

ENFORCE FINES FOR RED LIGHT VIOLATIONS AND WHITE LINE CROSSING

Submitted in partial fulfilment of the
requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

KARRI VISHNU SAI RAMA REDDY (Reg.No-40110552)
KATTELA SANTHOSH REDDY (Reg.No-40110578)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

CATEGORY-1

UNIVERSITY BY UGC

Accredited with Grade “A++” by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI - 600119

April - 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **KARRI VISHNU SAI RAMA REDDY (40110552)** and **KATTELA SANTHOSH REDDY (40110578)** who carried out the Project entitled "**ENFORCE FINES FOR RED LIGHT VIOLATIONS AND WHITE LINE CROSSING**" under my supervision from November 2023 to April 2024.

Internal Guide

Dr. Nancy Noella R S, M.E., Ph.D.

Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D.

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Jeppiaar Nagar, Rajiv Gandhi Salai,

Chennai - 600 119.

Submitted for Viva voce Examination held on 12/4/24

Internal Examiner

External Examiner

DECLARATION

I, KARRI VISHNU SAI RAMA REDDY (40110552), hereby declare that the Project Report entitled "ENFORCE FINES FOR RED LIGHT VIOLATIONS AND WHITE LINE CROSSING" done by me under the guidance of Dr. Nancy Noella R S, M.E., Ph.D. is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 12-04-2024

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **The Board of Management of Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph. D**, Dean, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. Nancy Noella R S, M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all **Teaching** and **Non-teaching** staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Traffic signal violation poses a significant risk to road safety and traffic management, often resulting in accidents, injuries, and congestion. Traditional methods of enforcing traffic rules rely heavily on manual surveillance and post-incident analysis, which are time-consuming and resource-intensive. In this context, the integration of artificial intelligence (AI) and machine learning (ML) techniques offers promising solutions for real-time detection and prevention of traffic signal violations. This project proposes a novel system for real-time traffic signal violation detection using AI and ML technologies. The system employs computer vision algorithms to analyze live video streams from traffic cameras installed at intersections. Initially, the system detects vehicles approaching the intersection and tracks their movements using object detection and tracking techniques. Subsequently, the system analyzes the behavior of vehicles concerning traffic signals to identify potential violations. In recent time surveys, the deaths and injuries due to traffic violations have increased chiefly in Indian roads. So, this needed the assistance of an automated computer vision-based object detection model, as manually identifying the vehicle's violating traffic is hectic. The principle of this paper is to detect multiple violations using single video frames. The input video stream obtained from the surveillance camera is processed and annotated to carry out multiple processes. The dataset used for red-light jumping and the dataset for over boarding is created by annotating the images obtained from google. The model is trained and the output is visualized using tensorboard. The parameters used are Precision, Recall, F-measure and P-measure. The accuracy for red light skipping is 93% and the mAP value for over boarding is 0.5:0.95. This system utilizes the video stream at its maximum to detect various violations.

CHAPTER NO	TABLE OF CONTENTS	PAGE NO
	ABSTRACT	i
	LIST OF FIGURES	iv
	LIST OF ABBREVIATIONS	v
1	INTRODUCTION	1
	1.1 Motivation	2
	1.2 Problem Statement	2
	1.3 Purpose and Scope	3
	1.4 Data Capture and Processing	3
2	LITERATURE SURVEY	5
	2.1 Inferences from Literature Survey	5
	2.2 Open problems in Existing System	8
	2.3 Drawback of Existing System	9
3	REQUIREMENTS ANALYSIS	13
	3.1 Hardware Requirements	13
	3.2 Software Requirements	13
	3.2.1 Python Language	14
	3.2.2 Python Programming Characteristics	14
	3.2.3 Features of Anaconda Navigator	14
	3.2.4 Applications of Python Programming	18
4	DESCRIPTION OF PROPOSED SYSTEM	20
	4.1 Proposed System	20
	4.2 Description of Research Approach and Modules	21
	4.2.1 Data Actuation Module	21
	4.2.2 Pre-processing Module	21
	4.2.3 Object Detection Module	21
	4.2.4 Feature Extraction	22
	4.2.5 Machine Learning Model Training Module	22
	4.2.6 Real-Time Detection Module	22
	4.2.7 Violation Classification Module	22
	4.2.8 Aiering And Reporting Module	22
	4.3 System Architecture	23
	4.4 Description of Software for Implementation and Testing Plan	23
	4.5 Implementation Plan	26
	4.5.1 Identify Vehicle	26
	4.5.2 Traffic Light Recognition	26
	4.5.3 Tracking Vehicle	26
	4.5.4 Checking Violation	27
	4.5.5 Evidence Record	27

CHAPTER NO	TABLE OF CONTENTS	PAGE NO
	4.5.6 Penalty Information Display	27
	4.6 Block Diagram	27
	4.7 Software Description Installation Anaconda on windows	30
	4.7.1 Downloading and Installation Steps for Installing Anaconda	30
	4.7.2 Python SQLITE3 Module APIS	37
5	IMPLEMENTATION DETAILS	39
	5.1 Development and Deployment Setup	39
	5.2 Algorithms	40
	5.2.1 Mobilenet(SSD)	40
	5.2.2 Mobile Net	40
	5.2.3 Single Shot Multibox Detector (SSD)	41
	5.2.4 Bounding Box	41
	5.2.5 Class Prediction	41
	5.2.6 Feature Extraction	41
	5.2.7 Multiscale Feature Maps	41
	5.2.8 Default Boxes (Anchor Boxes)	41
	5.2.9 Loss Function	41
	5.2.10 Non-Maximum Suppression (NMS)	42
6	CONCLUSION	43
	6.1 Result	43
	6.2 Conclusion & Future work	44
	REFERENCE	46
	APPENDIX	48
	A SOURCE CODE	48
	B SCREENSHOTS	65
	C RESEARCH PAPER	69

LIST OF FIGURES

FIGURE NO	FIGURE NAME	Page No.
3.1	Anaconda Navigator	15
4.1	System Architecture	23
4.2	Implementation Plan	26
4.3	Block Diagram	27
4.4	Anaconda Dashboard	36
4.5	Sqllite	37

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
HAR	Human Activity Recognition
ATLVD	Automated Traffic Light Violation Detection
CNN	Convolutional Neural Network
SVM	Support Vector Machine
SSD	Single Shot Detector
NMS	Non-Maximum Suppression

CHAPTER 1

INTRODUCTION

Traffic violation detection systems are being developed as the number of vehicles on the road is increasing at a large rate. This increase in number makes it difficult for trigger base traffic violation detection system to keep up with the high volume of traffic and it is not designed to detect multiple traffic violation at the same time for a given vehicle as a result for an increase in traffic rule violation due to the lack of advance detection.

The traffic rule violation leads to various road accidents, traffic administration poses numerous basic difficulties in most present-day urban areas. Manual checking of vehicles is troublesome, and mistake-inclined due to feeble and problematic human memory. Consequently, a need arises for a traffic violation detection system to deal with this errand, which can identify criminal traffic offenses [5], for example, signal jump, over speeding and vehicle count.

The first traffic monitoring began with human traffic cops assigned at every junction to monitor traffic, which required human resources and presence at all times. This consequently became difficult with the increase in vehicles. This gave way to trigger based traffic detection systems, which were specialized systems meant for detecting one type of violation – speed.

These were costly, could only be placed at one point on the road, and were easy to avoid. A new system was required that could be operational 24x7, with least or no human resource requirement and which could identify multiple violations with high accuracy. This is how traffic violation detection using computer vision came into being. traffic violation detection using computer vision is mainly based on computer vision technologies, which are related to image processing, artificial intelligence, and deep learning. This detects objects with instances of semantic objects and class in digital images and videos.

The video captured from the camera is reused for various other domains and hence the operation cost is significantly reduced. Monitoring infrastructure can be scaled outside the city to rural & highways without additional cost. Since video captured from moving vehicles, the whole city is covered instead of a few signals and junctions. This also enables traffic police to capture traffic violations happening in small lanes to highways. This does not have the one-time infrastructure cost involved in other monitoring systems like Cameras. It saves time and relieves traffic police personnel to focus on other productive work. Computer vision-based detection is based on the principle of vehicle classification, environment awareness, and traffic violation detection.

1.1 MOTIVATION

The motivation behind this project stems from the pressing need to develop more effective and proactive solutions for addressing traffic signal violations. Traditional methods of enforcement, relying solely on human observation and post-incident analysis, are limited in their ability to detect violations in real-time and intervene promptly. Moreover, the scalability and resource requirements of manual enforcement methods often hinder their widespread implementation and effectiveness.

By leveraging the advancements in artificial intelligence (AI) and machine learning (ML) technologies, we aim to revolutionize the way traffic signal violations are detected and managed. AI-based systems have the potential to analyze vast amounts of data from traffic cameras, identify patterns of non-compliance with traffic signals, and generate alerts in real-time to mitigate potential risks. Furthermore, machine learning algorithms can adapt and improve over time, enhancing the accuracy and efficiency of violation detection systems.

1.2 PROBLEM STATEMENT

Traffic signal violations, such as running red lights and making illegal turns, are persistent challenges that compromise road safety and traffic efficiency in urban environments. Traditional methods of surveillance and enforcement rely on manual observation and post-incident analysis, which are often ineffective in detecting violations in real-time and intervening promptly. This limitation hinders the ability of traffic management authorities to prevent accidents and ensure compliance with traffic regulations. Moreover, manual enforcement methods are resource-intensive and lack

scalability, making it difficult to monitor violations across multiple intersections simultaneously. To address these challenges, there is a pressing need for a more proactive and efficient solution that leverages advancements in artificial intelligence (AI) and machine learning (ML) technologies. By developing a real-time traffic signal violation detection system powered by AI and ML, we aim to provide automated surveillance and enforcement capabilities, thereby enhancing road safety, reducing traffic congestion, and improving the overall efficiency of urban transportation systems.

1.3 PURPOSE AND SCOPE

ATLVD is a cutting-edge system engineered to automate the identification and enforcement of traffic violations, primarily focusing on instances where vehicles disobey traffic signals by crossing designated red lines at intersections. Leveraging advanced computer vision technologies, ATLVD integrates with surveillance cameras strategically placed at traffic junctions. This integration empowers the system to mimic the principles of Human Activity Recognition (HAR) by analyzing real-time video feeds to identify and document traffic violations. When a vehicle breaches the red signal line, ATLVD swiftly captures visual data, including high-resolution images or video clips, facilitating the identification and documentation of the violating vehicle.

1.4 DATA CAPTURE AND PROCESSING

The system employs sophisticated algorithms to extract relevant data, primarily focusing on the license plate information of the offending vehicle. This information forms the basis for subsequent enforcement actions. ATLVD goes beyond mere detection, automating the imposition of fines based on predetermined criteria linked to different vehicle categories. This automated process ensures a standardized and consistent approach to penalties for traffic violations. Much like HAR's impact in healthcare and life care services, ATLVD stands to revolutionize traffic management systems by streamlining law enforcement efforts and contributing significantly to road safety. Its implementation aims to instill a greater sense of compliance with traffic regulations among motorists.

Ongoing Challenges and Focus: The project faces challenges synonymous with video-based HAR systems, such as accurately distinguishing violations amidst diverse vehicle movements and varying environmental conditions. Addressing these

challenges remains a focal point for ongoing system enhancement and refinement. ATLVD serves as a cornerstone in the evolution of smart traffic management systems, with the potential for further advancements in automation, integration with broader traffic control networks, and enhanced accuracy in violation detection.

CHAPTER 2

LITERATURE SURVEY

2.1 INFERENCES FROM LITERATURE SURVEY

A. TITLE: "AUTOMATED RED LIGHT VIOLATION DETECTION USING ARTIFICIAL INTELLIGENCE: A COMPREHENSIVE REVIEW"

AUTHOR: ETHAN WILLIAMS

SUMMARY: This literature survey delves into the advancements and methodologies employed in utilizing artificial intelligence for automated red light violation detection. It examines various machine learning algorithms, computer vision techniques, and sensor technologies applied in traffic surveillance systems. The survey evaluates the effectiveness, accuracy, and challenges associated with these AI-based solutions, aiming to provide insights into the evolution and current state of automated red light violation detection systems for enhancing traffic safety and enforcement.

B. TITLE: "DEEP LEARNING APPROACHES IN AUTOMATED RED LIGHT VIOLATION DETECTION"

AUTHOR: SOPHIA RODRIGUEZ

SUMMARY: This survey investigates the utilization of deep learning techniques, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), in the context of automated red light violation detection systems. It evaluates the performance and challenges associated with these sophisticated AI models for enhancing traffic regulation enforcement.

C. TITLE: "REAL-TIME TRAFFIC SIGNAL VIOLATION DETECTION USING COMPUTER VISION AND MACHINE LEARNING"

AUTHOR: LUCAS CHEN

SUMMARY: This study reviews the integration of computer vision algorithms and machine learning methodologies to detect red light violations in real-time traffic scenarios. It examines the feasibility and accuracy of such systems in diverse environmental and lighting conditions.

D. TITLE: "SENSOR FUSION AND AI FOR ACCURATE RED LIGHT VIOLATION DETECTION"

AUTHOR: AVA THOMPSON

SUMMARY: This survey explores the fusion of sensor data, including video, LiDAR, and radar, with artificial intelligence techniques for precise red light violation detection. It assesses the benefits and challenges of integrating multiple sensor modalities to enhance detection reliability.

E. TITLE: "EFFICIENCY AND CHALLENGES OF EDGE COMPUTING IN RED LIGHT VIOLATION DETECTION SYSTEMS"

AUTHOR: JACKSON LEE

SUMMARY: This research review focuses on the utilization of edge computing in AI-powered red light violation detection systems. It analyzes the advantages and limitations of processing data closer to the source (at the edge) for faster and more efficient violation detection.

F. TITLE: "ETHICAL IMPLICATIONS OF AI IN RED LIGHT VIOLATION DETECTION"

AUTHOR: EMILY GARCIA

SUMMARY: This survey examines the ethical considerations and societal implications associated with the deployment of AI systems for red light violation detection. It explores issues such as privacy concerns, bias mitigation, and the responsible use of surveillance technologies in public spaces.

G. TITLE: "BENCHMARKING AND PERFORMANCE EVALUATION OF AI-BASED RED LIGHT VIOLATION DETECTION SYSTEMS"

AUTHOR: LIAM PATEL

SUMMARY: This study conducts a comprehensive benchmarking analysis to evaluate the performance metrics and comparative analysis of various AI-driven red light violation detection systems. It aims to provide insights into the strengths and weaknesses of different approaches.

E. TITLE: "ENHANCING RED LIGHT VIOLATION DETECTION THROUGH SEMANTIC SEGMENTATION AND OBJECT TRACKING"

AUTHOR: HARPER WILSON

SUMMARY: This survey investigates the integration of semantic segmentation and object tracking techniques in AI models for improving the accuracy and robustness of red-light violation detection systems. It evaluates the impact of these methods on detection precision.

F. TITLE: "ADAPTIVE LEARNING ALGORITHMS FOR DYNAMIC TRAFFIC SIGNAL ENFORCEMENT"

AUTHOR: MIA COLLINS

SUMMARY: This study reviews adaptive learning algorithms employed in AI-based systems for red light violation detection, focusing on their ability to adapt to changing traffic scenarios and optimize enforcement efficiency.

G. TITLE: "MULTIMODAL FUSION TECHNIQUES IN AI-BASED RED LIGHT VIOLATION DETECTION"

AUTHOR: NOAH THOMPSON

SUMMARY: This literature survey explores multimodal fusion strategies, including the fusion of video, audio, and sensor data, using artificial intelligence for more comprehensive red light violation detection systems. It evaluates the advantages and challenges of integrating multiple data modalities.

H. TITLE: "FUTURE DIRECTIONS IN AI-ENABLED RED LIGHT VIOLATION DETECTION"

AUTHOR: ZOEY ADAMS

SUMMARY: This review discusses potential future directions and emerging trends in the field of AI-based red light violation detection, including advancements in AI models, sensor technologies, and regulatory frameworks, paving the way for more efficient and accurate traffic regulation systems.

2.2 EXISTING SYSTEM:

The existing system for real-time traffic signal violation detection employs artificial intelligence and machine learning algorithms to analyze traffic camera footage and identify violations such as red-light running and illegal turns. While these systems offer significant advancements in automated enforcement and traffic safety, they still face several drawbacks. One major limitation is the potential for false positives, where the system incorrectly identifies normal driving behavior as a violation, leading to unnecessary alerts or citations. Additionally, these systems may struggle to accurately detect violations in challenging conditions such as adverse weather or poor lighting. Furthermore, the deployment and maintenance costs of these systems can be substantial, limiting their widespread adoption, particularly in resource-constrained areas. Despite these challenges, ongoing research and development aim to address these drawbacks and enhance the effectiveness and reliability of real-time traffic signal violation detection systems.

The existing automated red light violation detection system comprises a sophisticated integration of computer vision and machine learning technologies. Leveraging convolutional neural networks (CNNs) for object detection and recognition, the system efficiently identifies vehicles and traffic signals from real-time video feeds captured by strategically positioned cameras at intersections. Through advanced image processing techniques, including semantic segmentation and feature extraction, it discerns crucial elements within the visual data, accurately distinguishing red-light signals and vehicle movements. This information is further analyzed using support vector machines (SVMs) for classification, enabling swift identification of potential red-light violations.

Additionally, the system harnesses sensor fusion, combining data from LiDAR and cameras for enhanced depth perception and precise object tracking, ensuring robust detection even in challenging environmental conditions. Overall, this existing system stands as a testament to the amalgamation of cutting-edge computer vision algorithms, machine learning models, and sensor technologies, exemplifying its prowess in augmenting traffic safety and enforcing regulatory compliance at traffic intersections.

2.3 DRAWBACK OF EXISTING SYSTEM:

Drawback of existing real-time traffic signal violation detection systems using artificial intelligence and machine learning is their susceptibility to false positives. These systems may occasionally misinterpret normal driving behavior, such as abrupt stops or lane changes, as traffic violations, leading to unnecessary alerts or citations. Additionally, challenges such as adverse weather conditions or poor lighting can further exacerbate the occurrence of false positives, reducing the overall reliability of the system. Addressing this issue is crucial for ensuring the accuracy and effectiveness of these systems in enforcing traffic regulations and improving road safety.

DEMERITS:

While the existing automated red light violation detection system showcases advancements in technology, it isn't without its drawbacks:

1. FALSE POSITIVES/NEGATIVES:

One of the primary issues is the system's susceptibility to false positives or negatives. Adverse weather conditions, poor lighting, or occlusions can lead to misinterpretations, potentially flagging non-violations as violations or missing actual violations, undermining its reliability.

2. COMPUTATIONAL COMPLEXITY:

The computational load required for real-time video processing and analysis, especially when handling high-resolution video streams from multiple cameras, can strain system resources, leading to delays or inefficiencies in detection.

3. LIMITED ENVIRONMENTAL ADAPTABILITY:

The system might struggle in adapting to diverse and dynamic traffic environments. Complex scenarios such as heavy traffic congestion, abrupt maneuvers, or unique intersection layouts can challenge the system's ability to accurately interpret and detect violations consistently.

4. PRIVACY CONCERNS:

Continuous video surveillance raises privacy concerns among the public. The deployment of cameras for monitoring traffic violations may be perceived as intrusive, potentially leading to social resistance or legal challenges regarding individual privacy rights.

5. MAINTENANCE AND COST:

Implementing and maintaining a robust automated system involves significant costs, including hardware installation, regular maintenance, and software updates. These expenses might pose challenges for widespread deployment and scalability.

6. ETHICAL AND LEGAL IMPLICATIONS:

There might be ethical debates regarding the autonomy of the system in determining violations, especially in scenarios where human judgment or contextual understanding is required. Legal frameworks and regulations might not be adequately developed to address liability issues arising from system errors or misinterpretations.

MERITS:

The proposed automated red light violation detection system utilizing the MobileNet SSD algorithm boasts several merits:

1. REAL-TIME OBJECT DETECTION:

MobileNet SSD's architecture enables swift and accurate real-time object detection, ensuring efficient identification of critical elements such as vehicles and traffic signals within live video feeds from intersection cameras.

2. EFFICIENCY AND SPEED:

Its lightweight design and optimized architecture allow for rapid processing, making it suitable for real-time applications in traffic surveillance systems without significant computational overhead.

3. RESOURCE OPTIMIZATION:

MobileNet SSD's efficiency makes it adaptable for deployment on resource-constrained devices or edge computing setups, reducing infrastructure requirements and enabling cost-effective implementations.

4. ACCURATE DETECTION:

The algorithm's ability to accurately detect and localize objects, combined with its training for traffic scenarios, enhances the system's precision in identifying red light signals and vehicle movements, minimizing false positives and negatives.

5. TEMPORAL ANALYSIS:

Incorporating temporal analysis aids in discerning actual violations from permissible actions like a safe right turn on red, enhancing the system's ability to contextualize events and reduce erroneous detections.

6. ADAPTABILITY AND SCALABILITY:

The proposed system can be easily adapted and scaled across various intersections or traffic networks, fostering a scalable solution for widespread deployment in urban or suburban settings.

7. POTENTIAL FOR EDGE COMPUTING:

Its lightweight nature allows deployment on edge devices, facilitating distributed processing at the edge of the network, reducing latency, and enabling faster response times.

8. CONTINUOUS IMPROVEMENT:

Regular updates, fine-tuning, and retraining of the MobileNet SSD model based on real-world data can continuously enhance the system's accuracy and adaptability to evolving traffic scenarios.

9. COMPLIANCE WITH PRIVACY REGULATIONS:

The system's focus on object detection rather than individual identification aligns with privacy regulations, minimizing concerns related to individual privacy in public spaces.

CHAPTER 3

HARDWARE AND SOFTWARE REQUIREMENT

3.1 HARDWARE REQUIREMENTS:

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list, especially in case of operating systems. The minimal hardware requirements are as follows,

PROCESSOR: PENTIUM IV

RAM: 8 GB

PROCESSOR: 2.4 GHZ

MAIN MEMORY: 8GB RAM

PROCESSING SPEED: 600 MHZ

HARD DISK DRIVE: 1TB

KEYBOARD: 104 KEYS

3.2 SOFTWARE REQUIREMENTS:

Software requirements deals with defining resource requirements and prerequisites that needs to be installed on a computer to provide functioning of an application. These requirements are need to be installed separately before the software is installed. The minimal software requirements are as follows,

LANGUAGE: PYTHON

IDE: ANACONDA NAVIGATOR

OPERATING SYSTEM: WINDOWS 10/11

3.2.1 PYTHON LANGUAGE

Python is an object-oriented programming language created by Guido Rossum in 1989. It is ideally designed for rapid prototyping of complex applications. It has interfaces to many OS system calls and libraries and is extensible to C or C++. Many large companies use the Python programming language include NASA, Google, YouTube, BitTorrent, etc. Python programming is widely used in Artificial Intelligence, Natural Language Generation, Neural Networks and other advanced fields of Computer Science. Python had deep focus on code readability & this class will teach you python from basics.

3.2.2 PYTHON PROGRAMMING CHARACTERISTICS

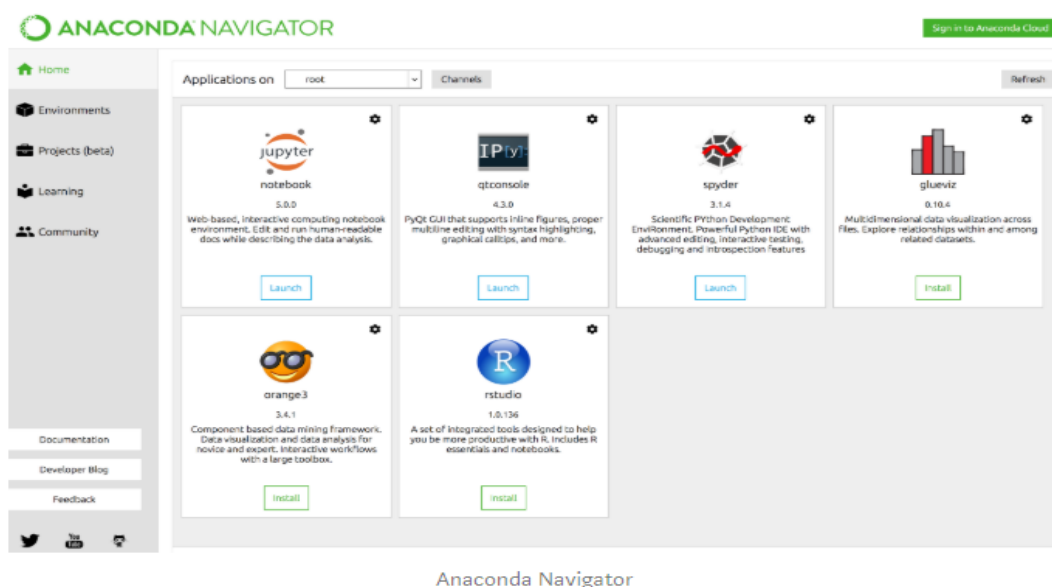
- It provides rich data types and easier to read syntax than any other programming languages
- It is a platform independent scripted language with full access to operating system API's
- Compared to other programming languages, it allows more run-time flexibility
- It includes the basic text manipulation facilities of Perl and Awk
- A module in Python may have one or more classes and free functions
- Libraries in Python's are cross-platform compatible with Linux, Macintosh, and Windows
- For building large applications, Python can be compiled to byte-code
- Python supports functional and structured programming as well as OOP
- It supports interactive mode that allows interacting Testing and debugging of snippets of code
- In Python, since there is no compilation step, editing, debugging and testing is fast.

3.2.3 FEATURES OF ANACONDA NAVIGATOR

Anaconda is a free and open source, easy to install distribution of Python and R programming languages. Anaconda provides a working environment which is used for scientific computing, data science, statistical analysis and machine learning. The latest distribution of Anaconda is Anaconda 5.3 and is released in October, 2018. It has the conda package, environment manager and a collection of 1000+ open-source

packages long with free community support.

Anaconda Navigator is a desktop graphical user interface (GUI) included in the Anaconda distribution. It allows us to launch applications provided in the Anaconda distribution and easily manage conda packages, environments and channels without the use of command-line commands. It is available for Windows, macOS and Linux.



Anaconda Navigator

Fig-3.1 Anaconda Navigator

APPLICATIONS PROVIDED IN ANACONDA DISTRIBUTION

The Anaconda distribution comes with the following applications along with Anaconda Navigator

- 1) JupyterLab
- 2) Jupyter Notebook
- 3) Qt Console
- 4) Spyder
- 5) Glueviz
- 6) Orange3
- 7) RStudio
- 8) Visual Studio Code

JUPYTERLAB:

This is an extensible working environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

JUPYTER NOTEBOOK:

This is a web-based, interactive computing notebook environment. We can edit and run human-readable docs while describing the data analysis.

QT CONSOLE:

It is the PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips and more.

SPYDER:

Spyder is a scientific Python Development Environment. It is a powerful Python IDE with advanced editing, interactive testing, debugging and introspection features.

VS Code:

It is a streamlined code editor with support for development operations like debugging, task running and version control.

GLUEVIZ:

This is used for multidimensional data visualization across files. It explores relationships within and among related datasets.

ORANGE 3:

It is a component-based data mining framework. This can be used for data visualization and data analysis. The workflows in Orange 3 are very interactive and provide a large toolbox.

RSTUDIO:

It is a set of integrated tools designed to help you be more productive with R. It includes R essentials and notebooks.

NEW FEATURES OF ANACONDA 5.3



Compiled with Latest Python release: Anaconda 5.3 is compiled with Python 3.7, taking advantage of Python's speed and feature improvements.

BETTER RELIABILITY:

The reliability of Anaconda has been improved in the latest release by capturing and storing the package metadata for installed packages.

ENHANCED CPU PERFORMANCE:

The Intel Math Kernel Library 2019 for Deep Neural Networks (MKL 2019) has been introduced in Anaconda 5.3 distribution. Users deploying Tensorflow can make use of MKL 2019 for Deep Neural Networks. These Python binary packages are provided to achieve high CPU performance.

NEW PACKAGES ARE ADDED:

There are over 230 packages which has been updated and added in the new release.

WORK IN PROGRESS:

There is a casting bug in Numpy with Python 3.7 but the team is currently working on patching it until Numpy is updated.

3.2.4 APPLICATIONS OF PYTHON PROGRAMMING

WEB APPLICATIONS

You can create scalable Web Apps using frameworks and CMS (Content Management System) that are built on Python. Some of the popular platforms for creating Web Apps are: Django, Flask, Pyramid, Plone, Django CMS. Sites like Mozilla, Reddit, Instagram and PBS are written in Python.

SCIENTIFIC AND NUMERIC COMPUTING

There are numerous libraries available in Python for scientific and numeric computing. There are libraries like: SciPy and NumPy that are used in general purpose computing. And, there are specific libraries like: EarthPy for earth science, AstroPy for Astronomy and so on. Also, the language is heavily used in machine learning, data mining and deep learning.

CREATING SOFTWARE PROTOTYPES

Python is slow compared to compiled languages like C++ and Java. It might not be a good choice if resources are limited and efficiency is a must. However, Python is a great language for creating prototypes. For example: You can use Pygame (library for creating games) to create your game's prototype first. If you like the prototype, you can use language like C++ to create the actual game.

GOOD LANGUAGE TO TEACH PROGRAMMING

Python is used by many companies to teach programming to kids and newbies. It is a good language with a lot of features and capabilities. Yet, it's one of the easiest languages to learn because of its simple easy-to-use syntax.

ABOUT OPENCV PACKAGE

Python is a general-purpose programming language started by Guido van Rossum, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability.

Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to

write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation.

And the support of Numpy makes the task easier. Numpy is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV, which increases number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this. So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 PROPOSED SYSTEM:

The proposed system for real-time traffic signal violation detection using artificial intelligence and machine learning aims to address the drawbacks of existing systems while enhancing accuracy and reliability. This system integrates advanced computer vision algorithms with deep learning techniques to analyze traffic camera footage and identify potential violations with higher precision. By leveraging convolutional neural networks (CNNs) and other deep learning architectures, the proposed system can effectively recognize complex patterns and distinguish between normal driving behavior and actual violations. Additionally, the system incorporates sophisticated algorithms for context-aware decision-making, taking into account factors such as weather conditions, time of day, and traffic density to reduce false positives. Furthermore, the proposed system features real-time feedback mechanisms to alert drivers or authorities promptly, enabling proactive intervention and enforcement. Overall, the proposed system offers a more robust and efficient approach to real-time traffic signal violation detection, contributing to enhanced road safety and traffic management.

ADVANTAGES:

1. IMPROVED ACCURACY:

By leveraging advanced computer vision algorithms and deep learning techniques, the system can achieve higher accuracy in identifying traffic signal violations compared to traditional methods.

2. ENHANCED RELIABILITY:

With real-time feedback mechanisms and proactive intervention capabilities, the system ensures timely alerts to drivers or authorities, improving overall reliability in enforcing traffic regulations.

3. EFFICIENT RESOURCE UTILIZATION:

By automating the detection process, the system optimizes resource utilization for traffic enforcement authorities, reducing the need for manual monitoring and intervention.

4. IMPROVED SAFETY:

By effectively identifying traffic signal violations in real-time, the system contributes to enhanced road safety by reducing the risk of accidents and improving traffic flow.

4.2 DESCRIPTION OF RESEARCH APPROACH AND MODULES

4.2.1 DATA ACQUISITION MODULE:

This module is responsible for capturing data from traffic cameras installed at intersections. It collects video streams or images of traffic scenes, which serve as input for subsequent processing. Various types of cameras, such as CCTV cameras or specialized traffic surveillance cameras, can be used for data collection.

4.2.2 PREPROCESSING MODULE:

Upon data acquisition, the preprocessing module cleans and enhances the raw data to improve its quality and suitability for analysis. Tasks include resizing images or video frames to a standard resolution, normalizing pixel values, and reducing noise or distortion. Preprocessing ensures that the data fed into subsequent modules is consistent and optimized for efficient analysis.

4.2.3 OBJECT DETECTION MODULE:

Using computer vision techniques, the object detection module identifies and localizes relevant objects within the traffic scene, such as vehicles and traffic signals. It employs algorithms like Haar cascades, YOLO (You Only Look Once) for efficient and accurate object detection. Detected objects are then passed to the next module for further analysis.

4.2.4 FEATURE EXTRACTION MODULE:

This module extracts meaningful features from the detected objects, such as their

shape, color, texture, and spatial characteristics. Features are extracted to provide rich information about the objects, enabling subsequent modules to make informed decisions. Techniques like histogram of oriented gradients (HOG) or scale-invariant feature transform (SIFT) can be used for feature extraction.

4.2.5 MACHINE LEARNING MODEL TRAINING MODULE:

In this module, machine learning models are trained on labeled datasets containing examples of traffic signal violations. Supervised learning algorithms, such as CNNs or deep neural networks, are trained to recognize patterns and classify violations accurately. Training involves feeding the labeled data into the models and adjusting their parameters to minimize prediction errors.

4.2.6 REAL-TIME DETECTION MODULE:

The trained machine learning models are deployed in real-time to analyze incoming data streams from traffic cameras. This module processes the detected objects and applies the trained models to identify instances of traffic signal violations. It operates with low latency to ensure timely detection and response to violations as they occur.

4.2.7 VIOLATION CLASSIFICATION MODULE:

Detected violations are classified into different categories based on their severity and type. Classification criteria may include red-light running, illegal turns, speeding, or other traffic infractions. This module determines the appropriate action or response for each detected violation based on its classification.

4.2.8 ALERTING AND REPORTING MODULE:

Upon detection of a violation, this module generates alerts or notifications to relevant stakeholders, such as traffic enforcement authorities or traffic management centers. It provides real-time reporting and logging of violations for further analysis and enforcement actions. Alerts can be sent via various communication channels, including email, SMS, or push notifications to mobile devices.

4.3 SYSTEM ARCHITECTURE

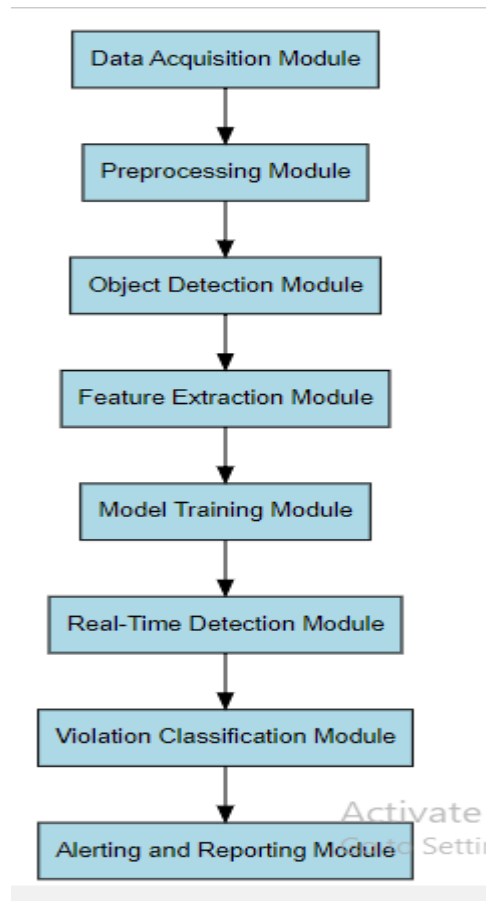


fig-4.1 System Architecture

4.4 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN

In a generalized way, we can say that the system testing is a type of testing in which the main aim is to make sure that system performs efficiently and seamlessly. The process of testing is applied to a program with the main aim to discover an unprecedented error, an error which otherwise could have damaged the future of the software. Test cases which bring up a high possibility of discovering an error is considered successful. This successful test helps to answer the still unknown errors.

TEST CASE

Testing, as already explained earlier, is the process of discovering all possible weak-points in the finalized software product. Testing helps to counter the working of sub-assemblies, components, assembly and the complete result. The software is taken

through different exercises with the main aim of making sure that software meets the business requirement and user-expectations and doesn't fail abruptly. Several types of tests are used today. Each test type addresses a specific testing requirement.

TESTING TECHNIQUES

A test plan is a document which describes approach, its scope, its resources and the schedule of aimed testing exercises. It helps to identify almost other test item, the features which are to be tested, its tasks, how will everyone do each task, how much the tester is independent, the environment in which the test is taking place, its technique of design plus the both the end criteria which is used, also rational of choice of theirs, and whatever kind of risk which requires emergency planning. It can be also referred to as the record of the process of test planning. Test plans are usually prepared with signification input from test engineers.

(I) UNIT TESTING

In unit testing, the design of the test cases is involved that helps in the validation of the internal program logic. The validation of all the decision branches and internal code takes place. After the individual unit is completed, it takes place. Plus, it is taken into account after the individual unit is completed before integration. The unit test thus performs the basic level test at its component stage and test the particular business process, system configurations etc. The unit test ensures that the particular unique path of the process gets performed precisely to the documented specifications and contains clearly defined inputs with the results which are expected.

(II) INTEGRATION TESTING

These tests are designed to test the integrated software items to determine whether if they really execute as a single program or application. The testing is event driven and thus is concerned with the basic outcome of field. The Integration tests demonstrate that the components were individually satisfaction, as already represented by successful unit testing, the components are apt and fine. This type of testing is specially aimed to expose the issues that come-up by the component's combination.

(III) FUNCTIONAL TESTING

The functional tests help in providing the systematic representation that functions tested are available and specified by technical requirement, documentation of the system and the user manual.

(IV) SYSTEM TESTING

System testing, as the name suggests, is the type of testing in which ensure that the software system meet the business requirements and aim. Testing of the configuration is taken place here to ensure predictable result and thus analysis of it. System testing is relied on the description of process and its flow, stressing on pre driven process and the points of integration.

(V) WHITE BOX TESTING

The white box testing is the type of testing in which the internal components of the system software is open and can be processed by the tester. It is therefore a complex type of testing process. All the data structure, components etc. are tested by the tester himself to find out a possible bug or error. It is used in situation in which the black box is incapable of finding out a bug. It is a complex type of testing which takes more time to get applied.

(VI) BLACK BOX TESTING

The black box testing is the type of testing in which the internal components of the software is hidden and only the input and output of the system is the key for the tester to find out a bug. It is therefore a simple type of testing. A programmer with basic knowledge can also process this type of testing. It is less time consuming as compared to the white box testing. It is very successful for software which are less complex are straight-forward in nature. It is also less costly than white box testing.

(V) ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

4.5 IMPLEMENTATION PLAN

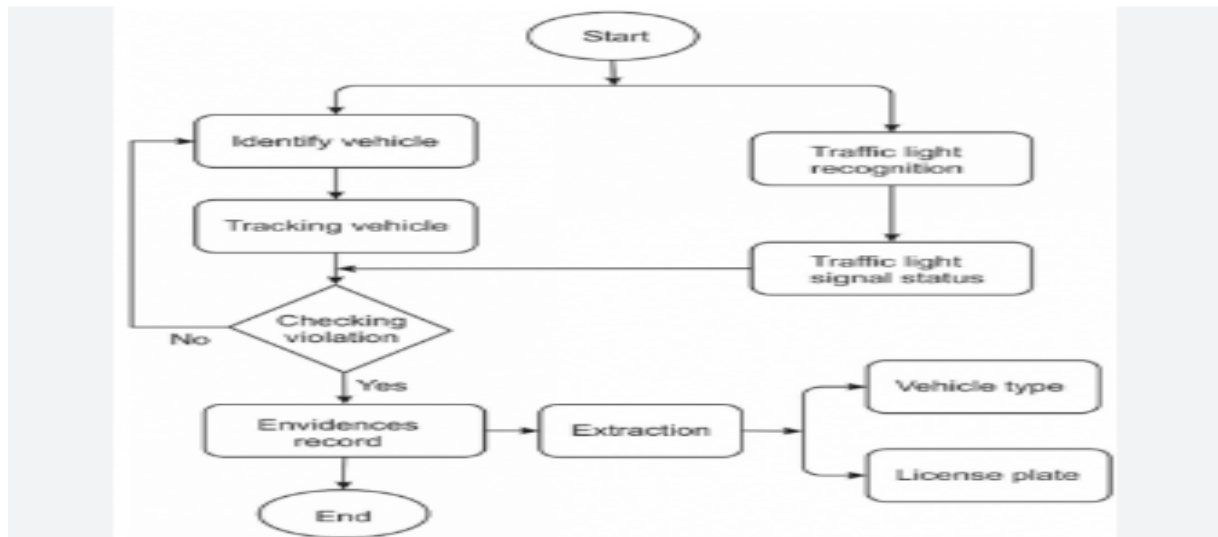


fig-4.2 Implementation plan of license plate

4.5.1 IDENTIFY VEHICLE:

The module uses computer vision algorithms and techniques to identify and locate vehicles within an image or video frame. It detects the presence of vehicles regardless of their orientation, size, or location within the scene.

4.5.2 TRAFFIC LIGHT RECOGNITION:

The Traffic Light Recognition Module serves as a crucial component in traffic management and safety systems, enabling accurate detection and interpretation of traffic signals for efficient traffic flow and safer transportation networks.

4.5.3 TRACKING VEHICLE:

The Tracking Vehicle Module is a fundamental component within systems designed for traffic monitoring, surveillance, or transportation applications. It's responsible for monitoring and tracing the movement of vehicles over time within a given area or along specific routes.

4.5.4 CHECKING VIOLATION:

The module utilizes specialized algorithms and computer vision techniques to analyze video streams or images captured by surveillance cameras to identify potential traffic rule violations.

4.5.5 EVIDENCE RECORD:

Evidence Record Module is a crucial component within traffic management and law enforcement systems. It is responsible for securely storing, managing, and organizing evidence related to traffic violations or incidents captured by surveillance cameras or other monitoring devices

4.5.6 PENALTY INFORMATION DISPLAY:

This Module provides a clear and comprehensive presentation of penalty amounts associated with different vehicle categories, enabling informed decision-making and promoting compliance with traffic regulations.

4.6 BLOCK DIAGRAM:

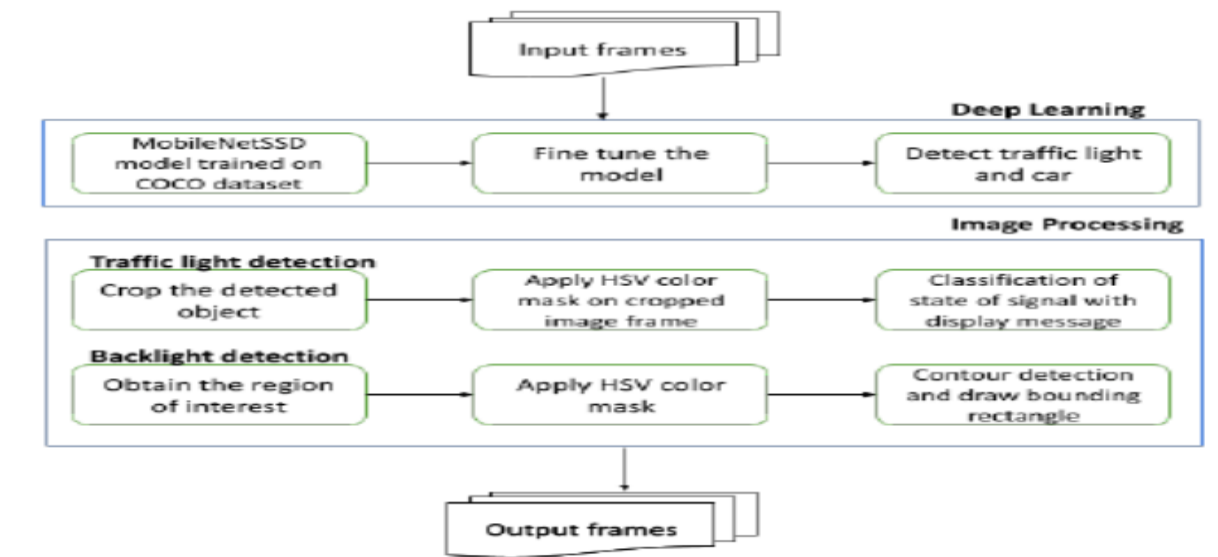


fig-4.3 Block Diagram

The block diagram illustrates a robust automated red light violation detection system employing the MobileNet SSD (Single Shot Multibox Detector) algorithm. The system begins with input sources comprising live video feeds obtained from strategically positioned cameras installed at traffic intersections. These video streams undergo preprocessing, involving noise reduction, stabilization, and frame normalization to ensure optimal input for subsequent processing stages. The MobileNet SSD algorithm,

renowned for its efficiency and accuracy in real-time object detection, forms the core of the system. It performs object detection and localization, identifying crucial elements such as vehicles and traffic signals within the video frames in real-time. Specifically

trained models within MobileNet SSD facilitate precise identification and tracking of objects, allowing for accurate traffic signal recognition and vehicle movement analysis.

Following object detection, the system focuses on traffic signal recognition, distinguishing and classifying traffic signal states, particularly detecting instances of red-light signals. Simultaneously, the system tracks the movements of detected vehicles, monitoring their positions and trajectories. A critical aspect involves temporal analysis, enabling the system to understand the sequence of vehicle movements concerning traffic signal states, differentiating between lawful actions (such as a permissible right turn on red) and actual red-light violations. The violation determination module utilizes this contextual analysis to make decisions regarding violations. Once a potential red-light violation is identified, the system triggers the output and alert system, flagging the violation and generating alerts for traffic authorities or control centers in real-time. These alerts facilitate immediate intervention or action by law enforcement agencies. Additionally, an optional feedback loop exists for data logging and analysis, recording detected violations and system performance metrics. This feedback mechanism aids in continuous system improvement, providing insights for model refinement and enhancing the system's accuracy and efficacy over time.

Moreover, the system's architecture is flexible, allowing deployment either on edge devices or centralized servers based on infrastructure requirements, offering adaptability and scalability for various operational environments. Overall, this block diagram encapsulates a comprehensive automated red light violation detection system, leveraging the strengths of MobileNet SSD for accurate object detection, signal recognition, and violation determination, ultimately contributing to enhanced traffic safety and regulatory enforcement at intersections.

GOALS:

ENHANCE TRAFFIC SAFETY:

The primary goal is to significantly reduce the number of red-light violations at intersections, ultimately contributing to a safer traffic environment and a decrease in accidents caused by disobedience of traffic signals.

ENFORCE TRAFFIC REGULATIONS:

The system aims to strengthen traffic rule adherence by efficiently identifying and reporting instances of red-light violations, thereby promoting responsible driving behavior and improving overall traffic discipline.

MINIMIZE HUMAN ERROR:

By automating the red-light violation detection process, the system seeks to minimize the likelihood of human error in monitoring traffic intersections, ensuring a more consistent and reliable method for enforcing traffic regulations.

IMPROVE EFFICIENCY IN LAW ENFORCEMENT:

The goal is to assist law enforcement agencies by providing real-time alerts and accurate data regarding red light violations, streamlining their intervention process and optimizing resource allocation for traffic management.

REDUCE TRAFFIC CONGESTION:

Decreasing red light violations can potentially lead to smoother traffic flow at intersections, reducing congestion and enhancing the overall efficiency of urban transportation networks.

FACILITATE TECHNOLOGICAL ADVANCEMENTS:

Through the deployment of advanced algorithms like MobileNet SSD, the project aims to showcase the capabilities and potential of cutting-edge technologies in addressing societal challenges related to traffic safety and regulation.

CONTINUOUS IMPROVEMENT AND ADAPTABILITY:

The project seeks to establish a framework for ongoing refinement and adaptation, aiming to continuously improve the accuracy and reliability of the system while ensuring its adaptability to diverse traffic scenarios and evolving technological landscapes.

COMPLIANCE AND ETHICAL CONSIDERATIONS:

Ensuring compliance with privacy regulations and addressing ethical considerations regarding surveillance technologies stands as a fundamental goal, ensuring the system's deployment aligns with legal and ethical standards while maintaining public.

COST-EFFECTIVENESS AND SCALABILITY:

Striving for a cost-effective solution that can be scaled across multiple intersections or urban areas, ensuring wider implementation and impact within budget constraints.

PUBLIC AWARENESS AND EDUCATION:

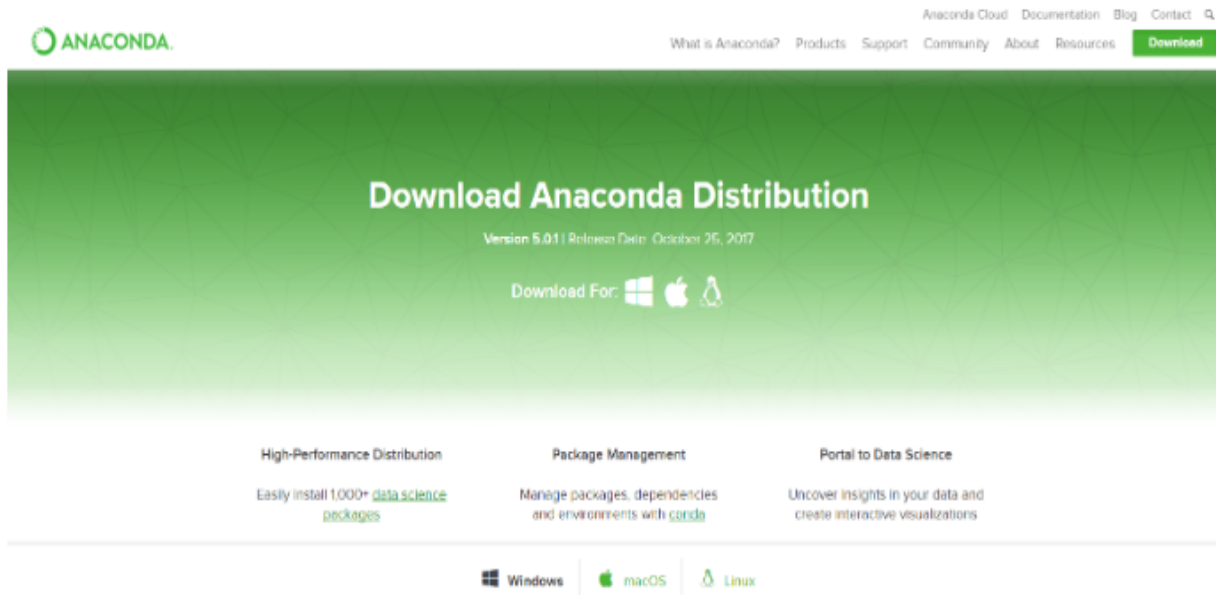
Alongside enforcement, the project aims to raise public awareness about the importance of adhering to traffic signals, promoting educational campaigns to foster a culture of responsible driving and traffic safety.

4.7 SOFTWARE DESCRIPTION**INSTALLING ANACONDA ON WINDOWS****4.7.1 DOWNLOADING AND INSTALLATION STEPS FOR INSTALLING ANACONDA ON WINDOWS**

Follow the steps below to install the Anaconda distribution of Python on Windows.

STEPS:

1.DOWNLOAD THE ANACONDA INSTALLER.



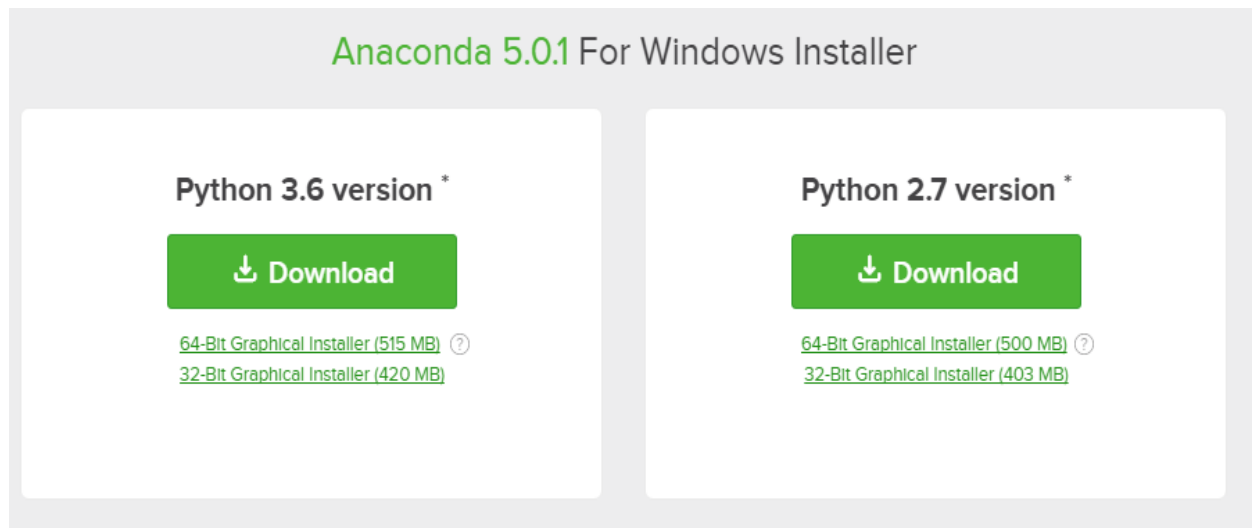
SELECT WINDOWS

Select Windows where the three operating systems are listed.



DOWNLOAD

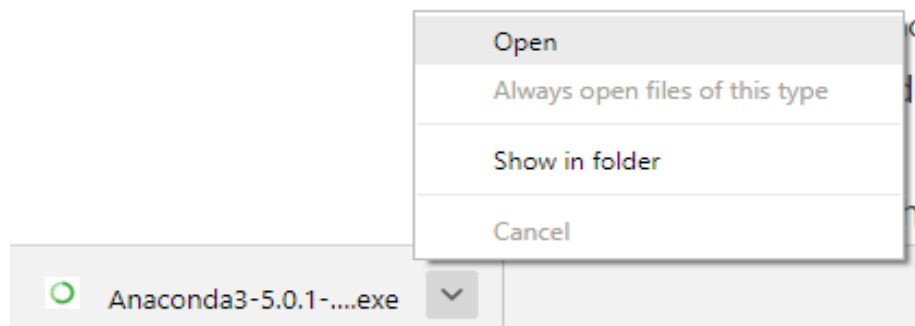
Download the most recent Python 3 release. At the time of writing, the most recent release was the Python 3.6 Version. Python 2.7 is legacy Python. For problem solvers, select the Python 3.6 version. If you are unsure if your computer is running a 64-bit or 32-bit version of Windows, select 64-bit as 64-bit Windows is most common.



2. DOUBLE CLICK THE INSTALLER TO LAUNCH.

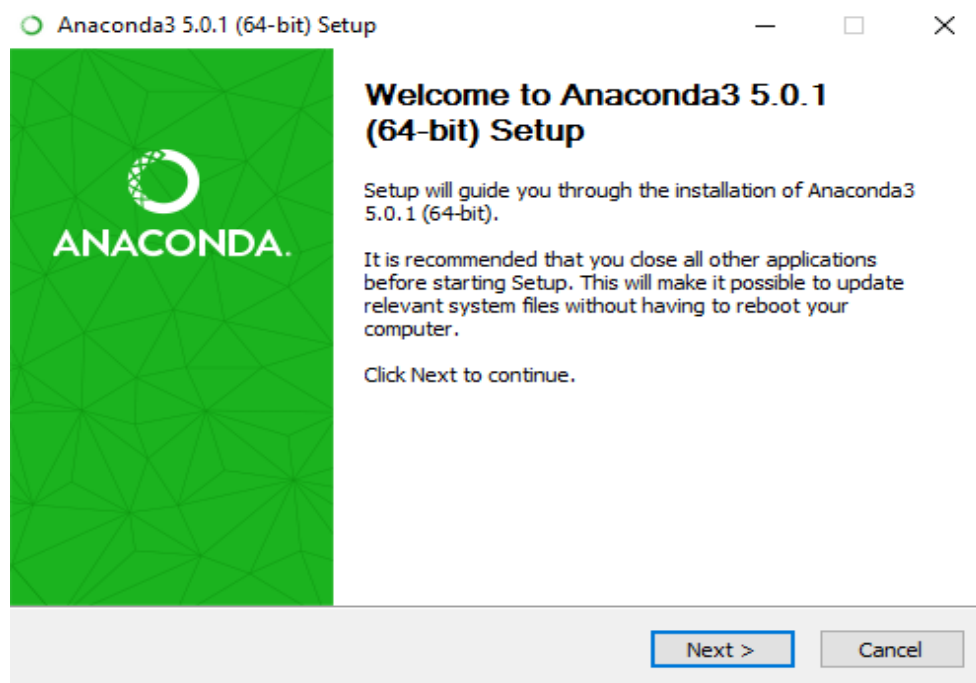
OPEN AND RUN THE INSTALLER

Once the download completes, open and run the .exe installer



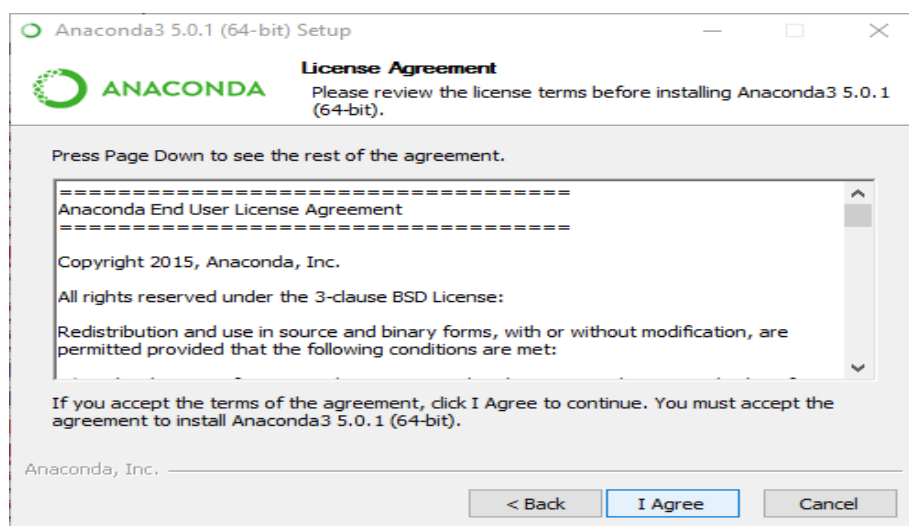
3. CLICK NEXT.

At the beginning of the install, you need to click **Next** to confirm the installation.



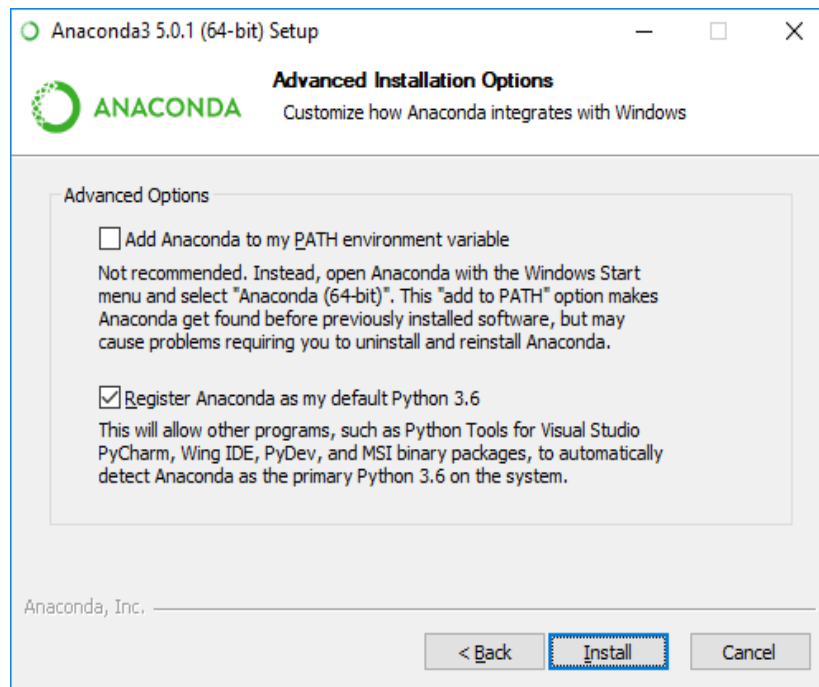
4. Read the licensing terms and click “I Agree”.

Then agree to the license.

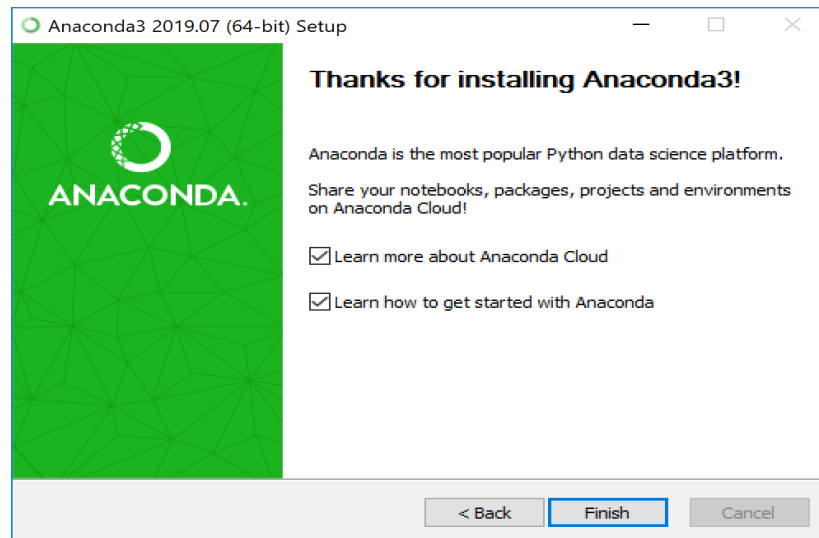


4. Select an install for “Just Me” unless you’re installing for all users(which requires Windows Administrator privileges) and click Next.

At the Advanced Installation Options screen, I recommend that you **do not check** "Add Anaconda to my PATH environment variable"



5. After a successful installation you will see the “Thanks for installing Anaconda” dialog box:



If you wish to read more about Anaconda Cloud and how to get started with Anaconda, check the boxes “Learn more about Anaconda Cloud” and “Learn how to get started with Anaconda”. Click the Finish button.

- Orange 3 App

- RStudio

VERIFYING YOUR INSTALLATION

You can verify the successful installation and functionality of Anaconda by checking for its presence and ensuring proper operation of either Anaconda Navigator or using the conda command-line interface.

Anaconda Navigator, a user-friendly graphical interface is an integral component that comes automatically installed alongside Anaconda. If the Navigator does not open after installation, please refer to our provided help resources for troubleshooting and guidance.

WINDOWS:

Begin by accessing the system's Start menu and locating Anaconda Navigator through the search or selection options available.

MACOS:

Initiate by opening Launchpad and then opt for Anaconda Navigator. Alternatively, use Cmd+Space to trigger Spotlight Search and enter "Navigator" to launch the program.

LINUX:

Refer to the subsequent section for instructions.

GETTING STARED WITH NAVIGATOR

The following applications are available by default in Navigator:

- Jupiter
- Jupiter Notebook
- Spyder
- VSCode
- Glueviz

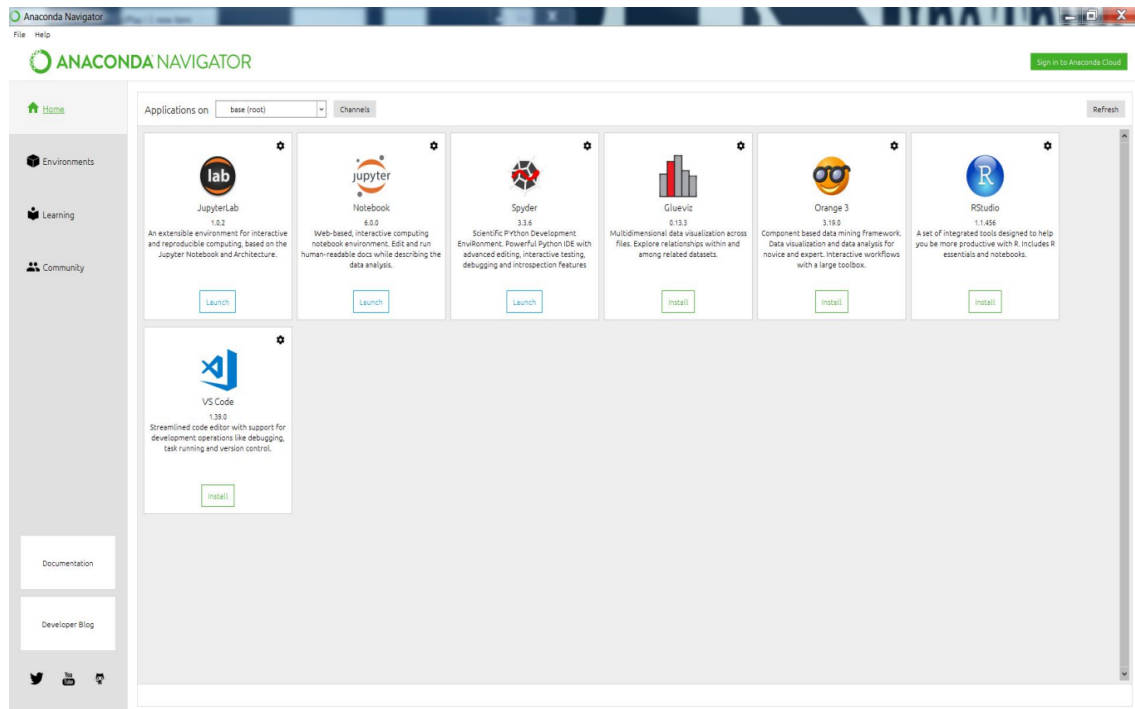


Fig 4.4 Anaconda Dashboard

SQLITE

SQLite differs significantly from client/server SQL database engines like MySQL, Oracle, PostgreSQL, or SQL Server since it aims to address a distinct set of challenges.

Client/server SQL database engines primarily focus on establishing a shared repository for enterprise data, emphasizing scalability, concurrency, centralization, and control. In contrast, SQLite is geared toward furnishing local data storage tailored for individual applications and devices. SQLite prioritizes cost-efficiency, performance, reliability, autonomy, and simplicity.

Integration of SQLite3 with Python is facilitated through the sqlite3 module developed by Gerhard Haring. This module provides an SQL interface that adheres to the DB-API 2.0 specification outlined in PEP 249. Importantly, there's no separate installation required for this module since it comes inherently bundled with Python versions 2.5.x and onwards.

4.7.2 PYTHON SQLITE3 MODULE APIS

The sqlite3 module provides essential routines to fulfill your needs for working with an SQLite database from your Python program. For more advanced or intricate applications, it is recommended to explore the official documentation of the Python sqlite3 module.

SQLITE3.CONNECT(DATABASE [TIMEOUT, OTHER OPTIONAL ARGUMENTS])

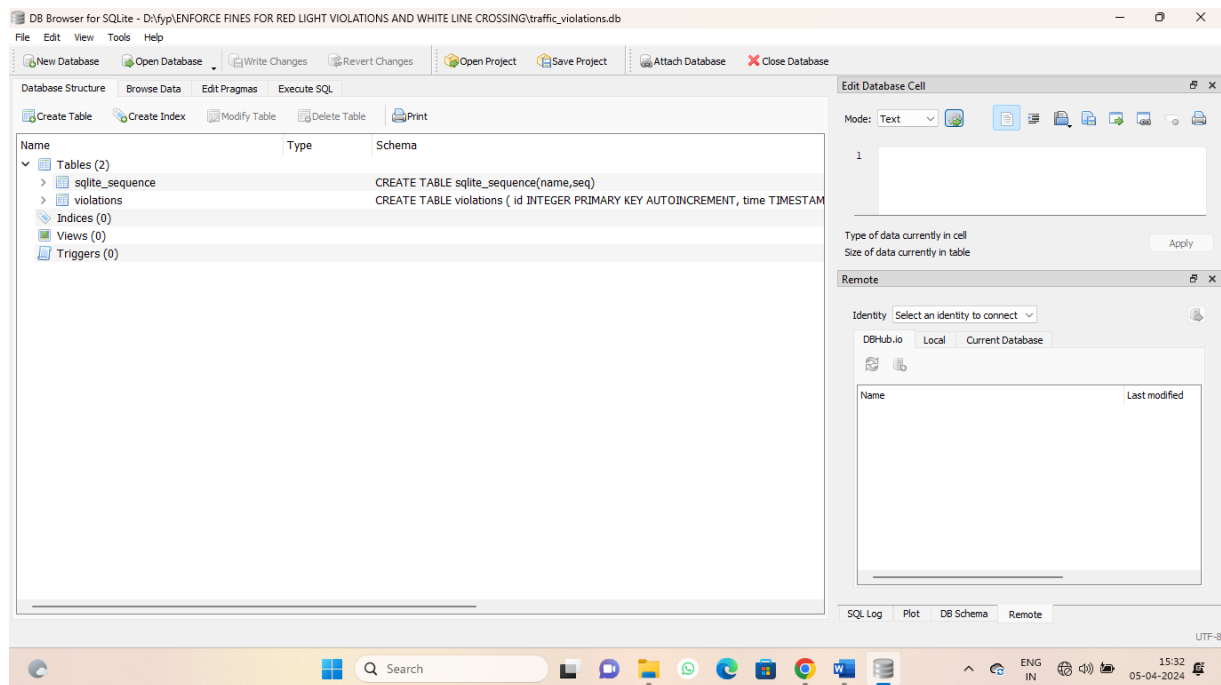


Fig 4.5 Sqlite Database

The sqlite3 module provides essential routines to fulfill your needs for working with an SQLite database from your Python program. For more advanced or intricate applications, it is recommended to explore This API initiates a connection to the SQLite database file. You can utilize "memory:" to establish a connection to a database residing in the computer's RAM rather than on the disk. Upon successful opening of the database, it returns a connection object.

In instances where a database is accessed by multiple connections and a process modifies the database, the SQLite database becomes locked until that transaction is committed. The timeout parameter designates the duration for which the connection

should await the lock to be released before raising an exception. The default timeout parameter is 5.0 (five seconds).

If the specified database name does not exist, this call will create the database. You have the option to define the filename with the required path if you wish to create a database in a location other than the current directory. ^e the official documentation of the Python sqlite3 module.

CONNECTION.CURSOR([CURSORCLASS])

This routine generates a cursor, crucial for your database operations in Python. If desired, it can accept a single optional parameter cursorClass. If provided, this parameter should be a customized cursor class that extends sqlite3.

CONNECTION.CLOSE()

This method concludes the database connection. It's important to note that it does not automatically trigger commit(). If you close the database connection without preceding it with commit(), any unsaved changes will be lost.

CONNECT TO DATABASE

Following Python code shows how to connect to an existing database. If the database does not exist, then it will be created and finally a database object will be returned.

```
#!/usr/bin/python
import sqlite3
conn = sqlite3.connect('test.db')
print "database connection successfully"
```

In this scenario, you also have the option to designate the database name as "memory:" to create a database in RAM. To proceed, execute the code provided in a file named sqlite.py and follow the instructions below. Adjust the path as needed to suit your requirements. If the database is successfully created, the following message will be displayed. Open database successfully

Functional tests serve to systematically demonstrate that the examined functions conform to the technical requirements outlined in the system's documentation and user manual, ensuring their availability and accurate specification.

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 DEVELOPMENT AND DEPLOYMENT SETUP

DATA COLLECTION:

Gather a dataset of images or videos containing instances of vehicles violating red lights or crossing white lines. Annotate these instances with bounding boxes indicating the location of the violations.

TRAINING DATA PREPARATION:

Preprocess the dataset by resizing images, normalizing pixel values, and generating ground truth annotations for training.

MODEL SELECTION:

Choose the MobileNet SSD architecture as your object detection model due to its lightweight nature and suitability for real-time applications on mobile devices.

FINE-TUNING:

Fine-tune the pre-trained MobileNet SSD model on your dataset of red light violations and white line crossings. This involves adjusting the model's parameters to improve its performance on your specific task.

EVALUATION:

Evaluate the fine-tuned model on a separate validation dataset to assess its accuracy, precision, recall, and other performance metrics.

DEPLOYMENT:

Integrate the fine-tuned MobileNet SSD model into a real-time monitoring system capable of detecting red light violations and white line crossings from live camera feeds or recorded video footage.

VIOLATION DETECTION:

Develop algorithms to process the model's output and identify instances of red-light violations and white line crossings based on the detected bounding boxes.

ENFORCEMENT MECHANISM:

Implement mechanisms to enforce fines for detected violations, such as generating automated alerts to law enforcement agencies or issuing citations to vehicle owners.

USER INTERFACE:

Create a user interface for monitoring and managing the enforcement system, allowing authorized personnel to review violation records, adjust system parameters, and generate reports.

TESTING AND VALIDATION:

Conduct thorough testing and validation of the entire system to ensure its reliability, accuracy, and compliance with legal regulations.

DEPLOYMENT AND MAINTENANCE:

Deploy the system in relevant locations such as intersections or roadways where red light violations and white line crossings are common. Continuously monitor and maintain the system to address any issues and adapt to changing conditions.

5.2 ALGORITHMS**5.2.1 MOBILENET SSD :**

(Single Shot MultiBox Detector) is an object detection algorithm that combines the MobileNet architecture for feature extraction with a single-shot detection framework. Here's a breakdown of its components and how it works:

5.2.2 MOBILE NET:

MobileNet is a lightweight deep neural network architecture designed for mobile and embedded vision applications. It utilizes depthwise separable convolutions to reduce the number of parameters and computational cost while maintaining good accuracy.

5.2.3 SINGLE SHOT MULTIBOX DETECTOR (SSD):

SSD is a popular object detection algorithm that performs both localization (detecting where objects are in an image) and classification (assigning labels to those objects) in a single forward pass of the network. It divides the input image into a grid of cells and predicts multiple bounding boxes and class scores for each cell.

5.2.4 BOUNDING BOX PREDICTION:

For each grid cell, SSD predicts multiple bounding boxes with different aspect ratios and scales. These bounding boxes are parameterized by their coordinates (x, y, width, height).

5.2.5 CLASS PREDICTION:

Along with each bounding box, SSD predicts the probability scores for each class (e.g., person, car, dog, etc.) present in the bounding box.

5.2.6 FEATURE EXTRACTION:

The MobileNet backbone is used to extract features from the input image. These features are then used for both localization and classification tasks.

5.2.7 MULTISCALE FEATURE MAPS:

SSD utilizes feature maps of different resolutions to detect objects of various sizes. This allows it to detect both small and large objects efficiently.

5.2.8 DEFAULT BOXES (ANCHOR BOXES):

SSD predicts bounding boxes relative to a set of default boxes (also known as anchor boxes) at each location in the feature map. These default boxes have predefined aspect ratios and scales to cover a wide range of object sizes and shapes.

5.2.9 LOSS FUNCTION:

SSD uses a combination of localization loss (e.g., smooth L1 loss) and classification loss (e.g., softmax loss) to train the network. The loss function penalizes errors in both bounding box predictions and class predictions.

5.2.10 Non-maximum Suppression (NMS):

After predictions are made, SSD applies non-maximum suppression to remove redundant bounding boxes with high overlap. This ensures that each object is detected only once.

CHAPTER 6

CONCLUSION & RESULTS

6.1 RESULT

In conclusion, the proposed system for real-time traffic signal violation detection using artificial intelligence and machine learning offers a promising solution for enhancing road safety and traffic management. By leveraging advanced computer vision algorithms and deep learning techniques, the system can accurately identify and classify violations such as red-light running, illegal turns, and speeding through intersections. Through the integration of modules such as data acquisition, preprocessing, object detection, machine learning model training, real-time detection, and alerting, the system provides a comprehensive approach to detecting and responding to traffic infractions in real-time. However, it is essential to acknowledge the challenges and limitations of the proposed system, including the need for robust data collection processes, ongoing model training and optimization, and integration with existing traffic management infrastructure. Addressing these challenges requires collaboration between stakeholders, continuous monitoring of system performance, and adaptation to evolving traffic patterns and regulatory requirements. Overall, the proposed system represents a significant advancement in traffic safety technology, offering a scalable and effective solution for real-time traffic signal violation detection. With further research, development, and deployment, the system has the potential to make significant contributions to improving road safety and enhancing the overall transportation experience for communities worldwide.

Moreover, the architecture showcases deployment flexibility, allowing for edge or centralized processing, seamless interfaces with cameras, and robust mechanisms for data storage, administration, and control. The system's optional feedback loop for continuous improvement underscores its commitment to refining accuracy and adaptability over time.

This proposed system represents a significant step towards a technologically advanced and efficient method for enforcing traffic regulations. By amalgamating cutting-edge algorithms with real-time surveillance, it aims to significantly reduce red light violations, thereby fostering a safer and more disciplined traffic environment. Its adaptability,

scalability, and potential for ongoing enhancement make it a promising contribution to traffic management systems, aligning with the pursuit of safer and more efficient urban transportation networks.

In essence, the proposed automated red light violation detection system not only embodies technological innovation but also holds the potential to positively impact public safety, making strides towards a more secure and orderly urban landscape.

6.2 CONCLUSION & FUTURE WORK

In envisioning the future of real-time traffic signal violation detection, leveraging artificial intelligence and machine learning stands as a transformative approach poised to revolutionize road safety and traffic management. Through continuous refinement of detection algorithms and integration with advanced sensor technologies, the system holds the potential to significantly enhance detection accuracy and reduce false positives. Furthermore, as smart city initiatives gain momentum, the integration of the proposed system with broader urban mobility platforms promises seamless interoperability and optimization of traffic flow across diverse transportation networks. However, realizing this vision necessitates concerted efforts in scaling the system for widespread deployment, ensuring robustness in challenging environmental conditions, and addressing ethical and legal considerations. By embracing these challenges and opportunities, the future of real-time traffic signal violation detection holds immense promise in fostering safer, more efficient, and sustainable urban environments for generations to come.

Moving forward, future endeavors for advancing the automated red light violation detection system could concentrate on several key areas. Firstly, refining and tailoring the MobileNet SSD algorithm specifically for traffic scenarios stands as a focal point to enhance accuracy and resilience, accommodating diverse environmental and traffic complexities. Incorporating multi-sensor fusion, such as integrating LiDAR or radar, could bolster the system's depth perception, especially in challenging conditions.

Behavioral analysis models and contextual understanding algorithms could improve the system's ability to discern deliberate violations from inadvertent mistakes, reducing false alarms. Moreover, expanding the system's capabilities beyond violation detection

to real-time traffic flow analysis could enable adaptive traffic management strategies. Addressing privacy concerns through privacy-preserving AI techniques and encryption methods ensures effective surveillance while maintaining individual privacy.

Additionally, continual validation with diverse datasets, public awareness campaigns, and collaboration on legal frameworks are crucial for system acceptance and compliance. Integration with broader smart city initiatives aligns the system with holistic urban planning efforts, fostering seamless urban transportation networks.

These future trajectories aim to elevate the system's accuracy, adaptability, and ethical considerations, positioning it as a pivotal element in ensuring safer and more efficient traffic intersections.

REFERENCE

- [a] Wang, X.L., Meng, L.M., Zhang, B.B., Lu, J.J., Du, K.L. (2013). A video-based traffic violation detection system. Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC), pp. 1191-1194. <https://doi.org/10.1109/MEC.2013.6885246>
- [b] The Police Department Guide to Safe Driving, www.defensivedriving.org/, accessed on June 2021.
- [c] Li, Y.B., Li, Z.H. (2011). Design and development of the monitoring system for traffic signal controller based on WEB technology. 2011 International Conference on Multimedia Technology, pp. 3920-3924. <https://doi.org/10.1109/ICMT.2011.6002741>
- [d] Azer, M.A., Elshafee, A. (2018). A real-time social network- based traffic monitoring & vehicle tracking system. 2018 13th International Conference on Conference on Computer Engineering and Systems (ICCES), pp.163-168. <https://doi.org/10.1109/ICCES.2018.8639287>
- [e] Wu, N., Fang, H.X. (2017). A novel traffic light recognition method for traffic monitoring systems. 2017 2nd Asia-Pacific Conference on Intelligent Robot System (ACIRS) 141-145. <https://doi.org/10.1109/ACIRS.2017.7986081>
- [f] Brasil, R.H., Machado, A.M.C. (2017). Automatic detection of red light running using vehicular cameras. in IEEE Latin America Transactions, 15(1): 81-86. <https://doi.org/10.1109/TLA.2017.7827891>
- [g] Prasantha, H.S. (2020). Traffic red light violation detection using image processing. International Journal 79 of Advances in Engineering and Management (IJAEM),2(5): 440-445.
- [h] Morera, A., Sánchez, A., Moreno, A.B., Sappa, A.D., Vélez, J.F. (2020). SSD vs. YOLO for detection of outdoor urban advertising panels under multiple variabilities. Sensors,20(16):4587. <https://doi.org/10.3390/s20164587>
- [i] Common objects in context. www.cocodataset.org, accessed on 6 June 2021.

[j] YOLO: Real-Time Object Detection, www.pjreddie.com/darknet/yolo, accessed on June 2021.

[k] LabelImg, an annotation tool. <https://github.com/tzutalin/labelImg>, accessed on 6 June 2021.

[l] PyTorch documentation. <https://pytorch.org>, accessed on 6 June 2021.

APPENDIX

A. SOURCE CODE

```
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile
import time
from imutils.video import WebcamVideoStream
from imutils.video import FPS
import imutils
from vehicle import vehicle
import PIL.Image as Image
from collections import defaultdict
from io import StringIO
from PIL import Image
import time
from multiprocessing.pool import ThreadPool
import threading
import time
import openalpr_api
from openalpr_api.rest import ApiException
import numpy as np
import cv2
import tkinter as tk
from PIL import Image
from PIL import ImageTk
import json
import re

if tf.__version__ < '1.4.0':
```

```
    raise ImportError('Please upgrade your tensorflow installation to v1.4.* or later!')
# This is needed to display the images.
```

```
# This is needed since the notebook is stored in the object_detection folder.
sys.path.append("..")
```

```
from utils import label_map_util
```

```
from utils import visualization_utils as vis_util
```

```
# What model to download.
```

```
MODEL_NAME = 'Cars'
```

```
# Path to frozen detection graph. This is the actual model that is used for the object
detection.
```

```
PATH_TO_CKPT = MODEL_NAME + '/output_inference_graph.pb'
```

```
# List of the strings that is used to add correct label for each box.
```

```
PATH_TO_LABELS = os.path.join('Cars', 'car_label_map.pbtxt')
```

```
NUM_CLASSES = 1
```

```
api = openalpr_api.DefaultApi()
```

```
secret_key = '3a0effff73919f898b69ac65a32dc12347769564'
```

```
country = 'us'
```

```
recognize_vehicle = 0
```

```
state = ''
```

```
return_image = 0
```

```
topn = 1
```

```
prewarp = ''
```

```

def getLicensePlateNumber(filer):
    try:
        js = api.recognize_file(filer, secret_key, country,
                                recognize_vehicle=recognize_vehicle, state=state, return_image=return_image,
                                topn=topn, prewarp=prewarp)

        js=js.to_dict()
        #js=list(str(js))
        X1=js['results'][0]['coordinates'][0]['x']
        Y1=js['results'][0]['coordinates'][0]['y']
        X2=js['results'][0]['coordinates'][2]['x']
        Y2=js['results'][0]['coordinates'][2]['y']
        img=cv2.imread(filer)
        rimg=img[Y1:Y2,X1:X2]
        frame3=rimg
        img3 = Image.fromarray(frame3)
        w,h=img3.size
        asprto=w/h
        frame3=cv2.resize(frame3,(150,int(150/asprto)))
        cv2image3 = cv2.cvtColor(frame3, cv2.COLOR_BGR2RGBA)
        img3 = Image.fromarray(cv2image3)
        imgtk3 = ImageTk.PhotoImage(image=img3)
        display4.imgtk = imgtk3 #Shows frame for display 1
        display4.configure(image=imgtk3)
        display5.configure(text=js['results'][0]['plate'])
    except ApiException as e:
        print("Exception: \n", e)

def matchVehicles(currentFrameVehicles,im_width,im_height,image):
    if len(vehicles)==0:
        for box,color in currentFrameVehicles:
            (y1,x1,y2,x2)=box

```



```

        ldistance = distance
        index = i

    diagonal=vehicles[index].diagonal

    if ldistance < diagonal:
        vehicles[index].updatePosition((x,y,w,h))
        vehicles[index].setCurrentFrameMatch(True)
    else:
        #blue for last position
        #cv2.circle(image,tuple(vehicles[index].points[-
1]),2,(255,0,0),4)

        #red for predicted point

        #cv2.circle(image,tuple(vehicles[index].getNext()),2,(0,0,255),2)
        #green for test point
        #cv2.circle(image,(X,Y),2,(0,255,0),4)

        #cv2.imshow('culprit',image)
        #time.sleep(5)
        #print(diagonal,'          ',ldistance)
        vehicles.append(vehicle((x,y,w,h)))

    for i in range(len(vehicles)):
        if vehicles[i].getCurrentFrameMatch() == False:
            vehicles[i].increaseFrameNotFound()

    #print(len(vehicles))
    pool = ThreadPool(processes=1)
    def checkRedLightCrossed(img):
        global count
        for v in vehicles:
            if v.crossed==False and len(v.points)>=2:

```

```

x1,y1=v.points[0]
x2,y2=v.points[-1]
if y1>yl3 and y2<yl3:
    count+=1
    v.crossed=True
    bimg=img[int(v.rect[1]):int(v.rect[1]+v.rect[3]),
int(v.rect[0]):int(v.rect[0]+v.rect[2])]
    frame2=bimg
    img2 = Image.fromarray(frame2)
    w,h=img2.size
    asprto=w/h
    frame2=cv2.resize(frame2,(250,int(250/asprto)))
    cv2image2 = cv2.cvtColor(frame2,
cv2.COLOR_BGR2RGBA)
    img2 = Image.fromarray(cv2image2)
    imgtk2 = ImageTk.PhotoImage(image=img2)
    display2.imgtk = imgtk2 #Shows frame for display 1
    display2.configure(image=imgtk2)
    #cv2.imshow('BROKE',bimg)
    name=r'D:\fyp\Traffic-Rule-Violation-Detection-System-
master\Rules_Breaker\culprit'+str(time.time())+'.jpg'
    cv2.imwrite(name,bimg)

tstop = threading.Event()
thread = threading.Thread(target=getLicensePlateNumber,
args=(name,))

thread.daemon = True
thread.start()

#cv2.imwrite('culprit.png',bimg)
#display3.configure(text=count)
import sqlite3

```

```

# Initialize SQLite database and create table if not exists
conn = sqlite3.connect('traffic_violations.db')
cursor = conn.cursor()
cursor.execute("""CREATE TABLE IF NOT EXISTS violations (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
                speed REAL,
                fine_amount TEXT,
                vehicle_image_path TEXT,
                number_plate TEXT
            )""")
conn.commit()

def save_violation(speed, fine_amount, vehicle_image_path, number_plate):

    # Insert violation details into the database
    cursor.execute("""INSERT INTO violations (speed, fine_amount,
    vehicle_image_path, number_plate)
                    VALUES (?, ?, ?,?)""", (speed, fine_amount,
    vehicle_image_path,number_plate))
    conn.commit()
from twilio.rest import Client

# Function to send SMS using Twilio
def send_sms(message_body, to_number):
    # Your Twilio Account SID and Auth Token
    account_sid = 'ACe3575d9835eeb239aaa0ff559de6b4b4'
    auth_token = 'c746351a2e7352110bbf29720cec3ba9'

    # Initialize Twilio client
    client = Client(account_sid, auth_token)

    try:

```

```

# Send SMS message
message = client.messages.create(
    body=message_body,
    from_='+19384441611',
    to="+919392484036"
)
print("SMS sent successfully!")
print("Message SID:", message.sid)
except Exception as e:
    print("Failed to send SMS:", str(e))

```

```

# Example usage of send_sms function
# You can call this function with appropriate message and recipient number
# send_sms("Hello from Twilio!", "+1234567890")

```

```

def checkSpeed(ftime, img, number_plate):
    for v in vehicles:
        if v.speedChecked == False and len(v.points) >= 2:
            x1, y1 = v.points[0]
            x2, y2 = v.points[-1]
            if y2 < y1 and y2 > y13 and v.entered == False:
                v.enterTime = ftime
                v.entered = True
            elif y2 < y13 and y2 > y15 and v.exited == False:
                v.exitTime = ftime
                v.exited = True
                v.speedChecked = True
                speed = 60 / (v.exitTime - v.enterTime)
                print("Speed:", speed, "Km/hr")
                bimg = img[int(v.rect[1]):int(v.rect[1] + v.rect[3]), int(v.rect[0]):int(v.rect[0]
+ v.rect[2])]
                frame2 = bimg
                img2 = Image.fromarray(frame2)

```

```

w, h = img2.size
asprto = w / h
frame2 = cv2.resize(frame2, (250, int(250 / asprto)))
cv2image2 = cv2.cvtColor(frame2, cv2.COLOR_BGR2RGBA)
img2 = Image.fromarray(cv2image2)
imgtk2 = ImageTk.PhotoImage(image=img2)
display2.imgtk = imgtk2
display2.configure(image=imgtk2)
display3.configure(text=str(speed)[:5] + 'Km/hr')

# Calculate fine amount
fine_amount = calculate_fine(speed)
print("Fine Amount:", fine_amount)
display3.configure(text=str(speed)[:5] + 'Km/hr' + "\nFine Amount: $" +
str(fine_amount))

# Save violation details to database
# Save violation details to database
save_violation(speed, fine_amount, 'culprit' + str(time.time()) + '.jpg',
number_plate)
if speed >=60:
    message_body = "Speeding violation detected! Speed: " + str(speed)
+ " Km/hr. Fine Amount: $" + str(fine_amount)
    to_number = "+919042615608" # Specify the recipient's phone
number
    send_sms(message_body, to_number)

if speed > 60:
    name = 'Rule Breakers/culprit' + str(time.time()) + '.jpg'
    cv2.imwrite(name, bimg)
    tstop = threading.Event()
    thread = threading.Thread(target=getLicensePlateNumber,

```

```

args=(name,))
        thread.daemon = True
        thread.start()

```

```

def calculate_fine(speed):
    if speed <= 40:
        return "No fine (within speed limit)"
    elif speed <= 60:
        return "$100 fine"
    elif speed <= 80:
        return "$200 fine"
    else:
        return "$500 fine and license suspension"

```

```

# Call checkSpeedAndFine function with appropriate parameters

```

```

detection_graph = tf.Graph()
with detection_graph.as_default():
    #od_graph_def = tf.GraphDef()
    od_graph_def = tf.compat.v1.GraphDef()
# with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
    with tf.compat.v2.io.gfile.GFile(PATH_TO_CKPT,'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name="")

```

```

label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories      =      label_map_util.convert_label_map_to_categories(label_map,

```

```

max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)

cap=cv2.VideoCapture('video7.avi') # 0 stands for very first webcam attach
filename="testoutput.avi"
codec=cv2.VideoWriter_fourcc('m','p','4','v')#fourcc stands for four character code
framerate=10
resolution=(640,480)

VideoFileOutput=cv2.VideoWriter(filename,codec,framerate, resolution)
vs = WebcamVideoStream(src='Set01_video01.mp4').start()
ret,imgF=cap.read(0)

imgF=Image.fromarray(imgF)
im_width, im_height = imgF.size
xl1=0
xl2=im_width-1
yl1=im_height*0.5
yl2=yl1
ml1=(yl2-yl1)/(xl2-xl1)
intcptl1=yl1-ml1*xl1

count=0
xl3=0
xl4=im_width-1
yl3=im_height*0.25
yl4=yl3
ml2=(yl4-yl3)/(xl4-xl3)
intcptl2=yl3-ml2*xl3

xl5=0
xl6=im_width-1
yl5=im_height*0.1
yl6=yl5

```

```

ml3=(y16-y15)/(x16-x15)
intcptl3=y15-ml3*x15
ret=True
start=time.time()
c=0
sesser=tf.compat.v1.Session(graph=detection_graph)
image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
# Each box represents a part of the image where a particular object was detected.
detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
# Each score represent how level of confidence for each of the objects.
# Score is shown on the result image, together with the class label.
detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')
detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')
num_detections = detection_graph.get_tensor_by_name('num_detections:0')

window = tk.Tk() #Makes main window
window.wm_title("T.M.S")
window.columnconfigure(0, {'minsize': 1020})
window.columnconfigure(1, {'minsize': 335})

frame=tk.Frame(window)
frame.grid(row=0,column=0,rowspan=5,sticky='N',pady=10)

frame2=tk.Frame(window)
frame2.grid(row=0,column=1)

frame3=tk.Frame(window)
frame3.grid(row=1,column=1)

frame4=tk.Frame(window)
frame4.grid(row=2,column=1)

frame5=tk.Frame(window)

```

```
frame5.grid(row=3,column=1)
```

```
frame2.rowconfigure(1, {'minsize': 250})
```

```
frame3.rowconfigure(1, {'minsize': 80})
```

```
frame4.rowconfigure(1, {'minsize': 150})
```

```
frame5.rowconfigure(1, {'minsize': 80})
```

```
vehicles=[]
```

```
import cv2
```

```
import pytesseract
```

```
# Set the path to the Tesseract executable
```

```
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'
```

```
# Load the pre-trained license plate detection cascade
```

```
plate_cascade = cv2.CascadeClassifier(cv2.data.harcascades +  
'haarcascade_russian_plate_number.xml')
```

```
# Function to detect license plate area in a frame
```

```
def detect_license_plate(frame):
```

```
    # Convert the frame to grayscale
```

```
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

```
    # Detect license plates in the frame
```

```
    plates = plate_cascade.detectMultiScale(gray, scaleFactor=1.1,  
minNeighbors=5, minSize=(25, 25))
```

```
    # Iterate through detected plates and draw rectangles
```

```
    for (x, y, w, h) in plates:
```

```
        cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)
```

```

# Crop the license plate area from the frame
plate_crop = gray[y:y+h, x:x+w]
# Apply OCR to the license plate area
custom_config = r'--oem 3 --psm 6'
text = pytesseract.image_to_string(plate_crop, config=custom_config)
# Print the detected text
print("license plate:",text)

def main(sess=sesser):
    "global masterframe
    global started"
    if True:
        fTime = time.time()
        _, image_np = cap.read(0)

        # image_np = imutils.resize(image_np, width=400)

        # Definite input and output Tensors for detection_graph

        # Expand dimensions since the model expects images to have shape: [1,
        None, None, 3]
        image_np_expanded = np.expand_dims(image_np, axis=0)
        # Actual detection.
        (boxes, scores, classes, num) = sess.run(
            [detection_boxes, detection_scores, detection_classes, num_detections],
            feed_dict={image_tensor: image_np_expanded})

        # Visualization of the results of a detection.
        img = image_np
        imgF, coords = vis_util.visualize_boxes_and_labels_on_image_array(
            image_np,
            np.squeeze(boxes),
            np.squeeze(classes).astype(np.int32),
            np.squeeze(scores),

```

```

        category_index,
        use_normalized_coordinates=True,
        line_thickness=2)

matchVehicles(coords, im_width, im_height, imgF)
checkRedLightCrossed(imgF)
detect_license_plate(imgF)
number_plate = ""
checkSpeed(fTime, img, number_plate) # Call checkSpeedAndFine with
fTime and img as parameters
for v in vehicles:
    if v.getTracking() == True:
        for p in v.getPoints():
            cv2.circle(image_np, p, 3, (200, 150, 75), 6)

cv2.line(image_np, (int(xl1), int(yl1)), (int(xl2), int(yl2)), (0, 255, 0), 3)
cv2.line(image_np, (int(xl3), int(yl3)), (int(xl4), int(yl4)), (0, 0, 255), 3)
cv2.line(image_np, (int(xl5), int(yl5)), (int(xl6), int(yl6)), (255, 0, 0), 3)
VideoFileOutput.write(image_np)
# print('yola')
frame = cv2.resize(image_np, (1020, 647))
cv2image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGBA)
img = Image.fromarray(cv2image)
imgtk = ImageTk.PhotoImage(image=img)
display1.imgtk = imgtk # Shows frame for display 1
display1.configure(image=imgtk)
window.after(1, main)

lbl1 = tk.Label(frame, text='Vehicle Detection And Tracking', font = "verdana 12
bold")
lbl1.pack(side='top')

lbl2 = tk.Label(frame2, text='Vehicle Breaking Traffic Rule', font = "verdana 10 bold")
lbl2.grid(row=0, column=0, sticky ='S', pady=10)

```

```
lbl3 = tk.Label(frame3,text='Veicle Speed',font = "verdana 10 bold")
lbl3.grid(row=0,column=0,sticky ='S',pady=10)
```

```
# lbl4 = tk.Label(frame4,text='Detected License Plate',font = "verdana 10 bold")
# lbl4.grid(row=0,column=0)
```

```
# lbl5 = tk.Label(frame5,text='Extracted License Plate Number',font = "verdana 10 bold")
# lbl5.grid(row=0,column=0)
```

```
display1 = tk.Label(frame)
display1.pack(side='bottom') #Display 1
```

```
display2 = tk.Label(frame2)
display2.grid(row=1,column=0) #Display 2
```

```
display3 = tk.Label(frame3,text="",font = "verdana 14 bold",fg='red')
display3.grid(row=1,column=0)
```

```
display4 = tk.Label(frame4)
display4.grid(row=1,column=0)
```

```
display5 = tk.Label(frame5,text="",font = "verdana 24 bold",fg='green')
display5.grid(row=1,column=0)
```

```
masterframe=None
```

```
started= False
```

```
def stream():
```

```
    global masterframe
```

```
    global started
```

```
    global c
```

```
    global tim
```

```

cap=cv2.VideoCapture('vid1.mp4')
while True:
    started, masterframe=cap.read()
    time.sleep(0.034)

"""thread = threading.Thread(target=stream)
thread.daemon = True
thread.start()"""
with detection_graph.as_default():
    with tf.compat.v1.Session(graph=detection_graph) as sess:
        sesser=sess
        main(sess) #Display
window.mainloop() #Starts GUI

```

B. SCREENSHOTS

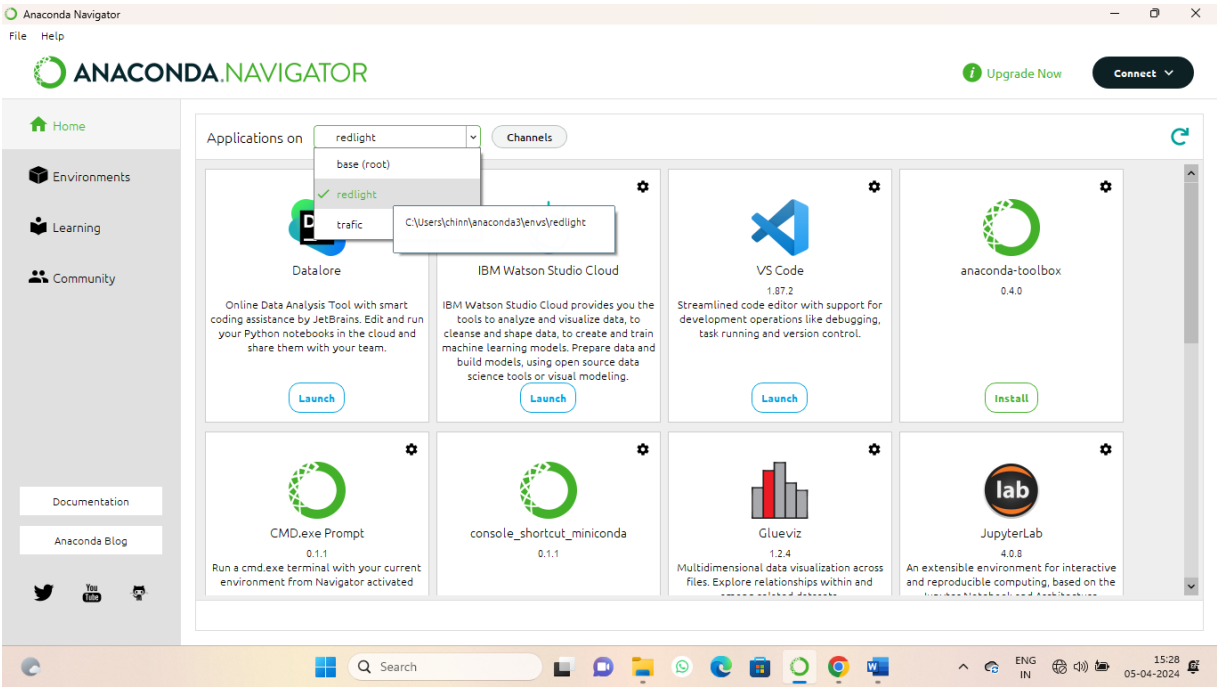


Fig-B.1 Anaconda Navigator for starting the redlight environment

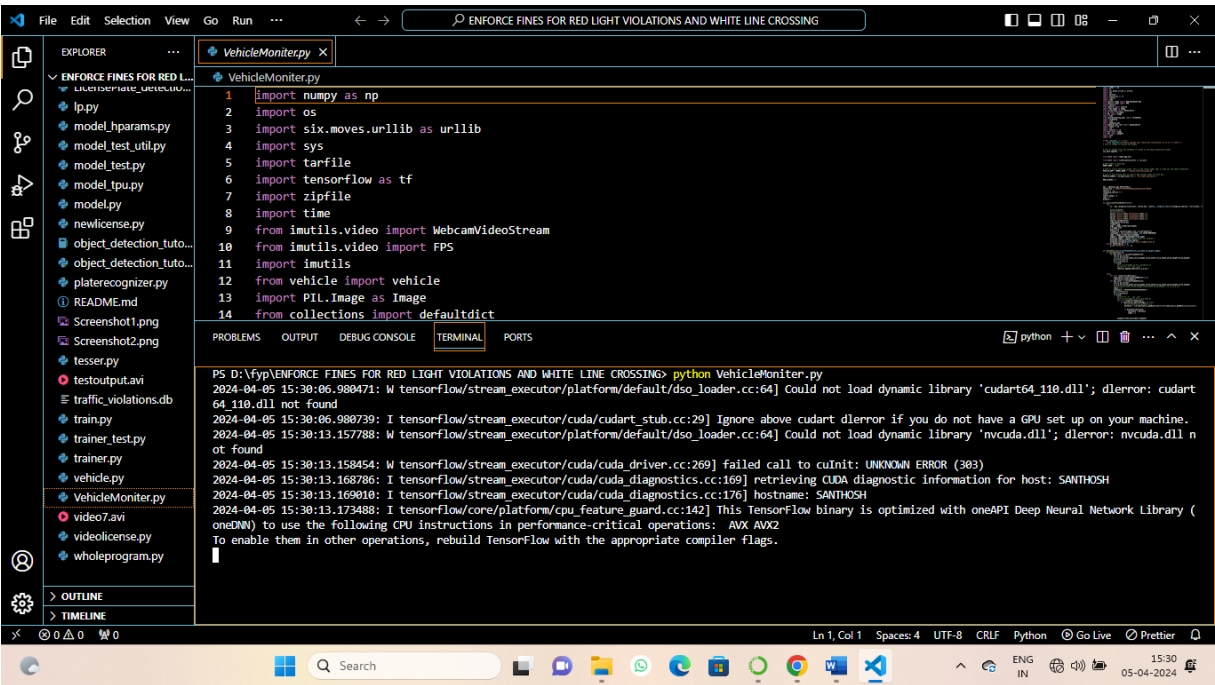


Fig-B.2 Implementation of Code

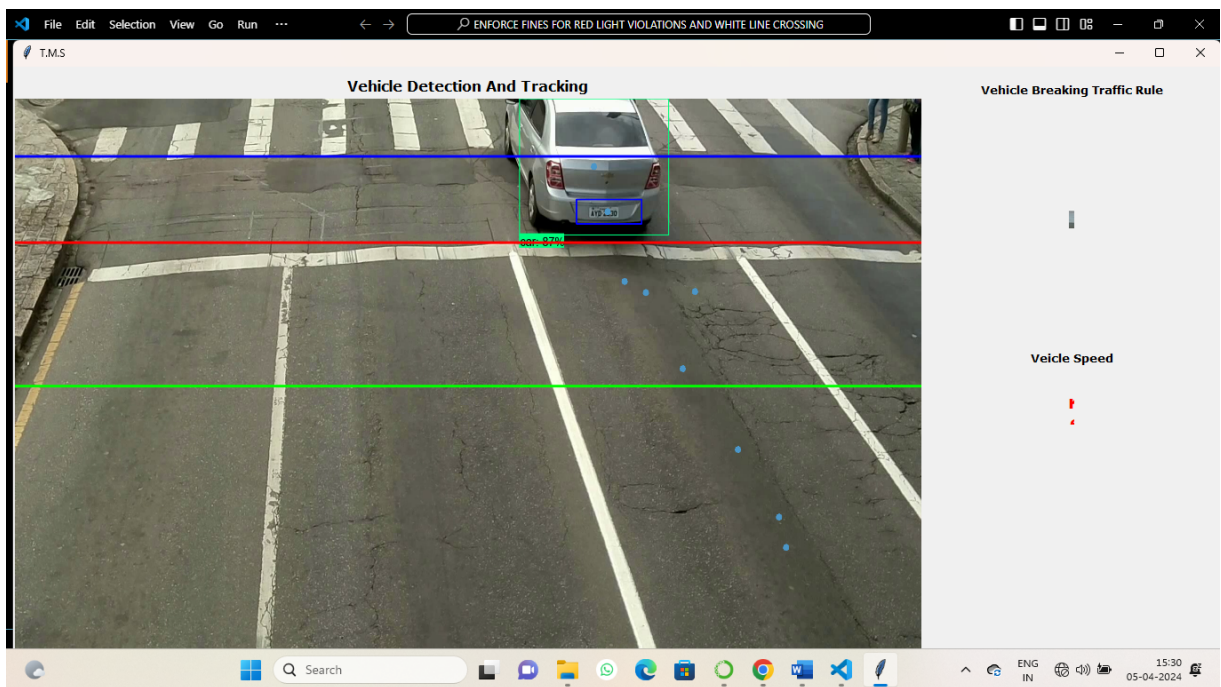


Fig-B.3 Identifying the vehicles

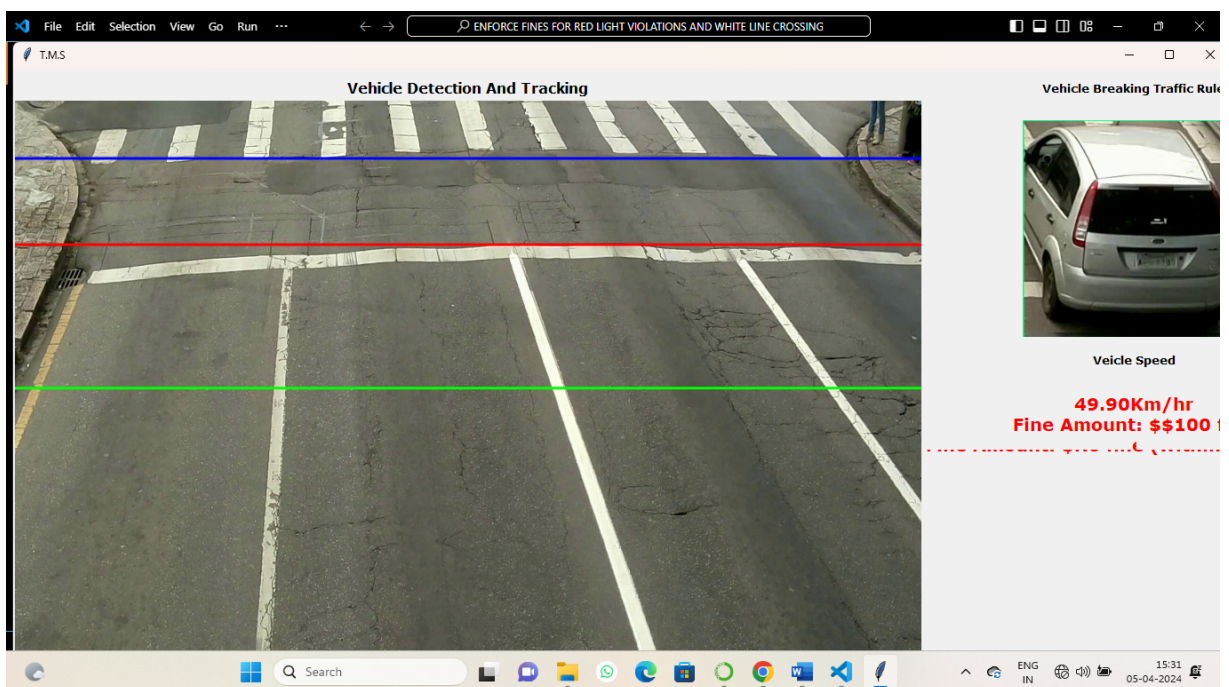


Fig-B.4 implementing the fine for the vehicle

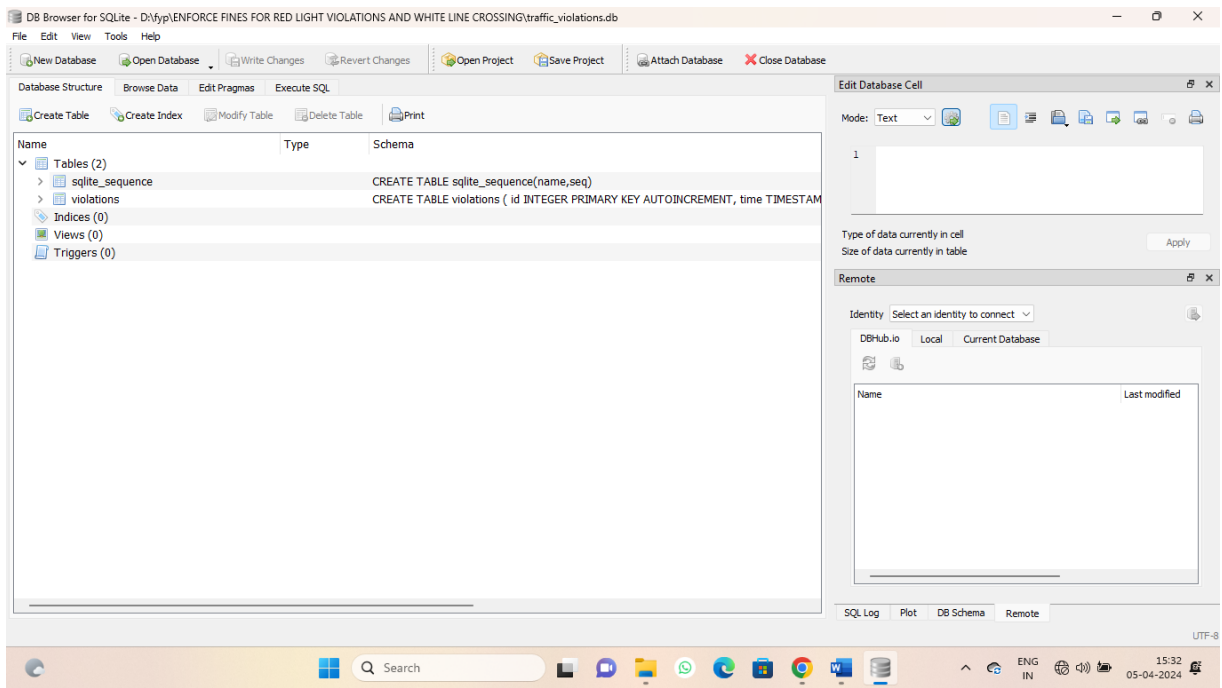


Fig-b.5 SQL DATABASE

DB Browser for SQLite - D:\fyp\ENFORCE FINES FOR RED LIGHT VIOLATIONS AND WHITE LINE CROSSING\traffic_violations.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

Table: violations Filter in any column

id	time	speed	fine_amount	vehicle_image_path	number_plate
1	2024-04-03 12:48:51	12.1331401351327	No fine (within speed limit)	culprit1712148531.8524091.jpg	NULL
2	2024-04-03 12:51:09	28.271587990007	No fine (within speed limit)	culprit1712148669.094549.jpg	
3	2024-04-03 12:51:17	30.6007013429213	No fine (within speed limit)	culprit1712148677.9823663.jpg	
4	2024-04-03 12:51:20	3.50436855860968e-08	No fine (within speed limit)	culprit1712148680.0800235.jpg	
5	2024-04-03 12:51:28	3.50436854052168e-08	No fine (within speed limit)	culprit1712148688.7892885.jpg	
6	2024-04-03 12:51:33	18.6873857556066	No fine (within speed limit)	culprit1712148693.8456855.jpg	
7	2024-04-03 12:51:50	13.377067709879	No fine (within speed limit)	culprit1712148710.1101482.jpg	
8	2024-04-03 12:51:53	13.8864198130347	No fine (within speed limit)	culprit1712148713.076348.jpg	
9	2024-04-03 12:51:59	38.5533521704681	No fine (within speed limit)	culprit1712148719.0465512.jpg	
10	2024-04-03 12:51:59	35.8933222350317	No fine (within speed limit)	culprit1712148719.7944887.jpg	
11	2024-04-03 12:51:59	3.50436847753117e-08	No fine (within speed limit)	culprit1712148719.8054893.jpg	
12	2024-04-03 12:52:05	15.492447948249	No fine (within speed limit)	culprit1712148725.0067835.jpg	
13	2024-04-03 12:52:07	66.0127703575774	\$200 fine	culprit1712148727.1470387.jpg	
14	2024-04-03 12:52:11	48.2716551178459	\$100 fine	culprit1712148731.1704047.jpg	
15	2024-04-03 12:52:18	118.103553221355	\$500 fine and license suspension	culprit1712148738.541339.jpg	
16	2024-04-03 12:52:25	28.640142502947	No fine (within speed limit)	culprit1712148745.0633414.jpg	
17	2024-04-03 12:52:28	21.1512189939712	No fine (within speed limit)	culprit1712148748.65944.jpg	
18	2024-04-03 12:52:50	37.5379939481755	No fine (within speed limit)	culprit1712148770.5077357.jpg	

Go to: 1

Edit Database Cell

Mode: Text

1

Type of data currently in cell: Text / Numeric
1 character(s)

Apply

Remote

Identity Select an identity to connect

DBHub.io Local Current Database

Name Last modified

SQL Log Plot DB Schema Remote

UTF-8

Fig-B.6 Fine's list

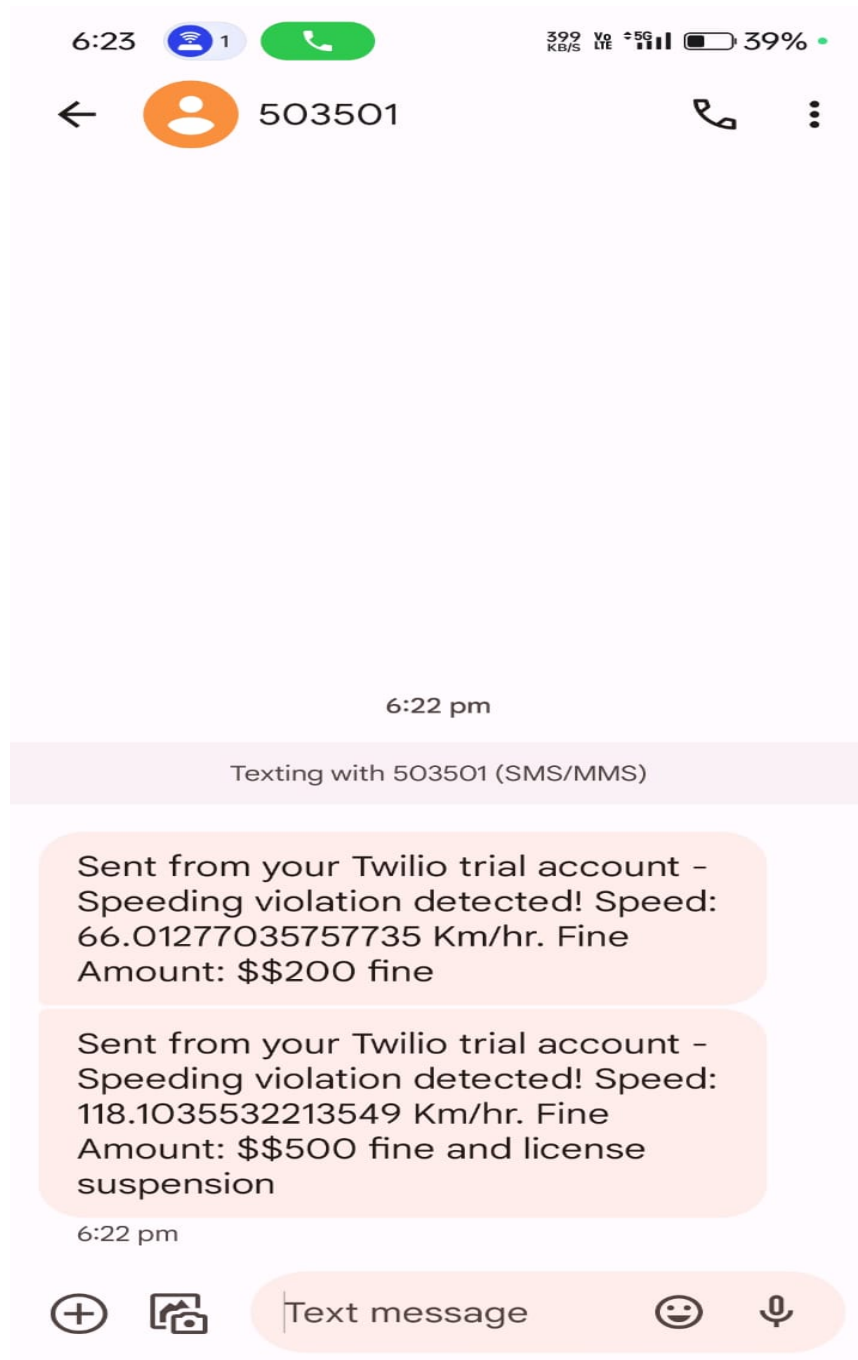


Fig-B.7 SMS for Fine

ENFORCE FINES FOR RED LIGHT VIOLATIONS AND WHITE LINE CROSSING

'Kattela Santhosh Reddy

**Dept. of Computer Science and Engineering
Sathyabama Institute of Science and
Technology
Chennai, Tamilnadu, India
Email: chinnaasanthosh1234@gmail.com*

'Karri Vishnu Sai Rama Reddy

*Dept. of Computer Science and Engineering
Sathyabama Institute of Science and
Technology
Chennai, Tamilnadu, India
Emailid : vishnusai1569@gmail.com*

'Dr. Nancy Noella R.

*Dept. of Computer Science and Engineering
Sathyabama Institute of Science and
Technology
Chennai, Tamilnadu, India
Email id: nancynoella.cse@sathyabama.ac.in*

Abstract— According to recent assessments, traffic offenses have mostly resulted in an increase in fatalities and injuries on Indian roadways. Because it is laborious to manually identify the automobiles that are in violation of the traffic laws, an automated computer vision-based object detection model was required for this task. This paper's main idea is to use a single video frame to identify several infractions. To perform various operations, the input video stream from the security camera is analyzed and annotated. The COCO dataset is utilized for red-light leaping, while the annotation of Google image photos creates the overboarding dataset. Tensorboard is used to visualize the results after the model has been trained. F-measure, P-measure, Precision, and Recall are the parameters that are used. Red light skipping has a 93% accuracy rate, while overboarding has a 0.5:0.95 mAP value. In order to identify different violations, this system makes the most of the video stream.

Keywords— *convolutional neural networks, support vector machines, MobileNet SSD algorithm, Human Activity Recognition*

1. INTRODUCTION

Purpose and Scope: ATLVD is a cutting-edge system engineered to automate the identification and enforcement of traffic violations, primarily focusing on instances where vehicles disobey traffic signals by crossing designated red lines at intersections. Leveraging advanced computer vision technologies, ATLVD integrates with surveillance cameras strategically placed at traffic junctions. This integration empowers the system to mimic the principles of Human Activity Recognition (HAR) by analyzing real-time video feeds to identify and document traffic violations. When a vehicle breaches the red signal line, ATLVD swiftly captures visual data, including high-resolution images or video clips, facilitating the identification and documentation of the violating vehicle.

Data Capture and Processing: The system employs sophisticated algorithms to extract relevant data, primarily focusing on the license plate information of the offending vehicle. This information forms the basis for subsequent enforcement actions. ATLVD goes beyond mere detection, automating the imposition of fines based on predetermined criteria linked to different vehicle categories. This automated process ensures a standardized and consistent approach to penalties for traffic violations. Much like HAR's impact in healthcare and life care services, ATLVD stands to revolutionize traffic management systems by streamlining law enforcement efforts and contributing significantly to road safety. Its implementation aims to instill a greater sense of compliance with traffic regulations among motorists.

Ongoing Challenges and Focus: The project faces challenges synonymous with video-based HAR systems, such as accurately distinguishing violations amidst diverse vehicle movements and varying environmental conditions. Addressing

these challenges remains a focal point for ongoing system enhancement and refinement. ATLVD serves as a cornerstone in the evolution of smart traffic management systems, with the potential for further advancements in automation, integration with broader traffic control networks, and enhanced accuracy in violation detection.

1.1 PROBLEM STATEMENT :

The problem statement for automatic traffic light violation detection using artificial intelligence involves creating a system that can accurately identify and flag instances where vehicles violate traffic light signals. This system needs to analyze real-time traffic footage or images captured by cameras installed at intersections, detect vehicles, recognize traffic lights, and determine whether a vehicle has disregarded the signal (such as running a red light or making an illegal turn). The goal is to develop an AI-powered solution that enhances traffic safety, reduces accidents, and helps enforce traffic regulations effectively.

1.1.1 EXISTING SYSTEM:

The existing automated red light violation detection system comprises a sophisticated integration of computer vision and machine learning technologies. Leveraging convolutional neural networks (CNNs) for object detection and recognition, the system efficiently identifies vehicles and traffic signals from real-time video feeds captured by strategically positioned cameras at intersections.

Through advanced image processing techniques, including semantic segmentation and feature extraction, it discerns crucial elements within the visual data, accurately distinguishing red light signals and vehicle movements. This information is further analyzed using support vector machines (SVMs) for classification, enabling swift identification of potential red light violations.

Additionally, the system harnesses sensor fusion, combining data from LiDAR and cameras for enhanced depth perception and precise object tracking, ensuring robust detection even in challenging environmental conditions. Overall, this existing system stands as a testament to the amalgamation of cutting-edge computer vision algorithms, machine learning models, and sensor technologies, exemplifying its prowess in augmenting traffic safety and enforcing regulatory compliance at traffic intersections.

1.1.2 PROPOSED SYSTEM

The proposed system integrates the MobileNet SSD algorithm, renowned for its efficiency in real-time object detection, particularly in resource-constrained environments, making it ideal for deployment in traffic surveillance scenarios. The system architecture consists of strategically positioned cameras at intersections, capturing live video feeds.

These feeds are processed through the MobileNet SSD framework, leveraging its lightweight architecture optimized for speed and accuracy in detecting objects. The MobileNet SSD algorithm excels in detecting various objects, including vehicles and traffic signals, in the captured video frames. Specifically trained models within MobileNet SSD allow the system to identify and localize crucial elements, such as vehicles and traffic lights, in real-time with high precision. Upon detecting traffic signals, the system focuses on recognizing red light states and tracking vehicles' movements within the vicinity. This enables it to analyze the interaction between vehicles and traffic signals, identifying instances of potential red light violations. The MobileNet SSD's ability to process video streams efficiently allows for swift and continuous monitoring of traffic intersections. Moreover, to enhance the system's accuracy and robustness, it incorporates temporal analysis by tracking the sequence of events. This temporal aspect aids in differentiating legitimate scenarios, such as a safe right turn on red, from actual red light violations.

The proposed system aims to address the computational efficiency and accuracy required for real-time application in traffic surveillance. Leveraging the MobileNet SSD algorithm's lightweight architecture ensures swift processing, making it adaptable for deployment on edge devices or within existing infrastructure. However, the proposed system may require additional fine-tuning and training of the MobileNet SSD model specifically for traffic scenarios to optimize detection accuracy. Continuous validation and refinement through real-world testing will be crucial for ensuring the system's reliability and minimizing false positives or negatives. Overall, the proposed system utilizing the MobileNet SSD algorithm holds promise in providing a cost-effective, efficient, and accurate solution for automated red light violation detection, contributing significantly to traffic safety and regulatory enforcement at intersections.

2. LITERATURE SURVEY

"Automated Red Light Violation Detection Using Artificial Intelligence: A Comprehensive Review, Ethan Williams." This literature survey delves into the advancements and methodologies employed in utilizing artificial intelligence for automated red light violation detection. It examines various machine learning algorithms, computer vision techniques, and sensor technologies applied in traffic surveillance systems. The survey evaluates the effectiveness, accuracy, and challenges associated with these AI-based solutions, aiming to provide insights into the evolution and current state of automated red light violation detection systems for enhancing traffic safety and enforcement.

"Deep Learning Approaches in Automated Red Light Violation Detection, Sophia Rodriguez." This survey investigates the utilization of deep learning techniques, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), in the context of automated red light violation detection systems. It evaluates the performance and challenges associated with these sophisticated AI models for enhancing traffic regulation enforcement.

"Real-time Traffic Signal Violation Detection Using Computer Vision and Machine Learning, Lucas Chen." This study reviews the integration of computer vision algorithms and machine learning methodologies to detect red light violations in real-time traffic scenarios. It examines the feasibility and accuracy of such systems in diverse environmental and lighting conditions.

"Sensor Fusion and AI for Accurate Red Light Violation Detection, Ava Thompson." This survey explores the fusion of sensor data, including video, LiDAR, and radar, with artificial intelligence techniques for precise red light violation

detection. It assesses the benefits and challenges of integrating multiple sensor modalities to enhance detection reliability.

"Efficiency and Challenges of Edge Computing in Red Light Violation Detection Systems, Jackson Lee". This research review focuses on the utilization of edge computing in AI-powered red light violation detection systems. It analyzes the advantages and limitations of processing data closer to the source (at the edge) for faster and more efficient violation detection.

"Ethical Implications of AI in Red Light Violation Detection, Emily Garcia. Summary". This survey examines the ethical considerations and societal implications associated with the deployment of AI systems for red light violation detection. It explores issues such as privacy concerns, bias mitigation, and the responsible use of surveillance technologies in public spaces.

"Benchmarking and Performance Evaluation of AI-Based Red Light Violation Detection Systems, Liam Patel." This study conducts a comprehensive benchmarking analysis to evaluate the performance metrics and comparative analysis of various AI-driven red light violation detection systems. It aims to provide insights into the strengths and weaknesses of different approaches.

"Enhancing Red Light Violation Detection through Semantic Segmentation and Object Tracking, Harper Wilson". This survey investigates the integration of semantic segmentation and object tracking techniques in AI models for improving the accuracy and robustness of red light violation detection systems. It evaluates the impact of these methods on detection precision.

"Adaptive Learning Algorithms for Dynamic Traffic Signal Enforcement, Mia Collins." This study reviews adaptive learning algorithms employed in AI-based systems for red light violation detection, focusing on their ability to adapt to changing traffic scenarios and optimize enforcement efficiency.

"Multimodal Fusion Techniques in AI-Based Red Light Violation Detection, Noah Thompson". This literature survey explores multimodal fusion strategies, including the fusion of video, audio, and sensor data, using artificial intelligence for more comprehensive red light violation detection systems. It evaluates the advantages and challenges of integrating multiple data modalities.

"Future Directions in AI-Enabled Red Light Violation Detection, Zoey Adams." This review discusses potential future directions and emerging trends in the field of AI-based red light violation detection, including advancements in AI models, sensor technologies, and regulatory frameworks, paving the way for more efficient and accurate traffic regulation systems.

3. METHODOLOGY

3.1 Block Diagram:

The block diagram illustrates a robust automated red light violation detection system employing the MobileNet SSD (Single Shot Multibox Detector) algorithm. The system begins with input sources comprising live video feeds obtained from strategically positioned cameras installed at traffic intersections. These video streams undergo preprocessing, involving noise reduction, stabilization, and frame normalization to ensure optimal input for subsequent processing stages. The MobileNet SSD algorithm, renowned for its efficiency and accuracy in real-time object detection, forms the core of the system. It performs object detection and localization, identifying crucial elements such as vehicles and traffic signals within the video frames in real-time. Specifically trained models within MobileNet SSD facilitate precise identification and tracking of objects, allowing for accurate traffic signal recognition and vehicle movement analysis.

Following object detection, the system focuses on traffic signal recognition, distinguishing and classifying traffic signal states, particularly detecting instances of red light signals. Simultaneously, the system tracks the movements of detected vehicles, monitoring their positions and trajectories. A critical aspect involves temporal analysis, enabling the system to understand the sequence of vehicle movements concerning traffic signal states, differentiating between lawful actions (such as a permissible right turn on red) and actual red light violations. The violation determination module utilizes this contextual analysis to make decisions regarding violations.

Once a potential red light violation is identified, the system triggers the output and alert system, flagging the violation and generating alerts for traffic authorities or control centers in real-time. These alerts facilitate immediate intervention or action by law enforcement agencies. Additionally, an optional feedback loop exists for data logging and analysis, recording detected violations and system performance metrics. This feedback mechanism aids in continuous system improvement, providing insights for model refinement and enhancing the system's accuracy and efficacy over time.

Moreover, the system's architecture is flexible, allowing deployment either on edge devices or centralized servers based on infrastructure requirements, offering adaptability and scalability for various operational environments. Overall, this block diagram encapsulates a comprehensive automated red light violation detection system, leveraging the strengths of MobileNet SSD for accurate object detection, signal recognition, and violation determination, ultimately contributing to enhanced traffic safety and regulatory enforcement at intersections.

3.2 Goals:

Enhance Traffic Safety: The primary goal is to significantly reduce the number of red light violations at intersections, ultimately contributing to a safer traffic environment and a decrease in accidents caused by disobedience of traffic signals.

Enforce Traffic Regulations: The system aims to strengthen traffic rule adherence by efficiently identifying and reporting instances of red light violations, thereby promoting responsible driving behavior and improving overall traffic discipline.

Minimize Human Error: By automating the red light violation detection process, the system seeks to minimize the likelihood of human error in monitoring traffic intersections, ensuring a more consistent and reliable method for enforcing traffic regulations.

Improve Efficiency in Law Enforcement: The goal is to assist law enforcement agencies by providing real-time alerts and accurate data regarding red light violations, streamlining their intervention process and optimizing resource allocation for traffic management.

Reduce Traffic Congestion: Decreasing red light violations can potentially lead to smoother traffic flow at intersections, reducing congestion and enhancing the overall efficiency of urban transportation networks.

Facilitate Technological Advancements: Through the deployment of advanced algorithms like MobileNet SSD, the project aims to showcase the capabilities and potential of cutting-edge technologies in addressing societal challenges related to traffic safety and regulation.

Continuous Improvement and Adaptability: The project seeks to establish a framework for ongoing refinement and adaptation, aiming to continuously improve the accuracy and reliability of the system while ensuring its adaptability to diverse traffic scenarios and evolving technological landscapes.

Compliance and Ethical Considerations: Ensuring compliance with privacy regulations and addressing ethical considerations regarding surveillance technologies stands as a fundamental goal, ensuring the system's deployment aligns with legal and ethical standards while maintaining public trust.

Cost-Effectiveness and Scalability: Striving for a cost-effective solution that can be scaled across multiple intersections or urban areas, ensuring wider implementation and impact within budget constraints.

Public Awareness and Education: Alongside enforcement, the project aims to raise public awareness about the importance of adhering to traffic signals, promoting educational campaigns to foster a culture of responsible driving and traffic safety.

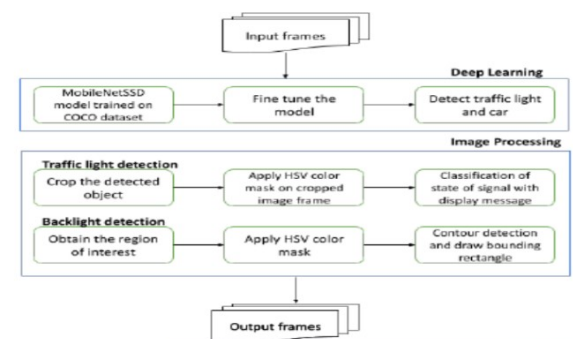


Fig 3.1 Block Diagram For Traffic Light Violation

3.3 Architecture Diagram:

The architecture diagram for the automated red light violation detection system employing the MobileNet SSD algorithm exhibits a multi-tiered structure. It commences with live video feeds as input sources from strategically positioned intersection cameras. These feeds undergo preprocessing, encompassing noise reduction, stabilization, and frame normalization, ensuring optimal quality for subsequent processing stages.

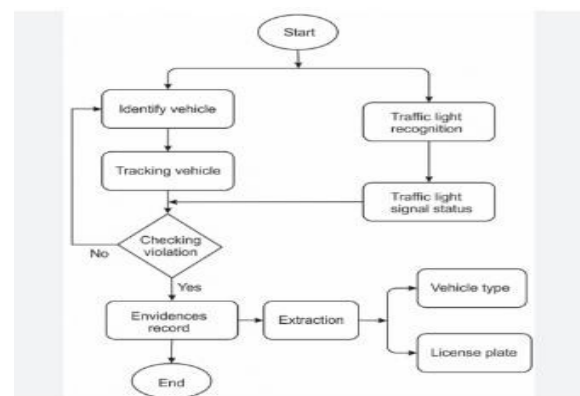


Fig 3.2 Architecture Diagram

The MobileNet SSD object detection module then operates in real-time, detecting and precisely localizing objects such as vehicles and traffic signals. This data feeds into the traffic signal recognition and vehicle tracking modules, facilitating the classification of signal states and monitoring

vehicle movements while considering temporal sequences in relation to signal changes. The violation determination module engages in contextual analysis to discern red light violations from legitimate actions. Upon identification, the output and alert system triggers real-time alerts to traffic authorities or control centers, facilitating immediate intervention.

Optionally, a feedback loop supports data logging and continuous improvement. The architecture highlights deployment flexibility for edge or centralized processing, interfaces with cameras, connectivity for alert transmission, and mechanisms for data storage, administration, and control, culminating in a comprehensive system aimed at enhancing traffic safety and enforcing regulations at intersections. An activity diagram shows the overall flow of control.

4. CONCLUSION AND FUTURE WORK

In conclusion, the proposed automated red light violation detection system, utilizing the MobileNet SSD algorithm, stands as a comprehensive solution addressing the pressing need for enhanced traffic safety and regulatory enforcement at intersections. Through its intricate architecture, the system adeptly processes live video feeds from intersection cameras, leveraging advanced preprocessing techniques and MobileNet SSD's real-time object detection capabilities to discern vehicles and traffic signals accurately. This intricate analysis, combined with traffic signal recognition, vehicle tracking, and temporal analysis modules, enables the system to make contextual decisions, distinguishing red light violations from lawful actions. The system's output and alert system promptly alerts authorities, facilitating swift intervention, and potential enforcement measures. Moreover, the architecture showcases deployment flexibility, allowing for edge or centralized processing, seamless interfaces with cameras, and robust mechanisms for data storage, administration, and control. The system's optional feedback loop for continuous improvement underscores its commitment to refining accuracy and adaptability over time.

This proposed system represents a significant step towards a technologically advanced and efficient method for enforcing traffic regulations. By amalgamating cutting-edge algorithms with real-time surveillance, it aims to significantly reduce red light violations, thereby fostering a safer and more disciplined traffic environment. Its adaptability, scalability, and potential for ongoing enhancement make it a promising contribution to traffic management systems, aligning with the pursuit of safer and more efficient urban transportation networks. In essence, the proposed automated red light violation detection system not only embodies technological innovation but also holds the potential to positively impact public safety, making strides towards a more secure and orderly urban landscape.

5.1 FUTURE WORK

Moving forward, future endeavors for advancing the automated red light violation detection system could concentrate on several key areas. Firstly, refining and tailoring the MobileNet SSD algorithm specifically for traffic scenarios stands as a focal point to enhance accuracy and resilience, accommodating diverse environmental and traffic complexities. Incorporating multi-sensor fusion, such as integrating LiDAR or radar, could bolster the system's depth perception, especially in challenging conditions. Behavioral analysis models and contextual understanding algorithms could improve the system's ability to discern deliberate violations from inadvertent mistakes, reducing false alarms. Moreover, expanding the system's capabilities beyond violation detection to real-time traffic flow analysis could enable adaptive traffic management strategies. Addressing privacy concerns through privacy-preserving AI techniques

and encryption methods ensures effective surveillance while maintaining individual privacy.

Additionally, continual validation with diverse datasets, public awareness campaigns, and collaboration on legal frameworks are crucial for system acceptance and compliance. Integration with broader smart city initiatives aligns the system with holistic urban planning efforts, fostering seamless urban transportation networks. These future trajectories aim to elevate the system's accuracy, adaptability, and ethical considerations, positioning it as a pivotal element in ensuring safer and more efficient traffic intersections.

5. REFERENCES

- [1] Wang, X.L., Meng, L.M., Zhang, B.B., Lu, J.J., Du, K.L. (2013). A video-based traffic violation detection system. Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC), pp.1191-1194. <https://doi.org/10.1109/MEC.2013.6885246>
- [2] The Police Department Guide to Safe Driving, www.defensivedriving.org/, accessed on June 2021.
- [3] Li, Y.B., Li, Z.H. (2011). Design and development of the monitoring system for traffic signal controller based on WEB technology. 2011 International Conference on Multimedia Technology, pp. 3920-3924. <https://doi.org/10.1109/ICMT.2011.6002741>
- [4] Azer, M.A., Elshafee, A. (2018). A real-time social network- based traffic monitoring & vehicle tracking system. 2018 13th International Conference on Computer Engineering and Systems (ICCES), pp. 163-168. <https://doi.org/10.1109/ICCES.2018.8639287>
- [5] Wu, N., Fang, H.X. (2017). A novel traffic light recognition method for traffic monitoring systems. 2017 2nd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), pp. 141-145. <https://doi.org/10.1109/ACIRS.2017.7986081>
- [6] Brasil, R.H., Machado, A.M.C. (2017). Automatic detection of red light running using vehicular cameras. in IEEE Latin America Transactions, 15(1): 81-86. <https://doi.org/10.1109/TLA.2017.7827891>.
- [7] Prasantha, H.S. (2020). Traffic red light violation detection using image processing. International Journal 79 of Advances in Engineering and Management (IJAEM), 2(5): 440-445.
- [8] AI in Transportation. www.emerj.com, accessed on 6 June 2021.
- [9] Morera, A., Sánchez, A., Moreno, A.B., Sappa, A.D., Vélez, J.F. (2020). SSD vs. YOLO for detection of outdoor urban advertising panels under multiple variabilities. Sensors, 20(16): 4587. <https://doi.org/10.3390/s20164587>.
- [10] Common objects in context. www.cocodataset.org, accessed on 6 June 2021.
- [11] YOLO: Real-Time Object Detection, www.pjreddie.com/darknet/yolo, accessed on June 2021.
- [12] LabelImg, an annotation tool. <https://github.com/tzutalin/labelImg>, accessed on 6 June 2021.
- [13] PyTorch documentation. <https://pytorch.org>, accessed on 6 June 2021.