TWO WHEELER TRAFFIC INFRINGEMENT RECOGNITION USING YOLO V7 AND AUTOMATED TICKETING

A PROJECT REPORT

Submitted by

ARUN KUMAR D [211419104022]
ASWIN V [211419104030]
HARI VIGNESH M [211419104095]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

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

APRIL 2023

PANIMALAR ENGINEERING COLLEGE

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

BONAFIDE CERTIFICATE

Certified that this project report **"TWO WHEELER TRAFFIC INFRINGEMENT RECOGNITION USING YOLO V7 AND AUTOMATED TICKETING"** is the bonafide work of **"ARUN KUMAR D (211419104022), ASWIN V (211419104030), HARI VIGNESH M (211419104095)"** who carried out the project work under my supervision.

SIGNATURE

Dr.L.JABASHEELA,M.E.,Ph.D.,
HEAD OF THE DEPARTMENT

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

Mr.THYAGARAJAN.C M.E.,(Ph.D),
SUPERVISOR
ASSISTANT PROFESSOR

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the End Semester Project Viva-Voce Examination held on.....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We **ARUN KUMAR D (211419104022) , ASWIN V (211419104030) , HARI VIGNESH M (211419104095)** hereby declare that this project report titled **"TWO WHEELER TRAFFIC INFRINGEMENT RECOGNITION USING YOLO V7 AND AUTOMATED TICKETING",** under the guidance of **Mr.THYAGARAJAN.C M.E.,(Ph.D.)**, is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

ARUN KUMAR D

ASWIN V

HARI VIGNESH M

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our beloved Directors **Tmt.C.VIJAYARAJESWARI**, **Dr.C.SAKTHI KUMAR, M.E.,Ph.D** and **Dr.SARANYASREE SAKTHI KUMAR B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.MANI, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. L.JABASHEELA, M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank our parents, friends, project Guide **Mr.THYAGARAJAN.C M.E.,(Ph.D.)**, and coordinator **Dr.N.PUGHAZENDI M.E., Ph.D.,** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

ARUN KUMAR D

ASWIN V

HARI VIGNESH M

ABSTRACT

India's excessive population, rising commuter population, poor management of traffic signals, and rider mentality make traffic violation monitoring and control a big challenge. It is clear that physical police-based traffic monitoring is insufficient to concurrently monitor such high traffic volumes and trace offences. This has caused numerous infringers to go undiscovered. The offenders, in turn bring more serious accidents on the road that endanger both their own life and the lives of others. Thus, it is necessary to incorporate artificial intelligence (AI) to automatically detect two-wheeler infractions on Indian roads, such as not wearing a helmet, using a phone while driving, triple-driving, wheeling, and illegal parking, and eventually automate the ticketing process by logging the infractions and associated vehicle number in a database. We suggest utilizing a specially trained yolo-v7 + deep sort for infringement recognition and tracking and YOLO-v7 + tesseract for number plate detection and extraction. The traffic infringement recognition system developed can be used to automate issuing tickets for vehicle violation. The created system will be especially helpful in developing various safety-related policies, assisting in the enforcement of strict traffic regulations, and contributing to the development of a smart city ecosystem through the automated AI-based traffic violation and ticketing system.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS

AI	Artificial Intelligence
YOLO	You Only Look Once
OCR	Optical Character Recognition
ROI	Region Of Interest
CNN	Convolutional Neural Network
OpenCV	Open Source Computer Vision
RCNN	Regions Convolutional Neural Networks
RTO	Road Traffic Officer
UML	Unified Modeling Language
FPS	Frame Per Second
GPU	Graphical Processing Unit
CCTV	Closed-Circuit Television

CHAPTER 1
INTRODUCTION

1. INTRODUCTION

1.1 OVERVIEW

The Indian government is very concerned about traffic monitoring and traffic violation control because of the excessive crowding, rising commuter numbers, flawed traffic system designs, and general human nature. Today, the majority of India's urban traffic management systems are still manually supervised. Both heavy traffic congestion and human error result from this. According to a report by the Ministry of Road Transport and Highways, riders of two-wheelers made up more than one-third (37%) of those killed in traffic accidents in 2019. Furthermore, not wearing a helmet contributed to 44,666 fatalities in road accidents in 2019 (30,148 drivers and 14,518 pillion passengers), or 29.82% of all fatalities in road accidents. Findings from the World Health Organization (WHO) indicate that proper helmet use alone could cut the risk of fatal injuries by 42% and head injuries by 69%. Furthermore, according to WHO data, drivers who are on their phones while driving are four times more likely to be in an accident than those who aren't. The inevitable results of using a phone while driving include slower reaction times. These statistics cover just two of the numerous two-wheeler traffic infractions that have been documented. According to a 2017 report from the Bureau of Police Research and Development, India's traffic police had a total staffing capacity of just over 72,000 while there are currently over 200 million vehicles on Indian roads .The volume and variety of vehicles are too many for traffic police monitoring to handle alone. Additionally, a lot of current solutions aren't flexible enough to recognise, examine, and track the wide range of vehicle types, licence types, dynamic traffic patterns, and street layouts. Numerous cities still use antiquated traffic control systems that can't be scaled up effectively to handle the volume and variety of traffic. Modern technologies must be implemented to monitor traffic and automate enforcement in order to handle these different challenges allowing for more intelligent traffic control systems.

Current techniques for detecting helmets and licence plates in traffic films have low efficiency and accuracy, rendering them unsuitable for use in the real world. Innovative methods are required in real-time circumstances to attain great accuracy and efficiency. The current approach makes it impossible for the system to distinguish between unclean number plates and corroded number plates, which might result in rule violations and the avoidance of fines. Using a collection of traffic photos, we tested the suggested method, and we outperformed conventional computer vision approaches with high accuracy and recall rates. Our strategy may

be utilised to increase traffic safety and enforce rules on the wearing of helmets and vehicle registration. The findings of our investigation show that deep learning is successful. In this paper, we use YOLO v7 model under the process of Convolutional Neural Network. In this YOLO model, we have the process of collecting data. By the means of collected data, the data sets are being trained and tested in modules. In the YOLO model, the labelling is done like whether the person on the bike or two-wheeler wears the helmet or not. In the labelling model, we give the input in the means of dimensions and array. By this method, the YOLO will be able to learn and process the function.

1.2 PROBLEM DEFINITION

Traffic monitoring and traffic violations pose a major challenge in India due to the increased commuter, and bad traffic system design. This leads to traffic congestion and a series of mishaps on the road. Thus there is a need for incorporating artificial intelligence-based techniques to eliminate manual intervention for the detection and catching of violators. This work proposed a pre-trained deep model. The proposed method would be helpful to enforce strong regulation of traffic rules.

CHAPTER 2
LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 A Novel Methodology for Vehicle Number Plate Recognition using Artificial Neural Network

Author Name : Sangita Kumari D. K. Gupta

Year of Publish : 2022

Number plate recognition system is used for vehicle management, security, congestion control, access control and in the vehicle behavior monitoring system. This work presents a real time number plate recognition system which is able to recognize the vehicle number plate in different illumination conditions, independent of orientation and scale of the plate. This research work begins by pre-processing, detecting plate region, segmentation, feature extraction and finally recognition of the character by using neural network. This system has been tested with other paper's data set that has different images with different illumination conditions and this system can recognize the number plates under different illumination conditions with a success rate of about 95%.

2.2 An Overview of the Tesseract OCR Engine

Author Name : Ray Smith

Year of Publish : 2020

The Tesseract OCR engine, as was the HP Research Prototype in the UNLV Fourth Annual Test of OCR Accuracy, is described in a comprehensive overview. Emphasis is placed on aspects that are novel or at least unusual in an OCR engine, including in particular the line finding, features/classification methods, and the adaptive classifier.

2.3 Object Detection And Tracking Algorithms For Vehicle Counting: A Comparative Analysis

Author Name : Vishal Mandal and Yaw Adu-Gyamfi

Year of Publish : 2021

The rapid advancement in the field of deep learning and high performance computing has highly augmented the scope of video-based vehicle counting system. In this paper, the authors deploy several state-of-the-art object detection and tracking algorithms to detect and track different classes of vehicles in their regions of interest (ROI). The goal of correctly

detecting and tracking vehicles' in their ROI is to obtain an accurate vehicle count. Multiple combinations of object detection models coupled with different tracking systems are applied to access the best vehicle counting framework. The models' addresses challenges associated to different weather conditions, occlusion and low-light settings and efficiently extracts vehicle information and trajectories through its computationally rich training and feedback cycles. The automatic vehicle counts resulting from all the model combinations are validated and compared against the manually counted ground truths of over 9 hours' traffic video data obtained from the Louisiana Department of Transportation and Development. Experimental results demonstrate that the combination of CenterNet and Deep SORT, Detectron2 and Deep SORT, and YOLOv4 and Deep SORT produced the best overall counting percentage for all vehicles.

2.4 An Intelligent License Plate Detection and Recognition Model Using Deep Neural Networks

Author Name **:** J.Andrew Onesimu1, Robin D.Sebastian1

Year of Publish **:** 2021

One of the largest automotive sectors in the world is India. The number of vehicles traveling by road has increased in recent times. In malls or other crowded places, many vehicles enter and exit the parking area. Due to the increase in vehicles, it is difficult to manually note down the license plate number of all the vehicles passing in and out of the parking area. Hence, it is necessary to develop an Automatic License Plate Detection and Recognition (ALPDR) model that recognize the license plate number of vehicles automatically. To automate this process, we propose a three-step process that will detect the license plate, segment the characters and recognize the characters present in it. Detection is done by converting the input image to a bi-level image. Using region props the characters are segmented from the detected license plate. A two-layer CNN model is developed to recognize the segmented characters. The proposed model automatically updates the details of the car entering and exiting the parking area to the database. The proposed ALPDR model has been tested in several conditions such as blurred images, different distances from the cameras, day and night conditions on the stationary vehicles. Experimental result shows that the proposed system achieves 91.1%, 96.7%, and 98.8% accuracy on license plate detection, segmentation, and recognition respectively which is superior to state-of-the-art literature models.

2.5 A Video-Based Traffic Violation Detection System

Author Name **:** Xiaoling Wang1,2 Li-Min Meng1

Year of Publish **:** 2020

Traffic violation detection systems are effective tools to help traffic administration to monitor the traffic condition. It can detect traffic violations, such as running red lights, speeding, and vehicle retrogress in real time. In this paper, we propose an improved background-updating algorithm by using wavelet transform on dynamic background, and then track moving vehicles by feature-based tracking method. A complete traffic violation detection system is realized in C++ with OpenCV.

2.6 Vehicle Licence Plate Recognition Using Artificial Neural Networks

Author Name **:** Cemil Öz1

Year of Publish **:** 2022

In this study, a vehicle license plate is recognizedusing artificial neural networks. Recognition of a vehicle license plate is usually important for many security and control systems. Artificial neural networks (ANN) are explained, and used image processing algoritms towards recognition are given with their result in this paper.

2.7 Helmet Presence Classification With Motorcycle Detection And Tracking

Author Name : J. Chiverton

Year of Publish : 2018

Helmets are essential for the safety of a motorcycle rider, however, the enforcement of helmet wearing is a time consuming labour intensive task. A system for the automatic classification and tracking of motorcycle riders with and without helmets is therefore described and tested. The system uses support vector machines trained on histograms derived from head region image data of motorcycle riders using both static photographs and individual image frames from video data. The trained classifier is incorporated into a tracking system where motorcycle riders are automatically segmented from video data using background subtraction. The heads of the riders are isolated and then classified using the trained classifier. Each motorcycle rider results in a sequence of regions in adjacent time frames called tracks. These tracks are then classified as a whole using a mean of the individual classifier results. Tests show that the classifier is able to accurately classify whether riders are wearing helmets or not on

static photographs. Tests on the tracking system also demonstrate the validity and usefulness of the classification approach.

2.8 Deep Learning Based License Plate Number Recognition for Smart Cities

Author Name : T. Vetriselvi, E. Laxmi Lydia

Year of Publish : 2022

Smart city-aspiring urban areas should have a number of necessary elements in place to achieve the intended objective. Precise controlling and management of traffic conditions, increased safety and surveillance, and enhanced incident avoidance and management should be top priorities in smart city management. At the same time, Vehicle License Plate Number Recognition (VLPNR) has become a hot research topic, owing to several real-time applications like automated toll fee processing, traffic law enforcement, private space access control, and road traffic surveillance. Automated VLPNR is a computer vision-based technique which is employed in the recognition of automobiles based on vehicle number plates. The current research paper presents an effective Deep Learning (DL)-based VLPNR called DLVLPNR model to identify and recognize the alphanumeric characters present in license plate. The proposed model involves two main stages namely, license plate detection and Tesseract-based character recognition. The detection of alphanumeric characters present in license plate takes place with the help of fast RCNN with Inception V2 model. Then, the characters in the detected number plate are extracted using Tesseract Optical Character Recognition (OCR) model. The performance of DL-VLPNR model was tested in this paper using two benchmark databases, and the experimental outcome established the superior performance of the model compared to other methods.

2.9 Detecting and Handling Traffic Violation

Author Name **:** M.Yogavalli, E.Arulmozhi, M.Rajeswari

Year of Publish **:** 2020

Safety and comfort of road users is becoming a matter of big concern. It is essential to build a safer and much more reliable system for traffic control and management. The main objective of this project is to introduce a system which detects stop line violation during red light running and to capture the invalid license, Road Tax, FC, insurance & chassis of a vehicle by using Active Radio-Frequency Identification (RFID), Global System for Mobile communication (GSM) and Programmable Interface Controller (PIC). This project consists of vehicle unit, traffic junction and Road Traffic Officer (RTO) unit. If the vehicle crosses the red

signal first time then message will be sent to user of the vehicle and RTO with penalty and vehicle details exist in the RFID available in vehicle unit. If the penalty is not paid within the timeline or the same vehicle crosses the red signal second time then vehicle will be slow down and stopped via GSM by the RTO unit. LCD's placed in the vehicle are used to display the RC number and to show the message for slow down and stop the vehicle. When the driver cut the connection and try to drive then the RC number will not be displayed on the vehicle which will help to capture the violated vehicle easily. A speed sensor is affixed to the vehicle, to control the speed of the vehicle when it violates the specified speed.

2.10 Effect Of Mixed Traffic On Capacity Of Two-Lane Roads: Case Study On Indian Highways

Author Name **:**Nabanita Roy, Rupali Roy, Hitesh Talukdar

Year of Publish **:** 2019

This paper focuses on effects of mixed traffic on capacity of two-lane roads. On the basis of field data collected on Indian highways, the present paper makes it clear that capacity reduces if the proportion of slower vehicles increases in the traffic stream. Since such vehicles are responsible for the formation of platoons, their increasing proportion in traffic would accordingly increase the equivalency factor of vehicles, thereby, resulting in variation in capacity. The present study therefore explicates the need of introducing the concept of dynamic passenger unit and anticipates that this would alleviate the current implication on capacity standards of such roads under mixed traffic.

CHAPTER 3
SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Traffic monitoring and traffic violation control is a major concern for the Indian Government, due to the excess crowd, increasing commuters, bad traffic system designs, and people mentality. Poor road safety is a crucial developmental issue, a public health concern, and a ruling cause of injury and demise in India. As per the Report of the Ministry of Road Transport and Highways on road accidents in India 2019, there were 1,51,113 accident-related deaths in India in 2019. Even though there are so many road laws introduced by our government for the safety of passengers and drivers, the number of accidents is still increasing alarmingly. The main reason behind these accidents is that people do not follow the basic traffic rules like wearing a helmet. In this existing approach, the physical traffic police-based monitoring alone is insufficient to monitor such a large traffic volume and simultaneously track violations.

3.2 DISADVANTAGES OF EXISTING SYSTEM

- The existing system has a detection accuracy of 88%.
- Implemented using YOLO v4 which has less pre trained datasets
- Number plate image is not used for ticketing.
- Normal ticketing is carried which is time consuming.

3.3 PROPOSED SYSTEM

In the proposed method, design and develop an AI model to detect and track automatically traffic violations. The system consists of three major components namely vehicle detection, helmet detection, number plate detection, triples detection, and sending the violated person's vehicle number plate through mail. We first obtain the frames from the live footage as input to perform violation detection using an Object detection module. The violations covered for our study are not wearing helmets, Triple riding. The object detection is carried out using YOLO-v7 object detector to detect one or more violations in each image frame.

3.4 ADVANTAGES OF PROPOSED SYSTEM

- The proposed system uses YOLO V7 which is the latest model
- This approach makes the riders to follow strict rules.
- This system tracks the detection and rises a ticket in mail.
- This system reduces the manual ticketing for the infringement.

3.5 FEASIBILITY STUDY

The massive population increase and the overcrowding of transport networks, traffic accidents are also increasing. These accidents cause colossal material and moral losses and are considered a hindrance to development. The existing traffic violation detection system has a detection of 88% . This system aims to improve the efficiently of detection and reduce the error of false detection . Our work provides an combination of pre trained algorithms which make the processing more efficiently. The work will be economically feasible than previous system.the main advantage of our approach is by increasing the accuracy and automatically the ticketing the number plate to mail.

3.6 HARDWARE ENVIRONMENT

- Hard Disk : 500GB and Above
- RAM : 4GB and Above
- Processor : I3 and Above

3.7 SOFTWARE ENVIRONMENT

- Operating System : Windows 10 (64 bit)
- Software : Python
- Tools : Anaconda

3.8 TECHNOLOGIES USED

- Python
- Deep Learning

3.8.1 Python

Python is a high-level, interpreted programming language that is widely used in various domains such as web development, data science, artificial intelligence, scientific computing, and more. It was first released in 1991 and has since become one of the most popular programming languages in the world. Some key features of Python include:

- Easy to Learn: Python has a simple and easy-to-learn syntax, which makes it an ideal language for beginners.
- Interpreted Language: Python is an interpreted language, which means that the code is executed line by line, making it easier to test and debug.
- Cross-Platform: Python can be run on various platforms, including Windows, macOS, and Linux.
- Large Standard Library: Python has a large standard library that provides a wide range of built-in modules for various tasks, such as file I/O, regular expressions, networking, and more.
- Open Source: Python is open-source software, which means that the source code is freely available to anyone and can be modified and redistributed.
- Object-Oriented: Python is an object-oriented language, which means that it supports object-oriented programming concepts such as encapsulation, inheritance, and polymorphism.

Python has a vast community of developers and users, which has contributed to its popularity and growth. It has a large number of third-party libraries and frameworks that make it easier to develop complex applications quickly. Some popular Python frameworks include Django, Flask, and Pyramid for web development, and NumPy, Pandas, and SciPy for scientific computing and data analysis.In summary, Python's simplicity, ease of use, and versatility have made it a popular programming language in various industries, from web development to scientific computing and artificial intelligence.

3.8.2 Deep Learning

Deep learning is a subfield of machine learning that uses neural networks to model and solve complex problems. Neural networks are a type of artificial intelligence modeled after the structure of the human brain. They consist of layers of interconnected nodes, or artificial neurons, that are trained to recognize patterns in data. In deep learning, neural networks are organized in layers, with each layer learning to identify and extract different features from the data. The layers are stacked one on top of the other, creating a deep neural network that can learn complex representations of the data. Deep learning has shown remarkable success in many areas of artificial intelligence, such as image recognition, natural language processing, speech recognition, and robotics. It has enabled breakthroughs in areas such as self-driving cars, virtual assistants, and medical diagnosis.Some of the popular deep learning architectures include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Generative Adversarial Networks (GANs). CNNs are commonly used in image recognition tasks, RNNs are used for sequential data such as natural language processing, and GANs are used for image generation. Deep learning requires large amounts of labeled data and computational power to train the neural networks. With the availability of big data and advancements in hardware, deep learning has become a widely used and powerful tool for solving complex problems in various industries.

CHAPTER 4
SYSTEM DESIGN

4. SYSTEM DESIGN

4.1 ENTITY-RELATIONSHIP DIAGRAM

An architecture diagram is a graphical representation of a set of concepts that are part of an architecture, including their principles, elements and components. It is also defined as a visual representation that maps out the physical implementation for components of a software system. It shows the general structure of the software system and the associations, limitations, and boundaries between each element.

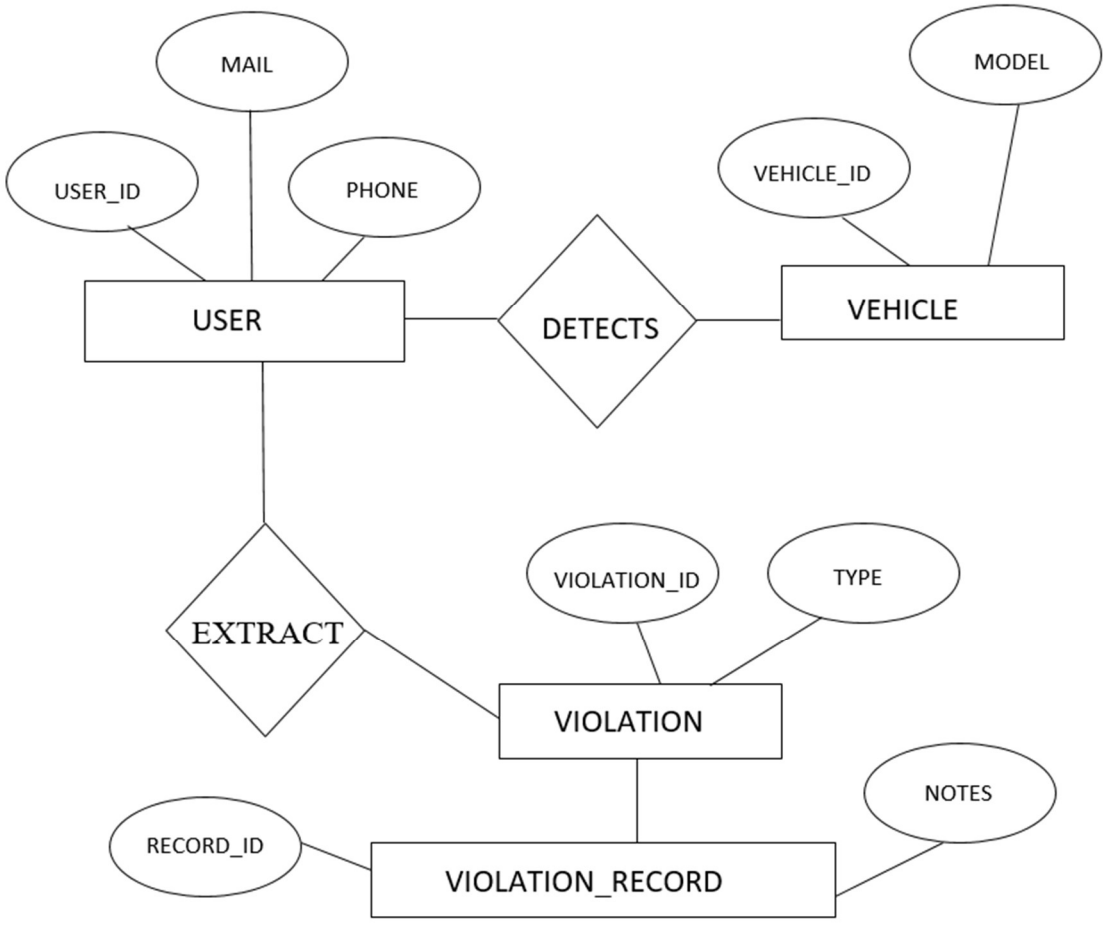

Fig 4.1 Entity Relationship Diagram

The User entity represents a user of the system who owns one or more vehicles. Each user has a unique user_id, and can be identified by their name, email, and phone_number.

The Vehicle entity represents a two-wheeler vehicle owned by a user. Each vehicle has a unique vehicle_id, and is associated with a User entity through a foreign key user_id.

The Violation entity represents a traffic violation that can be detected by the system, such as not wearing a helmet or carrying more than two people on a two-wheeler. Each violation has a unique violation_id, and includes information such as the violation_type, date, time, location, and the user_id of the user who committed the violation.

The Violation_Record entity represents a record of a violation that has been detected by the system. Each record has a unique record_id, and includes information such as the violation_id of the violation that was committed, the vehicle_id of the vehicle that was used, the officer_id of the officer who detected the violation, and any additional notes or comments.

4.2 DATA FLOW DIAGRAM (DFD)

A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement. They are often elements of a formal methodology such as Structured Systems Analysis and Design Method (SSADM). Superficially, DFDs can resemble flow charts or Unified Modeling Language (UML), but they are not meant to represent details of software logic. DFDs make it easy to depict the business requirements of applications by representing the sequence of process steps and flow of information using a graphical representation or visual representation rather than a textual description. When used through an entire development process, they first document the results of business analysis. Then, they refine the representation to show how information moves through, and is changed by, application flows. Both automated and manual processes are represented.

LEVEL 0

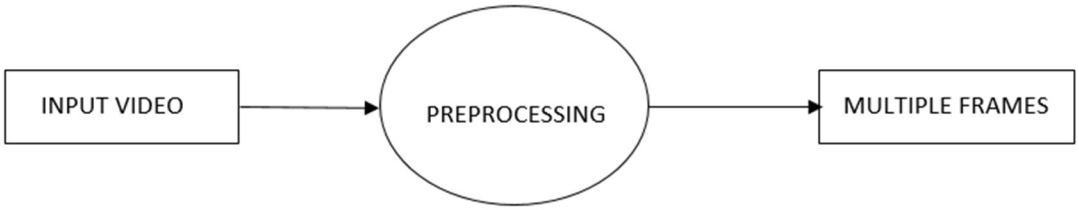

Fig 4.2.1 Level 0 of Data Flow Diagram

LEVEL 1

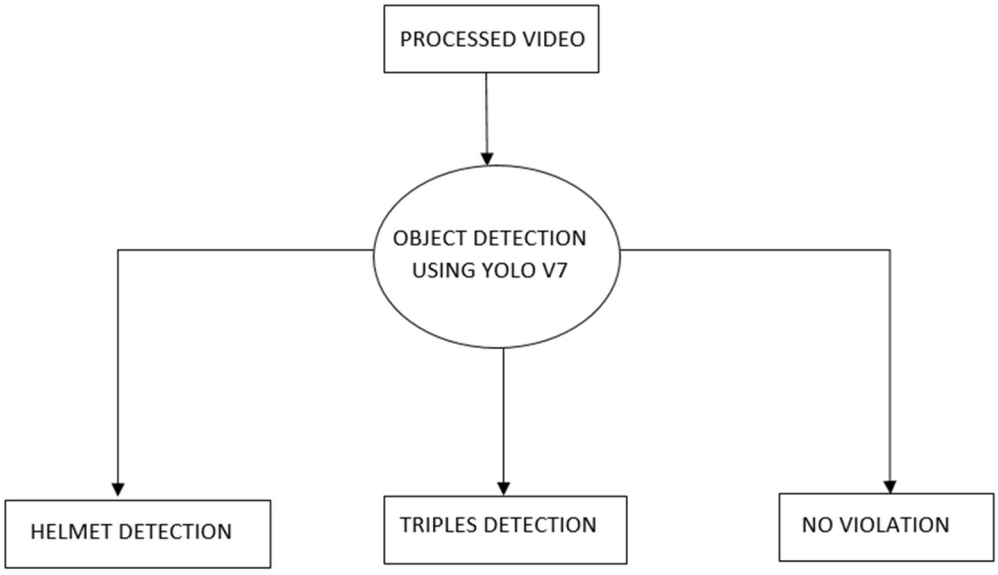

Fig 4.2.2 Level 1 of Data Flow Diagram

LEVEL 2

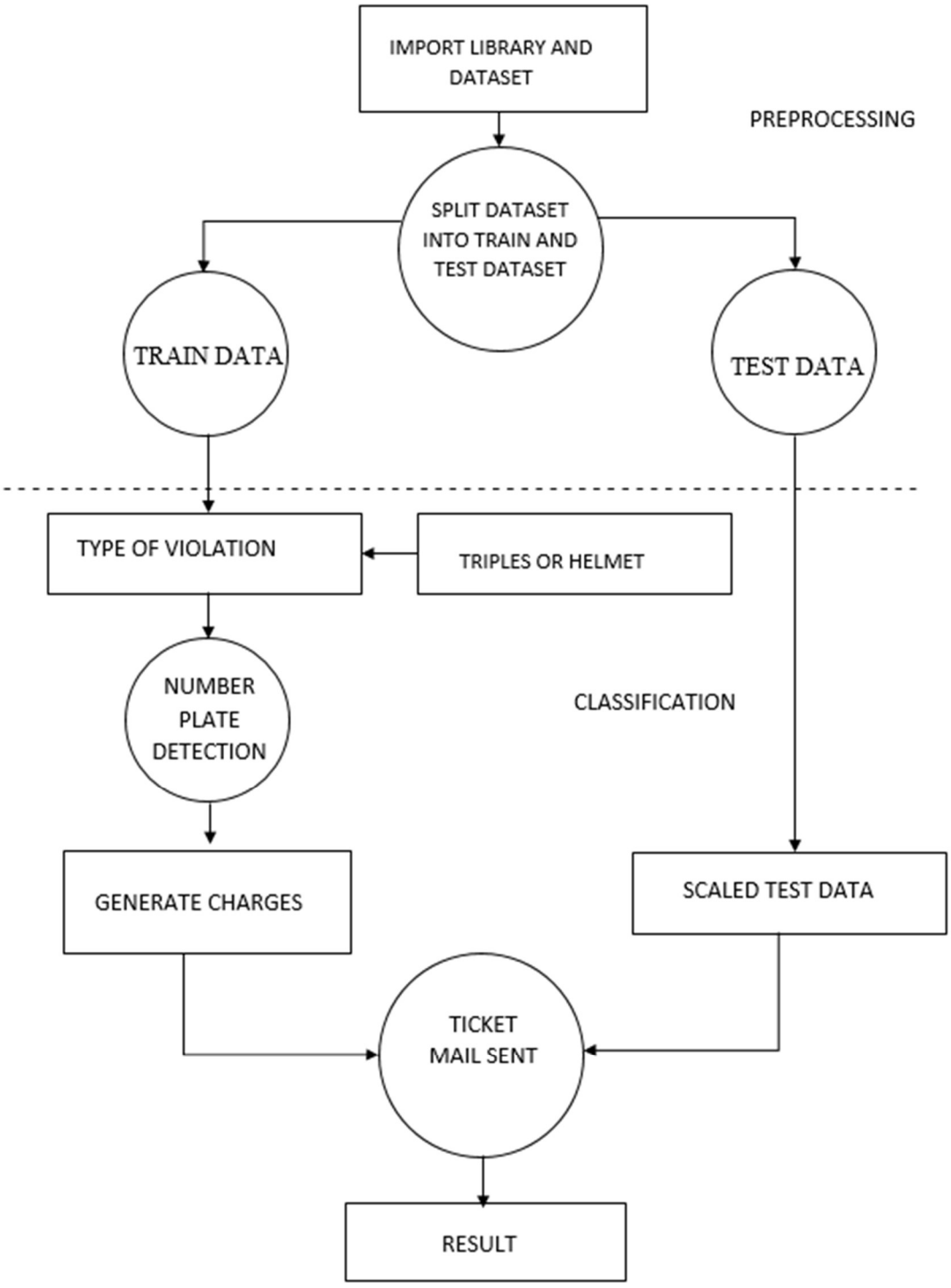

Fig 4.2.3 Level 2 of Data Flow Diagram

4.3 UML DIAGRAMS

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.OMG is continuously making efforts to create a truly industry standard.UML stands for Unified Modeling Language.UML is different from the other common programming languages such as C++, Java, COBOL, etc. UML is a pictorial language used to make software blueprints.UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system.Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc.UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization, UML has become an OMG standard.

4.3.1 Use Case Diagram

A use case diagram is a type of Unified Modeling Language (UML) diagram that represents the interactions between a system and its actors, and the various use cases that the system supports. It is a visual representation of the functional requirements of the system and the actors that interact with it. Use case diagrams typically include the following elements:

- Actors: Actors are external entities that interact with the system. They can be human users, other systems, or devices.
- Use Cases: Use cases are the specific functions or tasks that the system can perform. Each use case represents a specific interaction between an actor and the system.
- Relationships: Relationships are used to indicate how the actors and use cases are related to each other. The two main relationships in a use case diagram are "uses" and "extends". "Uses" relationship indicates that an actor uses a specific use case, while "extends" relationship indicates that a use case extends or adds functionality to another use case.
- System Boundary: The system boundary is a box that contains all the actors and use cases in the system. It represents the physical or logical boundary of the system being modeled.

- Use case diagrams are useful for identifying the functional requirements of a system, and for communicating these requirements to stakeholders. They can be used in the requirements gathering phase of software development, as well as in the design and testing phases.

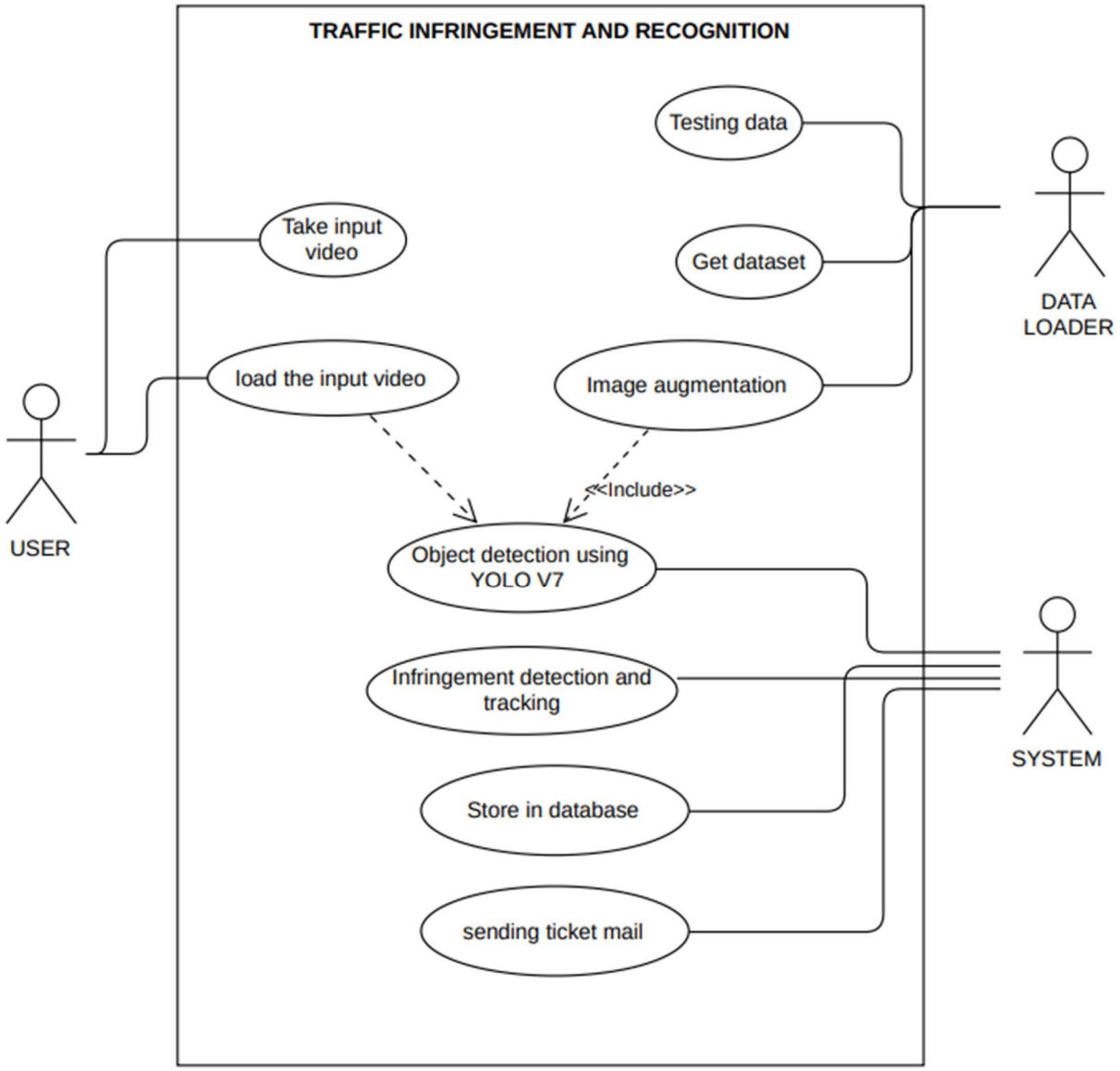

Fig 4.3.1 Use Case Diagram

4.3.2 Class Diagram

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.It shows the attributes, classes, functions, and relationships to give an overview of the software system. It constitutes class names, attributes, and functions in a separate compartment that helps in software development. Since it is a collection of classes, interfaces, associations, collaborations, and constraints, it is termed as a structural diagram.The main purpose of class diagrams is to build a static view of an application. It is the only diagram that is widely used for construction, and it can be mapped with object-oriented languages.

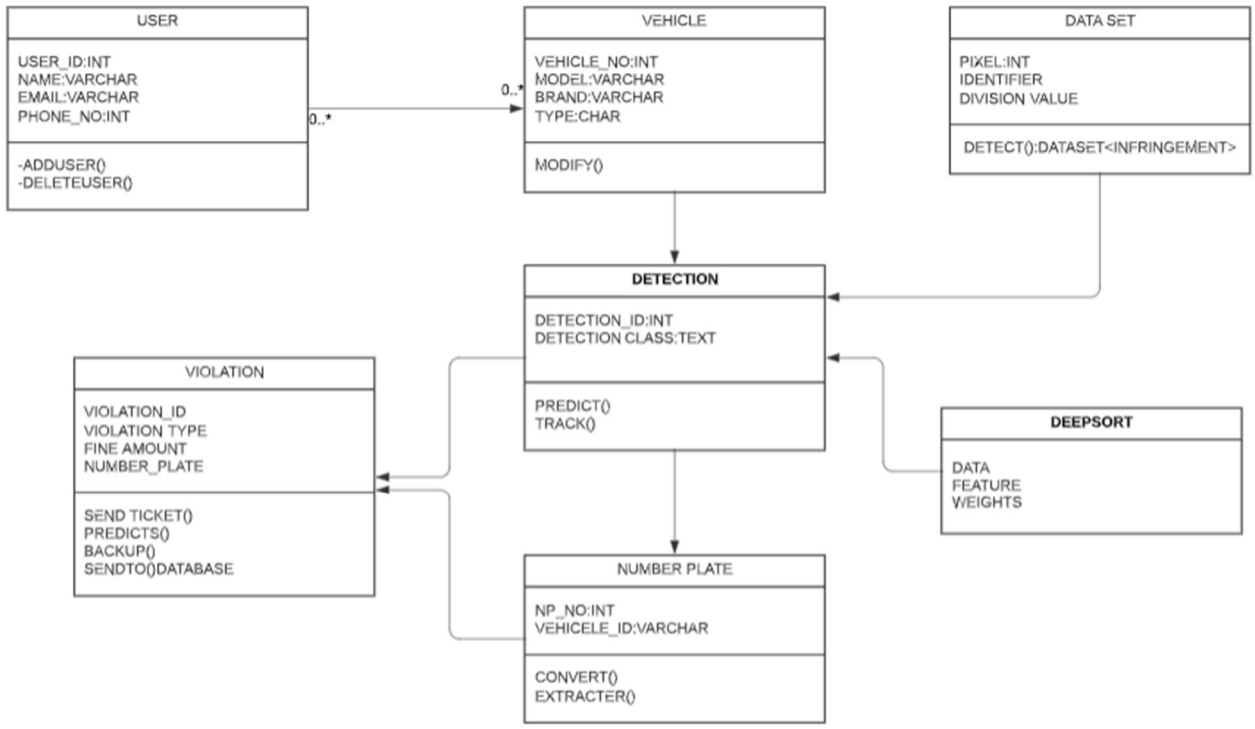

Fig 4.3.2 Class Diagram

4.3.3 State Chart Diagram

State chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of State chart diagram is to model lifetime of an object from creation to termination. State chart diagrams are also used for forward and reverse engineering of a system.

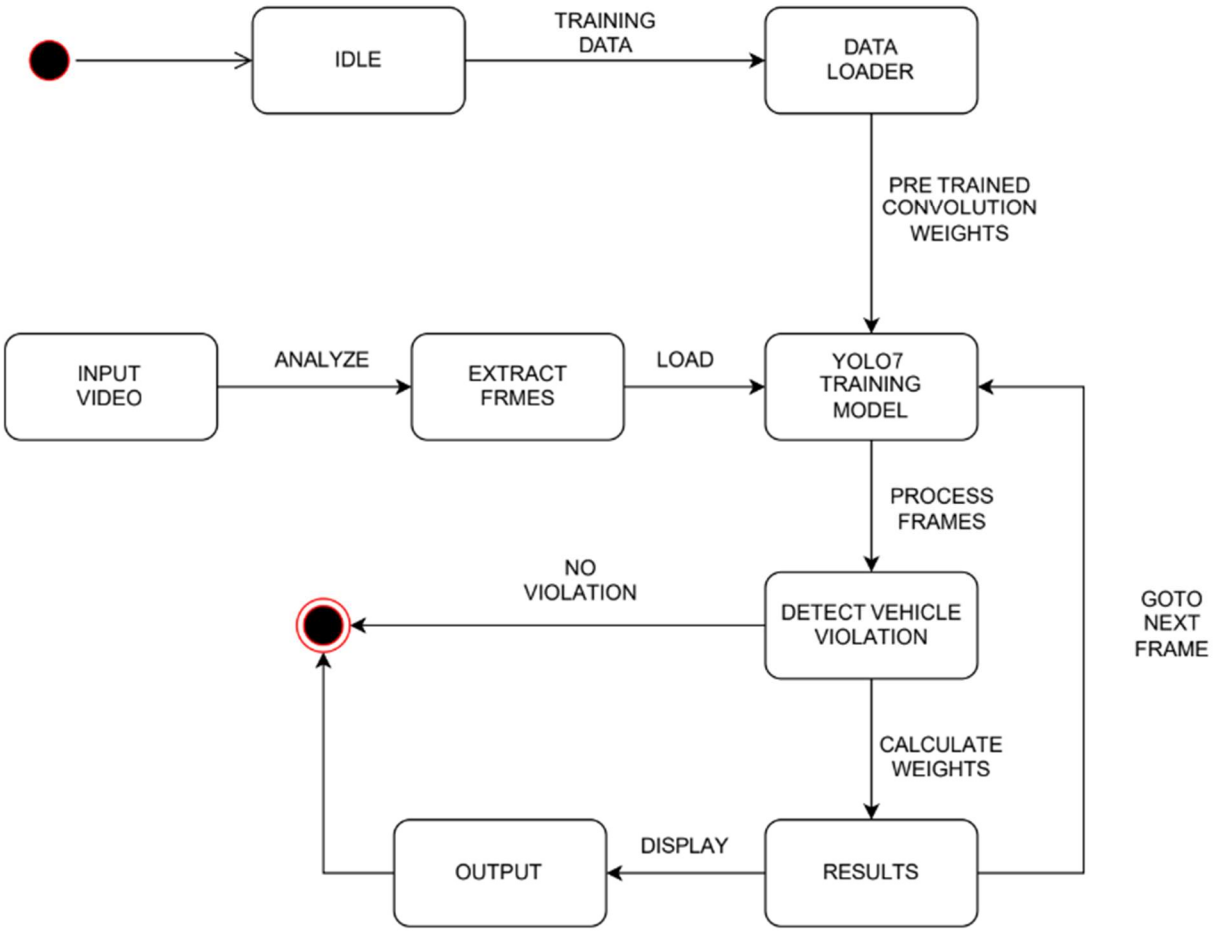

Fig 4.3.3 State Chart Diagram

4.3.4 Sequence Diagram

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram.

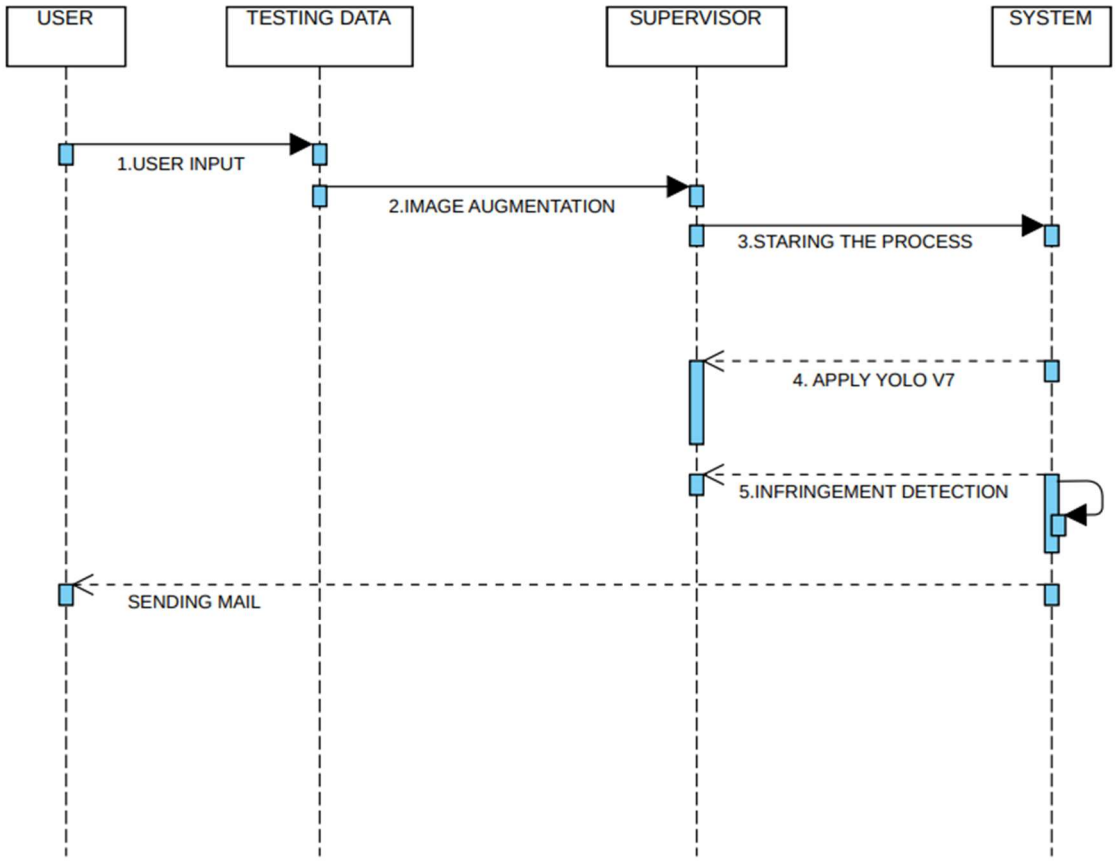

Fig 4.3.4 Sequence Diagram

4.3.5 Activity Diagram

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

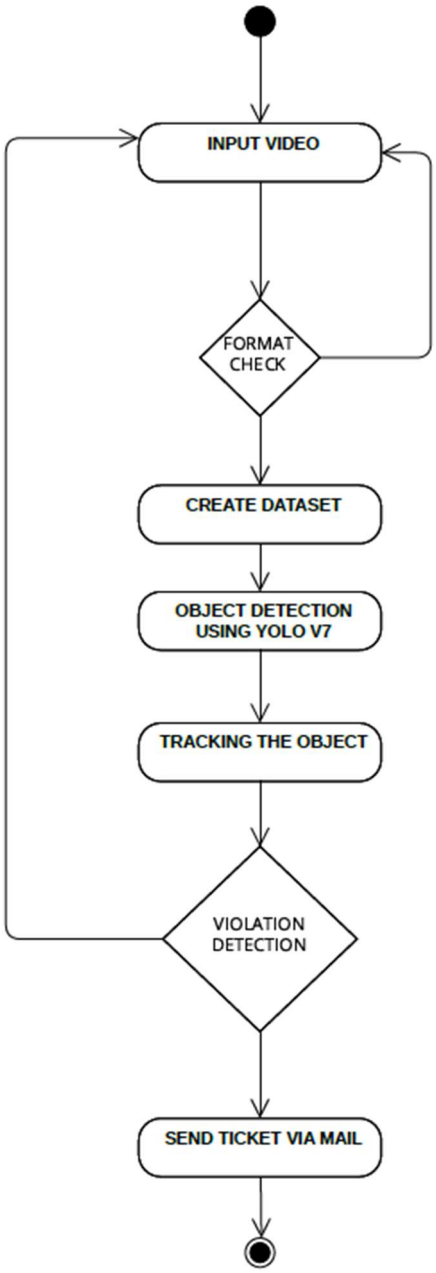

Fig 4.3.5 Activity Diagram

4.3.6 Component Diagram

A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node. It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces.

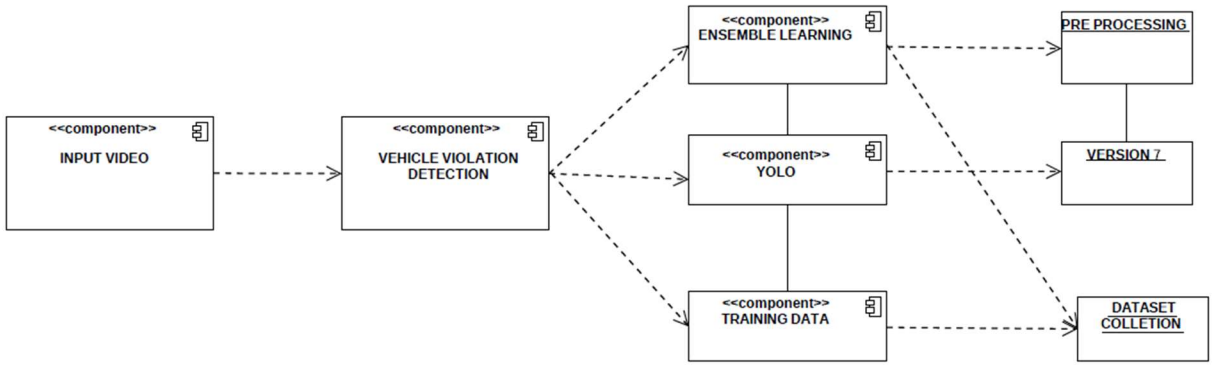

Fig 4.3.6 Component Diagram

4.3.7 Deployment Diagram

The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships. It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be

executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths.

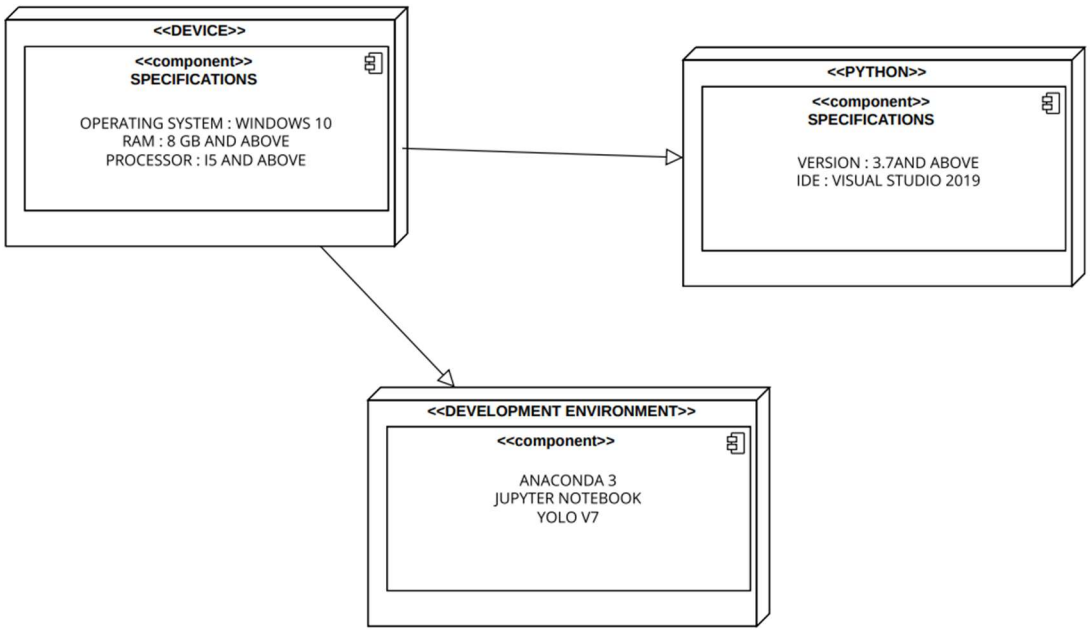

Fig 4.5.7 Deployment Diagram

4.4 DATA DICTIONARY DIAGRAM

A data dictionary provides terminology for all relevant data to be used by the developers in a project. It helps in performing analysis based on the impact of some data on the processing activities. It also helps the developers to determine the definition of different data structures in terms of their basic elements while designing activities. In the case of large systems, data dictionaries may become extremely voluminous and difficult to handle. In such case, CASE (Computer-Aided Software Engineering) tools are used, that capture all the data items appearing in the DFD and automatically generate the data dictionary.

Entity: User		
Attribute Name	Data Type	Description
user_id	int	Unique user identifier
name	varchar	User's full name
email	varchar	User's email address
Phone_number	varchar	User's phone number

Table 4.4.1 User Entity Table

Entity: Vehicle		
Attribute Name	Data Type	Description
vehicle_id	int	Unique vehicle identifier
Make	varchar	Vehicle make
Model	varchar	Vehicle model
year	varchar	Vehicle year of manufacture
user_id (FK)	int	Foreign key to user entity

Table 4.4.2 Vehicle Entity Table

Entity: Violation		
Attribute Name	Data Type	Description
violation_id	int	Unique violation identifier
Violation_type	varchar	Type of traffic violation
Fine_amount	decimal	Fine amount for violation

Table 4.4.3 Violation Entity table

Entity: Violation_Record		
Attribute Name	Data Type	Description
record_id	int	Unique violation_record identifier
Violation_id (FK)	varchar	Foreign key to violation entity
Vehicle_id (FK)	varchar	Foreign key to vehicle entity
Officer_id	varchar	Foreign key to Officer entity
notes	text	Notes or comments

Table 4.4.4 Violation Record Entity Table

CHAPTER 5
SYSTEM ARCHITECTURE

5. SYSTEM ARCHITECTURE

5.1 ARCHITECTURE DIAGRAM

An architecture diagram is a graphical representation of a set of concepts that are part of an architecture, including their principles, elements and components. It is also defined as a visual representation that maps out the physical implementation for components of a software system. It shows the general structure of the software system and the associations, limitations, and boundaries between each element.

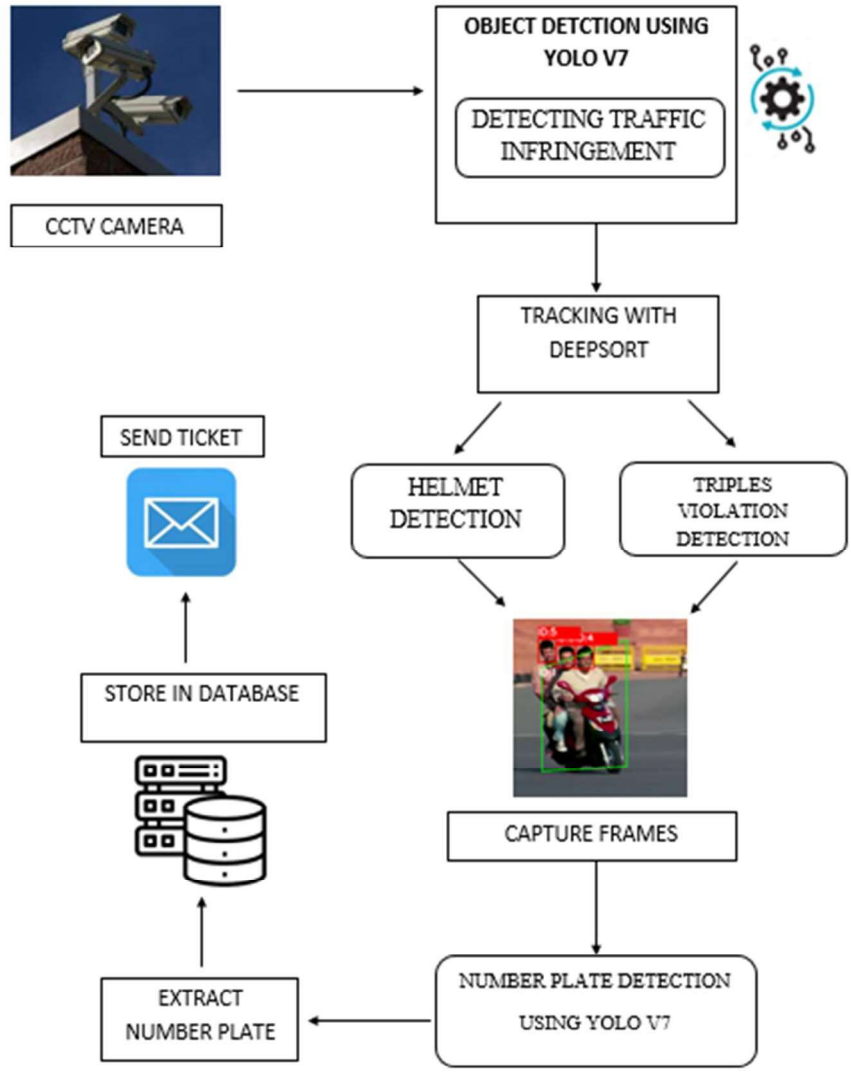

Fig 5.1 Architecture Diagram

The system architecture of the fig 5.1 clearly shows that the input is given as video then using the YOLO v7 model the objects are detected and classified objects are assigned with id .After the detection of objects the tracking of the objects are done using DeepSort algorithm which tracks the user and takes decision of the infringement carried or not. If the infringement is confirmed, the number plate is detected and extracted using YOLO v7 model and stores the output and number plate to the local database and send a ticket to the mail which is given to the database.

5.2 ALGORITHMS

5.2.1 YOLO V7

The YOLOv7 performance was evaluated based on previous YOLO versions (YOLOv4 and YOLOv5) and YOLOR as baselines. The models were trained with the same settings. The new YOLOv7 shows the best speed-to-accuracy balance compared to state-of-the-art object detectors. In general, YOLOv7 surpasses all previous object detectors in terms of both speed and accuracy, ranging from 5 FPS to as much as 160 FPS. The YOLO v7 algorithm achieves the highest accuracy among all other real-time object detection models – while achieving 30 FPS or higher using a GPU V100.

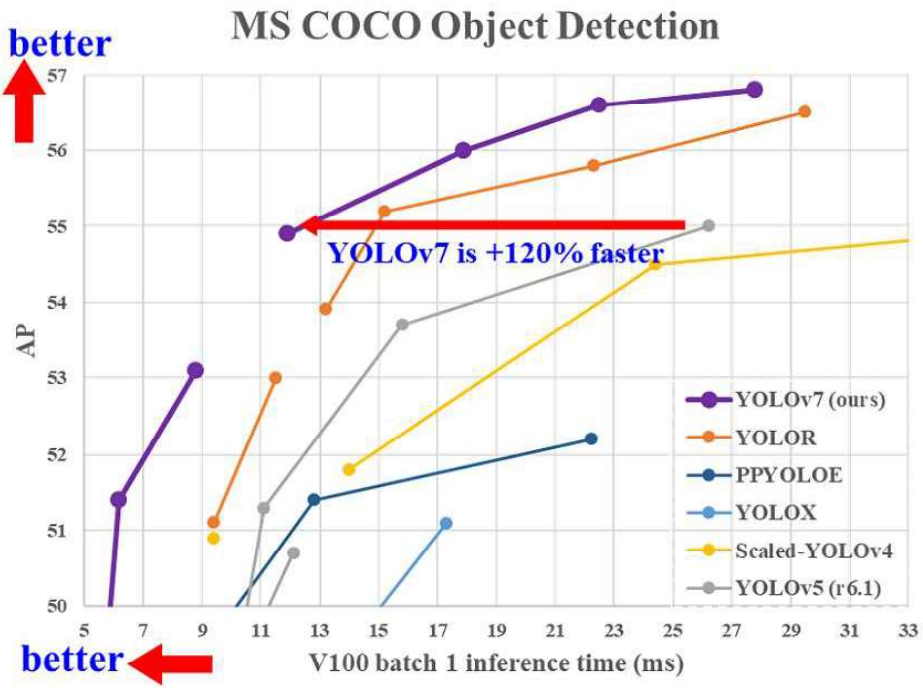

Fig 5.2.1.1 Comparison of Object detection Models

Compared to the best performing Cascade-Mask R-CNN models, YOLOv7 achieves 2% higher accuracy at a dramatically increased inference speed (509% faster). This is impressive because such R-CNN versions use multi-step architectures that previously achieved significantly higher detection accuracies than single-stage detector architectures. YOLOv7 outperforms YOLOR, YOLOX, Scaled-YOLOv4, YOLOv5, DETR, ViT Adapter-B, and many more object detection algorithms in speed and accuracy.

Model	#Param.	FLOPs	Size	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
YOLOv4 [3]	64.4M	142.8G	640	49.7%	68.2%	54.3%	32.9%	54.8%	63.7%
YOLOR-u5 (r6.1) [81]	46.5M	109.1G	640	50.2%	68.7%	54.6%	33.2%	55.5%	63.7%
YOLOv4-CSP [79]	52.9M	120.4G	640	50.3%	68.6%	54.9%	34.2%	55.6%	65.1%
YOLOR-CSP [81]	52.9M	120.4G	640	50.8%	69.5%	55.3%	33.7%	56.0%	65.4%
YOLOv7	36.9M	104.7G	640	**51.2%**	**69.7%**	**55.5%**	**35.2%**	**56.0%**	**66.7%**
improvement	-43%	-15%	-	+0.4	+0.2	+0.2	+1.5	=	+1.3
YOLOR-CSP-X [81]	96.9M	226.8G	640	52.7%	**71.3%**	57.4%	36.3%	57.5%	68.3%
YOLOv7-X	71.3M	189.9G	640	**52.9%**	71.1%	**57.5%**	**36.9%**	**57.7%**	**68.6%**
improvement	-36%	-19%	-	+0.2	-0.2	+0.1	+0.6	+0.2	+0.3
YOLOv4-tiny [79]	6.1	6.9	416	24.9%	42.1%	25.7%	8.7%	28.4%	39.2%
YOLOv7-tiny	6.2	5.8	416	**35.2%**	**52.8%**	**37.3%**	**15.7%**	**38.0%**	**53.4%**
improvement	+2%	-19%	-	+10.3	+10.7	+11.6	+7.0	+9.6	+14.2
YOLOv4-tiny-3l [79]	8.7	5.2	320	30.8%	47.3%	32.2%	**10.9%**	31.9%	51.5%
YOLOv7-tiny	6.2	3.5	320	**30.8%**	**47.3%**	**32.2%**	10.0%	**31.9%**	**52.2%**
improvement	-39%	-49%	-	=	=	=	-0.9	=	+0.7
YOLOR-E6 [81]	115.8M	683.2G	1280	55.7%	73.2%	60.7%	40.1%	**60.4%**	69.2%
YOLOv7-E6	97.2M	515.2G	1280	**55.9%**	**73.5%**	**61.1%**	**40.6%**	60.3%	**70.0%**
improvement	-19%	-33%	-	+0.2	+0.3	+0.4	+0.5	-0.1	+0.8
YOLOR-D6 [81]	151.7M	935.6G	1280	56.1%	73.9%	61.2%	**42.4%**	60.5%	69.9%
YOLOv7-D6	154.7M	806.8G	1280	56.3%	73.8%	61.4%	41.3%	60.6%	70.1%
YOLOv7-E6E	151.7M	843.2G	1280	**56.8%**	**74.4%**	**62.1%**	40.8%	**62.1%**	**70.6%**
improvement	=	-11%	-	+0.7	+0.5	+0.9	-1.6	+1.6	+0.7

Fig 5.2.1.2 Performance chart for YoloV7 model

5.2.2 DEEPSORT

SORT performs very well in terms of tracking precision and accuracy. But SORT returns tracks with a high number of ID switches and fails in case of occlusion. This is because of the association matrix used. DeepSORT uses a better association metric that combines both motion and appearance descriptors. DeepSORT can be defined as the tracking algorithm which tracks objects not only based on the velocity and motion of the object but also the appearance of the object.A well-discriminating feature embedding is trained offline just before implementing tracking. The network is trained on a large-scale person re-identification dataset making it suitable for tracking context. To train the deep association metric model in the DeepSORT cosine metric learning approach is used. According to DeepSORT's paper, "The cosine distance considers appearance information that is particularly useful to recover identities after long-term occlusions when motion is less discriminative." That means cosine distance is

a metric that helps the model recover identities in case of long-term occlusion and motion estimation also fails. Using these simple things can make the tracker even more powerful and accurate.

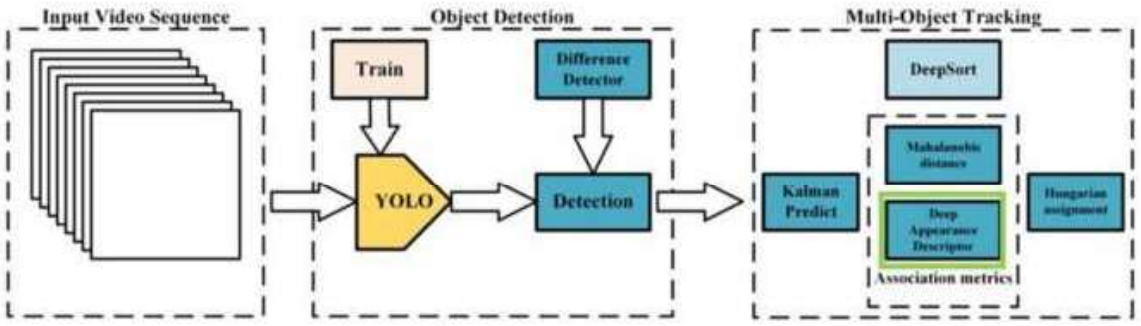

Fig 5.2.2 Deep sort working Diagram

5.2.3 TESSERACT

Tesseract — is an optical character recognition engine with open-source code, this is the most popular and qualitative OCR-library. OCR uses artificial intelligence for text search and its recognition on images. Tesseract is finding templates in pixels, letters, words and sentences. It uses two-step approach that calls adaptive recognition. It requires one data stage for character recognition, then the second stage to fulfil any letters, it wasn't insured in, by letters that can match the word or sentence context. The main task was to recognize receipts from photos. Tesseract OCR was used as a primary tool. Library pros are trained language models (>192), different kinds of recognition (image as word, text block, vertical text), easy to setup. 3rd party wrapper from GitHub was used as Tesseract OCR was written on C++.

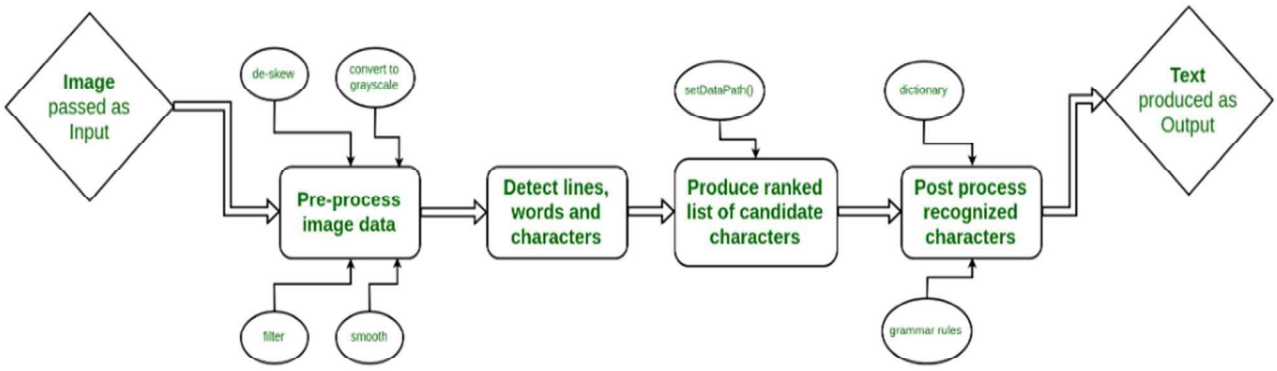

Fig 5.2.3 Tesseract flow Diagram

CHAPTER 6

SYSTEM IMPLEMENTATION

6. SYSTEM IMPLEMENTATION

6.1 MODULE DESIGN SPECIFICATION

We'll go over the suggested end-to-end system for automating ticketing and traffic infringement recognition. The system is made up of four main parts:

- **Object Detection Module**
- **Object Tracking Model**
- **License Plate Detection**
- **Ticketing**

• Using an **Object detection module**, we first obtain the frames from the live video as input to perform violation detection. In the context of India, the violations we looked at included not wearing helmets, using phones while riding, triple riding, wheeling, and not parking. Each image frame is subjected to object detection using the YOLO-v7 object detector, which can find one or more infringement.

• **Object tracking model:** The detected object is then tracked using the object tracking model to avoid repetitive counting and to ensure capturing the vehicles across multiple frames.

• **Licence plate detection:** After detecting and tracking the infringement, each violating vehicle is cropped out using the coordinates obtained from bounding boxes given by object detection module of vehicle detector module. YOLO-V7 is used for the detection of the number plate of a vehicle from cropped vehicle images. Tesseract is used for OCR (Optical Character Recognition) is used to extract the license numbers from the number plate and store them in a database.

• **Ticketing:** Vehicle number plates of violated users are mailed with associated violations from the database. The database can be used further for statistical analysis on traffic rules violations.

6.1.1 Object Detector Module

Object detection is concerned with the identification and localization of objects, in our case the violations. We have used YOLO v7 with pre-trained weights, as this is a pre-trained setup capable of identifying 9000 classes of objects and can be fine-tuned to capture YOLO v7 surpasses all known object detectors in both speed and accuracy in the range from 5 FPS to 160 FPS and has the highest accuracy 56.8% AP among all known real-time object detectors with 30 FPS or higher on GPU V100. YOLOv7-E6 object detector (56 FPS V100, 55.9% AP)

outperforms both transformer-based detector SWIN-L Cascade-Mask R-CNN (9.2 FPS A100, 53.9% AP) by 509% in speed and 2% in accuracy, and convolutional-based detector ConvNeXt-XL Cascade-Mask R-CNN (8.6 FPS A100, 55.2% AP) by 551% in speed and 0.7% AP in accuracy, as well as YOLOv7 outperforms: YOLOR, YOLOX, Scaled-YOLOv4,YOLOv5, DETR, Deformable DETR, DINO-5scale-R50, ViT-Adapter-B and many other object detectors in speed and accuracy. Moreover, we train YOLOv7 only on MS COCO dataset from scratch without using any other datasets or pre-trained weights.

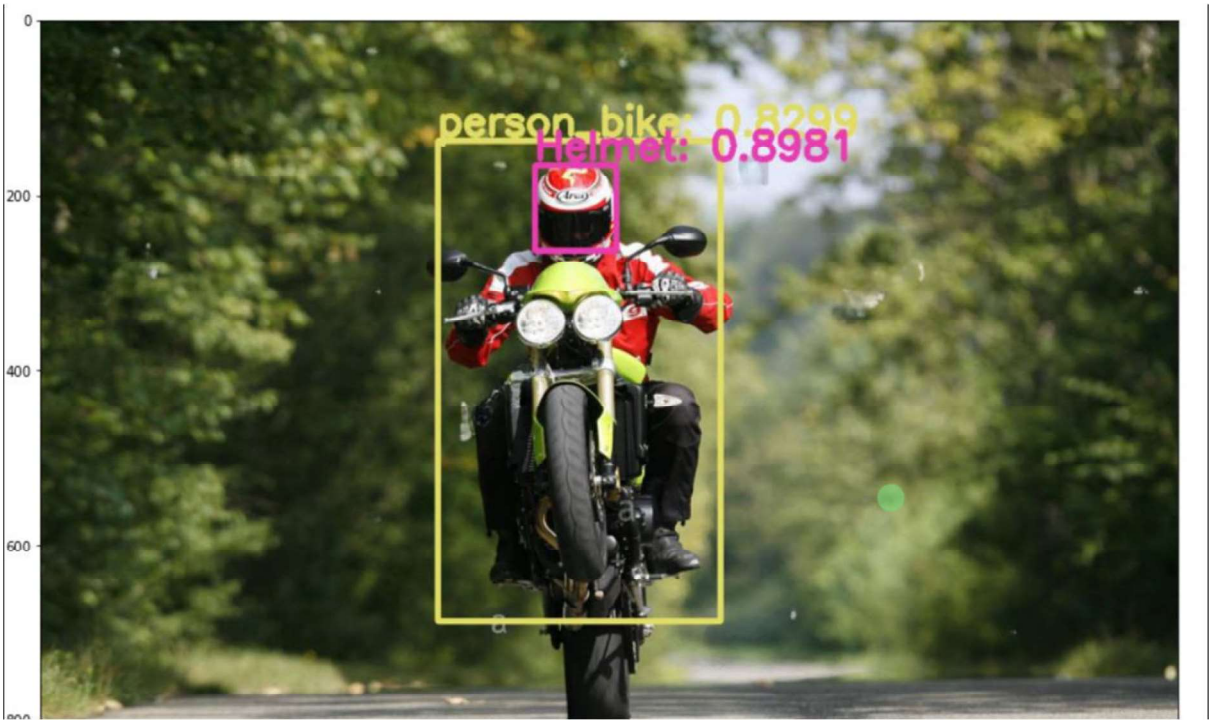

Fig 6.1.1 Object detection and classification

In figure 6.1.1 the given input is processed and the objects are detected and classes are filtered using YOLO V7 thus making the detection more accurate.

6.1.2 Object Tracking Module

Object tracking involves the process of tracking an object across a series of frames. In fig 6..1.2 we start with all possible object detections in a frame and give them an ID. In subsequent frames, we try to carry forward an object's ID. If the object has moved away from the frame then that ID is dropped. If a new object appears then they start with a fresh ID. This is important for us to remove duplication of violations count across frames. For tracking objects, we are using DeepSORT ,which is an improved variant of Kalman filter tracker .Deep sort uses position-velocity-measured Kalman filter tracking to effectively handle Multiple

Object Tracking issues like occlusion, distortion etc. DeepSORT consists of the following three steps:

- Yolo-v7 (Object detector) is used to perform the initial object detections per frame.
- Kalman filter-based estimation of the tracks of existing objects in the current frame. This uses the state of each track as a vector of eight quantities, that is, box center (x, y), box scale (s), box aspect ratio (a), and their derivatives with time as velocities. If there is no detection of the tracking object for a threshold for consecutive frames, it is considered to be out of frame or lost. Thus for every freshly detected box, a new track begins. Kalman filter further helps the problem of sudden occlusions.
- Finally, an association is made from the predicted states from Kalman filtering for the new detection with old object tracks in the previous frame using the Hungarian algorithm on bipartite graph matching. This is made robust by setting the weights of the matching with distance formulation.

Fig 6.1.2 Object tracking of input

6.1.3 License Plate Detection Module

The process of Licence Plate capturing consists of three important steps;

• Detection: An object detector, in our case Yolov7 is used to identify and localize the number plate and provide coordinates of the bounding box of the number plate in the image. The exact bounding box is cropped by image processing.

• Pre-processing: The main reason for pre-processing is to enhance the quality and readability of the cropped number plate characters before undergoing segmentation and recognition.

1) RGB to greyscale conversion

2) Gaussian blur for noise removal

3) Binarization of image to improve detection quality. The adaptive Thresholding method is used and found to give better results as we calculate the threshold of the smaller region in the image rather than the global threshold.

• Segmentation and Recognition: Tesseract is used for segmentation and character recognition. The Connected Component Analysis gathers the component outline outlines into Blobs in the first step, solely by nesting. Text lines are formed from blobs, and the lines and regions are then examined for fixed pitch or proportional text. The type of character spacing determines how the lines are divided into words. Character cells immediately chop fixed pitch text. Words in proportional text are separated using either definite or fuzzy spaces. Recognition proceeds as a two-pass process. The first pass includes an attempt to identify each word. The words that are correctly recognised are passed to an adaptive classifier as training data. As a result, the adaptive classifier has a chance to enhance outcomes for text that is located further down on the page. In order to locate small-cap text, the final phase resolves the ambiguous spaces and tests alternate x-height hypotheses.

CHAPTER 7
SYSTEM TESTING

7. SYSTEM TESTING

7.1 BLACK BOX TESTING

Black box testing is a software testing technique that focuses on testing the functionality of a software system without knowing the internal workings of the system. In this type of testing, the tester is not concerned with the code, architecture, or design of the system but instead focuses on the inputs and outputs of the system. The goal of black box testing is to determine whether the software system behaves correctly based on its specifications, requirements, and business logic. The tester typically creates test cases that simulate different scenarios and inputs to verify the expected results. Black box testing can be performed at different levels of software testing, including unit testing, integration testing, system testing, and acceptance testing. It can be automated or performed manually. The advantages of black box testing include the ability to test the software system from the end user's perspective, the ability to detect defects that may be missed in other types of testing, and the fact that it does not require knowledge of the internal workings of the system. The disadvantages of black box testing include the possibility of incomplete testing due to the limited knowledge of the system, and the fact that it may be difficult to determine the root cause of defects.

7.2 WHITE BOX TESTING

White box testing is a software testing technique that focuses on testing the internal workings of a software system. In this type of testing, the tester has access to the source code, architecture, and design of the system and uses this knowledge to create test cases that verify the correctness and quality of the system's implementation. The goal of white box testing is to ensure that the code is written correctly, follows best practices and coding standards, and meets the design specifications. The tester typically creates test cases that target specific areas of the code, such as loops, conditionals, and error handling, to ensure that all possible scenarios have been tested. White box testing can be performed at different levels of software testing, including unit testing, integration testing, and system testing. It can be automated or performed manually.

The advantages of white box testing include the ability to test the system thoroughly and ensure that all code paths have been exercised, the ability to detect defects that may be missed in other types of testing, and the ability to optimize the code for performance and efficiency.

7.3 TEST CASES

TEST REPORT: 01

PRODUCT : Detecting traffic infringement in roads

USECASE : Upload video

TEST CASE ID	TESTCASE / ACTION TO BE PERFORMED	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1	Upload the video as an input	Uploaded	Uploaded	PASS
2	Upload the video as an input	File format must be changed	File format must be changed	PASS

Table-7.3.1 Test Case For Video Upload

TEST REPORT : 02

PRODUCT : Detecting traffic infringement in roads

USECASE : Search and detect type of traffic infringement

TEST CASE ID	TESTCASE/ ACTION TO BE PERFORMED	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1	Search the appropriate type of violation from the recorded video	Searched Successfully	Searched Successfully	PASS
2	Search the appropriate type of violation from the recorded video	Vehicle not detected	Vehicle not detected	PASS
3	Show the detected type of violation from the recorded video	Detected Successfully	Detected Successfully	PASS
4	Show the detected type of violation from the recorded video	Vehicle not detected	Vehicle not detected	PASS

Table-7.3.2 Test Case For Search And Detect Type Of Traffic Infringement

TEST REPORT : 03

PRODUCT : Detecting number plates in vehicles

USECASE : Search and detect number plate

TEST CASE ID	TESTCASE/ ACTION TO BE PERFORMED	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
1	Search the appropriate number plate image from the recorded video frame by frame.	Searched Successfully	Searched Successfully	PASS
2	Search the appropriate number plate image from the recorded video frame by frame.	Number plate not detected	Number plate not detected	PASS
3	Show the detected number plates from the recorded video frame by frame.	Detected Successfully	Detected Successfully	PASS
4	Show the detected number plates from the recorded video frame by frame.	Number plate not detected	Number plate not detected	PASS

Table-7.3.3 Test Case For Search And Detect Number Plates In Vehicles

CHAPTER 8

CONCLUSION & FUTURE ENHANCEMENT

8. CONCLUSION & FUTURE ENHANCEMENT

8.1 CONCLUSION

We present a complete system for automating the detection of two-wheeler violations so that issuing tickets requires either no or very little human involvement. For the purpose of tracking and detecting violations, Yolo-v7 + DeepSORT and Yolo-v7 + Tesseract, respectively, are recommended. On test data, this implementation's mean average precision (mAP) for violation detection was 98.09% and its accuracy for number plate detection was 99.41%. The system detected 77 out of 93 violations with zero False Positive detection when tested on eight live feeds for real-life scenarios. In order to enable automatic ticketing of violations, the system was able to identify violations, effectively capture the number plate of the offenders, and write the same to a database with all pertinent data. As a result, using AI-based systems, strict regulations of traffic rule violations can be implemented to raise awareness among vehicle users and improve road safety standards. Constructing a very reliable violation model involves a number of difficulties. A lot of labelled data is needed for custom model training. It takes a lot of time and human resources to complete the task, which entails creating, collecting, and labelling numerous proprietary data sets. Additionally, there are issues with image quality because the majority of CCTV footage has low resolution, rendering the majority of the data useless. Another issue seen was inconsistent number plates with fancy fonts, number plates in local languages, and occasionally no number plates at all. These are a few of the difficulties that must be overcome in addition to the substantial amount of computational power needed to run the system effectively.

8.2 FUTURE ENCHANCEMENT

For upcoming work, the system can be strengthened by adding more categories of road vehicles and identifying corresponding vehicle-related traffic infractions. Additionally, architectures can be created to handle multiple camera views and handle more different types of violations. The OCR for detecting licence plates can be further trained to recognise number plates with unusual fonts and designs. Additionally, experiments with more recent object detection models can be used to compare models' accuracy and speed. Install high-definition cameras: High-definition cameras can capture clear images and videos, which will help in better detecting and identifying helmet and triple riding violations. These cameras can be

installed at key intersections and on patrol vehicles to provide a more comprehensive view of the traffic situation. Use real-time data analysis: Analyse the data collected by the cameras in real-time to identify potential violations. This could involve using computer vision algorithms to recognize patterns and identify potential helmet or triple riding violations, and alerting traffic police officers immediately. Integrate with existing traffic management systems: Integrate the helmet and triple riding detection system with existing traffic management systems. This would enable the system to automatically issue fines and summon notices to the violators, and also help traffic police officers in better managing traffic flow. Conduct awareness campaigns: Conduct awareness campaigns among two-wheeler riders to educate them about the importance of following traffic rules and regulations. This would help in reducing the number of violations and making roads safer for everyone.

REFERENCES

REFERENCES

[1] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Mark Liao, “YOLOv4: Optimal speed and accuracy of object detection”, 2020, arXiv:2004.10934.

[2] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime racking with a deep association metric,” in Proc. IEEE Int. Conf. Image Process. (ICIP), Sep. 2017, pp. 3645–3649, doi: 10.1109/ICIP.2017.8296962.

[3] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple online and realtime tracking,” in Proc. IEEE Int. Conf. Image Process. (ICIP), Sep. 2016, pp. 3464–3468, doi: 10.1109/ICIP.2016.7533003.

[4] R. Smith, “An overview of the Tesseract OCR engine,” in Proc. 9th Int. Conf. Document Anal. Recognit. (ICDAR), Sep. 2007, pp. 629–633, doi: 10.1109/ICDAR.2007.4376991.

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, JitendraMalik(2014), “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation.” The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 580-587.

[6] Xiaoling Wang “A Video-based Traffic Violation Detection System”, https://www.researchgate.net/publication/271545712, December 2013 DOI: 10.1109/MEC.2013.6885246

[7] S. Awang, and NMAN Azmi, "Vehicle counting system based on vehicle type classification using deep learning method." In IT Convergence and Security 2017, pp. 52-59. Springer, Singapore, 2018.

[8] Maharsh Desai, "Automatic Helmet Detection on Public Roads",in International Journal of Engineering Trends and Technology · May 2016 DOI: 10.14445/22315381/IJETT-V35P241

[9] Deeksha N , "Detection of Bike Riders without Helmet and Triple Riding using YOLO v3 model", International Journal of Scientific Research in Engineering and Management (IJSREM), Volume: 06 Issue: 06 | June - 2022 Impact Factor: 7.185 ISSN: 2582-3930

[10] Chien-Yao Wang, “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors.”, https://arxiv.org/abs/2207.02696, 16-07-2022

[11] Debby Ratna Daniel E-Traffic Operational Information System Based on Automatic Number Plate Recognition (ANPR) System as a Tool to Detect Traffic Violation and to

Manage the Traffic Fines in Indonesia", JCAE Symposium 2018 – Journal of Contemporary Accounting and Economics Symposium 2018 on Special Session for Indonesian Study

[12] Sangita Kumari," A Novel Methodology for Vehicle Number Plate

Recognition using Artificial Neural Network", Third International Symposium on Computer Vision and the Internet (VisionNet), September 2016

[13] Chirag Patel," Automatic Number Plate Recognition System (ANPR): A Survey",

in International Journal of Computer Applications · May 2013

[14] M. S. Sarfraz et al., "Real-Time automatic license plate recognition for CCTV forensic applications," Journal of Real-Time Image Processing- Springer Berlin/Heidelberg, 2011.

[15] Aishwarya Agrawal, N. P. (2017). AUTOMATIC LICENCE PLATE RECOGNITION USING RASPBERRY PI. International Interdisciplinary Conference on Science Technology Engineering Management Pharmacy and Humanities. Singapore.

[16] Xinyu Hou ," Vehicle Tracking Using Deep SORT with Low Confidence Track Filtering", 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 25 November 2019,INSPEC Accession Number: 19173920, DOI: 10.1109/AVSS.2019.8909903.

[17] K. Saho, "Kalman Filter for Moving Object Tracking: Performance Analysis and Filter Design," Kalman Filters - Theory for Advanced Applications, Feb. 2018, doi: 10.5772/intechopen.71731.

APPENDICES

APPENDICES

A.1 CODING

```
import os
from operator import index
from number_plate import npDetectionTest
# comment out below line to enable tensorflow logging outputs
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import time
import tensorflow as tf
import tensorflow.keras as keras
physical_devices = tf.config.experimental.list_physical_devices('GPU')
if len(physical_devices) > 0:
    tf.config.experimental.set_memory_growth(physical_devices[0], True)
from absl import app, flags, logging
from absl.flags import FLAGS
import core.utils as utils
from core.yolov4 import filter_boxes
from core.video import det_to_vid, interpolation
from core.association import *
from tensorflow.python.saved_model import tag_constants
from core.config import cfg
from PIL import Image
import cv2
import numpy as np
import pandas as pd
import pickle
from shapely.geometry import Polygon
import matplotlib.pyplot as plt
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
# deep sort imports
```

```
from deep_sort import preprocessing, nn_matching

from deep_sort.detection import Detection

from deep_sort.tracker import Tracker

from tools import generate_detections as gdet

flags.DEFINE_string('framework', 'tf', '(tf, tflite, trt')

#flags.DEFINE_string('weights', './checkpoints/yolov4-416',

#                    'path to weights file')

#--------------------change the model paths

flags.DEFINE_string('weights_L4', r'C:\Users\aruns\OneDrive\Desktop\Two-Wheeler Vehicle Traffic Violations Detection M1\m3\checkpoints\yolov4-416_L4/','path to weights file')

flags.DEFINE_string('weights_RHNH',r'C:\Users\aruns\OneDrive\Desktop\Two-Wheeler Vehicle Traffic Violations Detection M1\m3\checkpoints\yolov4-416-FG/','path to weights file')

flags.DEFINE_string('classes', './data/classes/4_class_detector.names','path to manes file')

flags.DEFINE_integer('size', 416, 'resize images to')

flags.DEFINE_float('rider_pred_threshold', 1.5, 'IOU/NIOU area threshold')

flags.DEFINE_boolean('tiny', False, 'yolo or yolo-tiny')

flags.DEFINE_string('model', 'yolov4', 'yolov3 or yolov4')

flags.DEFINE_string('trapezium_pred_model', r'C:\Users\aruns\OneDrive\Desktop\Two-Wheeler Vehicle Traffic Violations Detection M1\m3\data\Trapezium_Prediction_Weights.pickle', 'add the model weights for predicting trapezium bounding box as a post-processing step')

flags.DEFINE_string('video', './data/video/3idiots.mp4', 'path to input video or set to 0 for webcam')

flags.DEFINE_string('output', './outputs/detections/3idiots.mp4', 'path to raw output video')

flags.DEFINE_string('output_format', 'mp4v', 'codec used in VideoWriter when saving video to file')

flags.DEFINE_float('iou', 0.45, 'iou threshold')

flags.DEFINE_float('score', 0.50, 'score threshold')

flags.DEFINE_boolean('dont_show', False, 'dont show video output')

flags.DEFINE_boolean('info', False, 'show detailed info of tracked objects')

flags.DEFINE_boolean('count', False, 'count objects being tracked on screen')

flags.DEFINE_boolean('interpolation', True, 'interpolate the missing bounding boxes based on future frames')

def get_instance(rider, motorcycle, iou_threshold):
```

```
    """
    args:
    rider, motorcycle : pd.DataFrame
    output:
    rider, motorycle : pd.DataFrame with a column named 'instance_id'
    """
    rider['instance_id'] = -1
    motorcycle['instance_id'] = -1
    for i in range(len(motorcycle)):
        motorcycle.iat[i,motorcycle.columns.get_loc('instance_id')] = i
        for j in range(len(rider)):
            if (motor_rider_iou(motorcycle.iloc[i], rider.iloc[j]) > iou_threshold):
                if (rider.iloc[j]['instance_id'] == -1):
                    rider.iat[j,rider.columns.get_loc('instance_id')] = i
                else:
                    instance = int(rider.iloc[j]['instance_id'])
                    instance_final = motor2_rider_iou(motorcycle.iloc[i], motorcycle.iloc[instance],
rider.iloc[j], i, instance)
                    rider.iat[j,rider.columns.get_loc('instance_id')] = instance_final
    return rider, motorcycle
            if FLAGS.interpolation:
                print('Improving tracking quality by interpolation process....')
                dir_path_ = os.path.dirname(FLAGS.output)
                path_ = os.path.join(dir_path_,
os.path.basename(FLAGS.output).split(".")[0]+"_interpolated."+os.path.basename(FLAGS.o
utput).split(".")[1])
                det_to_vid(FLAGS.video, interpolated_detections, path_)
            else:
                print('Generation of camera ready video in progress....')
                dir_path_ = os.path.dirname(FLAGS.output)
                path_ = os.path.join(dir_path,
os.path.basename(FLAGS.output).split(".")[0]+"_interpolated"+os.path.basename(FLAGS.ou
tput).split(".")[1])
                det_to_vid(FLAGS.video, path, path_)
```

```
            break
        frame_num +=1
        print('Frame #: ', frame_num)
        frame_size = frame.shape[:2]
        image_data = cv2.resize(frame, (input_size, input_size))
        image_data = image_data / 255.
        image_data = image_data[np.newaxis, ...].astype(np.float32)
        start_time = time.time()

        # run detections on tflite if flag is set
        if FLAGS.framework == 'tflite':
            interpreter.set_tensor(input_details[0]['index'], image_data)
            interpreter.invoke()
            pred = [interpreter.get_tensor(output_details[i]['index']) for i in
range(len(output_details))]
            # run detections using yolov3 if flag is set
            if FLAGS.model == 'yolov3' and FLAGS.tiny == True:
                boxes, pred_conf = filter_boxes(pred[1], pred[0], score_threshold=0.25,
                                                input_shape=tf.constant([input_size, input_size]))
            else:
                boxes, pred_conf = filter_boxes(pred[0], pred[1], score_threshold=0.25,
                                                input_shape=tf.constant([input_size, input_size]))
        else:
            batch_data = tf.constant(image_data)
            pred_bbox = infer_L4.predict(batch_data)
            pred_bbox1 = infer_RHNH.predict(batch_data)
            for value in pred_bbox:
                temp_value = np.expand_dims(value, axis=0)
                boxes = temp_value[:, :, 0:4]
                pred_conf = temp_value[:, :, 4:]
        if FLAGS.count:
            #cv2.putText(frame, "Objects being tracked: {}".format(count), (5, 35),
cv2.FONT_HERSHEY_COMPLEX_SMALL, 2, (0, 255, 0), 2)
```

```
        print("Objects being tracked: {}".format(num_objects))
    # Bounding boxes are in normalized ymin, xmin, ymax, xmax
    original_h, original_w, _ = frame.shape
     #getting rider, motorcycle dataframe
    df = pd.DataFrame(classes, columns=['class_id'])
    ymin = bboxes[:, 0]
    xmin = bboxes[:, 1]
    ymax = bboxes[:, 2]
    xmax = bboxes[:, 3]
    df['x'] = pd.DataFrame(xmin + (xmax-xmin)/2, columns=['x'])
    df['y'] = pd.DataFrame(ymin + (ymax-ymin)/2, columns=['y'])
    df['w'] = pd.DataFrame(xmax-xmin, columns=['w'])
    df['h'] = pd.DataFrame(ymax-ymin, columns=['h'])
    rider = df.loc[df['class_id']==0]
    motorcycle = df.loc[df['class_id']==3]
   #predicting trapezium
    rider, motorcycle = get_instance(rider, motorcycle, 0.01)
    y = np.zeros((len(motorcycle), 8))
    num = 0
    tracker_instance = []
    for i in range(len(motorcycle)):
        input = []
        motor = motorcycle.iloc[i]
        instance = motor['instance_id']
        input.extend([float (motor['x']),float (motor['y']),float (motor['w']),float (motor['h'])])
        rider_ins = rider.loc[rider['instance_id']==instance]
        if (len(rider_ins)==0):
            tracker_instance.append([-1, -1])
            continue
            for j in range(len(rider_ins)):
            input.extend([float (rider_ins.iloc[j]['x']),float (rider_ins.iloc[j]['y']),float
(rider_ins.iloc[j]['w']),float (rider_ins.iloc[j]['h'])])
```

```
            tracker_instance.append([num, len(rider_ins)])
            x=np.zeros((1,24))
            x[0,:len(input)] = np.array(input).reshape((1,-1))
            predict = trapez_model.predict(x)
            a = predict[0]
            a = heuristic_on_pred(a, motor, rider_ins)
            y[num][0], y[num][1], y[num][2], y[num][3], y[num][4], y[num][5],y[num][6]
,y[num][7]  = (a[0] - a[4]/2)*original_w,(a[5]+x[0][1]-x[0][3]/2)*original_h, (a[0] -
a[4]/2)*original_w,(a[2]+x[0][1]+x[0][3]/2)*original_h, (a[0] + a[4]/2)*original_w,
(a[3]+x[0][1]+x[0][3]/2)*original_h, (a[0] + a[4]/2)*original_w, (a[6]+x[0][1]-
x[0][3]/2)*original_h
            y[num] = corner_condition(y[num], original_w, original_h)
            num = num+1
        trapez_bboxes = y[:num,:]
        deleted_indx = []
        instance = np.zeros((len(bboxes), 2), dtype = int)
        k = 0
        for i in range(len(bboxes)):
            if (classes[i]==3):
                if (int(tracker_instance[k][1]) == -1):
                    deleted_indx.append(i)
                else:
                    instance[i][:] = int(tracker_instance[k][0]), int(tracker_instance[k][1])
                    k = k+1
            else:
                instance[i][:] = -1, -1
        bboxes = np.delete(bboxes, deleted_indx, axis=0)
        scores = np.delete(scores, deleted_indx, axis=0)
        classes = np.delete(classes, deleted_indx, axis=0)
        num_objects = len(classes)
        # format bounding boxes from normalized ymin, xmin, ymax, xmax ---> xmin, ymin,
width, height
        original_h, original_w, _ = frame.shape
        bboxes = utils.format_boxes(bboxes, original_h, original_w)
```

```
        # store all predictions in one parameter for simplicity when calling functions
        pred_bbox = [bboxes, scores, classes, num_objects]
        # read in all class names from config
        class_names = utils.read_class_names(cfg.YOLO.CLASSES)
        # by default allow all classes in .names file
        allowed_classes = list(class_names.values())
        # custom allowed classes (uncomment line below to customize tracker for only people)
        #allowed_classes = ['person']
        # loop through objects and use class index to get class name, allow only classes in allowed_classes list
        names = []
        deleted_indx = []
        # encode yolo detections and feed to tracker
        features = encoder(frame, bboxes)
        detections = [Detection(bbox, score, class_name, feature) for bbox, score, class_name, feature in zip(bboxes, scores, names, features)]

        #initialize color map
        cmap = plt.get_cmap('tab20b')
        colors = [cmap(i)[:3] for i in np.linspace(0, 1, 20)]

        # run non-maxima supression
        boxs = np.array([d.tlwh for d in detections])
        scores = np.array([d.confidence for d in detections])
        classes = np.array([d.class_name for d in detections])
        indices = preprocessing.non_max_suppression(boxs, classes, nms_max_overlap, scores)
        detections = [detections[i] for i in indices]

        # Call the tracker
        tracker.predict()
        tracker.update(detections)
        # update tracks
        num = 0
```

```
j=0
track_ids = []
nums = []
riders = []
vals = []
for track in tracker.tracks:
    if not track.is_confirmed() or track.time_since_update > 1:
        continue
    bbox = track.to_tlbr()
    class_name = track.get_class()
    if (class_name != 'Motorcycle'):
        continue
    num, num_riders, val = find(bbox, instance, bboxes, classes)
    if (num==-1):
        continue
    if (num not in nums):
        track_ids.append(track.track_id)
        nums.append(num)
        riders.append(num_riders)
        vals.append(val)
        continue
    idx = nums.index(num)
    if (val<=vals[idx]):
        track_ids[idx] = track.track_id
        vals[idx] = val
        riders[idx] = num_riders
    for track in tracker.tracks:
    if not track.is_confirmed() or track.time_since_update > 1:
        continue
    bbox = track.to_tlbr()
    class_name = track.get_class()
    color = [255, 0, 0]
```

```
            if (class_name == 'No-Helmet'):
                if track not in HNH_violation:
                    HNH_violation[track] = 1
                if track in HNH_violation:
                    HNH_violation[track]+=1
                if HNH_violation[track]==2:
                    HNH_violated.append(track)
             if track in HNH_violated:
                color = [255, 0, 0]
                cv2.rectangle(frame, (int(bbox[0]), int(bbox[1])), (int(bbox[2]), int(bbox[3])), color, 2)
                cv2.rectangle(frame, (int(bbox[0]), int(bbox[1]-30)), (int(bbox[0])+(5+len(str(track.track_id)))*17, int(bbox[1])),(255,0,0), -1)
                cv2.putText(frame,  "ID:" + str(track.track_id),(int(bbox[0]), int(bbox[1]-10)),0, 0.75, (255,255,255),2)
                detections_tovid.append([str(frame_num), "No-Helmet", str(track.track_id), "0", "0", "0", "0", "0", "0", "0", "0", str(int(bbox[0])), str(int(bbox[1])), str(int(bbox[2])), str(int(bbox[3])) ])
                continue
            if (class_name == "Helmet"):
                color = [0,255, 0]
                cv2.rectangle(frame, (int(bbox[0]), int(bbox[1])), (int(bbox[2]), int(bbox[3])), color, 2)
                cv2.rectangle(frame, (int(bbox[0]), int(bbox[1]-30)), (int(bbox[0])+(5+len(str(track.track_id)))*17, int(bbox[1])),(0,255,0), -1)
                cv2.putText(frame,  "ID:" + str(track.track_id),(int(bbox[0]), int(bbox[1]-10)),0, 0.75, (255,255,255),2)
                detections_tovid.append([str(frame_num), "Helmet", str(track.track_id), "0", "0", "0", "0", "0", "0", "0", "0", str(int(bbox[0])), str(int(bbox[1])), str(int(bbox[2])), str(int(bbox[3])) ])
                continue
            if class_name == 'Motorcycle':
                if (track.track_id not in track_ids):
                    continue
```

A.2 SAMPLE SCREENS

In fig A.2.1 for processing the input we need to activate the virtual environment in the anaconda prompt using the command conda activate yolov7-cpu

Fig A2.1 Virtual Environment Activation

In fig A.2.2 The video is given as input to the python run file which starts the processing of the video frame by frame which runs the algorithm in it.

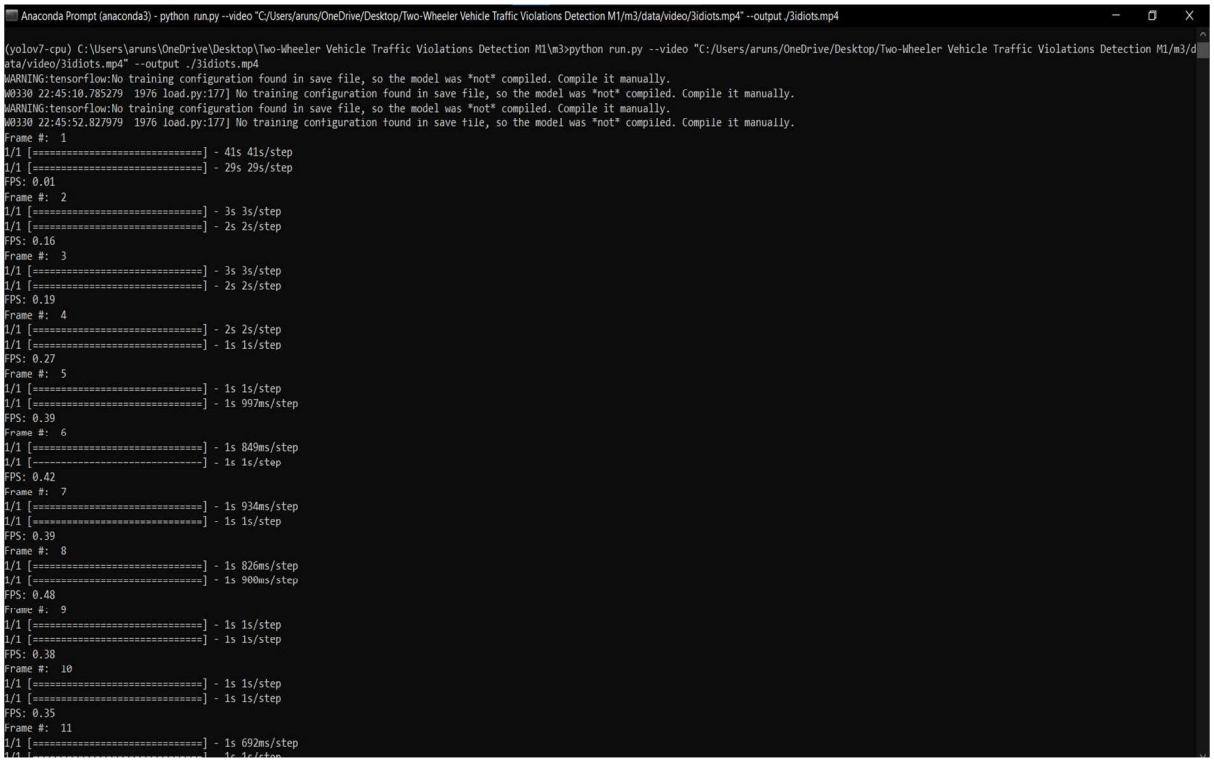

Fig A2.2 Executing The Input

In fig A2.3 The application gui is opened which displays frame by frame detection of the infringement such as helmet and triple rider violation.

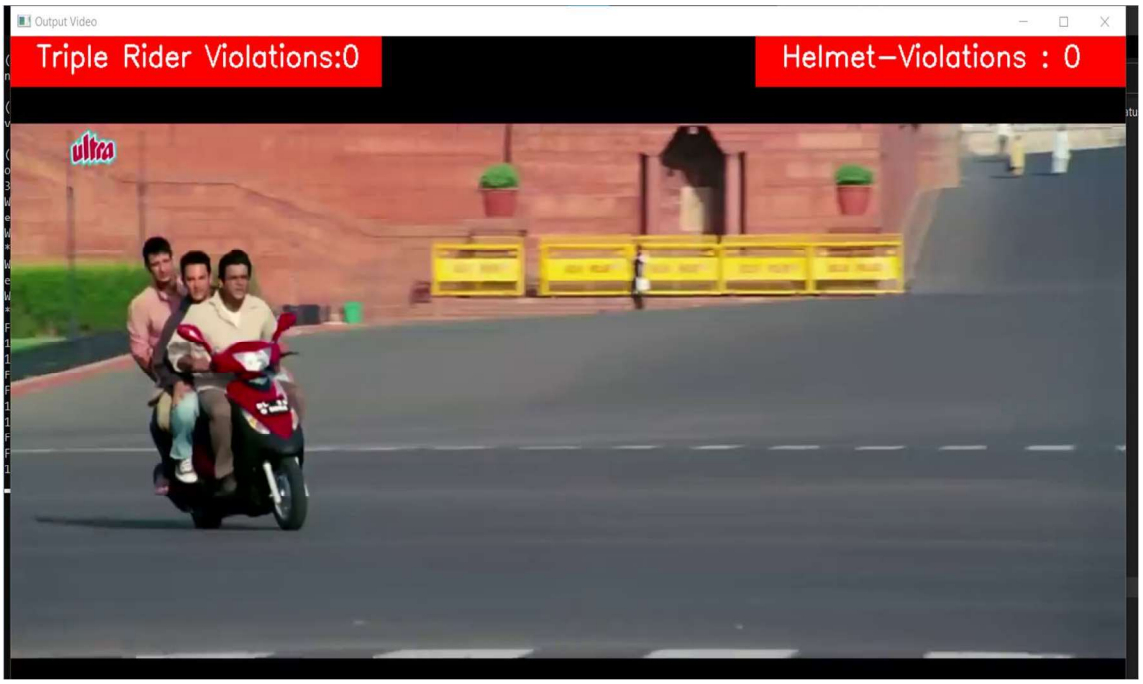

Fig A2.3 Frame By Frame Detection

In A2.4 The application that tracks which makes the markings of the detected infringement and saves the process as an output video.

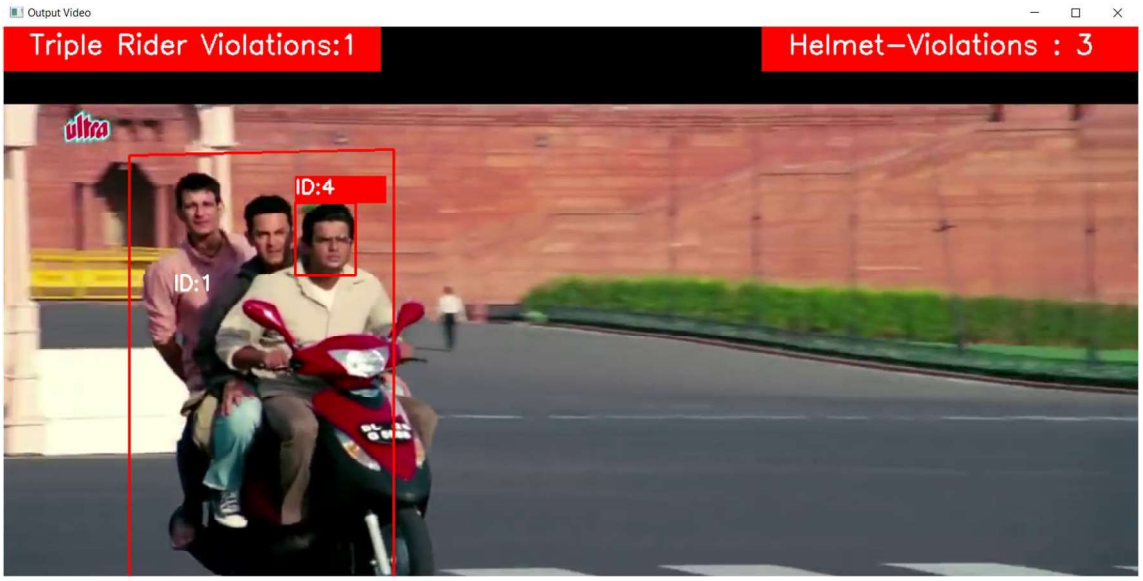

Fig A2.4 Infringement Detection In The Video

In fig A2.5 The ticket is issued to the mailed where the number plate of the violated vehicle is clipped and attached.

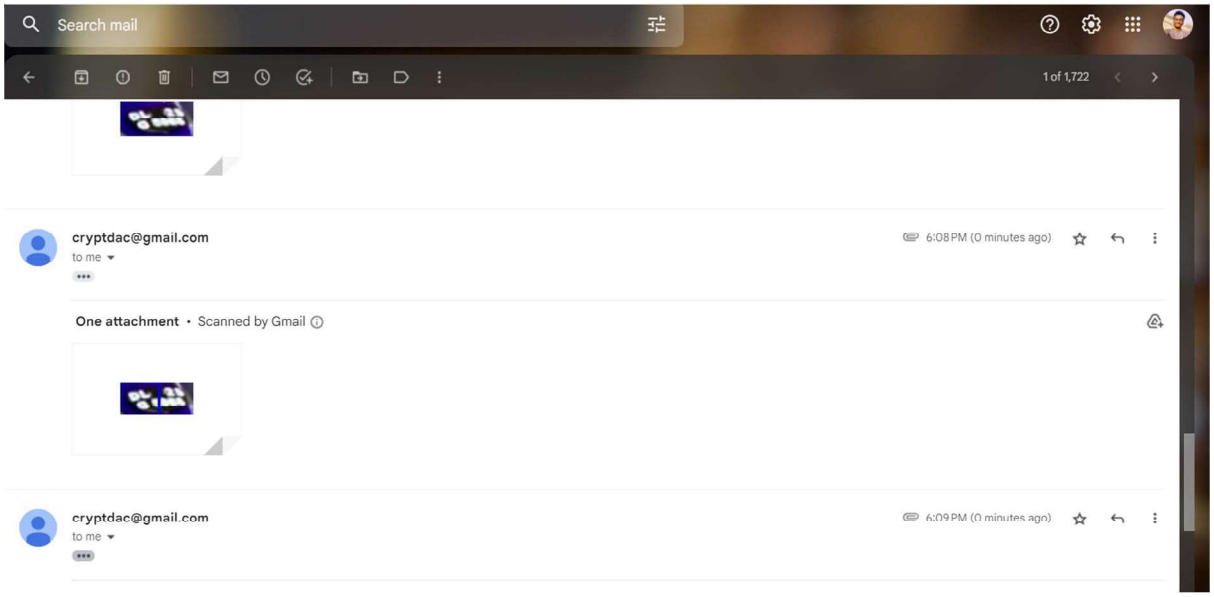

Fig A2.5 Alert Notification To Mail