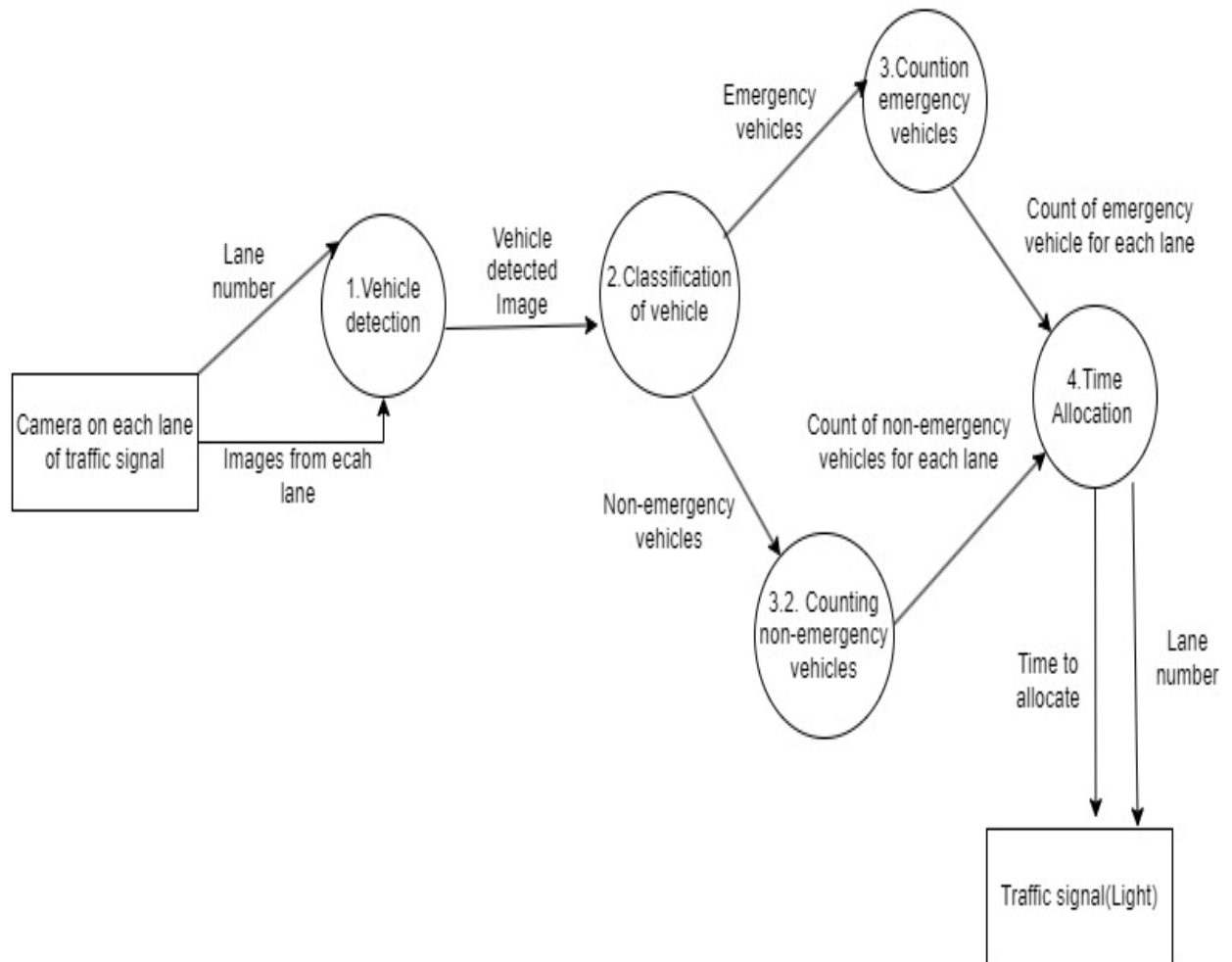
Diagrams and Figures

4.1 Data flow diagram:- DFD represents the data flow of a system or process. It also provides information on each entity's inputs, outputs, and the process itself. There are no loops, control flows, or decision rules in DFD. A flowchart can illustrate certain procedures based on the kind of data.

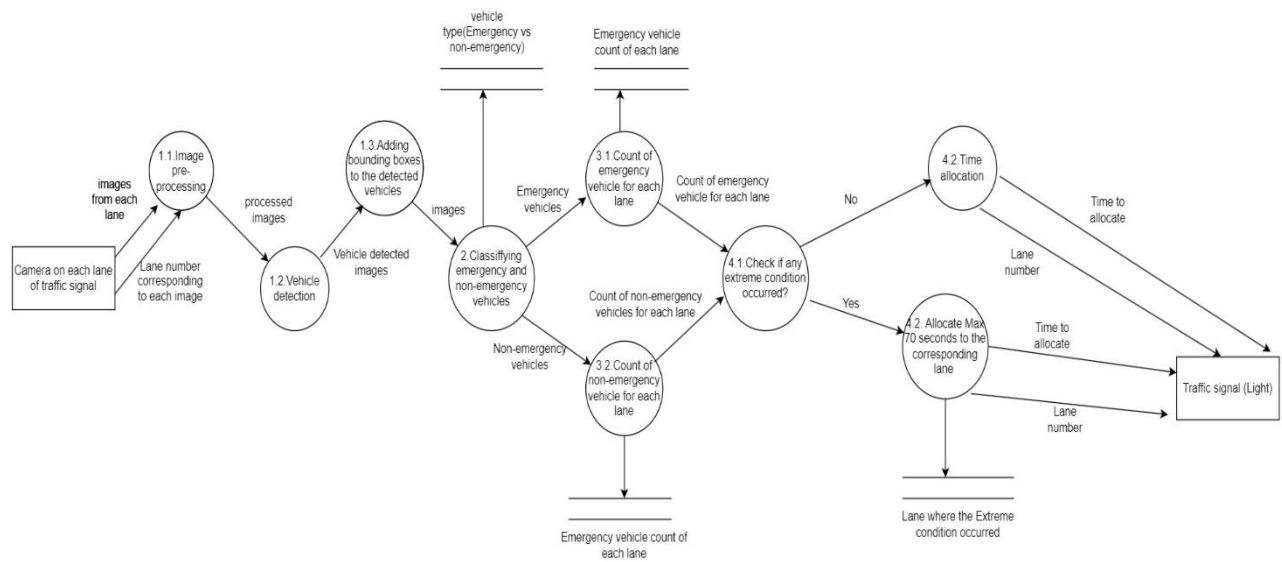
4.1.1 DFD (level-0)



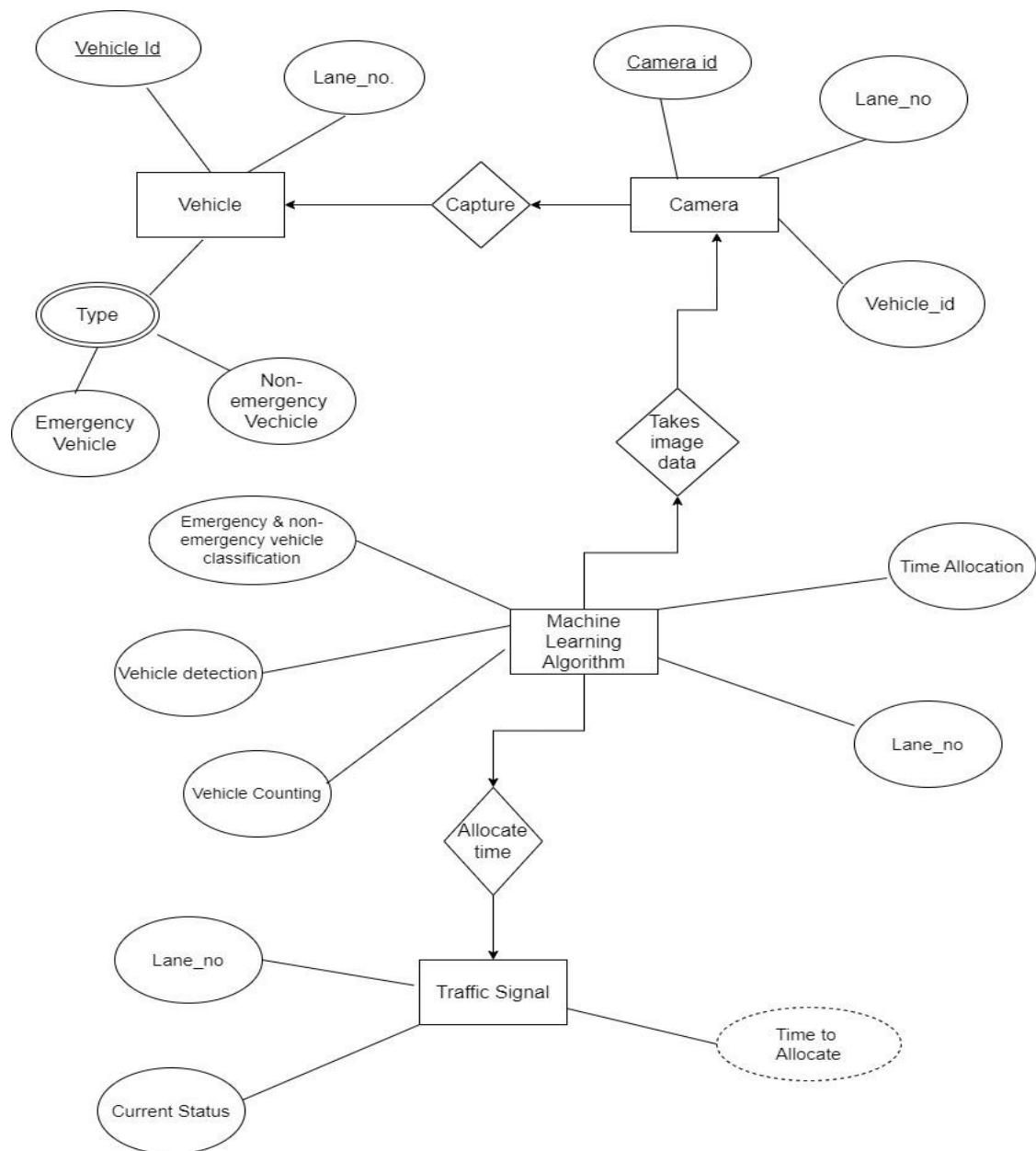
4.1.2 DFD (level –1)



4.1.3 DFD (level-2)

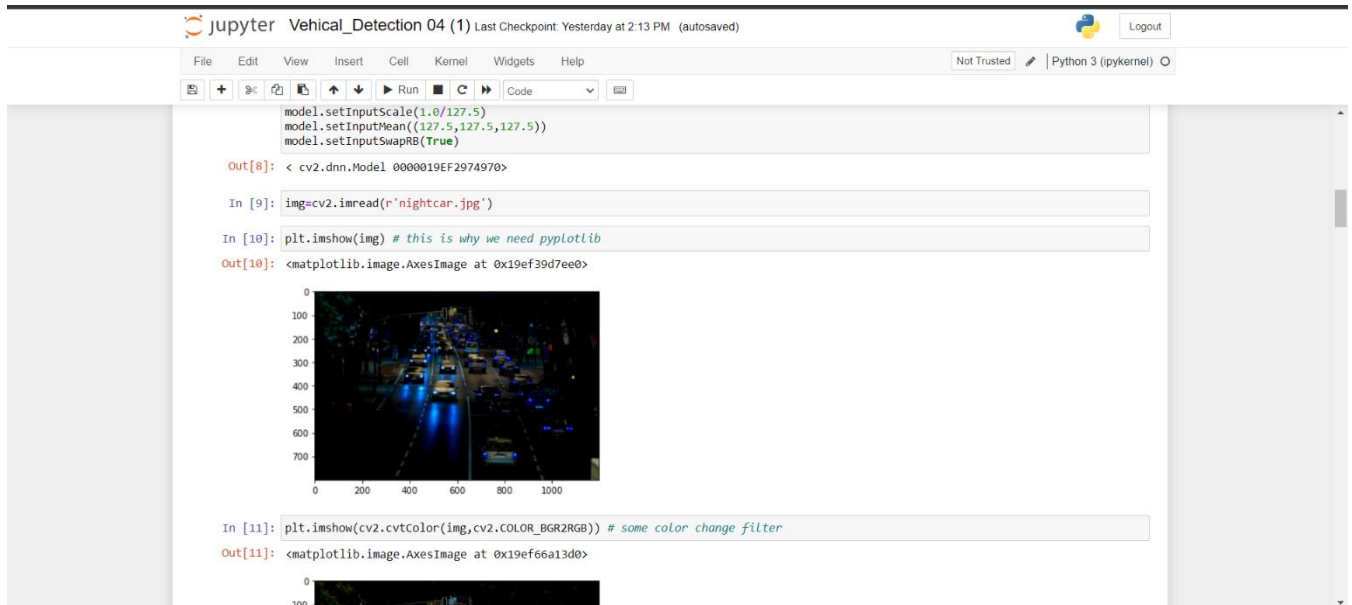


4.2 Entity- relation diagram (ERD)

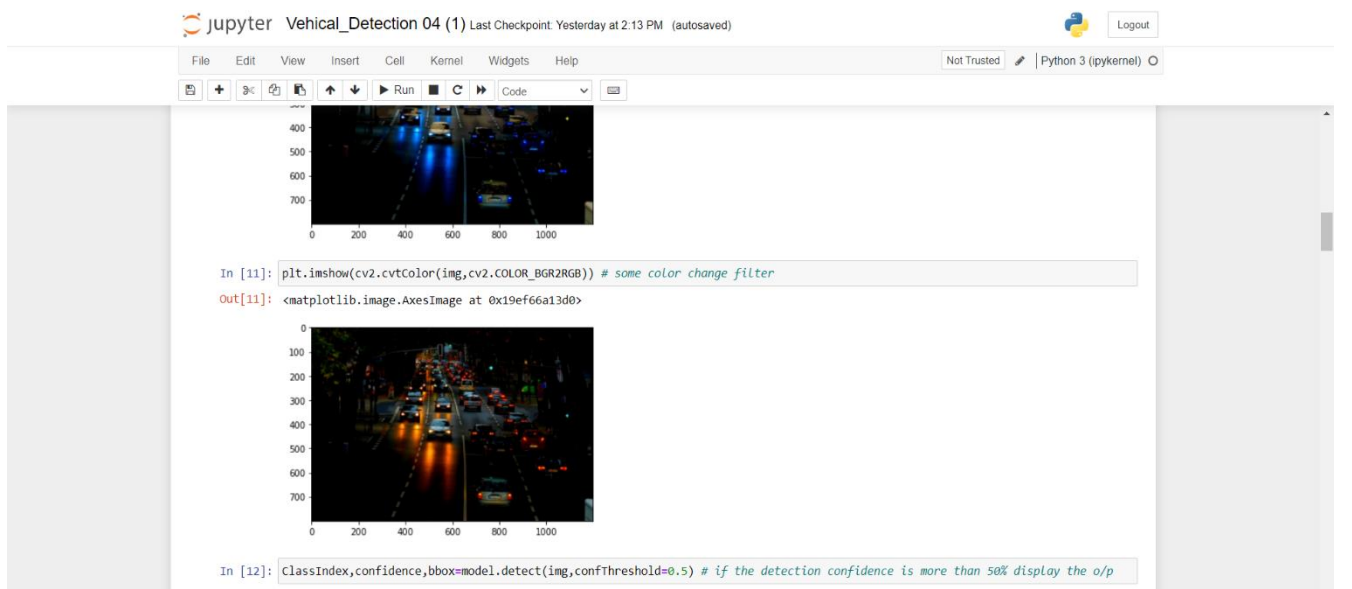


4.3 Results and outputs

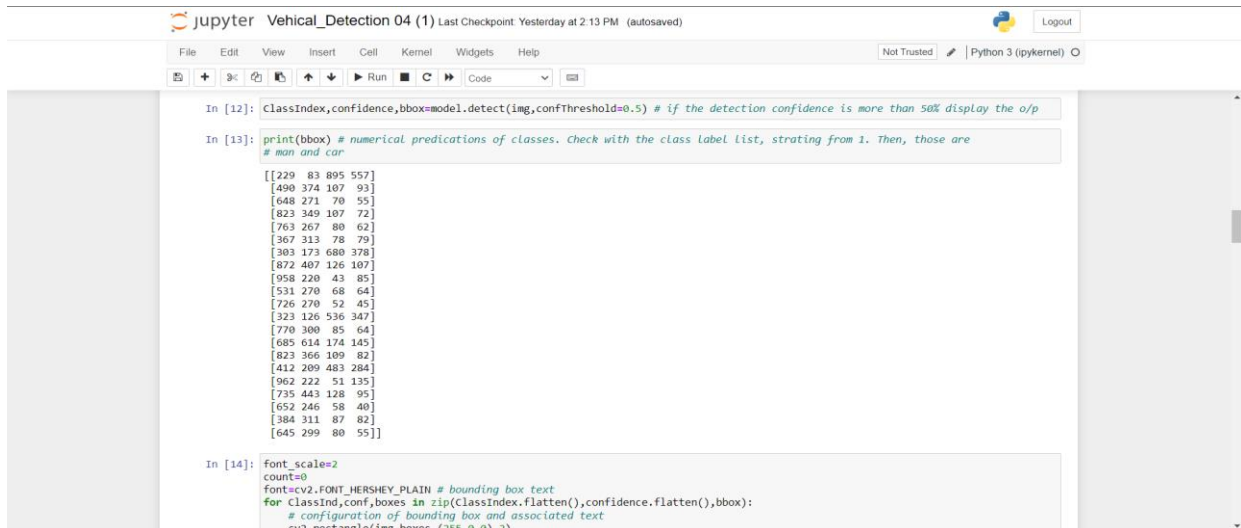
4.3.1 Resizing image to size (320,320)



4.3.2 Changing the colour format to BGR to RGB



4.3.3 Generating the position (X, Y) co-ordinates and dimensions of the objects detected inside the image.



```
Jupyter Vehical_Detection 04 (1) Last Checkpoint: Yesterday at 2:13 PM (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

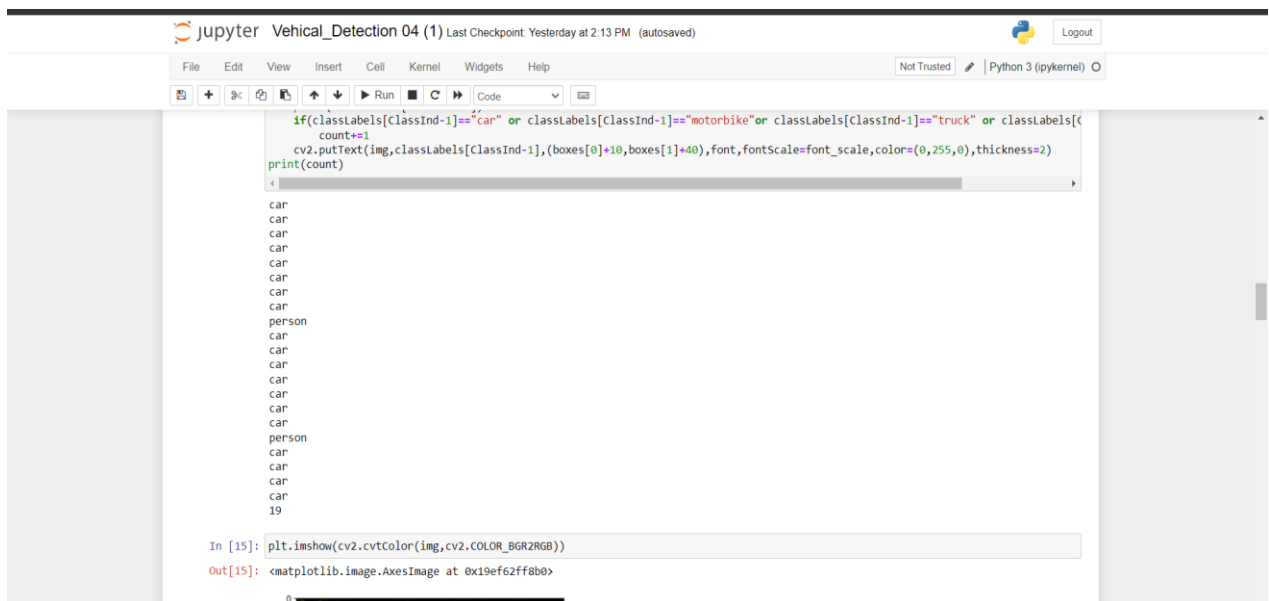
In [12]: classIndex,confidence,bbox=model.detect(img,confThreshold=0.5) # if the detection confidence is more than 50% display the o/p

In [13]: print(bbox) # numerical predications of classes. Check with the class label List, strating from 1. Then, those are
# man and car

[[229  83 895 557]
 [490 374 107  93]
 [648 271  70  55]
 [823 349 107  72]
 [763 267  80  62]
 [367 313  78  79]
 [303 173 680 378]
 [872 407 126 107]
 [958 220  43  85]
 [531 270  68  64]
 [726 270  52  45]
 [323 126 536 347]
 [770 300  85  64]
 [685 614 174 145]
 [823 366 109  82]
 [412 209 483 284]
 [962 222  51 135]
 [735 443 128  95]
 [652 246  58  40]
 [384 311  87  82]
 [645 299  80  55]]

In [14]: font_scale=2
count=0
font=cv2.FONT_HERSHEY_PLAIN # bounding box text
for ClassInd,conf,boxes in zip(ClassIndex.flatten(),confidence.flatten(),bbox):
    # configuration of bounding box and associated text
    cv2.rectangle(img,boxes,(255,0,0),2)
```

4.3.4 Name of the objects detected in the image



```
Jupyter Vehical_Detection 04 (1) Last Checkpoint: Yesterday at 2:13 PM (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

if(classLabels[ClassInd-1]=="car" or classLabels[ClassInd-1]=="motorbike" or classLabels[ClassInd-1]=="truck" or classLabels[ClassInd-1]=="person"):
    count+=1
cv2.putText(img,classLabels[ClassInd-1],(boxes[0]+10,boxes[1]+40),font,fontScale=font_scale,color=(0,255,0),thickness=2)
print(count)

car
car
car
car
car
car
car
person
car
car
car
car
car
car
person
car
car
car
car
car
19

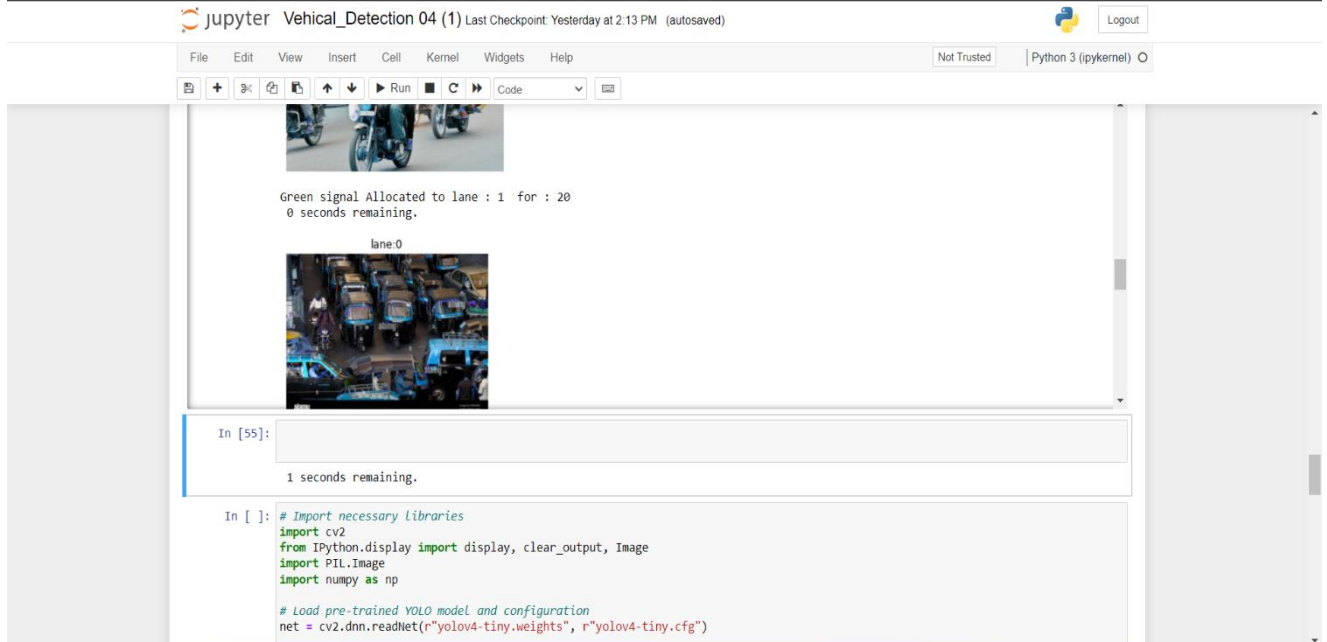
In [15]: plt.imshow(cv2.cvtColor(img,cv2.COLOR_BGR2RGB))

Out[15]: <matplotlib.image.AxesImage at 0x19ef62ff8b0>
```

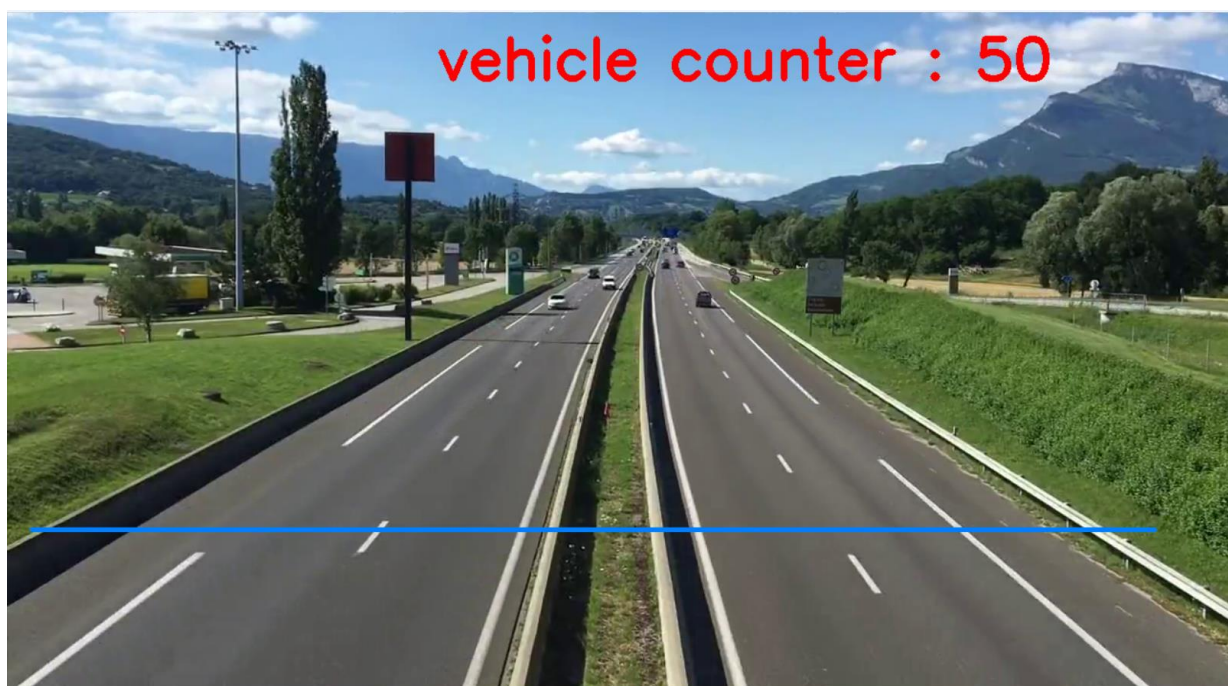
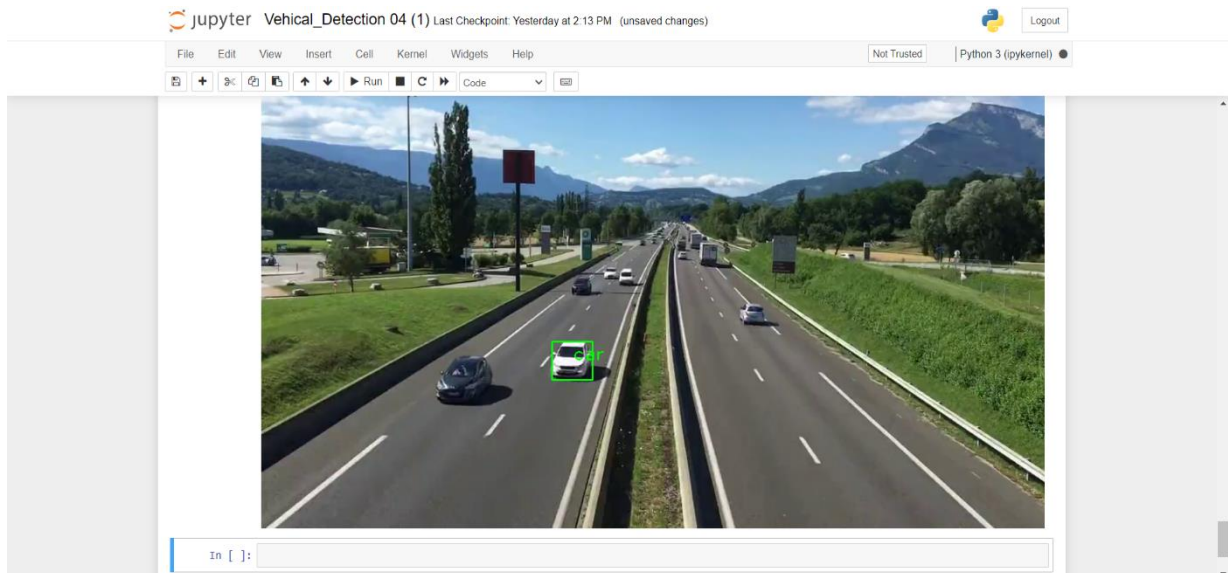
4.3.5 Detected objects in the image



4.3.6 Time allocation to lane



4.3.7 Testing the model on video input (Based on the need You can select either the image or video model)



4.3.8 Importing the necessary library (TensorFlow, etc)

Importing the necessary library

```
In [1]: !pip install --upgrade tensorflow-hub
```

```
In [1]: conda install -c conda-forge cudatoolkit=11.2 cudnn=8.1.0
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

## Package Plan ##

  environment location: C:\Users\aksha\anaconda3

  added / updated specs:
    - cudatoolkit=11.2
    - cudnn=8.1.0

The following packages will be SUPERSEDED by a higher-priority channel:

ca-certificates    pkgs/main::ca-certificates-2024.3.11~ --> conda-forge::ca-certificates-2024.2.2-h56e8100_0
certifi            pkgs/main/win-64::certifi-2024.2.2-py~ --> conda-forge/noarch::certifi-2024.2.2-pyhd8ed1ab_0
```

4.3.9 Installing OpenCV module and importing various modules like, matplotlib, pandas, NumPy, PIL, Keras, sklearn, etc.

```
In [4]: ! pip install numpy~=1.20
import tensorflow as tf
```

```
Requirement already satisfied: numpy~=1.20 in c:\users\aksha\anaconda3\lib\site-packages (1.26.4)
```

```
In [5]: import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import cv2
%matplotlib inline
from PIL import Image
from tqdm import tqdm
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dropout, Conv2D, MaxPooling2D, Flatten, Dense, BatchNormalization
import tensorflow_hub as hub
import keras.utils as image
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.metrics import accuracy_score, classification_report
```

```
WARNING:tensorflow:From C:\Users\aksha\anaconda3\lib\site-packages\tf_keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

```
In [6]: !pip install opencv-python
```

```
Requirement already satisfied: opencv-python in c:\users\aksha\anaconda3\lib\site-packages (4.9.0.80)
Requirement already satisfied: numpy>=1.19.3 in c:\users\aksha\anaconda3\lib\site-packages (from opencv-python) (1.26.4)
```


4.3.10 Reading the first image which have to be trained

Directory assignment

```
In [7]: train_dir = "train"
        test_dir = "test"
```

Reading the first image

```
In [8]: image = cv2.imread(r"train/1.jpg")
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # converting to RGB
```

4.3.11 Checking or Verifying the first image which have to be trained

```
In [9]: plt.imshow(image)
```

```
Out[9]: <matplotlib.image.AxesImage at 0x1df62940c70>
```



4.3.12 Verifying the shape of first image that has to be trained

```
In [10]: image.shape # This shows the shape of the image (length, width,channel)
```

```
Out[10]: (224, 224, 3)
```

4.3.13 Pre-processing of Image data to fit in the model

Preprocessing

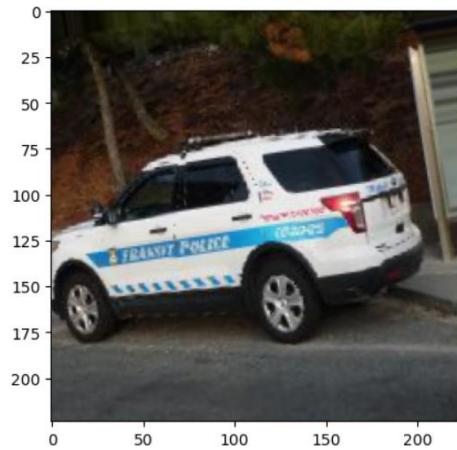
For preprocessing, the ImageDataGenerator is used. It is used for the generation of batches containing the data of tensor images and is used in the domain of real-time data augmentation.

```
In [11]: from tensorflow.keras.preprocessing.image import ImageDataGenerator # Keras image data generator is used for the generation
```

```
In [12]: image_gen = ImageDataGenerator(
    rotation_range=20, # specifies the rotation to be 20%
    width_shift_range=0.1, # specifies the width shift of 10%
    height_shift_range=0.1, # specifies the height shift of 10%
    shear_range=0.20, # crops part of the image
    zoom_range=0.20, # zooms the image by 20%
    fill_mode='nearest', # fills using the nearest pixel
    horizontal_flip=True, # Specifies the horizontal flip
    rescale=1/255 # scales the image
)
```

4.3.14 Verifying the Image from different axis

```
In [13]: plt.imshow(image_gen.random_transform(image))  
Out[13]: <matplotlib.image.AxesImage at 0x1df629b9fd0>
```



4.3.15 Uploading the Training and Testing data into the variable

```
In [14]: train_df=pd.read_csv("train.csv")  
test_df=pd.read_csv("test.csv")
```

```
In [15]: test_df
```

```
Out[15]:
```

	image_names
0	1960.jpg
1	668.jpg
2	2082.jpg
3	808.jpg
4	1907.jpg
...	...
701	674.jpg
702	1027.jpg
703	447.jpg
704	2176.jpg
705	1014.jpg

706 rows × 1 columns

4.3.16. Classified Image (1 denoting Emergency and 0 denoting non-emergency vehicle) in the train dataset

```
In [16]: train_df
```

```
Out[16]:
```

	image_names	emergency_or_not
0	1503.jpg	0
1	1420.jpg	0
2	1764.jpg	0
3	1358.jpg	0
4	1117.jpg	0
...
1641	1638.jpg	0
1642	1095.jpg	0
1643	1130.jpg	0
1644	1294.jpg	0
1645	860.jpg	1

1646 rows × 2 columns

4.3.17 Checking the information of training dataframe

```
In [17]: if 'emergency_or_not' in train_df.columns:
          train_df['emergency_or_not'] = train_df['emergency_or_not'].astype(str)

          if 'emergency_or_not' in test_df.columns:
              test_df['emergency_or_not'] = test_df['emergency_or_not'].astype(str)
```

```
In [18]: train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1646 entries, 0 to 1645
Data columns (total 2 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   image_names      1646 non-null   object
1   emergency_or_not 1646 non-null   object
dtypes: object(2)
memory usage: 25.8+ KB
```

4.3.18 Validating the images

```
In [19]: train_generator=image_gen.flow_from_dataframe(dataframe=train_df[:1150], # specify the dataset used for training
directory=train_dir, # specify the path to the directory
x_col='image_names', # specify the names of the images
y_col='emergency_or_not', # specifies the class labels
class_mode='binary', # specifies the kind of classification
target_size=(224,224), # specifies the dimension to resize the im
batch_size=50 # defines the batch size
)
validation_generator=image_gen.flow_from_dataframe(dataframe=train_df[1150:], # specify the dataset used for tr
directory=train_dir, # specify the path to the directory
x_col='image_names', # specify the names of the images
y_col='emergency_or_not', # specifies the class labels
class_mode='binary', # specifies the kind of classification
target_size=(224,224), # specifies the dimension to resize the im
batch_size=50 # defines the batch size
)
```

Found 1150 validated image filenames belonging to 2 classes.
Found 496 validated image filenames belonging to 2 classes.

4.3.19 Constructor Stage

Constructor stage

```
In [20]: model = Sequential([
    Conv2D(filters=32, kernel_size=(3, 3), input_shape=(224, 224, 3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(filters=32, kernel_size=(3, 3), activation='relu', padding='valid'),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(filters=32, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    BatchNormalization(),
    Flatten(),
    Dense(50, activation='relu'),
    Dropout(0.2),
    Dense(1, activation='sigmoid')
])
```

C:\Users\aksha\anaconda3\lib\site-packages\keras\src\layers\convolutional\base_conv.py:99: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.

```
super().__init__()
```

4.3.20 Compilation stage and verifying the summary of the model

Compilation stage

```
In [21]: model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
```

```
In [22]: model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 32)	896
max_pooling2d (MaxPooling2D)	(None, 112, 112, 32)	0
conv2d_1 (Conv2D)	(None, 110, 110, 32)	9,248
max_pooling2d_1 (MaxPooling2D)	(None, 55, 55, 32)	0
conv2d_2 (Conv2D)	(None, 53, 53, 32)	9,248
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 32)	0
batch_normalization (BatchNormalization)	(None, 26, 26, 32)	128
flatten (Flatten)	(None, 21632)	0
dense (Dense)	(None, 50)	1,081,650
dropout (Dropout)	(None, 50)	0
dense_1 (Dense)	(None, 1)	51

Total params: 1,101,221 (4.20 MB)

Trainable params: 1,101,157 (4.20 MB)

4.3.21 Training stage and fitting the model on the algorithm

Training stage

```
In [23]: history=model.fit(train_generator,epochs = 23,validation_data = validation_generator)
```

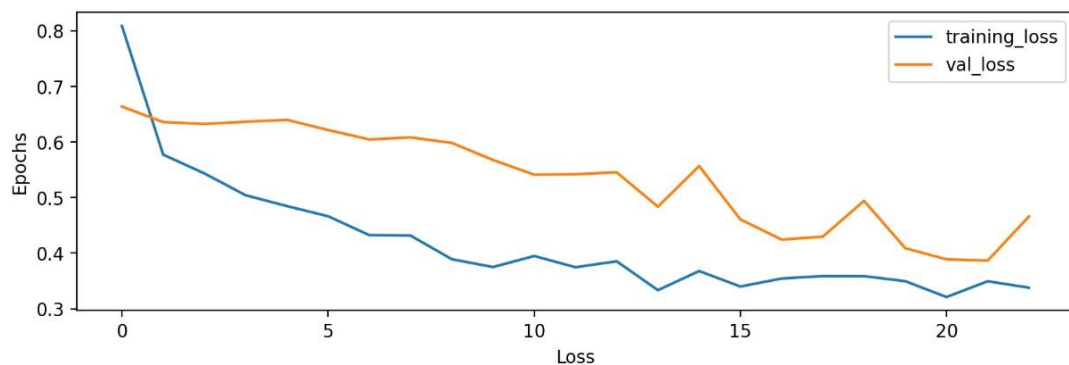
Epoch 1/23

C:\Users\aksha\anaconda3\lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:120: UserWarning: Your PyDataset class should call 'super().__init__(**kwargs)' in its constructor. '**kwargs' can include 'workers', 'use_multiprocessing', 'max_queue_size'. Do not pass these arguments to 'fit()', as they will be ignored.
self._warn_if_super_not_called()

```
23/23 ————— 29s 1s/step - accuracy: 0.5627 - loss: 1.0578 - val_accuracy: 0.6694 - val_loss: 0.6636
Epoch 2/23
23/23 ————— 21s 788ms/step - accuracy: 0.7129 - loss: 0.5477 - val_accuracy: 0.6069 - val_loss: 0.6358
Epoch 3/23
23/23 ————— 23s 883ms/step - accuracy: 0.7346 - loss: 0.5528 - val_accuracy: 0.6069 - val_loss: 0.6323
Epoch 4/23
23/23 ————— 25s 951ms/step - accuracy: 0.7758 - loss: 0.4949 - val_accuracy: 0.6129 - val_loss: 0.6363
Epoch 5/23
23/23 ————— 24s 937ms/step - accuracy: 0.8078 - loss: 0.4619 - val_accuracy: 0.6069 - val_loss: 0.6398
Epoch 6/23
23/23 ————— 20s 757ms/step - accuracy: 0.7817 - loss: 0.4811 - val_accuracy: 0.6230 - val_loss: 0.6215
Epoch 7/23
23/23 ————— 23s 894ms/step - accuracy: 0.8164 - loss: 0.4344 - val_accuracy: 0.6794 - val_loss: 0.6045
Epoch 8/23
23/23 ————— 26s 1s/step - accuracy: 0.8062 - loss: 0.4271 - val_accuracy: 0.6613 - val_loss: 0.6082
Epoch 9/23
23/23 ————— 29s 1s/step - accuracy: 0.8198 - loss: 0.3930 - val_accuracy: 0.7500 - val_loss: 0.5983
Epoch 10/23
23/23 ————— 25s 912ms/step - accuracy: 0.8267 - loss: 0.3795 - val_accuracy: 0.7056 - val_loss: 0.5676
Epoch 11/23
23/23 ————— 23s 896ms/step - accuracy: 0.7967 - loss: 0.3951 - val_accuracy: 0.7903 - val_loss: 0.5411
Epoch 12/23
23/23 ————— 24s 917ms/step - accuracy: 0.8428 - loss: 0.3657 - val_accuracy: 0.7722 - val_loss: 0.5420
Epoch 13/23
23/23 ————— 23s 886ms/step - accuracy: 0.8126 - loss: 0.4086 - val_accuracy: 0.7097 - val_loss: 0.5453
```

4.3.22 Plotting the graph of training and loss value for checking the overfitting

```
In [25]: ## checking for overfitting
history_df = pd.DataFrame(history.history)
plt.figure(dpi=200, figsize = (10,3))
plt.plot(history_df['loss'],
         label='training_loss')
plt.plot(history_df['val_loss'],
         label='val_loss')
plt.xlabel('Loss')
plt.ylabel('Epochs')
plt.legend()
plt.show()
```



4.3.23 Testing datasets to the model

Evaluation stage ¶

```
In [26]: test_df.index
```

```
Out[26]: RangeIndex(start=0, stop=706, step=1)
```

```
In [27]: test_dir_image = []
for i in tqdm(test_df.index):
    img_path = 'test/' + test_df['image_names'][i] # Adjust the path to match the directory structure
    img = Image.open(img_path).convert('RGB') # Load image using PIL
    img = img.resize((224, 224)) # Resize image
    img = np.array(img) # Convert image to NumPy array
    # img = img / 255.0 # Normalize pixel values
    test_dir_image.append(img)

test = np.array(test_dir_image)
```

```
100%|██████████| 706/706 [00:01<00:00, 429.57it/s]
```

```
In [28]: pred = model.predict(test)
```

```
23/23 ————— 5s 154ms/step
```

```
In [29]: num_ = np.floor(pred)
```

4.3.24 Testing a random image and predicting the output as emergency or non-emergency vehicle

```
In [30]: submission = pd.read_csv('sample_submission.csv')
submission['emergency_or_not'] = num_
```

```
In [31]: submission.to_csv('submission29.csv', index = False)
```

```
In [82]: img_path = 'ambulance 1.jpeg' # Adjust the path to match the directory structure
img = Image.open(img_path).convert('RGB') # Load image using PIL
img = img.resize((224, 224)) # Resize image
img = np.array(img) # Convert image to NumPy array
# img = img / 255.0 # Normalize pixel values
l=[img]

test = np.array(l)
```

```
In [83]: pred = model.predict(test)
```

```
1/1 ————— 0s 20ms/step
```

```
In [84]: print(np.floor(pred))
```

```
[[1.]]
```

```
In [76]: submission.head(5)
```

```
Out[76]:
```

	image_names	emergency_or_not
0	1960.jpg	0.0
1	668.jpg	1.0
2	2082.jpg	0.0
3	808.jpg	1.0
4	1907.jpg	0.0

CHAPTER 5

CONCLUSION

An important development in the area of intelligent transportation systems is the " Smart Traffic Regulation using Machine Learning " project. Through the integration of cutting-edge technologies like TensorFlow and Keras for machine learning and OpenCV2 for real-time image processing, we have created a resilient and adaptable system that can optimize traffic flow at crossings. Through the reduction of wait times and enhancement of overall transportation efficiency, this initiative successfully tackles the problems associated with traffic congestion.

5.1 Key Achievements:

1. **Real-Time Traffic Analysis:** Utilizing OpenCV2, the system can accurately detect and count vehicles, providing essential data for traffic management.
2. **Machine Learning Integration:** Our machine learning models can anticipate traffic trends and dynamically modify traffic signals with TensorFlow and Keras, guaranteeing more efficient traffic flow.
3. **Environmental Impact:** By optimizing traffic signal timings, the system reduces vehicle idling time, leading to lower fuel consumption and emissions, contributing to a more sustainable urban environment.
4. **Scalability and Flexibility:** The system lessens vehicle idle time, which lowers fuel consumption and emissions and promotes a more sustainable urban environment. This is accomplished by optimizing traffic signal timings.

CHAPTER 6

FUTURE SCOPE

The Smart Traffic Regulation using Machine Learning project offers significant potential for future development and applications. Building on the current implementation using OpenCV2, TensorFlow, and Keras for data collection and processing, the following future scopes can be considered:

6.1. Integration with Real-time Video Data

Future iterations of the Smart Traffic Regulation using Machine Learning can integrate real-time video feeds from street cameras. This will enable dynamic adjustment of traffic signals based on live traffic conditions, enhancing responsiveness and reducing congestion more effectively. The use of advanced computer vision techniques to analyse vehicle density, speed, and types of vehicles (e.g., emergency vehicles) can further optimize traffic flow.

6.2. Enhanced Machine Learning Models

Current implementations using TensorFlow and Keras can be expanded with more sophisticated machine learning models. For instance, employing deep reinforcement learning can help in making the system more adaptive and efficient. Models can be trained on larger datasets to improve their accuracy and reliability in predicting traffic patterns and optimizing signal timings.

6.3. Vehicle-to-Infrastructure (V2I) Communication

Implementing V2I communication can significantly enhance the smart traffic light system. This involves vehicles communicating with traffic signals to inform them of their presence, speed, and intended direction. This data can help in making more informed decisions, such as extending green light durations for heavy traffic flow or prioritizing emergency vehicles.

6.4. Public Transport Signal Priority

Integrating public transport signal priority (TSP) can improve the efficiency of public transport systems. By giving priority to buses and trams at traffic lights, delays can be

minimized, and the reliability of public transportation can be improved. This is crucial for encouraging the use of public transport and reducing overall traffic congestion.

6.5. Eco-driving and Emissions Reduction

Future systems can incorporate eco-driving support features that optimize traffic signal timings to reduce vehicle idling time, leading to lower fuel consumption and reduced emissions. This aligns with smart city initiatives aimed at making urban areas more sustainable and environmentally friendly.

6.6. Scalability to Larger Networks

Extending the system to synchronize traffic signals across a broader network can further enhance traffic management. Synchronization ensures smoother traffic flow across multiple intersections, reducing bottlenecks and improving overall traffic efficiency. Research indicates that such synchronization can significantly alleviate congestion in metropolitan areas ([MDPI](#)) ([MDPI](#)).

6.7. Adaptive Traffic Signal Control (ATSC)

Traffic flow may be greatly enhanced by creating adaptive traffic signal control systems that modify signal timings in response to real-time traffic data. This involves taking into account variables like the day of the week, the time of day, and particular traffic incidents. Adaptive systems are more efficient than static signal timing schemes because they can adjust to changes in traffic volumes and patterns more effectively.

The Smart Traffic Regulation using Machine Learning project can develop into a more complete and reliable traffic control system by implementing these developments. Urban transportation systems that are safer, more sustainable, and more efficient will be the result of ongoing research and development in this area.

CHAPTER 7

REFERENCES

Books:

- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* (adaptive computation and machine learning series). MIT press.
- Géron, A. (2017). *Hands-on machine learning with Scikit-Learn, Keras & TensorFlow* (Concepts, Tools, and Techniques to Build Intelligent Systems). O'Reilly Media.

Research Papers:

1. “Traffic Light Control with Reinforcement Learning:
https://www.researchgate.net/publication/373451408_Traffic_Light_Control_with_Reinforcement_Learning”
2. “Deep Reinforcement Learning for Traffic Light Control in Intelligent Transportation Systems.”
3. “Intelligent Traffic System Using Machine Learning Techniques: A Review:
https://www.researchgate.net/publication/279917067_Intelligent_Traffic_Light_Control_System_for_Isolated_Intersection_Using_Fuzzy_Logic”
4. “Technologies | Free Full-Text | Traffic Flow Prediction for Smart Traffic Lights Using Machine Learning Algorithms: <https://www.mdpi.com/2227-7080/10/1/5>”
5. “A Survey on Machine Learning for Intelligent Traffic Light Control:
https://www.researchgate.net/publication/370650716_Intelligent_Traffic_System_Using_Machine_Learning_Techniques_A_Review”

Websites and Articles:

- TensorFlow: <https://www.tensorflow.org/>
- Keras: <https://keras.io/>
- OpenCV: <https://opencv.org/>
- tqdm: <https://github.com/tqdm/tqdm>
- **Machine Learning for Smart Traffic Lights:** <https://link.springer.com/article/10.1007/s44163-023-00087-z> (This article provides a good introduction to using ML for smart traffic lights)
- **Smart Traffic Light System by Using Artificial Intelligence:** https://www.researchgate.net/publication/321067358_Artificial_intelligence_for_traffic_signal_control_based_solely_on_video_images (This website showcases an example of an AI-based traffic light system)
- **How Machine Learning Can Improve Traffic Flow:** <https://www.uber.com/en-IN/blog/scaling-ai-ml-infrastructure-at-uber/> (This article discusses the potential of ML for improving traffic management)

Plagiarism Report

Smart Traffic Light Regulation using Machine Learning

ORIGINALITY REPORT

9%

SIMILARITY INDEX

PRIMARY SOURCES

1	www.cabincrew.com Internet	184 words — 3%
2	www.ijmcer.com Internet	33 words — < 1%
3	Ietezaz Ul Hassan, Krishna Panduru, Joseph Walsh. "Review of Data Processing Methods Used in Predictive Maintenance for Next Generation Heavy Machinery", Data, 2024 Crossref	28 words — < 1%
4	Sadashiv Shirabur, Shivaling Hunagund, Suresh Murgd. "VANET Based Embedded Traffic Control System", 2020 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT), 2020 Crossref	19 words — < 1%
5	www.databridgemarketresearch.com Internet	18 words — < 1%
6	Liu, Fangqi. "Leveraging Mobility to Enhance IoT Applications", University of California, Irvine, 2023 ProQuest	17 words — < 1%
7	www.coursehero.com Internet	17 words — < 1%