

**URBAN TRAFFIC ADMINISTRATION SYSTEM**  
**A MAJOR PROJECT REPORT**

*Submitted by*

**Gevariya Meet B [Reg No: RA1911028010131]**

**Pratiksha Ghosh [Reg No: RA1911028010142]**

*Under the Guidance of*

**Dr. Murugaanandam S**

(Associate Professor, Department of Networking and Communications)

*in partial fulfillment of the requirements for the degree of*

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE ENGINEERING**  
**with specialization in CLOUD COMPUTING**



**DEPARTMENT OF NETWORKING AND COMMUNICATIONS**  
**COLLEGE OF ENGINEERING AND TECHNOLOGY**  
**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**  
**KATTANKULATHUR- 603 203**

**MAY 2023**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**  
**KATTANKULATHUR- 603203**

**BONAFIDE CERTIFICATE**

Certified that 18CSP109L major project report titled “**Urban Traffic Administration System**” is the Bonafide work of “**Mr. Gevariya Meet [RA1911028010131] and Ms. Pratiksha Ghosh [RA1911028010142]**” who carried out the major project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

**SIGNATURE**

**Dr. Murugaanandam S.**

**SUPERVISOR**

Associate Professor,  
Department of Networking and  
Communications

**SIGNATURE**

**Dr. Annapurani K.**

**HEAD OF THE DEPARTMENT**

Professor,  
Department of Networking and  
Communications

**Signature Of Internal Examiner**

**Signature Of External Examiner**



**Department of Networking and Communications**  
**SRM Institute of Science and Technology**  
**Own Work\* Declaration Form**

**Degree/ Course : B.Tech. - Computer Science and Engineering**  
**Student 1 Name : Gevariya Meet B**  
**Registration Number : RA1911028010131**  
**Student 2 Name : Pratiksha Ghosh**  
**Registration Number : RA1911028010142**  
**Title of Work : Urban Traffic Administration System**

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism\*\*, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that we have met the following conditions:

- Clearly references / listed all sources as appropriate
- Given the sources of all pictures, data etc. that are not my own
- Acknowledged in appropriate places any help that we have received from others (e.g.fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook /University website

We understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

**DECLARATION:**

We are aware of and understand the University's policy on Academic misconduct and plagiarism and we certify that this assessment is our own work, except where indicated by referring, and that we have followed the good academic practices noted above.

**DATE:**

Pratiksha Ghosh  
 RA1911028010142

Gevariya Meet B  
 RA1911028010131

## ACKNOWLEDGEMENT

We express our gratitude to **Dr C. Muthamizhchelvan**, Vice-Chancellor and **Dr T.V. Gopal**, Dean-CET, SRM IST, for the facilities extended for the project work and their continued support.

We wish to thank **Dr Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work. We are immensely thankful to **Dr. Annapurani K**, Professor and Head of the Department, Department of Networking and Communications, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the work.

We want to convey our thanks to our project coordinator **Dr TYJ Nagamalleswari**, Associate Professor, Department of Networking and Communications, SRM Institute of Science and Technology, and Panel Head, **Dr N. Prasath**, Associate Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We wish to convey our respect and thanks to our guide, **Dr. Murugaanandam S**, Associate Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for providing us with all the support to research on the topics of our interest. His passion for solving world problems has inspired us.

We sincerely thank our Faculty Advisor, **Dr Balamurugan P**, Assistant Professor, Department of Networking and Communications, SRM Institute of Science and Technology and all the other Networking and Communications Department staff and students for their help during our project. Finally, we would like to thank parents, family members, and friends for their constant support and encouragement.

**Pratiksha Ghosh [RA1911028010142]**  
**Gevariya Meet B [RA1911028010131]**

## **ABSTRACT**

Traffic congestion is a major problem in many cities, and the fixed-cycle light signal controllers are not resolving the high waiting time in the intersection. It is seen that policemen manage the movements instead of the traffic light. This human achievement encourages us to create a smart Traffic control system, considering the real time traffic condition. In the previous models we have observed a high level of focus given to vehicular movements. But as per our daily life observation, traffic jams and accidents are also increased due to the interference of pedestrians.

In order to ensure safety of pedestrians, our model suggests building subways or footbridges so that the on-road traffic can be tackled. In areas with high traffic density, a subway can eliminate the on-foot traffic and thus the cars can travel without worrying much about the pedestrians. Our main aim is to reduce the manual workload of the traffic police and automate the process. So the detection of vehicular movements and Helmets has been performed using YOLO v3, and pedestrian detection using HOG and SVM classification models.

# TABLE OF CONTENTS

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
	<b>BONAFIDE CERTIFICATE</b>	<b>ii</b>
	<b>ACKNOWLEDGEMENT</b>	<b>iv</b>
	<b>ABSTRACT</b>	<b>v</b>
	<b>TABLE OF CONTENTS</b>	<b>vi</b>
	<b>LIST OF FIGURES</b>	<b>viii</b>
	<b>LIST OF TABLES</b>	<b>x</b>
	<b>LIST OF ABBREVIATIONS</b>	<b>xi</b>
	<b>LIST OF SYMBOLS</b>	<b>xii</b>
<b>1.</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 MOTIVATION FOR WORK	4
	1.2 PROBLEM STATEMENT	4
<b>2.</b>	<b>LITERATURE SURVEY</b>	<b>6</b>
<b>3.</b>	<b>SYSTEM DESIGN</b>	<b>10</b>
	3.1 SYSTEM REQUIREMENTS	10
	3.2 SYSTEM ARCHITECTURE	13
<b>4.</b>	<b>METHODOLOGY</b>	<b>16</b>
	4.1 PROPOSED SYSTEM	16
	4.2 COMPONENTS	17
	4.3 UML DIAGRAMS	25
	4.4 ALGORITHMS USED	30

<b>5.</b>	<b>CODING AND TESTING</b>	<b>49</b>
<b>6.</b>	<b>RESULTS AND DISCUSSION</b>	<b>54</b>
<b>7.</b>	<b>CONCLUSION AND FUTURE WORK</b>	<b>60</b>
	<b>REFERENCES</b>	<b>62</b>
	<b>APPENDIX</b>	<b>65</b>
	SOURCE CODE	65
	PLAGIARISM REPORT	87
	JOURNAL PUBLICATION	89

## LIST OF FIGURES

FIGURE NO.	DESCRIPTION	PAGE NO.
3.1	Traffic Signal Violation Model	14
3.2	HOG and SVM Integration	15
4.1	Bounding Box	18
4.2	CNN Working	19
4.3	Object Identification	21
4.4	Number Plate Extraction	24
4.5	Block Diagram Module 1	25
4.6	Sequence Diagram- Module 2	26
4.7	Block Diagram- Module 2	27
4.8	Use Case Diagram- Module 3	28
4.9	Block Diagram- Module 3	29
4.10	Feature Pyramid Networks	31
4.11	Comparison Graph	31
4.12	YOLO Architecture	33
4.13	Working of HOG	36
4.14	SVM Algorithm	40
4.15	SVM Architecture	40
4.15.1	SVM Sample Image	42



4.15.2	SVM Sample Result	44
4.16	OCR Implementation	46
4.17	Representation of image using 0's and 1's	47
4.18	Text Extraction in OCR	48
5.1	Home Page	49
5.2	Input Selection	50
5.3	Video Preview	51
5.4	Marking Region of Interest	52
5.5	Detecting Pedestrians	53
6.1	Result of Module 1	54
6.2	Result of Module 2	55
6.3	Result of Module 3	56
6.4	Algorithm Comparison- FPS	58
6.5	Algorithm Comparison- Accuracy	59

## LIST OF TABLES

Table 4.1: Darknet 53

Table 6.1: Comparison of Algorithms

## LIST OF ABBREVIATIONS

<b>ATS</b>	Automated Traffic System
<b>ML</b>	Machine Learning
<b>SVM</b>	Support Vector Machine
<b>AI</b>	Artificial Intelligence
<b>NN</b>	Neural Networks
<b>DRL</b>	Deep Reinforcement Learning
<b>UML</b>	Unified Modelling Language
<b>HOG</b>	Histogram of Gradients
<b>OCR</b>	Optical Character Recognition System
<b>YOLO</b>	You Only Look Once
<b>V3</b>	Version 3
<b>CNN</b>	Convolutional Neural Network
<b>ANPR</b>	Automatic Number Plate Recognition
<b>PTPS</b>	Public Transport Priority Systems
<b>NLP</b>	Natural Language Processing
<b>RNN</b>	Recurrent Neural Network
<b>RCNN</b>	Region based Convolutional Neural Network

## LIST OF SYMBOLS

$X_t$	Input at current state
$X(t-1)$	Input at Previous state
$C_t$	Current State
$C(t-1)$	Previous State
$h_t$	Current hidden/output State
$h(t-1)$	Previous hidden/output State
$\sigma$	Sigmoid Function
$\tanh$	Hyperbolic tangent function

# CHAPTER 1

## INTRODUCTION

Many cities struggle with traffic congestion, and the fixed-cycle light signal controllers are unable to reduce the lengthy wait times at intersections. In place of the traffic signal, police officers are often seen controlling traffic. He determines how the route is doing and the length of a given signal. This human accomplishment motivates us to develop a smart traffic management system that considers the current traffic situation.

Agreeing to the overview, seven hours of travel time were included to the ordinary person's annually plan in 1982 when they lived in one of the 75 major cities within the country. By 2001, that number had expanded to 26 hours of delays every year, and what was once known as the "surge hour" had extended to about six hours each day, with the normal "surge hour" trip taking about 40% longer than the same trip at other times of the day.

In spite of changes in activity blockage shirking over the past a few decades, numerous communities still have trouble legitimately overseeing it. Pre-COVID-19 clog was greater than it was within the early 1980s [1], concurring to the USDOT, in spite of the US having developed more than 870,000 path miles of roadway between 1980 and 2021.

Since the effective operation of the economy and instructive frameworks depend on individuals working, going to school, and indeed running errands amid generally the same hours so they can associated with one another, activity blockage isn't fundamentally a issue but or maybe the arrangement to our fundamental portability issue, which is that as well numerous individuals need to move at the same times

each day. Without harming our economy and culture, changing that crucial need is inconceivable. Each critical city on soil has the same issue.

Urban Activity Organization Framework takes activity designs and vehicle movements in developing countries under consideration and responds in genuine time. To encourage day-to-day operations, it offers a user-friendly interface and leverages downstream discovery. The Versatile Activity Control Framework (VATCF) calculation creates enhanced red-green stages of activity lights and combines intersection activity information into a central activity framework to achieve junctions' green-green synchronization over the arrangement region. ATCS quickly alters in genuine time to changing activity circumstances. Flag timings that are best for the current activity circumstance are decided by ATCS utilizing machine learning calculations to look at real-time activity information from vehicle locators [2]. By evaluating the stream of activity at intersections along the corridors or over the whole area of deployment, the length of the red-green phases and green waves of the traffic light is automatically altered every cycle.

A profoundly common kind of transportation in essentially each country is the two-wheeler. Due to the need of security, there is a considerable threat related. The rider of a two-wheeler included in a mischance gets flung from the vehicle owing to a fast deceleration. Whereas the head stops moving when it collides with a thing, the brain, which has its claim mass, proceeds to move until the question impacts the insides locale of the cranium. This kind of head harm might some of the time be dangerous. Protective caps spare lives in such circumstances. The movement of the head is about irrelevant since wearing a protective cap diminishes the probability that the cranium will be moderated down [4].

As time passes, the head comes to a halt after a contact is padded inside the protective cap. Moreover, it scatters the constrain over a more prominent locale,

securing the cranium from genuine harm. More essentially, it serves as a mechanical shield between the rider's head and any adjacent objects. In case a high-quality total head protector is utilized, wounds may be diminished.

In arrange to radically diminish the peril of fatalities and wounds, activity laws are outlined to instill a feeling of discipline. Be that as it may, there's a need of exacting regard to these controls. To illuminate these issues, successful and viable strategies must be created. It is strongly suggested for bicycle riders to wear a head protector to reduce the peril related. It is concerning to note that India leads the world in terms of fatalities from activity mischances. Agreeing to investigations by specialists, this propensity is due in portion to fast urbanization and a need of utilize of situate belts, protective caps, and other security gear when driving. India guaranteed to bringing down activity fatalities to 50% by 2020 when it marked the Brasilia Statement on Street Security in 2015.

Governments have pronounced it illicit to ride a bicycle without a protective cap after figuring it out their esteem and have executed manual requirement strategies to capture guilty parties. The right now utilized advances based on video checking, in any case, are inactive and intensely dependent on human work. Since individuals are included and their productivity decreases with time, such frameworks are regularly not practicable.

For precise and exhaustive checking of these infractions, as well as for the expansive decrease within the amount of human assets required, robotization of this handle is especially wanted.

Recent studies have effectively completed this assignment utilizing highlights from CNN, R-CNN, LBP, Hoard, HaaR, etc. Be that as it may, these endeavors are obliged in terms of viability, accuracy, or the rate at which question discovery and categorization is carried out.

## **1.1 MOTIVATION FOR WORK**

A time saving is one of many reasons for why a traffic control system should be developed. Additionally, it greatly improves convenience for the manual labor-intensive traffic managers, lessens the need for workers to be stationed on-site, etc.

It is much simpler for the traffic guards to make choices in the moment, minimize accidents, reduce pollution, etc., when they have a single tool to monitor and keep track of the numerous components of traffic management.

## **1.2 PROBLEM STATEMENT**

In regular traffic signals, the timings for each lane are either automated and programmed to certain periods of a few minutes, or they are manually handled by the traffic cops. Consequently, a lane with a comparatively lower vehicle density may be left open for an unnecessary long period, while a lane that needs more time to clear out its traffic may not receive that extra time.

The majority of the solutions now in use in this field are centred on road vehicles and use their technology. The traffic controllers, though, take on much more than that. Managing pedestrian density, instructing them to utilise zebra crossings exclusively, urging them to use the tube instead of physically crossing major highways, etc.

The project's goal is to automate the system for detecting visitors signal violations and make it simple for the visitors' police branch to monitor traffic and take swift action against the owner of the offending vehicle. The system's first goal is accurately detecting and monitoring the automobile and its occupants' behaviours.



In addition, our technology will be able to identify people crossing roadways not marked with zebras. Based on the typical pedestrian density in a given location, our algorithm will also provide recommendations about the need for overbridges or subway.

## CHAPTER 2

### LITERATURE SURVEY

For proper traffic light control, authors proposed a system that integrates image processing algorithms with real-time traffic light management [1].

In order to effectively regulate traffic, he presented an FPGA (field Programmable Gate Array) controller based on the Neuro-Fuzzy device idea. It may be utilized to reduce the shortcomings of conventional traffic controllers by accurately applying a provided variation during early cycle times based on the high traffic masses that changed at each lane of a four-lane junction. A real-time assessment of queue length and the hiring of a set of green signal coordination rules with an APTTCA-based device were both accomplished by the adaptive predictive sign manage system developed [3].

Some others studied adaptive traffic management structures with VANET, focused on reliable traffic prediction strategies, and. One of the greatest options for an adaptive site visitors control system for India is advised crowd sourcing [4].

In order to address the issue of traffic control machines, they devised a number of light computing solutions. which include simulation model, ant colony set of rules, genetic and neural network algorithms, fuzzy methodologies, and fuzzy algorithms [5].

In user balance traffic, he introduced a new method to optimize the timing of traffic lights across the area. An optimized version was developed as a multidimensional search problem aimed at finding the minimum value resulting from the variation of total tour time [7] and tour time per unit tour distance of an urban street community. A genetic algorithm was developed to generate the model solution.

The judgement body and characteristic module of the location-extensive site visitors sign manage machine are designed using a simulation control protocol that is integrated into PARAMICS software programme tool, which can engage in area-wide micro simulation [8]. According to his findings, mobility can be improved by using the suggested model in conjunction with a genetic algorithm to optimize the timing of signals across a large area.

Earlier to exceptionally as of late, the larger part of strategies for question location and question classification utilized highlight extraction methods like Haar, Hoard, nearby double designs (LBP), the scale invariant highlight change (Filter), or speeded up strong highlights (SURF), and after that classifier methods like SVM, arbitrary woodlands, or AdaBoost. In arrange to recognize between bike riders who are wearing head protectors and those who are not, Silva et al. [1] utilize procedures counting highlight extraction and histograms of arranged slope (Hoard), LBP, and the wavelet change (WT).

They combine numerous of the basic characteristics, such as HOG+LBP+WT, to supply seven particular include sets. In [5], the author created a strategy for recognizing head protectors in observation film utilizing SVM classifiers to recognize between bike riders and non-riders and between those wearing head protectors and those not wearing them.

Hoard, Filter, and LBP, three habitually utilized highlights, were actualized for both classifiers, and their particular exhibitions were compared with those of the other two features. They came to the conclusion that the Hoard depiction was supportive in getting the leading comes about.

SSD does not utilize a arrange for appointed locale proposition. It closes up being a reasonably clear way instep. Little convolution channels are utilized to calculate the area and lesson scores.

SSD uses 3 3 convolution channels for each cell to supply forecasts after

extricating the highlight maps. (These filters calculate the comes about in a way comparative to the standard CNN channels.) There are 25 channels created by each channel: 21 scores for each course and one border box. Make a forecast for the location and the lesson employing a 3x3 convolution channel.

Area Proposals (R-CNN, Fast R-CNN, Faster R-CNN): In arrange to discover the thing of intrigued interior the picture, calculations endeavor to build a bounding box around it. Also, in an protest location situation, you'd not essentially make one bounding box; instead, there might be a number of them speaking to different things of intrigued within the picture, and you wouldn't know how numerous at the beginning. The most reason you can't illuminate this issue by to begin with making a ordinary convolutional organize, at that point including a completely associated layer is since the yield layer's length is changeable, not consistent, since the recurrence of the objects of intrigued isn't settled. Utilizing different zones of intrigued from the picture and a CNN to decide in the event that an thing is show in those regions would be a unrefined arrangement to this issue. This strategy has the disadvantage of the potential for changing spatial arrangements and viewpoint proportions for the things of intrigued. As a result, selecting a huge number of ranges would be vital, which might cause a computational fiasco. In arrange to rapidly find these occasions, calculations like R-CNN and YOLO have been made.

To prepare the arrange demonstrate more effectively, we use the same approach as for DSSD (the remaining arrange performs superior than the VGG organize in this case). Exactness upgrade is the point. Be that as it may, the primary modification that was made was ResNet being used in lieu of the VGG network that was utilized within the unique SSD. At the conclusion of the essential arrange, we are going also include numerous convolution highlight layers. The estimate of these highlight layers will be dynamically diminished to permit for a few scales of discovery result prediction. It is experimentally known that the ResNet-101 layer, which is more profound than the VGG-16 layer when the input estimate is 300 and 320, replaces the SSD's basic convolution organize

with a leftover organize, which does not increment the exactness of the SSD but or maybe declines it. RetinaNet, a one-stage locator, employments centered misfortune to decrease misfortune from "simple" negative information so that misfortune is concentrated on "difficult" tests, expanding forecast exactness. RetinaNet outperforms Speedier R-CNN, the well-known two-stage locators, and accomplishes state-of-the-art execution utilizing ResNet+FPN as the spine for include extraction and two task-specific subnetworks for classification and bounding box relapse. Profound highlight extraction is finished utilizing ResNet.

The most reason you can't illuminate this issue by to begin with making a ordinary convolutional organize, at that point including a completely associated layer is since the yield layer's length is changeable, not consistent, since the recurrence of the objects of intrigued isn't settled. Utilizing different zones of intrigued from the picture and a CNN to decide in the event that an thing is show in those regions would be a unrefined arrangement to this issue. This strategy has the disadvantage of the potential for changing spatial arrangements and viewpoint proportions for the things of intrigued. As a result, selecting a huge number of ranges would be vital, which might cause a computational fiasco.

## **CHAPTER 3**

### **SYSTEM ANALYSIS AND ARCHITECTURE DESIGN**

A critical element of the layout segment is the structure layout, which describes the system's hardware, software program, and community surroundings. The architecture design flows usually from the nonfunctional requirements, together with operational, overall performance, safety, cultural, and political necessities. The deliverables from architecture design consist of the architecture design and the hardware and software program specification.

Objectives of Architecture Design and Requirement Analysis:

- Describe the fundamental components of an information system.
- Describe server-based, client-based, and client–server architectures.
- Describe newer architectural options, such as cloud computing.
- Explain how operational, performance, security, cultural, and political requirements affect the architecture design.
- Create an architectural design.
- Create a hardware and software specification.

#### **3.1 SYSTEM REQUIREMENTS**

##### **PROJECT HARDWARE SPECIFICATIONS**

RAM: 4 GB, storage: 500 GB, CPU: at least 2 GHz, architecture 32- or 64-bit, and a minimum of 2 GHz.

##### **IT REQUIREMENTS FOR THE PROJECT**

- Python 3 is used for model training, data pre-processing, and prediction.
- Operating system: Windows 7. MAC OS and Linux are also supported.
- Installing YOLOv3 and the essential Python libraries.
- Tkinter is used to create the system's graphical user interface.

**PYTHON:** Python is an easy-to-understand programming language that also has a variety of strong classes. Additionally, Python may easily be combined with other programming languages like C or C++. It has benefits of its own as an effective programming language:

- Open source
- Simple and simple to learn
- Ability to scale

**OPENCV:** A programming work library called OpenCV (Open-source computer vision) is basically centered on real-time computer vision. The open-source BSD permit for the library makes it cross-platform and free to utilize. The profound learning systems Torch/PyTorch, caffe, and TensorFlow are bolstered by OpenCV.

**NUMPY:** The word "array" refers to a kind of data in Python. NumPy is a crucial tool for data analysis and calculation when using Python to build the array data type. All of them are open-source libraries. The majority of matrix calculations utilise NumPy 22.

**PANDAS:** It is a Python programming language extension, is a quick, potent, adaptable, and user-friendly open-source tool for data analysis and manipulation.

**MATPLOTLIB:** For building inactive, energized, and intuitively visualisations in Python, consider Matplotlib. Matplotlib makes both basic and troublesome errands conceivable.

**PIL:** Python Imaging Library may be a free and open-source expansion library for the Python programming dialect.

It gives bolster for perusing, handling, and putting away a wide range of picture record sorts. PIL is additionally known as Pad in later forms.

**TENSORFLOW:** TensorFlow may be a free and open-source library for dataflow programming utilized for a assortment of purposes. It serves as a typical math library and is utilized in machine learning programs like neural systems. At Google, it is utilized for both inquire about and fabricating. Its versatile plan makes it conceivable to send compute rapidly over a extend of stages (CPUs, GPUs, and TPUs), from PCs to server clusters to portable and edge gadgets. Stateful dataflow charts are how TensorFlow calculations are spoken to. The activities that these neural networks take on multidimensional information clusters are the source of the term TensorFlow. The term "tensors" is utilized to depict these clusters.

TensorFlow, too known as the second-generation machine learning framework, is used for question location since it bolsters profound learning and works by performing numerical calculations utilizing information stream charts.

- It is open source and taken a toll nothing.
- It is tried and true (and free of serious bugs).
- A great community and Google back it.
- It is simple to utilize
- Numerous employers see it as a ability.
- With the capacity to function on CPUs and GPUs, it may be utilized in a assortment of Google applications, counting Discourse Acknowledgment, Google Photographs, Gmail, and indeed look.

**IMAGEAI:** It may be a Python library planned to empower software engineers, analysts, and understudies to form systems and applications with self-contained Profound Learning and Computer Vision capabilities by composing as it were a couple of brief lines of code.

**KERAS:** Python-based Keras is an open-source neural organize library. It is congruous with Theano, Microsoft Cognitive Toolkit, and TensorFlow. It centers on being user-friendly, measured, and extendable Keras in arrange to



encourage fast experimentation with profound neural networks. The keras preprocessing picture is utilized for this purpose class for picture information generator. We make increased picture batch generators (and their names) utilizing the flow (data, names) strategy by designing arbitrary adjustments and standardization strategies to be performed on your picture information amid preparing. These generators may at that point be used with the Keras demonstrate method's `fit_generator`, `evaluate_generator`, and `predict_generator`, which all take information inputs.

- Installing Numpy 1.16.x with pip
- Installing TensorFlow and TensorFlow-GPU need pip versions of 1.14.0 and 1.14.0, respectively.
- Install keras==2.2.4 using pip keras
- Installing OpenCV in Python using pip

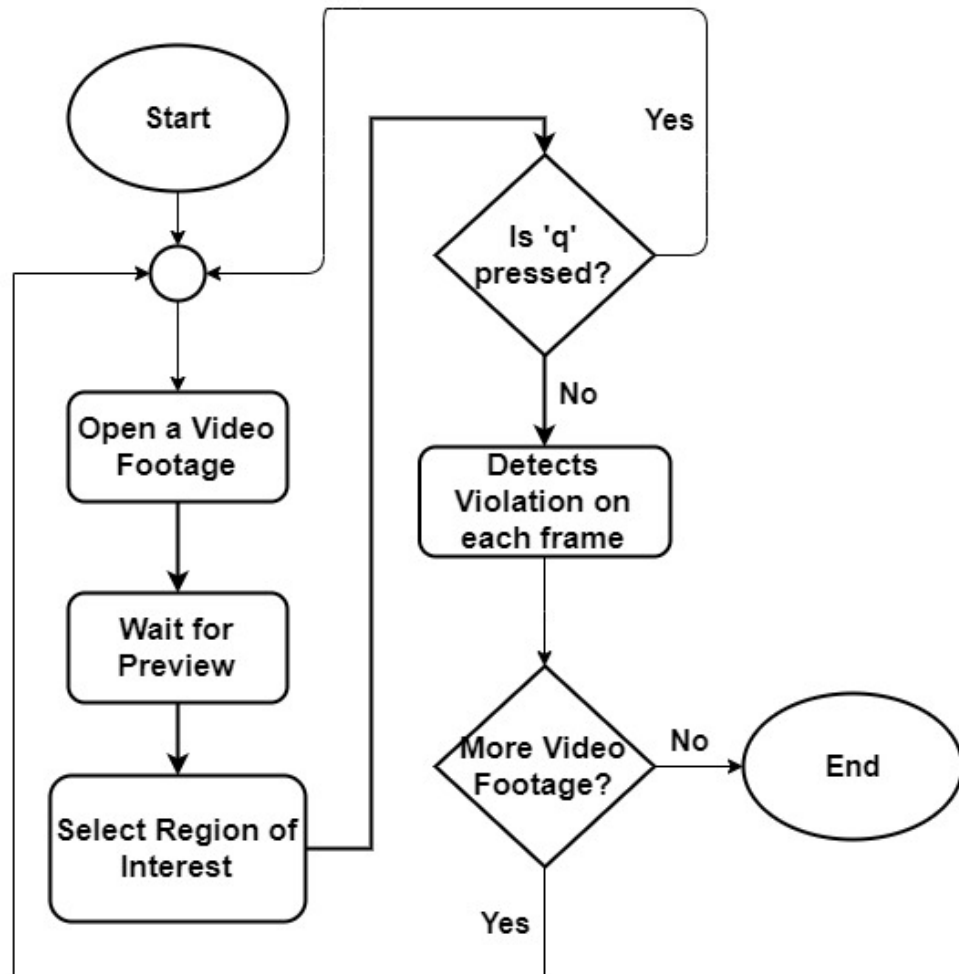
The ImageAI library may now be installed by using the following command at the prompt after all of those libraries have been installed:

```
install imageai -upgrade with pip
```

## **3.2 SYSTEM ARCHITECTURE**

### **3.2.1 MODULE 01**

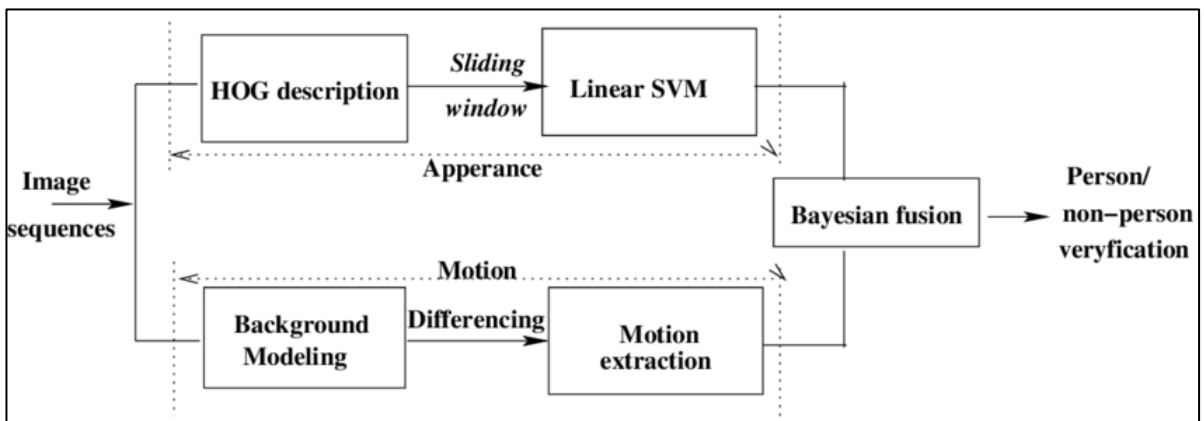
Fig 3.1 shows the flow diagram of the Vehicle detection model, and it helps to understand the decision-making process going on in the backend of the system.



**FIG 3.1 Flow Diagram of Traffic Signal Violation Module**

### 3.2.2 MODULE 2

From a picture of a human body, we may extract elements like the head, two hands, legs, and many others and provide them to an integrated model. The built-in image and video streams might then be improved upon using the model. The way that OpenCV works with pedestrian movement is one method. It can detect pedestrians in images and video streams using a pre-trained integrated HOG (Histogram of orientated Gradients) and built-in SVM model as shown below in Fig 3.2.



**FIG 3.2 HOG and SVM integration**

This module performs a couple of functions:

- It identifies the pedestrians at a particular crossroad
- It calculates the approximate weekly crowd of pedestrians at that location
- It suggests the control room to take the necessary actions based on the calculated data.

First, the video film is fed into the frame from the side of the road. Pedestrians are recognized from the film and stamped on the input image. A graphical client interface (GUI) makes the framework intuitive to clients. Customers can view activity films where they receive personalized recommendations based on herd status [10].

## **CHAPTER 4**

### **METHODOLOGY**

A project management methodology is a machine of concepts, strategies, and tactics utilized by folks that paintings in a field. not most effective do the pinnacle methodologies fluctuate in how they are structurally organized, but in addition they require distinct deliverables, workflows, or even mission control software program improvement.

#### **4.1 PROPOSED SYSTEM**

The system is first provided with the crossroads' video footage. As necessary, vehicles are detected. The technology tracks the movement of the cars and assesses whether there has been a violation. The diagram referred to as Fig. 1 shows how the system works. The system is interactive for the user thanks to the Graphical User Interface (GUI). The user may watch traffic video and get a warning when a violation is identified using the vehicle's bounding box. Then, a user in the police control room may use our platform to take further action.

The moment module utilizes direct back vector machines (SVM) for human classification and Histogram of situated slopes (Hoard) for highlight extraction within the human discovery strategy. To distinguish the classifiers that amplify review whereas recognizing individuals in perceptible video arrangements, an arrangement of tests are carried out.

We are paying particular attention to two-wheelers in the third module. We have seen how the traffic police must personally hunt down, apprehend, and fine bike riders who are not wearing helmets. However, this technique is time-consuming and prone to errors. Therefore, our algorithm leverages You Only Look Once (YOLO) to automate this procedure by identifying bike riders without helmets

and identifying the license plates of those cars. The owners of the vehicles are fined when their license plates are examined using optical character recognition (OCR).

## 4.2 COMPONENTS

### 4.2.1 OBJECT CLASSIFICATION

Moving objects are recognized in the provided video footage. The moving objects are categorized into the appropriate groups, such as automobiles, people, motorcycles, etc., using the object detection model YOLOv3.

Through the use of several methods, it increased accuracy and became more effective at identifying things. Using Darknet-53 architecture, the classifier model was created.

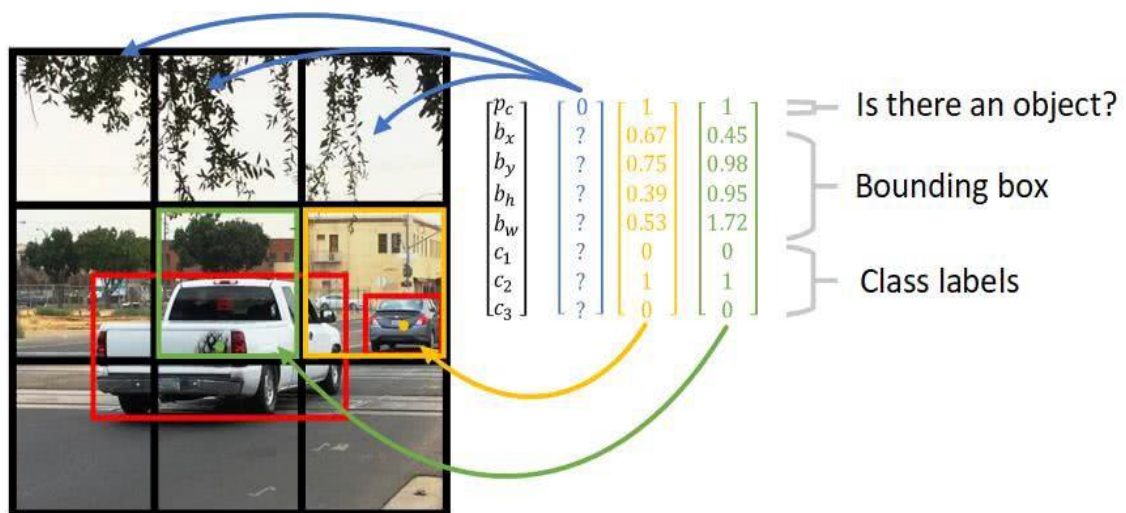
**TABLE 4.1 Darknet 53**

	Type	Filters	Size	Output
	Convolutional	32	$3 \times 3$	$256 \times 256$
	Convolutional	64	$3 \times 3 / 2$	$128 \times 128$
1x	Convolutional	32	$1 \times 1$	
	Convolutional	64	$3 \times 3$	
	Residual			$128 \times 128$
	Convolutional	128	$3 \times 3 / 2$	$64 \times 64$
2x	Convolutional	64	$1 \times 1$	
	Convolutional	128	$3 \times 3$	
	Residual			$64 \times 64$
	Convolutional	256	$3 \times 3 / 2$	$32 \times 32$
8x	Convolutional	128	$1 \times 1$	
	Convolutional	256	$3 \times 3$	
	Residual			$32 \times 32$
	Convolutional	512	$3 \times 3 / 2$	$16 \times 16$
8x	Convolutional	256	$1 \times 1$	
	Convolutional	512	$3 \times 3$	
	Residual			$16 \times 16$
	Convolutional	1024	$3 \times 3 / 2$	$8 \times 8$
4x	Convolutional	512	$1 \times 1$	
	Convolutional	1024	$3 \times 3$	
	Residual			$8 \times 8$
	Avgpool		Global	
	Connected		1000	
	Softmax			

## FEATURES

### A. Bounding Box Predictions

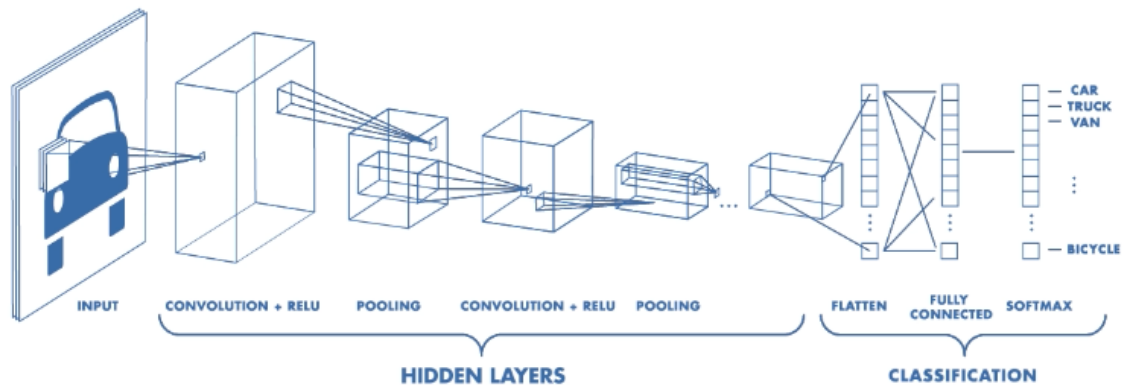
Because YOLOv3 is a single network, the loss for objectivity and categorization must be determined independently but from the same network. YOLOv3 uses logistic regression to predict the objectiveness score inaccuracy here will result in both classification and detection loss. Additionally, there would be alternative bounding box priors with objectiveness scores higher than the cutoff but lower than the best one, as shown in Fig 4.1. These mistakes won't affect the classification loss; only the detection loss will be affected [11].



**FIG 4.1 Bounding Box**

### B. Class Predictions:

A regular SoftMax layer is replaced with separate logistic classifiers for each class in YOLOv3. The purpose of this is to create a multi-label categorization. With the use of multilabel categorization, each box forecasts the classes that the bounding box could include, as shown in Fig 4.2.



**FIG 4.2 CNN working**

### C. Predictions across scales

YOLOv3 predicts boxes at three distinct scales to aid with scale-dependent detection. YOLOv3 is able to forecast more accurately at various sizes.

There are three scales that the bounding box priors constructed using dimension clusters are separated into, totaling nine bounding box priors.

### D. Feature Extractor

The new Darknet-53 network is used by YOLOv3. Darknet-53 is deeper than YOLOv2 and features 53 convolutional layers in addition to residuals or shortcut connections. It is more effective than ResNet-101 or ResNet-152 and more potent than Darknet.

The point of protest location, a subfield of computer vision and picture handling, is to distinguish occasions of semantic objects of a certain course (such as individuals, buildings, or cars) in advanced pictures and recordings. Two well-researched protest discovery ranges incorporate the acknowledgment of faces and people on foot. A few applications for question recognizable proof are found within the computer vision region, which incorporates picture recovery

and video observation. It is frequently utilized for computer vision applications such as portioning video objects and confront distinguishing proof, acknowledgment, and confront recognizable proof. A football amid a diversion or a individual in a movie are a number of cases of moving objects that will be seen with this innovation. One of the distinctive qualities that differentiates each item type, for example, is the fact that all circles are round. During the recognition of object classes, several distinct properties are employed. When looking for circles, for example, one looks for things that are distant from the centre, or a point. Equal-length objects with perpendicular corners must also be sought for in order to identify squares. The skin tone, the space between the eyes, the shape of the nose, the lips, and the eyes are all characteristics that may be used to identify faces using a similar technique.

The "You Simply See Once," or YOLO, family of models was created by Joseph Redmon and others, and was to begin with described within the 2015 article "You Merely See Once:

Bound together, Real-Time Protest Location." A set of end-to-end profound learning models are included in this framework, which points to perceive objects rapidly.

It parts the input into a lattice of cells and predicts a bounding box and an thing classification for each cell independently employing a single profound convolutional neural arrange (initially a form of GoogleNet, afterward overhauled and called DarkNet based on VGG). Because of this, there are a few elective bounding boxes, which are at that point coordinates within the post-processing organize to supply the ultimate forecast.

As of this composing, the approach has been actualized in three distinctive cycles: YOLOv1, YOLOv2, and YOLOv3. The common design was given within the to begin with emphasis, the plan was refined within the moment emphasis utilizing pre-defined stay boxes to progress bounding box proposition,

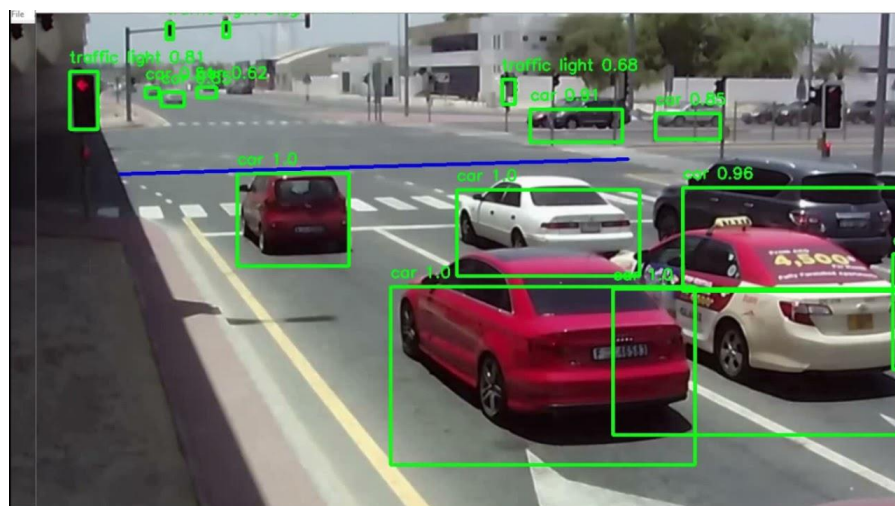


and the show design and preparing prepare were assist moved forward within the third cycle.

The models are widely used for object identification despite the fact that their accuracy is comparable to but lower than that of Region-Based Convolutional Neural Networks (R-CNNs). This is because of how quickly they can identify objects, which is frequently displayed in real-time on video or using camera feed input.

### 4.2.2 VIOLATION DETECTION

Using the YOLOv3 model, the cars are found. Infraction cases are examined when the cars have been located. In the user-provided preview of the video footage, a traffic queue is drawn across the street. The line indicates that the light is red for the intersection. Any car that crosses the center line when it is red is in violation [12]. A green bounding box surrounds the items that were detected. There is a violation if any vehicle proceeds through a red signal. The surrounding box of the car becomes red when a violation is detected.



### FIG 4.3 Object Identification

Fig 4.3 shows how the model helps in detecting the vehicles from an image containing multiple varieties of objects.

### 4.2.3 NUMBER PLATE ANALYSIS

The procedure for automatic number plate recognition includes the following steps:

Identify and localise a licence plate in an input picture or frame as the first step.

Next, take the characters off the license plate.

Utilize any kind of optical character recognition (OCR) to identify the characters that were extracted.

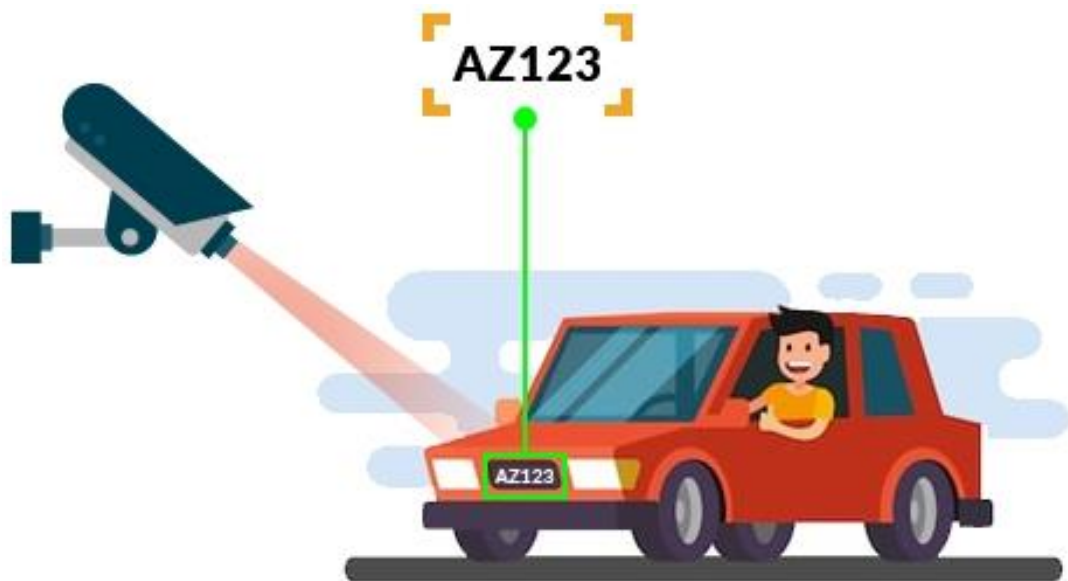
Number plate identification systems are made much more difficult by:

- Variable lighting conditions including reflections, shadows, and blurring
  - Rapidly moving objects
  - Obstructions
  - Additionally, obtaining big and reliable ANPR datasets for training and testing is challenging because of ANPR firms and governmental organizations tightly manage these databases as proprietary information since they include sensitive, private information, such as the time and position of a vehicle and its driver.
- 
- After applying a Gaussian Blur to the input image, we need to convert it to grayscale in order to decrease noise.
  - Look for the image's vertical edges.
  - After binarizing the image, the plate must be exposed. The Thresholding at the Vertical Edge image by Otsu should be used for this. In contrast to other thresholding techniques, Otsu's Thresholding calculates the cost automatically. In other thresholding techniques, we must choose a threshold price before we can binarize the image. Practise the final morphological transformation on a thresholded image. When an image has been thresholded, remains may be used to fill in tiny black spots within white areas. The white

box-shaped licence plate is well-known.

We need to look for contours in the picture to find the plate. Before detecting contours, the image must be binarized and morphed so that the contour finder can locate more useful contours and less overall contours. Find the smallest area rectangle that each contour may include, then check the aspect ratios and locations of each rectangle. Our descriptions of the plate's minimum and maximum regions are 4500 and 30000, respectively.

Next, find the contours in the region that has been verified, and confirm the facet ratios and location of the largest contour's bounding rectangle. You will get an extremely detailed registration code after verifying. Extraction of that contour from the source image is now complete. We will be provided with a plate image.



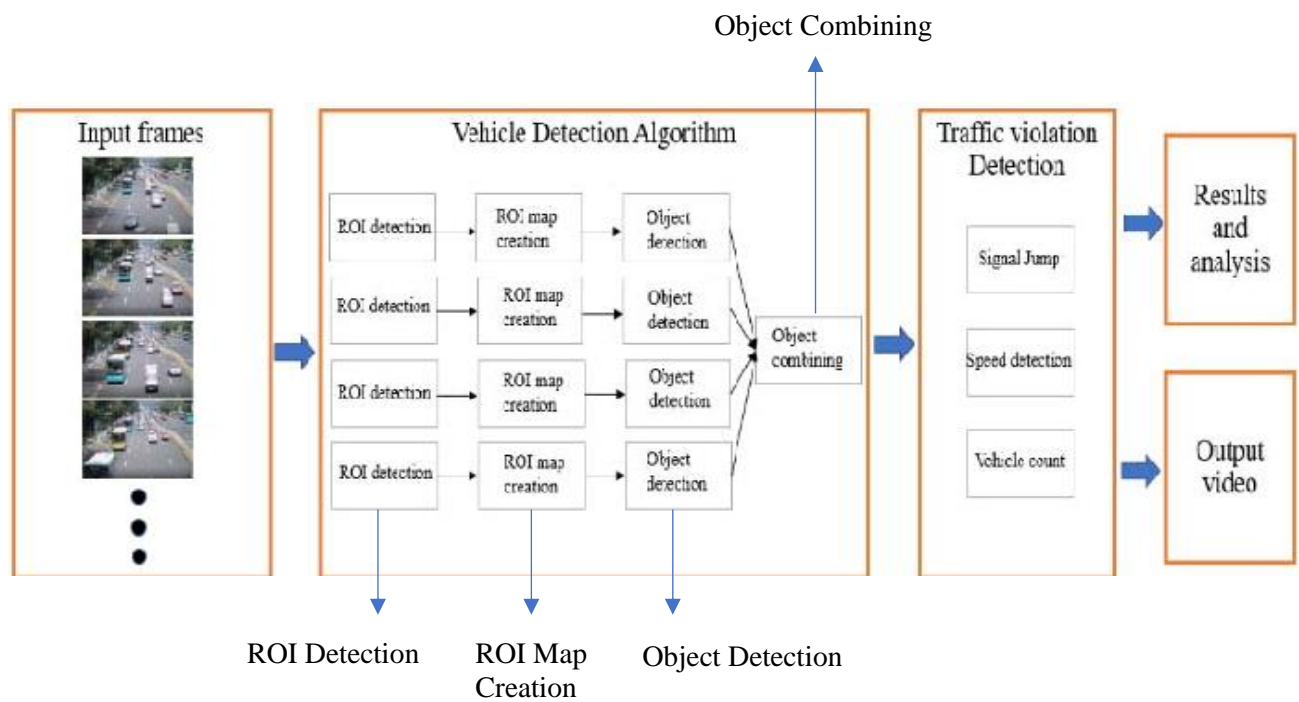
**FIG 4.4 Number Plate Extraction**

In the figure given above (Fig 4.4), we can see how the roadside camera scans the image of the detected vehicle, then the number plate detection model detects the number plate and extracts the text out of it for further processing.

### 4.3 UML DIAGRAMS

#### Module 1- Traffic Signal Violation Detection

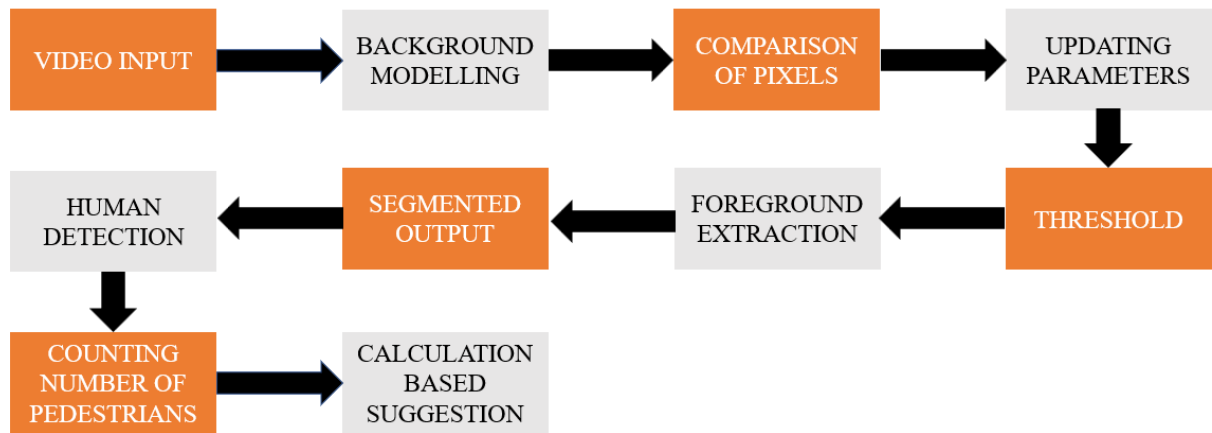
Here in Fig 4.5, the flow of steps in the detection of signal violating vehicles is shown.



**FIG 4.5 Block Diagram**

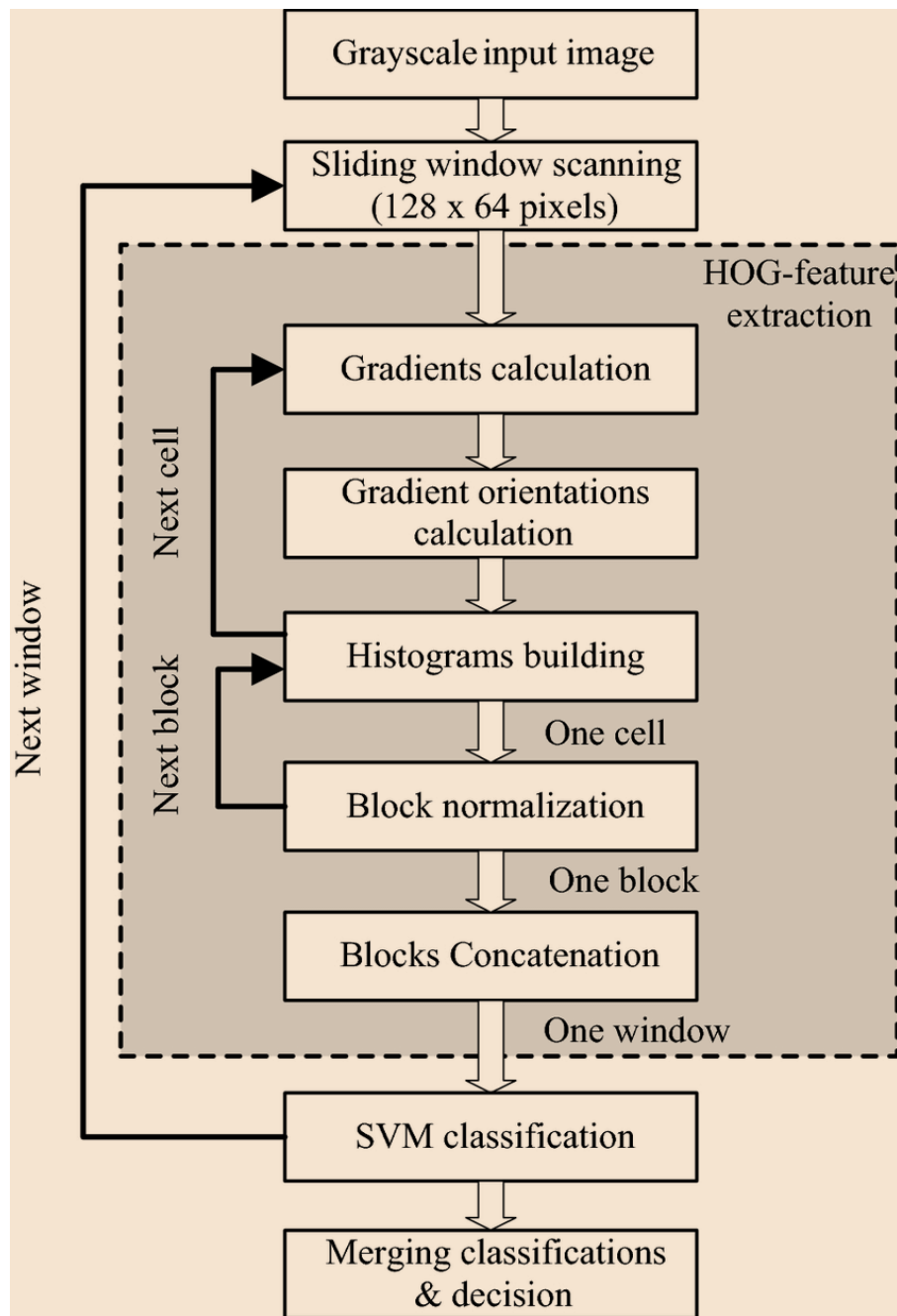
In Fig 4.5, first we open video footage, wait for preview, then select region of interest, and then each frame is monitored to check for violations. If the footage shows a violating vehicle, it is marked and a snapshot is saved. This process continues unless 'q' is pressed to end the process.

## Module 2- Pedestrian Detection



**FIG 4.6 Sequence Diagram- Module 2**

Fig 4.6 shows the sequential diagram of the processes involved in identifying pedestrians. So, from the input image, first the pedestrians are detected using background modelling and comparison of pixels, and then the detected humans are counted for performing calculation-based suggestions.

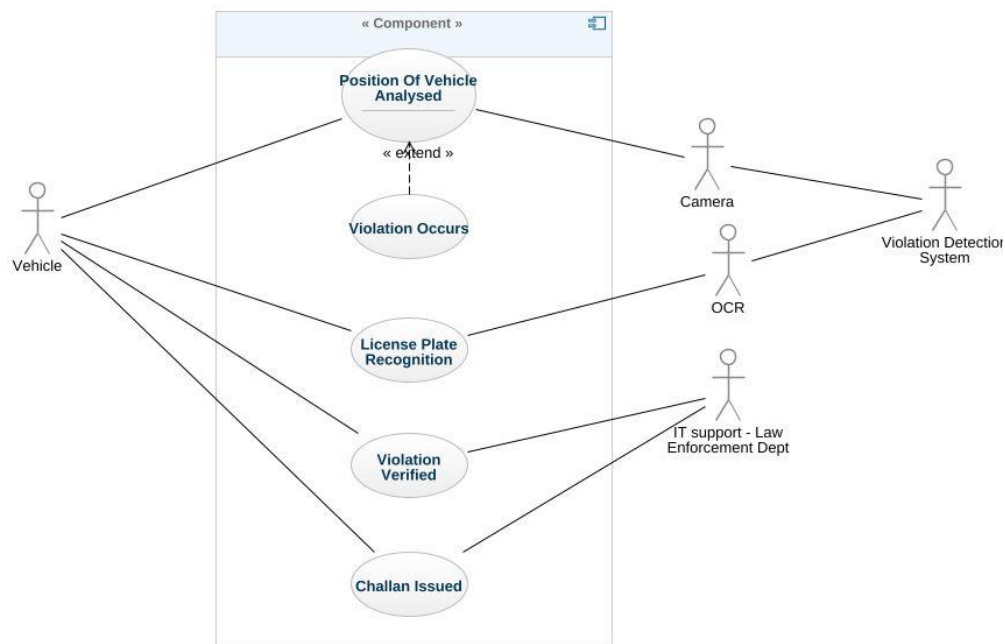


**FIG 4.7 Block Diagram- HOG and SVM**

In the figure shown above (Fig 4.7), the link between HOG and SVM algorithms is shown. So, the HOG model is used for feature extraction, and SVM is used for further classification and decision making.

### Module 3: Helmet Detection and Challan Generation

Fig 4.8 shows the use case diagram describing the process of Helmet and Number plate detection.

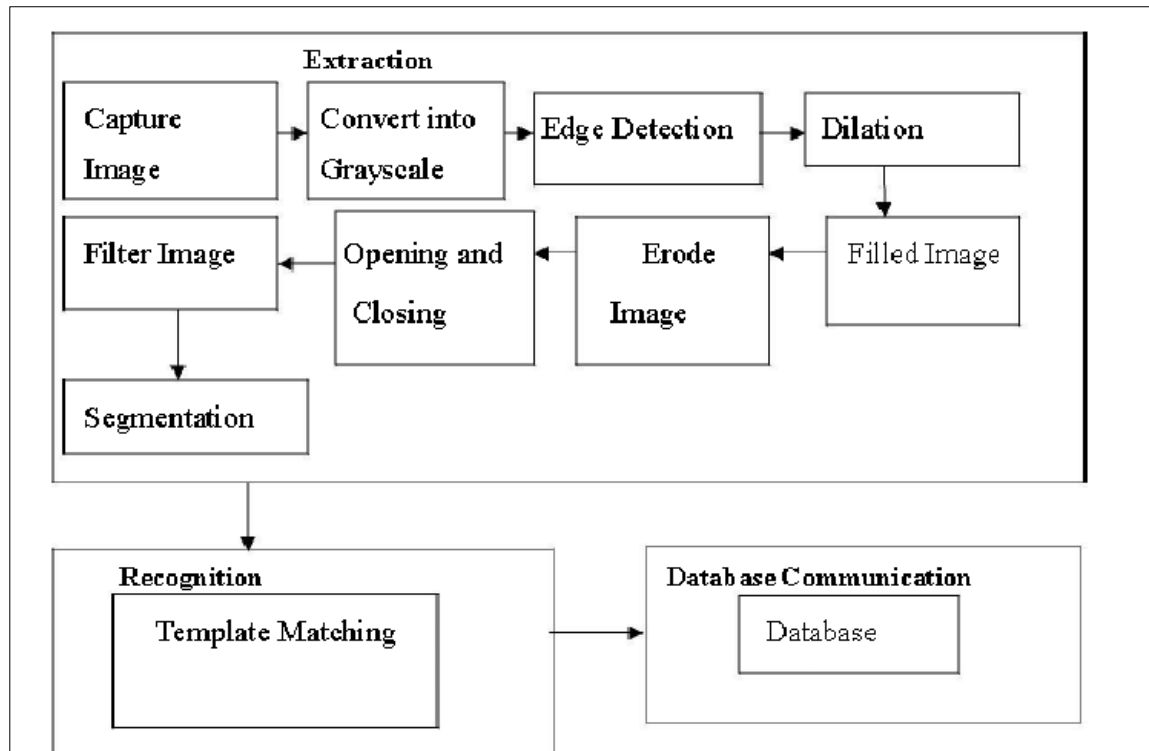


**FIG 4.8 Use-Case Diagram**

So, in Fig 4.8, each vehicle is detected and then, it is checked if its driver has a helmet on. If not, the number plate of the vehicle is detected and characters are extracted for challan generation.



Fig 4.9 shows the process blocks which are a part of the license plate detection process.



**FIG 4.9 Block Diagram of Number Plate Analysis**

In Fig 4.9, it shows how the process of Number Plate Detection and Analysis is divided into subparts. So once the number plate is detected, OCR extracts the characters and presents the text. Now this text is searched for in the Database to find out the driver/owner details of that vehicle. Post that, Challan is generated on his/her name.

## 4.4 ALGORITHMS

### 4.4.1 YOLO

You Only Look Alone (YOLO) is claimed to predict container amounts and classroom capacity simultaneously by using an end-to-end neural network. It does not display particular things, in contrast to prior detection approaches.

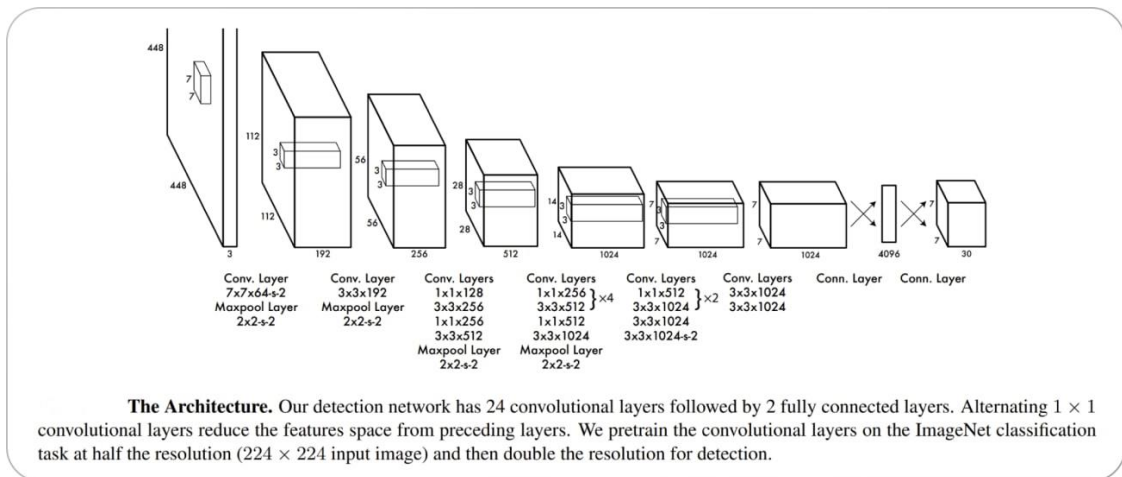
By using a whole unique method to item discovery, YOLO presents cutting-edge discoveries and performs far better than current real-time object recognition technologies.

While YOLO uses a full network to help with all of its predictions, speedier RCNN work first examines potential research topics using a regional network concept before doing validation in each region.

The method for exploiting the communication area in the region utilises the same photo several times, unlike YOLO, which only does one iteration. The architecture has been described in Fig 4.13.

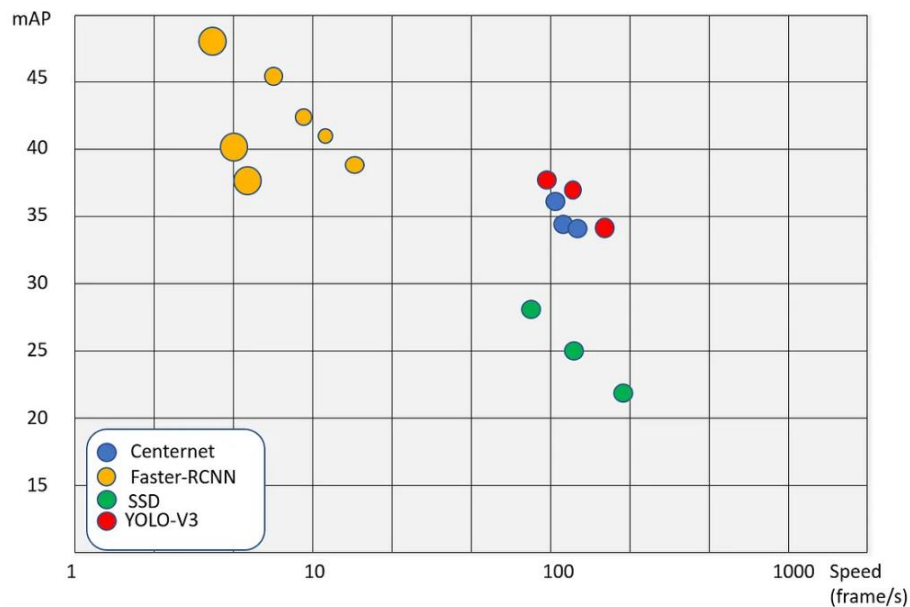
In order to forecast bounding boxes and class probabilities simultaneously, You Only Look Once (YOLO) suggests employing an end-to-end neural network. It contrasts from prior object detection algorithms' strategies, which reused classifiers to carry out detection.

YOLO executes all of its predictions with the aid of a single fully connected layer, in contrast to algorithms like Faster RCNN that operate by employing the Region Proposal Network to identify potential areas of interest before doing recognition on each of those regions independently. While methods that use region proposal networks go through many rounds for the same picture, YOLO just needs one.



**FIG 4.10 Feature Pyramid Networks**

The YOLO object identification method has been updated to version 3. "Feature pyramid networks" (FPN) shown in Fig 4.11 is a concept that is introduced in YOLO v3. the model is better equipped to recognize tiny things since it can see them from different angles [13][14].



**FIG 4.11 Comparison Graph**

Along with these upgrades, YOLO v3 now supports a broader variety of object sizes and aspect ratios. In comparison to earlier versions (Fig 4.12), it is also more accurate and reliable. Despite having a fairly straightforward working concept and architecture, YOLO's loss function is highly intricate. The profound meanings of the traits come from this. Consequently, a loss function with careful design may fit a lot of data into a tiny feature map.

The following list of key characteristics of the YOLO loss function:

**Augmenting input resolution** It could input photos of any size since it was a completely-convolutional network and didn't have any absolutely-related layers as earlier detectors had for the class project. However, since various input resolutions drive some network parameters higher, the community is trained using resolution augmentation: To enable the network to generalise its predictions for various resolutions, the scientists employed 10 entry resolution increments between 384x384 and 672x672 pixels that switched randomly every few training batches.

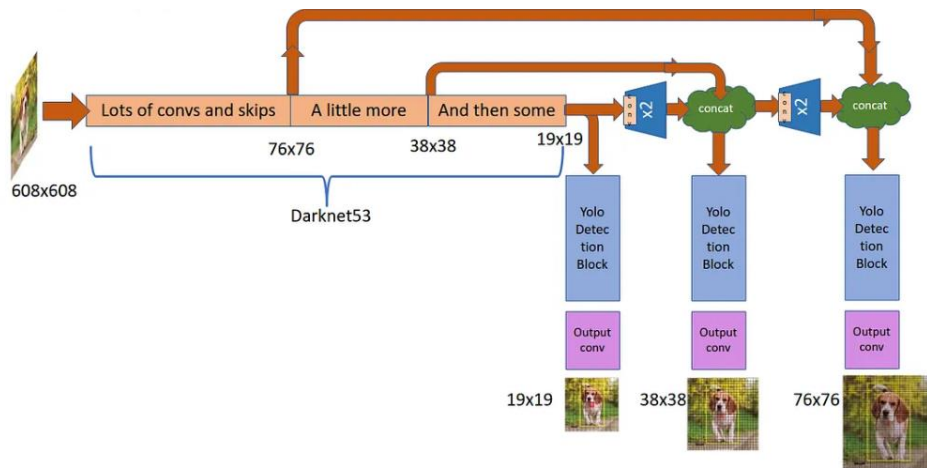
### **Loss Coefficients - Divide and Conquer**

The loss function approaches various boxes in various ways.

In the network's output layers, each spatial cell predicts numerous boxes, all centered in that cell (three in YOLO-V3, five in earlier versions). This is done through a method known as anchoring.

The fundamental terms that make up the YOLO loss for each box prediction are:

- A box prediction not completely enclosing an item, results in coordinate loss.
- Box-object IoU prediction error causes objectness loss.



**FIG 4.12 YOLO Architecture**

### Working of YOLO

Here, model's first 20 convolutional layers—all layers plus a brief intermediate pooling layer—are pre-trained using ImageNet. This pre-trained pattern translates to search by introducing a combinatorial process and a combinatorial process to the connection before the task, which improves performance. The YOLO final link algorithm predicts the class and bounding box coordinates that will result.

By using the supplied photo, YOLO generates a  $S \times S$  grid for  $B$  and associated confidence levels. This confidence score is a measure of the model's confidence in the box's likelihood to contain the item as well as its opinion on the forecasting accuracy of the box.

Every grid cell will have a large number of connected boxes, according to YOLO. We just need a single container that can hold everything for the lesson. In YOLO, the expert is held "responsible" for forecasting the performance of a certain product based on the largest feasible IOU between the prognosis and the actual circumstance.

This leads to a specialisation in bounding box searching. All predictors are more proficient in forecasting item size, ratio, and class, which increases overall score recovery.

NMS, or Non-Maximum Suppression, is the main approach used to solve the

YOLO model. NMS is the subsequent stage, which will When an object is detected, many boxes are often added to the picture of the object. The objective of each of these boxes is the same, even if some of them may overlap or have different functions. After NMS has found and deleted any incorrect or inaccurate boxes, each box that represents a component of the image is shown.

### **YOLO v3 (version 3)**

The YOLO object identification algorithm's third iteration is known as YOLO v3. As an upgrade from YOLO v2, it was released in 2018 with the goal of enhancing the algorithm's precision and speed.

Utilizing a new CNN architecture called Darknet-53 is one of the primary upgrades in YOLO v3. With a focus on object detection tasks, Darknet-53 is a ResNet architectural version. It is capable of achieving cutting-edge performance on several object identification benchmarks and includes 53 convolutional layers.

The term "feature pyramid networks" (FPN) is also made public by YOLO v3. For the purpose of detecting objects at various scales, CNNs use FPNs. They build a pyramid-shaped arrangement of feature maps, with each level utilised to identify things at various scales. Given that the model can see the items at various sizes, this helps to enhance the detection performance for tiny things.

A larger variety of item sizes and aspect ratios may be handled by YOLO v3 in addition to these upgrades. Compared to earlier YOLO versions, it is also more reliable and accurate.

A bounding box that may contain an object, its area, and tools, the probability that the object appears in each bounding box, and the probability that each object within that bounding box has a particular lesson location. Discovery, that runs in each of the three distinctive layers. More experienced YOLO neural planning plans have difficulty recognizing small objects, so the question is understood in three different sizes. These site-layer yield tensors have the same width and height as the input, but the depth is characterized by-

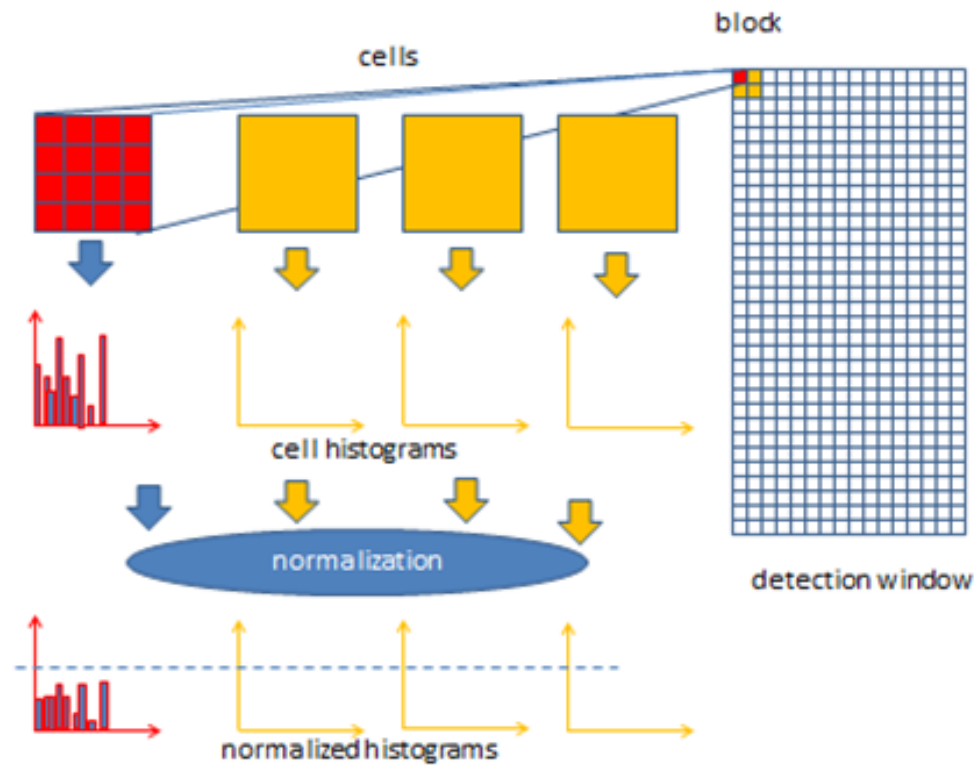
The number of properties of the bounding box (e.g. B. Width (bw), height (bh), and x and y position in the frame (bx,by) in the image. 1 means the box contains a recognizable question (pc)). course probability for each class (c1, c2, ..., c5). The raster can represent 3 bounding boxes, so the total for each cell is increased by 3. The result is 10 647 bounding box predictions from the array.

This configuration allows the simultaneous recognition of many objects using a single input image. The entire input image is analyzed and predicted to capture highlights as trains are placed. Compared to frameworks that use a sliding window approach, the organization receives data for the entire environment, including views and objects, resulting in significantly improved performance and clearer accuracy results. Compared to previous strategies for distinguishing protest evidence, YOLO takes a unique approach of dividing the image into grid cells.

This strategy discards numbers with a pc value less than 0.5, thus discarding most of the incorrect predictions. The same is periodically predicted within the image within the remaining bounding boxes. They are killed using a non-maximum concealment strategy.

#### **4.4.2 HOG (Histogram of Oriented Gradients)**

Each and every pixel is examined by this algorithm's immediate neighbours. To determine how much darker a given pixel is than those around it is the objective. The program creates arrows that indicate which way the picture is darkening. Every pixel in the picture goes through the same procedure, as shown in Fig 4.14.



**FIG 4.13 Working of HOG Algorithm**

Finally, each pixel would be replaced by an arrow; this new kind of arrow is known as a gradient. The progression of light from light to dark is shown by these gradients. Further analysis is carried out by these gradient-based methods.

Instructions for calculating HOG Features:

1. HOG characteristics you wish to compute. Make the picture 128x64 pixels in size (that is, 128 pixels in height and 64 pixels in width). As the authors' main goal with this sort of detection was to improve outcomes on the job of pedestrian detection, they employed this dimension and recommended it in the publication [16].

The MIT pedestrian database provided the authors of this research with extremely excellent findings, so they chose to create the "INRIA" dataset, a brand-new dataset that was substantially more difficult.



The first step is to compute  $G_x$  and  $G_y$  using the formulas below.

$$G_x(r, c) = I(r, c + 1) - I(r, c - 1) \text{-----} (1)$$

$$G_y(r, c) = I(r - 1, c) - I(r + 1, c) \text{-----} (2)$$

$$Magnitude(\mu) = \sqrt{G_x^2 + G_y^2} \text{-----} (3)$$

$$Angle(\theta) = |\tan^{-1}(G_y/G_x)| \text{-----} (4)$$

2. Once the slope for each pixel is decided, the slope network (network speaking to the greatness and point of each angle) is assembled into squares of 8x8 cells. A 9-point histogram is created for each square. A 9-point histogram makes a histogram with 9 containers and a precise extend of 20 degrees. Each of these 9-point histograms can be plotted as a histogram with canisters speaking to the quality of the slope inside that canister.

Considering the reality that a square can contain up to 64 diverse values, the taking after calculations are performed for each size and angle esteem. Since we're employing a 9-point histogram, we will say:

$$Step\ size(\Delta\theta) = 180^\circ / Number\ of\ bins = 20^\circ \text{--}(5)$$

$$[\Delta\theta \cdot j, \Delta\theta \cdot (j + 1)] \text{-----} (6)$$

$$C_j = \Delta\theta(j + 0.5) \text{-----} (7)$$

$$j = \lfloor \left( \frac{\theta}{\Delta\theta} - \frac{1}{2} \right) \rfloor \text{-----} (8)$$

$$V_j = \mu \cdot \left[ \frac{\theta}{\Delta\theta} - \frac{1}{2} \right] \text{-----}(9)$$

$$V_{j+1} = \mu \cdot \left[ \frac{\theta - C_j}{\Delta\theta} \right] \text{-----}(10)$$

3. A 9-point histogram matrix are clubbed together to form a new block (2x2). This clubbing is done in an overlapping manner with a stride of 8 pixels. For all 4 cells in a block, we concatenate all the 9-point histograms for each constituent cell to form a 36-feature vector.

$$f_{bi} = [b_1, b_2, b_3, \dots, b_{36}] \text{-----}(11)$$

4. Values of fb for each block is normalized by the L2 norm:

$$f_{bi} \leftarrow \frac{f_{bi}}{\sqrt{\|f_{bi}\|^2 + \varepsilon}} \text{-----} (12)$$

$$k = \sqrt{b_1^2 + b_2^2 + b_3^2 + \dots + b_{36}^2} \text{-----}(13)$$

$$f_{bi} = \left[ \left( \frac{b_1}{k} \right), \left( \frac{b_2}{k} \right), \left( \frac{b_3}{k} \right), \dots, \left( \frac{b_{36}}{k} \right) \right] \text{-----}(14)$$

5. The reason of this standardization is to reduce the effect of varieties in differentiate over photos of the same thing. each piece, from. It accumulates a 36-point highlight vector. 7 squares are found within the level course, whereas 15 blocks are found within the vertical heading. This implies that the by and large length of Hoard highlights will be  $7 \times 15 \times 36$ , or 3780. The desired image's Hoard characteristics are found.

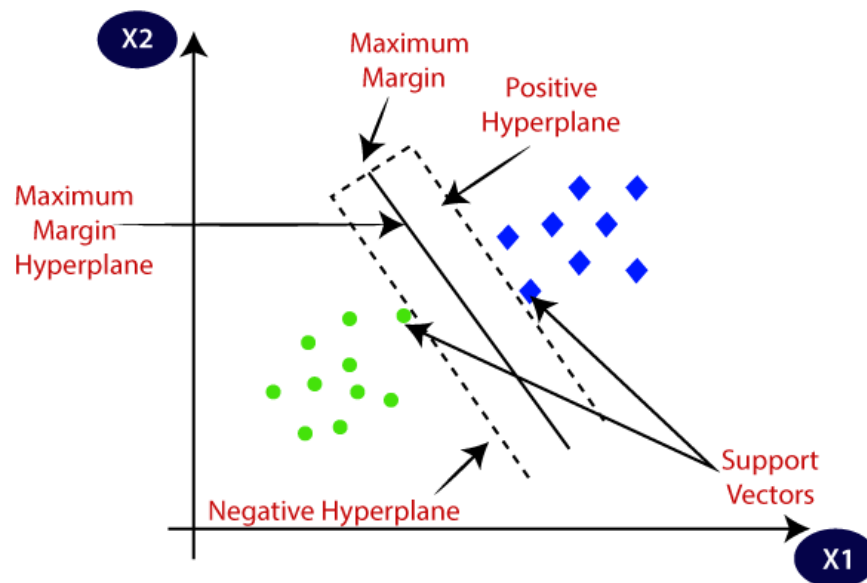
#### **4.4.3 SVM (Support Vector Machine)**

For the two-group classification problem, the Bolster Vector Machine (SVM) can be a supervised machine learning show using classification method. After receiving a set of preliminary information with each category named, the SVM broadcast can categorize future content.

They are more efficient and perform much better on fewer tests (within a few thousand) than post-computation like neural systems. For content classification problems where creating datasets of thousands of named tests is common, this method is well suited for such challenges.

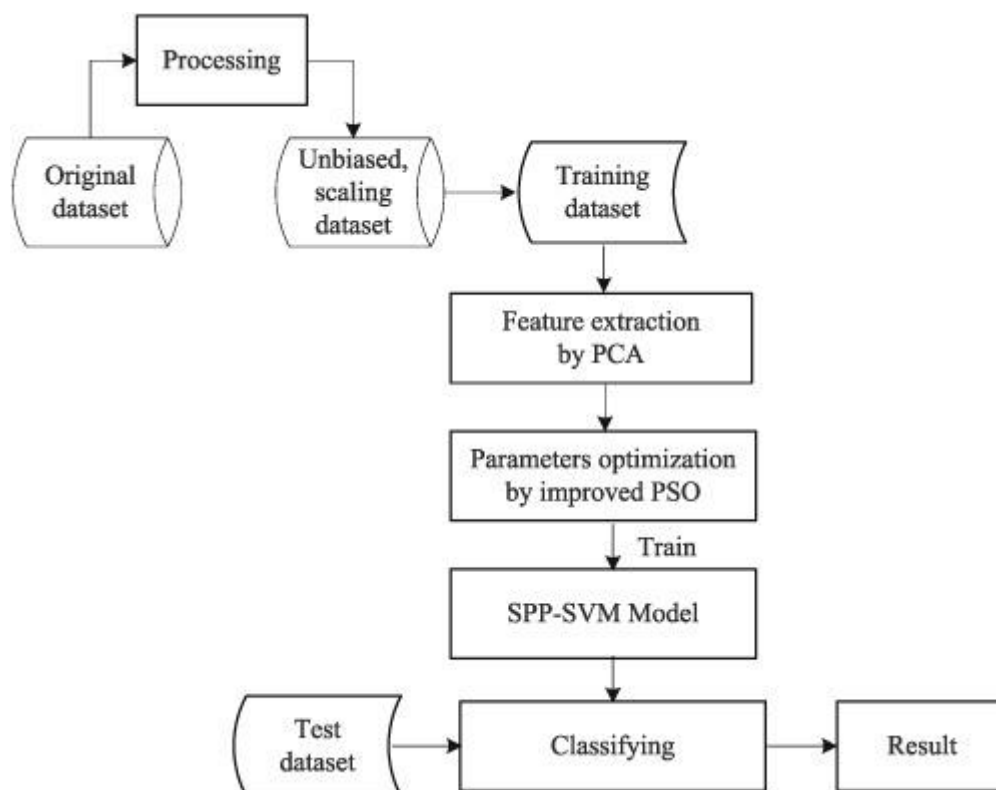
It is used for protest detection in computer vision and image processing. Within a limited area of the image, the approach looks for opportunities for gradient introduction. Using the size and angle leads, a histogram of the regions of the image is created.

To create the hyperplane, SVM chooses anomalous focal points and vectors. The back vectors used to represent these extraordinary events give the Bolster Vector Machine strategy its title. Consider the following illustration. In this illustration, two different categories are separated by a selection boundary or hyperplane.



**FIG 4.14 SVM Algorithm**

Above figure Fig. 4.14 clearly explains the SVM Algorithm, wherein the ideal hyperplane is shown for a particular dataset.



**FIG 4.15 SVM Architecture**

Fig 4.15 shown above, represents the architecture of the Support Vector Machine and explains its workflow.

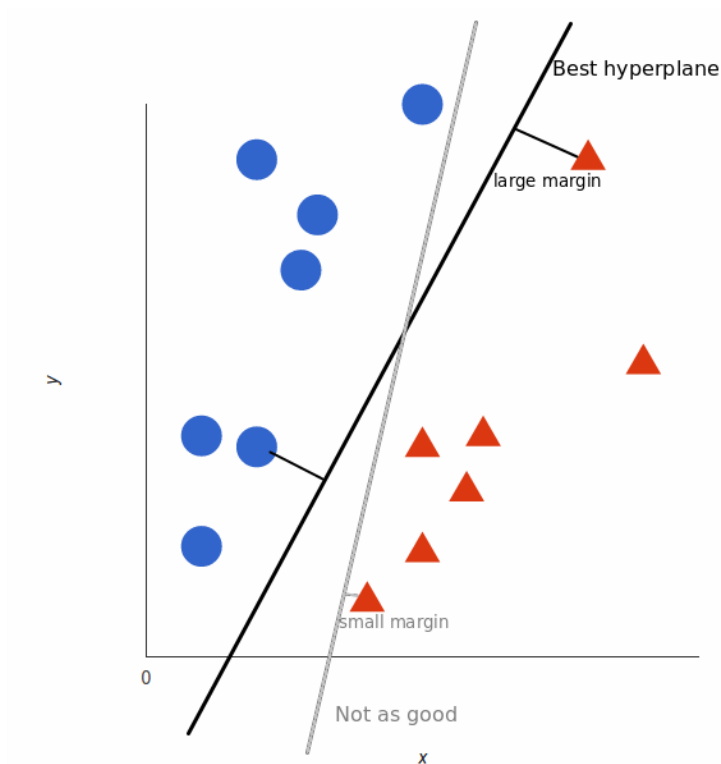
The hyperplane (smooth line in the two measurements) that ideally separates the labels is provided by a back-vector machine that takes these focal points of information as input. This line acts as a selection boundary, with anything on either side of it classified as blue, and anything else as reddish. SVM stands for support vector machines.

If a graph is plotted,  $x$  is the independent variable and  $y$  is the dependent variable. The hyperplane has been plotted in a way that it best separates the two classes of objects. The best 2D hyperplane is just a line.

The ideal hyperplane is the one that, according to SVM, maximizes the margins from both tags. In other words, the hyperplane (keep in mind that it's a line in this situation) that is furthest from the closest element of each tag.

For the two-group classification problem, the Bolster Vector Machine (SVM) can be a supervised machine learning show using classification method. After receiving a set of preliminary information with each category named, the SVM broadcast can categorize future content.

They are more efficient and perform much better on fewer tests (within a few thousand) than post-computation like neural systems. For content classification problems were creating datasets of thousands of named tests is common, this method is well suited for such challenges.



**Fig 4.15.1 SVM Sample Image**

In Fig 4.15.1,  $x$  is the independent variable and  $y$  is the dependent variable. It shows the different hyperplanes possible for the classification process.

There are several types of hyperplanes, as shown in Fig 4.15.1.

To find out how to find this ideal hyperplane precisely, watch this video instruction.

### **For Data that are Non-Linear**

Examining the following scenario:

The MIT pedestrian database provided the authors of this research with extremely excellent findings, so they chose to create the "INRIA" (National Institute for Research in Digital Science and Technology) dataset, a brand-new dataset that was substantially more difficult.

The absence of a linear decision boundary- a single line dividing the two tags is rather obvious. However, it seems as if it should be simple to separate the

vectors because of how clearly, they are divided.

In order to create a third dimension, this is what we will do. We only have the two dimensions of  $x$  and  $y$  before. We establish a new  $z$  dimension and decree that it be computed in a manner that is handy for us:  $z = x^2 + y^2$  (note that this is the equation for a circle).

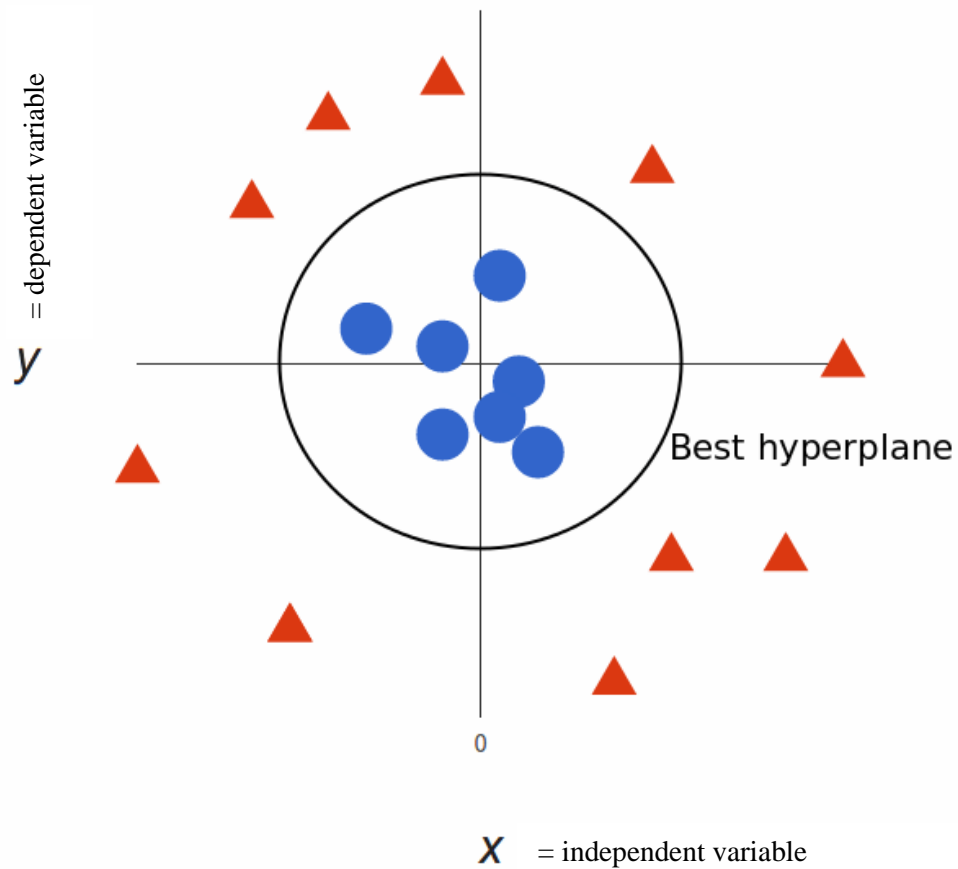
As a result, we shall have a three-dimensional space.

In Fig 4.15.4 and 4.15.5,  $x$  is the independent variable and  $y$  is the dependent variable. A portion of the graph is selected for best results.

The data is now divided into two groups that are separated linearly from one another.

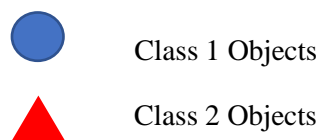
The hyperplane, which is a plane perpendicular to the  $x$  axis at a certain  $z$  (let us assume  $z = 1$ ), is important to note since we are now in three dimensions.

Mapping it back to two dimensions is all that is left.



**Fig 4.15.2 SVM Sample Result**

In Fig 4.15.2,  $x$  is the independent variable and  $y$  is the dependent variable. The best hyperplane is plotted in order to classify the different types of objects in the best possible way.



So, Fig 4.15.2 shows the final plotting of hyperplane based on the given datapoints. Now based on this training of the dataset, further predictions and classifications can be made using this model.



#### **4.4.4 IOU (Intersection Over Union)**

The intersection over Union metric is often used to determine local accuracy and local error in sample detection.

To compute the IoU of the estimated and actual ground bounding boxes, we first find the intersection of the two bounding boxes for the same object. Next, we calculate the total areas filled by the two connected boxes, also known as the "join," and their overlap, also known as the "intersection."

The junction divided by join result, which displays a uniform overlap throughout the whole area, is a good measure of how close the box is to the original box.

#### **4.4.5 Period Reliability**

The mean precision (AP) is calculated by computing the area under the accuracy-recall curve for a set of predictions.

Return is determined by dividing all predictions generated by the model for a class by all records that are available for that class. The ratio of the model's predictions to their actual quality is known as precision.

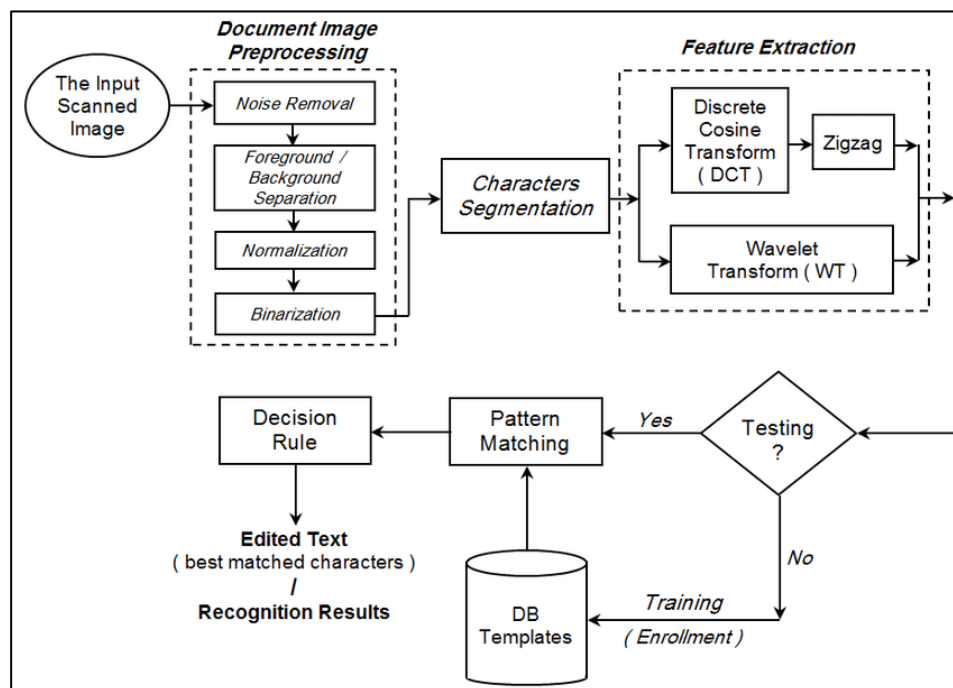
Recall and accuracy are sacrificed in order to reduce the variance of the distribution, which is represented graphically as a curve.

By analysing the region beneath this precision-recall curve, we can establish the model's typical accuracy for each class. The average of these numbers across all classes is known as mean precision, or mAP.

#### 4.4.6 OCR (Optical Character Recognition)

OCR is the process that converts a text image into a readable and editable text format. The text extraction process is explained in Fig 4.17.

After the number plates of the vehicles are detected, the letters are scanned and processing is done in the following steps:



**FIG 4.16 OCR Implementation**

- The taking of images

Documents are read and converted to binary data using a scanner. The bright regions of the scanned picture are categorized by the OCR software once it has examined them.

- Preprocessing

In order to make an image suitable for reading, OCR software first cleans it and fixes any defects. Its cleaning procedures include the following:

- Adjusting the scanned document's alignment after it has been skewered or slightly tilted to correct alignment problems.
- Despecking, i.e., eradicating any blemishes from digital photos, and rounded text image corners.
- Removing unnecessary lines and boxes from the picture.
- Script recognition for OCR technology that supports several languages
  - Recognising text

The terms "pattern matching" and "feature extraction" refer to the two primary OCR algorithms or computer processes that an OCR program utilizes to recognise text.

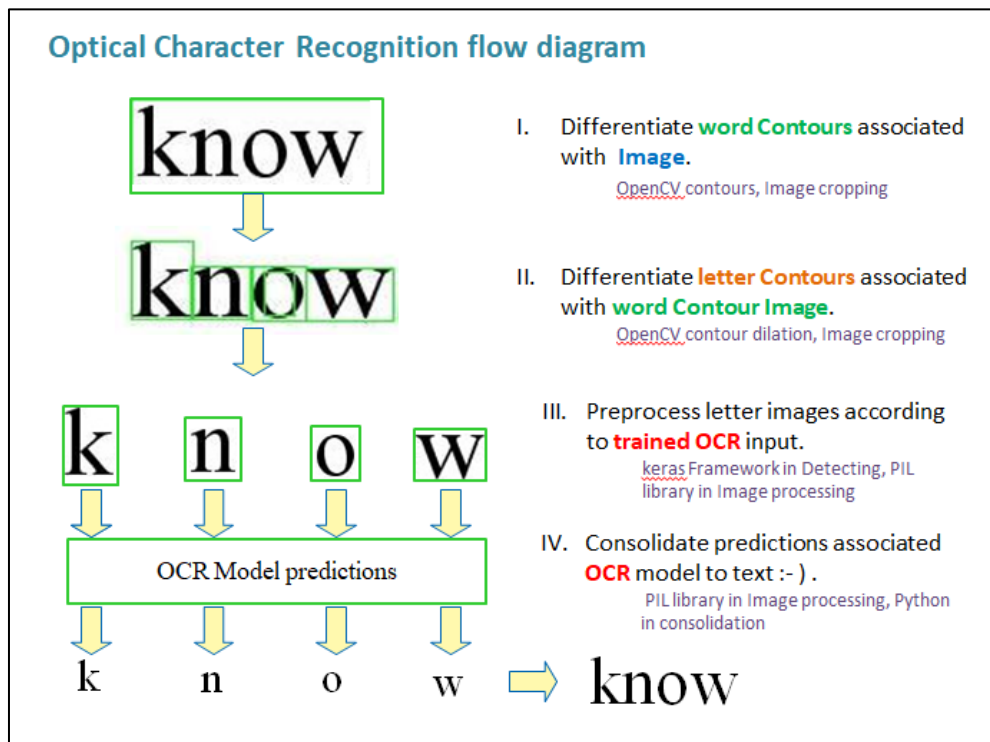
- Combining patterns

A character picture, known as a glyph, is isolated and compared to a previously stored glyph to perform pattern matching provide good candidates for this technique.



**FIG 4.17 Representation of image using 0's and 1's**

The above figure, Fig 4.17 shows how the character is traced using 0's for inactive areas and 1's for active areas.



**Fig 4.18- Text Extraction using OCR**

The above figure, Fig 4.18 describes how a word is segmented in order to distinctly identify the characters. So first the document is separated into paragraphs, then into lines, then into words, and further into individual characters.

## CHAPTER 5

### CODING AND TESTING

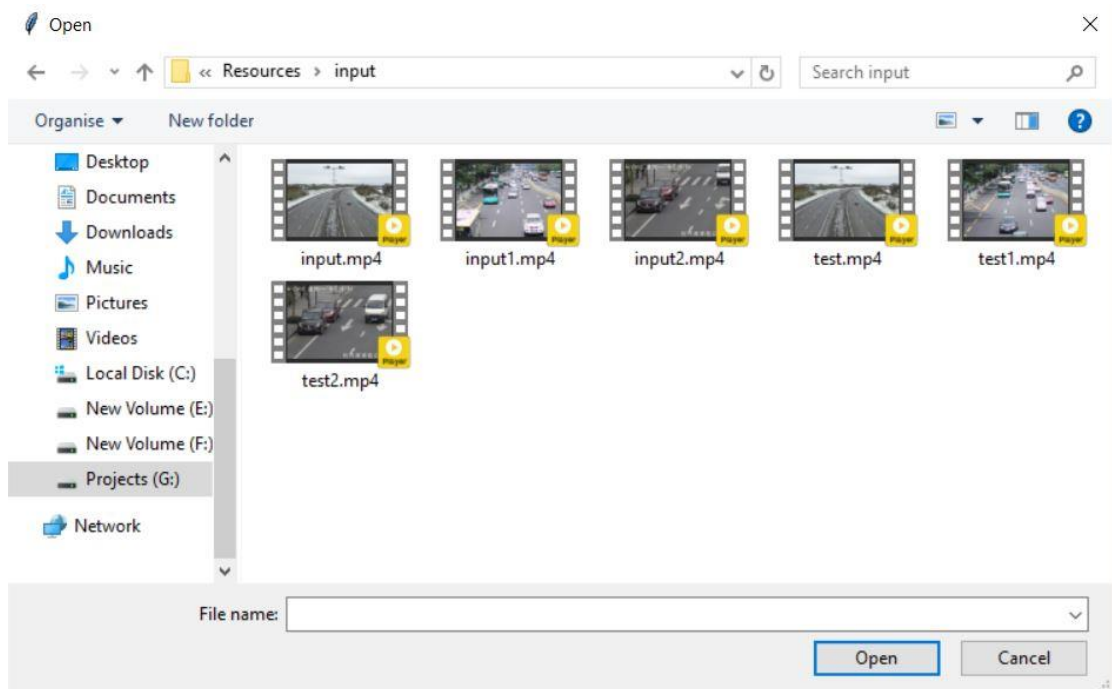
This project uses the open-source OpenCV image processing software package for computer vision and machine learning. The vehicle classifier with Darknet-53 is implemented using Tensorflow.

All of the software's choices are available on the graphical user interface. The program is used for administration and many debugging tasks. No management requires us to alter code.



**FIG 5.1 Home Page**

**HOME PAGE:** Fig 5.1 shows the first screen that we get to see on executing our first module. Here we have two options, 'File' and 'Analyze'. Using the file option, we can choose our input video.



**FIG 5.2 Input Selection**

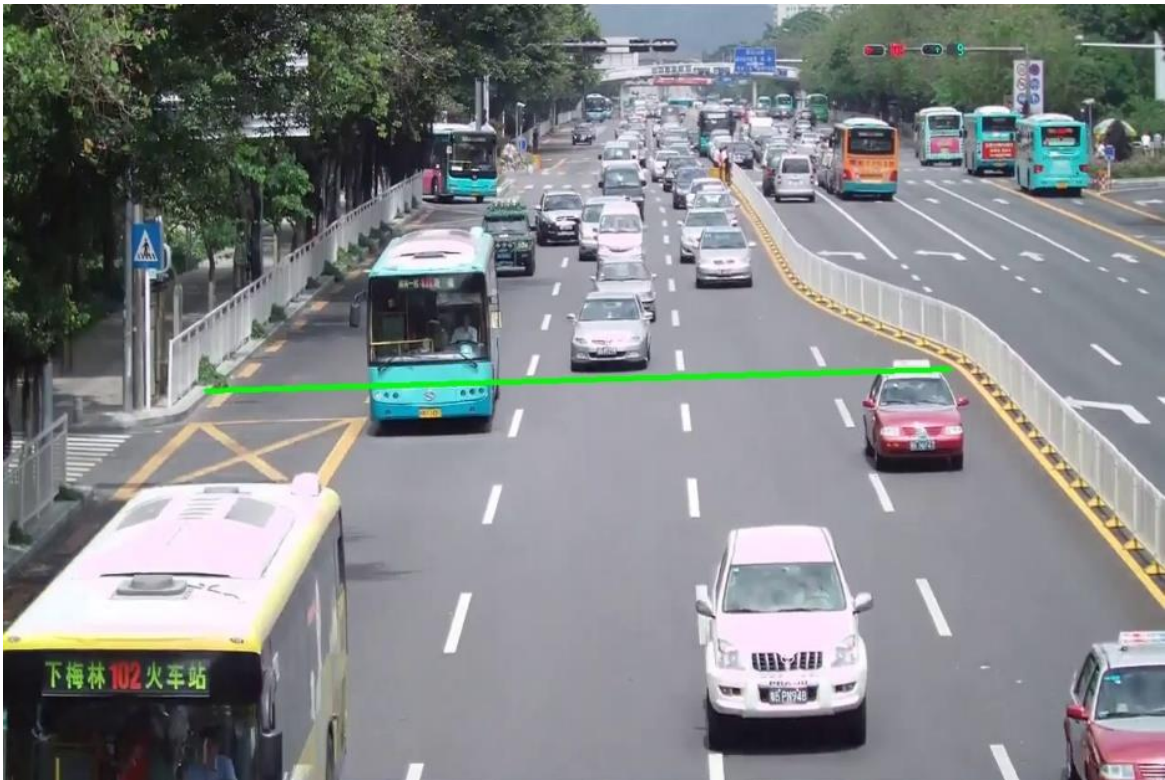
**SELECTING VIDEO FROM STORAGE:** Fig 5.2 represents the file selection window. From here, we can select the video footage which we want to monitor. However, the video footage needs to be present in the local system.



**FIG 5.3 Video preview**

**PREVIEW OF THE SELECTED VIDEO:** For demo purpose, we have chosen this sample video shown below in Fig 5.3, on which we will be marking regions to detect the vehicles violating the traffic signal.

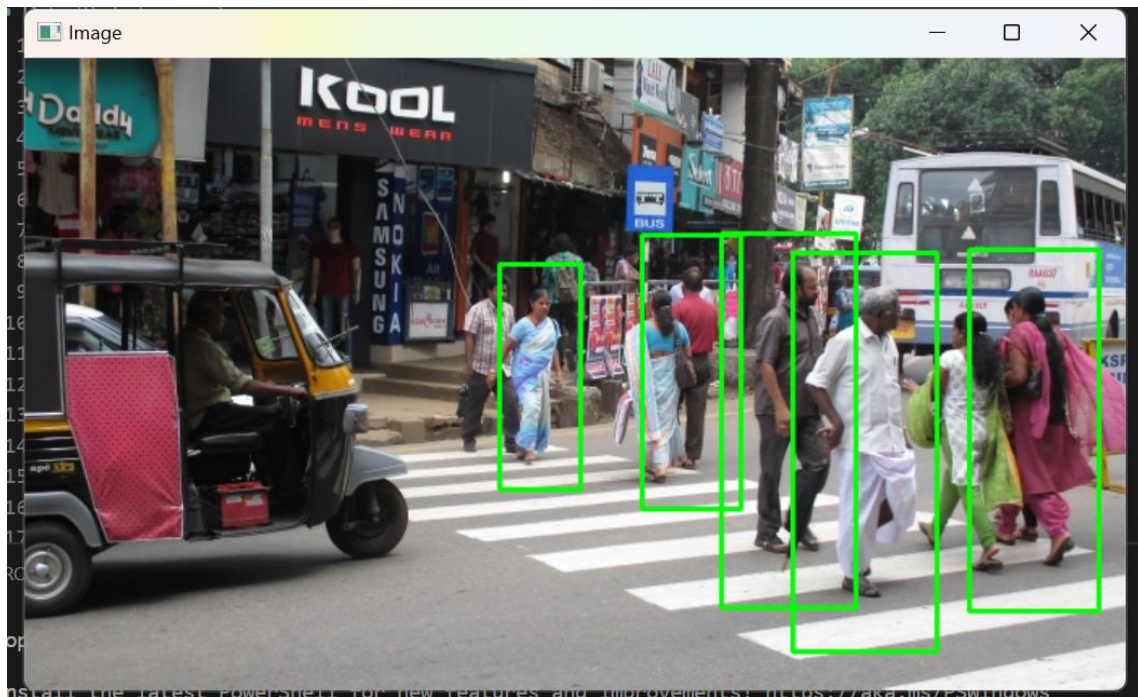




**FIG 5.4 Region of Interest**

**MARKING REGION OF INTEREST:** In the image below, we can see the green line that is known as the “Region of Interest”. To choose that, we need to click on “Analyse” and then select “Mark Region of Interest”. Post that we can click on the starting point and ending point of the line of interest and then the line will appear automatically, as shown in Fig 5.4.





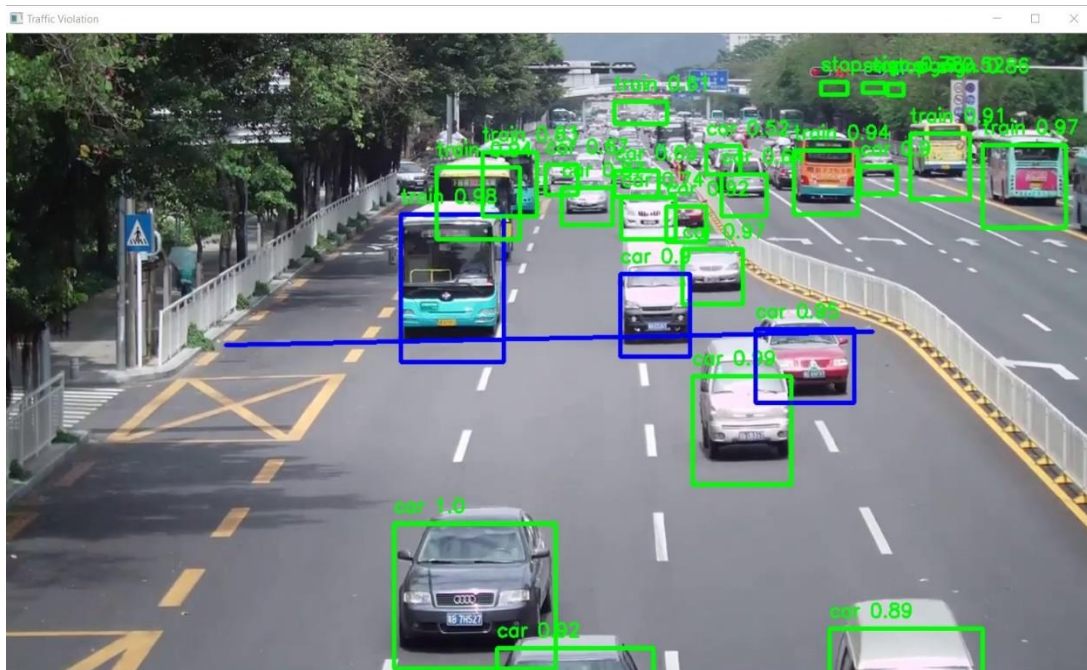
**FIG 5.5 Detecting Pedestrians**

**DETECTING PEDESTRIANS:** This image below shows the partial output of the second module. So here we can see that the pedestrians in the available footage are detected using the green bounding boxes, as shown in Fig 5.5.

## CHAPTER 6:

### RESULTS AND DISCUSSIONS

**FINAL OUTPUT OF MODULE 1:** As the output of the first module of our project, we see in the image below how the vehicles are detected using boxes of different colours. The cars which are not violating the traffic signal, according to our region of interest, are marked in green boxes, whereas the vehicles that our doing so are marked in blue boxes.

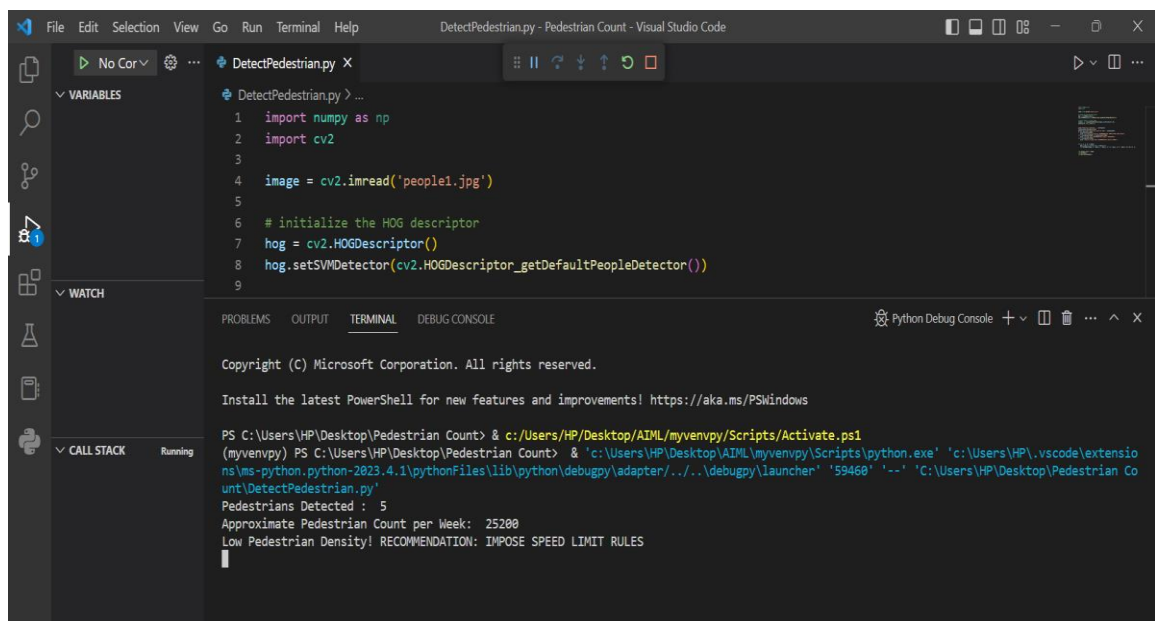


**FIG 6.1 Result of Module 1**

The method for detecting violations is launched after choosing the area of interest. A console display will display the line's coordinates. Following the line's drawing, the mechanism for detecting violations will launch. The initial step will be to load the weights. Objects will then be detected by the system, and any violations will be checked. Frame by frame, the output from the GUI will be shown. Up to the last frame of the video, the machine will output. An

"output.mp4" file will be created in the background. The output folder of "Resources" will include the file. By selecting "q," the current procedure will end instantly. The administrator may upload more video material from the original file manager after processing the first one. Once all work has been completed, the administrator may exit by selecting the 'Exit' option from the File menu.

**FINAL OUTPUT OF MODULE 2:** Below, we have the final output of our second module, in which our project suggests which area requires a subway, or a footbridge, and which other areas require speed restrictions.



```

DetectPedestrian.py X
1 import numpy as np
2 import cv2
3
4 image = cv2.imread('people1.jpg')
5
6 # initialize the HOG descriptor
7 hog = cv2.HOGDescriptor()
8 hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
9

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Python Debug Console

Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

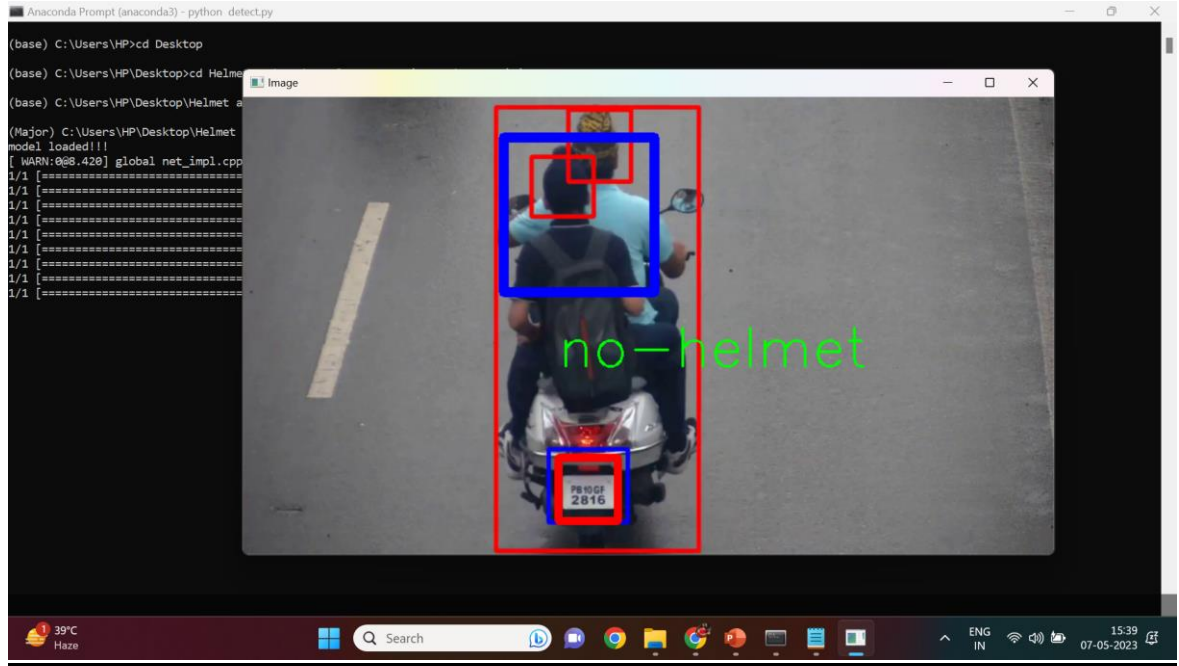
PS C:\Users\HP\Desktop\Pedestrian Count> & c:\Users\HP\Desktop\AIML\myvenvpy\Scripts\Activate.ps1
(myvenvpy) PS C:\Users\HP\Desktop\Pedestrian Count> & 'c:\Users\HP\Desktop\AIML\myvenvpy\Scripts\python.exe' 'c:\Users\HP\.vscode\extensions\ms-python.python-2023.4.1\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '59468' '--' 'C:\Users\HP\Desktop\Pedestrian Count\DetectPedestrian.py'
Pedestrians Detected : 5
Approximate Pedestrian Count per Week: 25200
Low Pedestrian Density! RECOMMENDATION: IMPOSE SPEED LIMIT RULES

```

**FIG 6.2 Result of Module 2**

The daily pedestrian count acquired from the submitted photos is used to estimate the estimated weekly pedestrian density of a certain location. After that, a suggestion is sent to the control room advising either building a subway or a footbridge in that region, depending on how crowded it is. This promotes safety for pedestrians and helps to avoid accidents. In order to prevent cars from driving too fast or recklessly, the system advises installing speed limitation metres at preset intervals if the pedestrian density is not too high.

**FINAL OUTPUT OF MODULE 3:** The final module of our project detects the bikes (two wheelers) in the input footage. The vehicles with their drivers wearing helmets are marked in green and “helmet” is shown on screen. For the vehicles with their drivers not having helmets on are marked in red, and “no helmet” is displayed on the screen.



**FIG 6.3 Result of Module 3**

We have got the photographs of the identified automobiles as the final product. The bike riders who are wearing helmets are denoted by green boxes, while those who are not are denoted by red boxes. Additionally, the license plate is identified individually and given additional processing.

**Analysis of our model and comparisons:** We performed a few calculations and compared the results of our model with the existing models using different algorithms and process structures. The analysis of different algorithms is shown below in Table 6.1.

**TABLE 6.1 Comparison of Algorithms**

	<b>Faster R-CNN</b>	<b>YOLO v3</b>	<b>SSD</b>
<b>Phases</b>	RPN + Fast R-CNN detector	Concurrent bounding-box regression and classification	Concurrent bounding-box regression and classification
<b>Neural Network Type</b>	Fully convolutional	Fully convolutional	Fully convolutional
<b>Backbone Feature Extractor</b>	VGG-16 or other feature extractors	Darknet-53 (53 convolutional layers)	VGG-16 or other feature extractors
<b>Location Detection</b>	Anchor-based	Anchor-Based	Prior boxes/Default boxes
<b>Anchor Box</b>	9 default boxes with different scales and aspect ratios	K-means from coco and VOC, 9 anchors boxes with different size	A fixed number of bounding boxes with different scales and aspect ratios in each feature map
<b>IoU Thresholds</b>	Two (at 0.3 and 0.7)	One (at 0.5)	One (at 0.5)
<b>Loss Function</b>	Softmax loss for classification; Smooth L1 for regression	Binary cross-entropy loss	Softmax loss for confidence; Smooth L1 Loss for localization

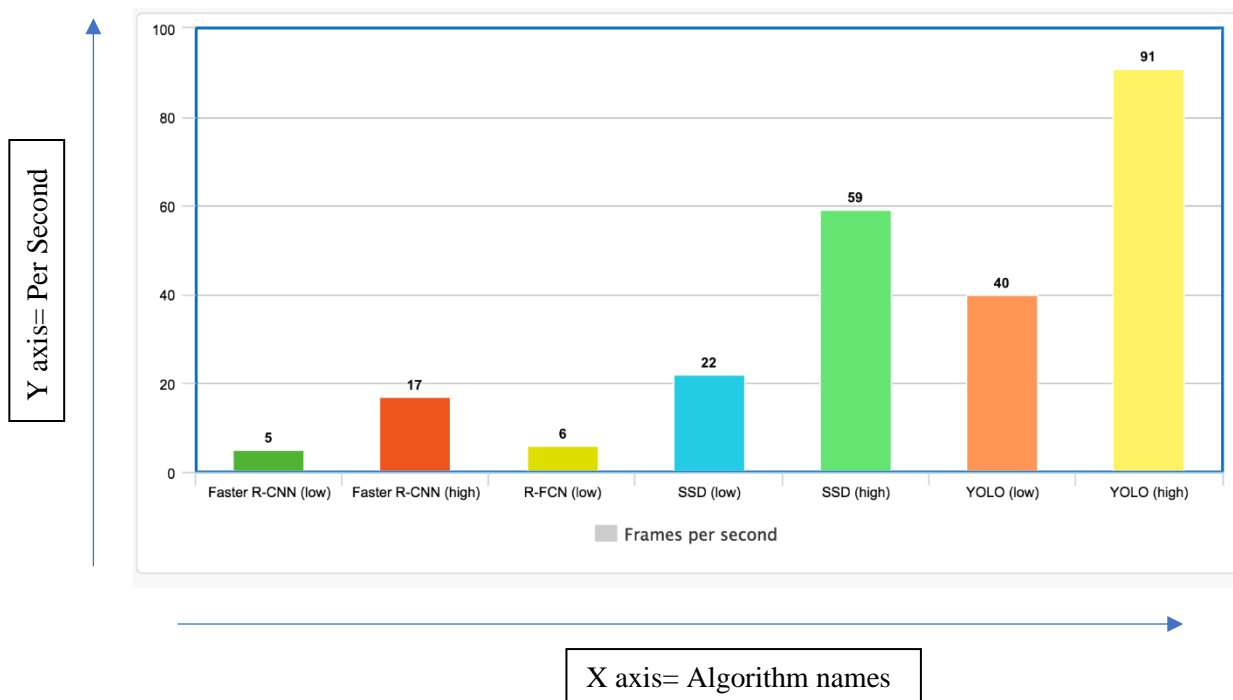
So, in the Table 6.1, we have compared the algorithms, namely Faster RCNN, Yolo v3 and SSD on the basis of parameters like Phases, Neural Network Type, Backbone Feature extractor, Location Detector, Anchor Box types, IoU Thresholds, and Loss Function.

The performance evaluation criteria will be based on 3 key points:

- Accuracy of the models in terms of Mean Average Precision (mAP)
- Speed of Inference in terms of Frames Per Second (FPS)
- Type of GPU Used: Gaming or AI GPUs. **GTX 1080 Ti, RTX 4090, Tesla V100, and Tesla P100** GPUs to be specific.

The image shown below compares the performances of some popular algorithms and represents them in graphical format-

Comparison factor 1: Frames processed per second

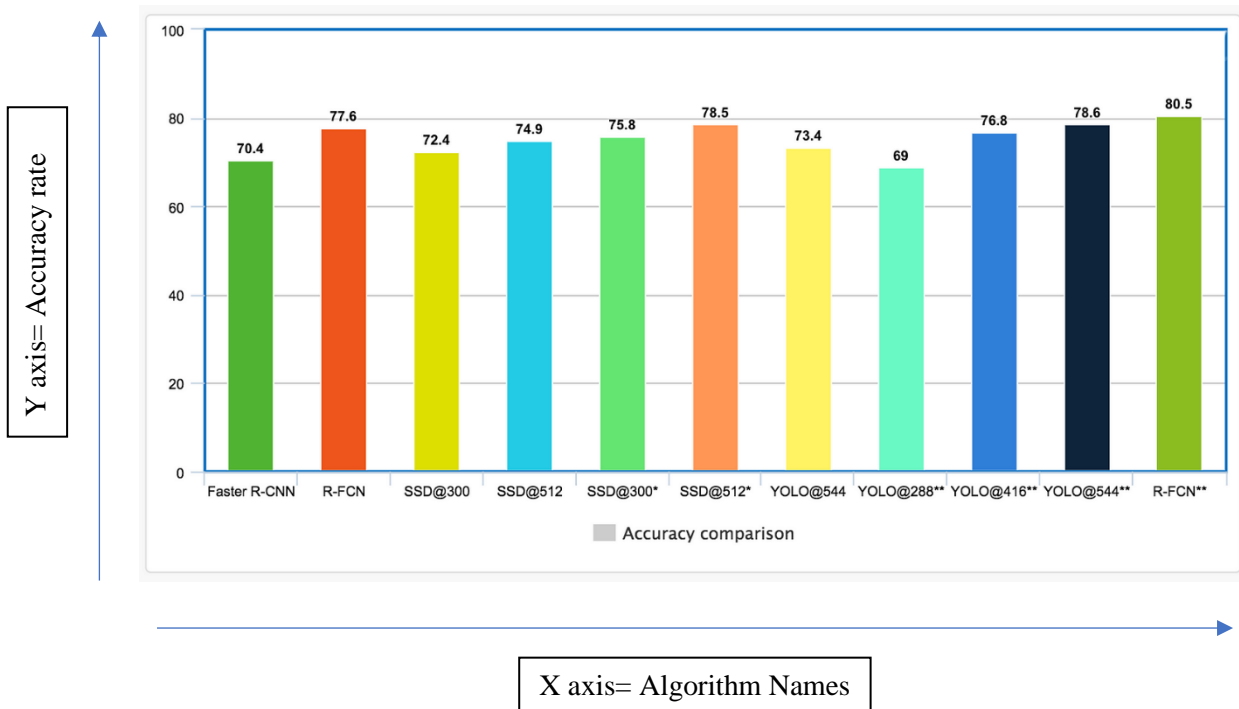


**FIG 6.4 Algorithm Comparison- FPS**

In Fig 6.1, the X axis represents the various algorithms, and the Y axis represents number of seconds. So, we can conclude that the new versions of YOLO have the highest FPS rate as compared to the previous algorithms. This makes the process faster in a real-time scenario.



### Comparison factor 2: Accuracy of the different algorithms



**FIG 6.5 Algorithm Comparison- Accuracy**

In Fig 6.2, X axis represents the different algorithms and the Y axis represents the accuracy. So, we can conclude that YOLO provides moderate accuracy as compared to SSD or R-FCN, so future models could make use of the newer algorithms for better accuracy in results.

## **CHAPTER 7:**

### **CONCLUSION AND FUTURE WORK**

The use of YOLO v3 in our project model has helped us in successfully achieving our goal of automating the process of urban traffic administration system.

After the implementation and execution, we have also performed manual testing in order to check for exceptions and correct them so that our model performs well in any scenario. The YOLO v3 model has been found to be quite precise and the efficiency was calculated using different parameters:

The overall accuracy of our project model was found to be 60%, which could be reduced further by using more advanced technologies like YOLO v7 and above.

As a part of our testing process, we have taken 5 overall test- cases, out of which two test cases worked with 100% accuracy. Out of the other three test cases, the efficiency varied, and we have considered an average of the data and come up with the average overall accuracy of our model as 60%.

As a part of future work, we intend to increase the efficiency of the model. Our project at this stage, due to lack of time and resources, is not a consolidated model yet which could be downloaded at once and be ready to use. Our model is still broken down into three modules, which need to be integrated with the help of comprehensive and easy to use User Interface. By incorporating cutting-edge services like remote parking spot reservations, online parking charge payments, and NFC, as well as enhancing our automobile algorithm, we will further build our suggested system in accordance with customer demands in future work.



- (1) Parking space surveillance. The motorist may use it to find out how many parking spots are available in a certain location.
- (2) Tracking traffic density. It enables the motorist to be aware of traffic density and jam rates before reaching a certain location.
- (3) Offering suggestions for other routes. It enables other paths to be provided around the specified location.

To further reduce energy use and lengthen the life of the WSN, we suggested a self-organizing algorithm. That will certainly serve our objective of helping automate the traffic monitoring process and thus help the traffic control rooms and departments to manage the huge volume of traffic- both cars and pedestrians, and thus prevent a high number of accidents and mishaps.

## REFERENCES

- [1] Gang Pan, Guande Qi, Wangsheng Zhang, Shijian Li, and Zhaohui Wu, Laurence Tianruo Yang, Trace Analysis and Mining for Smart Cities: Issues, Methods, and Applications. IEEE Trans. Communications Magazine. 51(6). pp 120-126.
  
- [2] Ganesh S. Khekare Apeksha V. Sakhare (2013, March). A Smart City Framework for Intelligent Traffic System Using VANET. Presented at International Multi Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s).
  
- [3] Malik, J., Weber, J., Luong, Q.T., Roller, D., “Smart Cars and Smart Roads”, Proceedings 6th. British Machine Vision Conference, 1995, pp 367-381.
  
- [4] Y. Kutsuma, H. Yaguchi and T. Hamamoto, "Real-time lane line and forward vehicle detection by smart image sensor," Communications and Information Technology, 2004. ISCIT 2004. IEEE International Symposium on, 2004, pp. 957-962 vol.2.
  
- [5] Sinhmar, P., “Intelligent Traffic Light and Density Control using IR Sensors and Microcontroller”, International Journal of Advanced Technology and Engineering Research (IJATER), ISSN No: 2250- 3536, volume 2, issue 2, March 2012.
  
- [6] Kanungo, A., Sharma, A., Singla, C., “Smart Traffic Lights Switching and Traffic Density Calculation using Video Processing”, Proceedings of 2014 RACES UIET Punjab University Chandigarh, 06-08 March 2014

- [7] Tyagi, V., Kalyanaraman, S., Krishnapuram, R., “Vehicular Traffic Density State Estimation Based on Cumulative Road Acoustics”, IEEE Transactions on Intelligent Transportation Systems, Volume 13, No.3, September 2012
- [8] S. Baldi, I. Michailidis, V. Ntampasi, E. Kosmatopoulos, I. Papamichail, and M. Papageorgiou, “A simulation-based traffic signal control for congested urban traffic networks,” *Transportation Science*, vol. 53, no. 1, pp. 6–20, 2019.
- [9] “The national transportation communications for ITS protocol,” 2013, <http://www.ntcip.org/>.
- [10] Tettamanti, Tamás., “Advanced Methods for Measurement and Control in Urban Road Traffic Networks,” 2013.
- [11] Arel, C. Liu, T. Urbanik, and A. G. Kohls, “Reinforcement learning-based multi-agent system for network traffic signal control,” *IET Intell. Transp. Syst.*, vol. 4, no. 2, p. 128, 2010.
- [12] Menon, R. Sinha, D. Ediga, I. Technology, and O. A. Road, “IMPLEMENTATION OF INTERNET OF THINGS IN BUS TRANSPORT SYSTEM OF SINGAPORE ”, 2013.
- [13] J. Park, Y. L. Murphey, R. McGee, J. G. Kristinsson, M. L. Kuang, and A. M. Phillips, “Intelligent trip modeling for the prediction of an origin-destination traveling speed profile,” *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 3, pp. 1039–1053, 2014.
- [14] S. K. Shukla and A. Agrawal, “Present and Future Perspective on Optimization of Road Network Management,” vol. 22, no. 2, pp. 64–67, 2015.

- [15] C. Chen, D. Zhang, N. Li, and Z. H. Zhou, “B-planner: Planning bidirectional night bus routes using large-scale taxi GPS traces,” *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 4, pp. 1451–1465, 2014.
- [16] Vu, S. Member, A. Ramanandan, A. Chen, J. A. Farrell, M. Barth, and S. Member, “Real-Time Computer Vision / DGPS-Aided Inertial Navigation System for Lane-Level Vehicle Navigation,” vol. 13, no. 2, pp. 899–913, 2012.
- [17] Srinivasan, M. C. Choy, and R. L. Cheu, “Neural networks for real-time traffic signal control,” *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 3, pp. 261–272, 2006.
- [18] Transportation, S. Its, J. Program, and O. Jpo, “Intelligent Transportation Systems (ITS) Strategic Plan 2015-2019,” 2014.

## APPENDIX

### SOURCE CODE

#### Module 1: Traffic Signal Violation Detection

##### *Object Detection.py*

```
import numpy as np
from keras.layers import Conv2D, Input, BatchNormalization, LeakyReLU,
ZeroPadding2D, UpSampling2D
from keras.layers.merge import add, concatenate
from keras.models import Model
import struct
import cv2

class WeightReader:
    def __init__(self, weight_file):
        with open(weight_file, 'rb') as w_f:
            major, = struct.unpack('i', w_f.read(4))
            minor, = struct.unpack('i', w_f.read(4))
            revision, = struct.unpack('i', w_f.read(4))

            if (major*10 + minor) >= 2 and major < 1000 and minor <
1000:
                w_f.read(8)
            else:
                w_f.read(4)

            transpose = (major > 1000) or (minor > 1000)

            binary = w_f.read()

            self.offset = 0
            self.all_weights = np.frombuffer(binary, dtype='float32')

    def read_bytes(self, size):
        self.offset = self.offset + size
        return self.all_weights[self.offset-size:self.offset]

    def load_weights(self, model):
        for i in range(106):
            try:
                conv_layer = model.get_layer('conv_' + str(i))
```

```

        print("loading weights of convolution #" + str(i))

        if i not in [81, 93, 105]:
            norm_layer = model.get_layer('bnorm_' + str(i))

            size = np.prod(norm_layer.get_weights()[0].shape)

            beta  = self.read_bytes(size) # bias
            gamma = self.read_bytes(size) # scale
            mean  = self.read_bytes(size) # mean
            var   = self.read_bytes(size) #
variance

            weights = norm_layer.set_weights([gamma, beta,
mean, var])

            if len(conv_layer.get_weights()) > 1:
                bias =
self.read_bytes(np.prod(conv_layer.get_weights()[1].shape))
                kernel =
self.read_bytes(np.prod(conv_layer.get_weights()[0].shape))

                kernel =
kernel.reshape(list(reversed(conv_layer.get_weights()[0].shape)))
                kernel = kernel.transpose([2,3,1,0])
                conv_layer.set_weights([kernel, bias])
            else:
                kernel =
self.read_bytes(np.prod(conv_layer.get_weights()[0].shape))
                kernel =
kernel.reshape(list(reversed(conv_layer.get_weights()[0].shape)))
                kernel = kernel.transpose([2,3,1,0])
                conv_layer.set_weights([kernel])
            except ValueError:
                print("no convolution #" + str(i))

        def reset(self):
            self.offset = 0

class BoundBox:
    def __init__(self, xmin, ymin, xmax, ymax, objness = None, classes
= None):
        self.xmin = xmin
        self.ymin = ymin
        self.xmax = xmax
        self.ymax = ymax

        self.objness = objness

```

```

        self.classes = classes

        self.label = -1
        self.score = -1

    def get_label(self):
        if self.label == -1:
            self.label = np.argmax(self.classes)

        return self.label

    def get_score(self):
        if self.score == -1:
            self.score = self.classes[self.get_label()]

        return self.score

def _conv_block(inp, convs, skip=True):
    x = inp
    count = 0

    for conv in convs:
        if count == (len(convs) - 2) and skip:
            skip_connection = x
            count += 1

            if conv['stride'] > 1: x = ZeroPadding2D(((1,0),(1,0)))(x) #
peculiar padding as darknet prefer left and top
            x = Conv2D(conv['filter'],
                        conv['kernel'],
                        strides=conv['stride'],
                        padding='valid' if conv['stride'] > 1 else 'same', #
peculiar padding as darknet prefer left and top
                        name='conv_' + str(conv['layer_idx']),
                        use_bias=False if conv['bnorm'] else True)(x)
            if conv['bnorm']: x = BatchNormalization(epsilon=0.001,
name='bnorm_' + str(conv['layer_idx']))(x)
            if conv['leaky']: x = LeakyReLU(alpha=0.1, name='leaky_' +
str(conv['layer_idx']))(x)

        return add([skip_connection, x]) if skip else x

def _interval_overlap(interval_a, interval_b):
    x1, x2 = interval_a
    x3, x4 = interval_b

    if x3 < x1:
        if x4 < x1:

```

```

        return 0
    else:
        return min(x2,x4) - x1
else:
    if x2 < x3:
        return 0
    else:
        return min(x2,x4) - x3

def _sigmoid(x):
    return 1. / (1. + np.exp(-x))

def bbox_iou(box1, box2):
    intersect_w = _interval_overlap([box1.xmin, box1.xmax], [box2.xmin,
box2.xmax])
    intersect_h = _interval_overlap([box1.ymin, box1.ymax], [box2.ymin,
box2.ymax])

    intersect = intersect_w * intersect_h

    w1, h1 = box1.xmax-box1.xmin, box1.ymax-box1.ymin
    w2, h2 = box2.xmax-box2.xmin, box2.ymax-box2.ymin

    union = w1*h1 + w2*h2 - intersect

    return float(intersect) / union

def make_yolov3_model():
    input_image = Input(shape=(None, None, 3))

    # Layer 0 => 4
    x = _conv_block(input_image, [{'filter': 32, 'kernel': 3, 'stride':
1, 'bnorm': True, 'leaky': True, 'layer_idx': 0},
                                {'filter': 64, 'kernel': 3, 'stride':
2, 'bnorm': True, 'leaky': True, 'layer_idx': 1},
                                {'filter': 32, 'kernel': 1, 'stride':
1, 'bnorm': True, 'leaky': True, 'layer_idx': 2},
                                {'filter': 64, 'kernel': 3, 'stride':
1, 'bnorm': True, 'leaky': True, 'layer_idx': 3}])

    # Layer 5 => 8
    x = _conv_block(x, [{'filter': 128, 'kernel': 3, 'stride': 2,
'bnorm': True, 'leaky': True, 'layer_idx': 5},
                        {'filter': 64, 'kernel': 1, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 6},
                        {'filter': 128, 'kernel': 3, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 7}])

```



```

# Layer 9 => 11
x = _conv_block(x, [{'filter': 64, 'kernel': 1, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 9},
                    {'filter': 128, 'kernel': 3, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 10}])

# Layer 12 => 15
x = _conv_block(x, [{'filter': 256, 'kernel': 3, 'stride': 2,
'bnorm': True, 'leaky': True, 'layer_idx': 12},
                    {'filter': 128, 'kernel': 1, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 13},
                    {'filter': 256, 'kernel': 3, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 14}])

# Layer 16 => 36
for i in range(7):
    x = _conv_block(x, [{'filter': 128, 'kernel': 1, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 16+i*3},
                        {'filter': 256, 'kernel': 3, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 17+i*3}])

skip_36 = x

# Layer 37 => 40
x = _conv_block(x, [{'filter': 512, 'kernel': 3, 'stride': 2,
'bnorm': True, 'leaky': True, 'layer_idx': 37},
                    {'filter': 256, 'kernel': 1, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 38},
                    {'filter': 512, 'kernel': 3, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 39}])

# Layer 41 => 61
for i in range(7):
    x = _conv_block(x, [{'filter': 256, 'kernel': 1, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 41+i*3},
                        {'filter': 512, 'kernel': 3, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 42+i*3}])

skip_61 = x

# Layer 62 => 65
x = _conv_block(x, [{'filter': 1024, 'kernel': 3, 'stride': 2,
'bnorm': True, 'leaky': True, 'layer_idx': 62},
                    {'filter': 512, 'kernel': 1, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 63},
                    {'filter': 1024, 'kernel': 3, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 64}])

```

```

# Layer 66 => 74
for i in range(3):
    x = _conv_block(x, [{'filter': 512, 'kernel': 1, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 66+i*3},
                        {'filter': 1024, 'kernel': 3, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 67+i*3}])

# Layer 75 => 79
x = _conv_block(x, [{'filter': 512, 'kernel': 1, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 75},
                    {'filter': 1024, 'kernel': 3, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 76},
                    {'filter': 512, 'kernel': 1, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 77},
                    {'filter': 1024, 'kernel': 3, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 78},
                    {'filter': 512, 'kernel': 1, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 79}], skip=False)

# Layer 80 => 82
yolo_82 = _conv_block(x, [{'filter': 1024, 'kernel': 3, 'stride':
1, 'bnorm': True, 'leaky': True, 'layer_idx': 80},
                        {'filter': 255, 'kernel': 1, 'stride':
1, 'bnorm': False, 'leaky': False, 'layer_idx': 81}], skip=False)

# Layer 83 => 86
x = _conv_block(x, [{'filter': 256, 'kernel': 1, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 84}], skip=False)
x = UpSampling2D(2)(x)
x = concatenate([x, skip_61])

# Layer 87 => 91
x = _conv_block(x, [{'filter': 256, 'kernel': 1, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 87},
                    {'filter': 512, 'kernel': 3, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 88},
                    {'filter': 256, 'kernel': 1, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 89},
                    {'filter': 512, 'kernel': 3, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 90},
                    {'filter': 256, 'kernel': 1, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 91}], skip=False)

# Layer 92 => 94
yolo_94 = _conv_block(x, [{'filter': 512, 'kernel': 3, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 92},
                        {'filter': 255, 'kernel': 1, 'stride': 1,
'bnorm': False, 'leaky': False, 'layer_idx': 93}], skip=False)

```

```

    # Layer 95 => 98
    x = _conv_block(x, [{'filter': 128, 'kernel': 1, 'stride': 1,
'bnorm': True, 'leaky': True, 'layer_idx': 96}], skip=False)
    x = UpSampling2D(2)(x)
    x = concatenate([x, skip_36])

    # Layer 99 => 106
    yolo_106 = _conv_block(x, [{'filter': 128, 'kernel': 1, 'stride':
1, 'bnorm': True, 'leaky': True, 'layer_idx': 99},
{'filter': 256, 'kernel': 3, 'stride':
1, 'bnorm': True, 'leaky': True, 'layer_idx': 100},
{'filter': 128, 'kernel': 1, 'stride':
1, 'bnorm': True, 'leaky': True, 'layer_idx': 101},
{'filter': 256, 'kernel': 3, 'stride':
1, 'bnorm': True, 'leaky': True, 'layer_idx': 102},
{'filter': 128, 'kernel': 1, 'stride':
1, 'bnorm': True, 'leaky': True, 'layer_idx': 103},
{'filter': 256, 'kernel': 3, 'stride':
1, 'bnorm': True, 'leaky': True, 'layer_idx': 104},
{'filter': 255, 'kernel': 1, 'stride':
1, 'bnorm': False, 'leaky': False, 'layer_idx': 105}], skip=False)

    model = Model(input_image, [yolo_82, yolo_94, yolo_106])
    return model

def preprocess_input(image, net_h, net_w):
    new_h, new_w, _ = image.shape

    # determine the new size of the image
    if (float(net_w)/new_w) < (float(net_h)/new_h):
        new_h = (new_h * net_w)/new_w
        new_w = net_w
    else:
        new_w = (new_w * net_h)/new_h
        new_h = net_h

    # resize the image to the new size
    resized = cv2.resize(image[:,:,:-1]/255., (int(new_w),
int(new_h)))

    # embed the image into the standard letter box
    new_image = np.ones((net_h, net_w, 3)) * 0.5
    new_image[int((net_h-new_h)//2):int((net_h+new_h)//2), int((net_w-
new_w)//2):int((net_w+new_w)//2), :] = resized
    new_image = np.expand_dims(new_image, 0)

    return new_image

```

```

def decode_netout(netout, anchors, obj_thresh, nms_thresh, net_h,
net_w):
    grid_h, grid_w = netout.shape[:2]
    nb_box = 3
    netout = netout.reshape((grid_h, grid_w, nb_box, -1))
    nb_class = netout.shape[-1] - 5

    boxes = []

    netout[..., :2] = _sigmoid(netout[..., :2])
    netout[..., 4:] = _sigmoid(netout[..., 4:])
    netout[..., 5:] = netout[..., 4][..., np.newaxis] * netout[...
5:]

    netout[..., 5:] *= netout[..., 5:] > obj_thresh

    for i in range(grid_h*grid_w):
        row = i / grid_w
        col = i % grid_w

        for b in range(nb_box):
            # 4th element is objectness score
            objectness = netout[int(row)][int(col)][b][4]
            #objectness = netout[..., :4]

            if(objectness.all() <= obj_thresh): continue

            # first 4 elements are x, y, w, and h
            x, y, w, h = netout[int(row)][int(col)][b][:4]

            x = (col + x) / grid_w # center position, unit: image width
            y = (row + y) / grid_h # center position, unit: image
height
            w = anchors[2 * b + 0] * np.exp(w) / net_w # unit: image
width
            h = anchors[2 * b + 1] * np.exp(h) / net_h # unit: image
height

            # last elements are class probabilities
            classes = netout[int(row)][col][b][5:]

            box = BoundBox(x-w/2, y-h/2, x+w/2, y+h/2, objectness,
classes)
            #box = BoundBox(x-w/2, y-h/2, x+w/2, y+h/2, None, classes)

            boxes.append(box)

    return boxes

```

```

def correct_yolo_boxes(boxes, image_h, image_w, net_h, net_w):
    if (float(net_w)/image_w) < (float(net_h)/image_h):
        new_w = net_w
        new_h = (image_h*net_w)/image_w
    else:
        new_h = net_h
        new_w = (image_w*net_h)/image_h

    for i in range(len(boxes)):
        x_offset, x_scale = (net_w - new_w)/2./net_w,
float(new_w)/net_w
        y_offset, y_scale = (net_h - new_h)/2./net_h,
float(new_h)/net_h

        boxes[i].xmin = int((boxes[i].xmin - x_offset) / x_scale *
image_w)
        boxes[i].xmax = int((boxes[i].xmax - x_offset) / x_scale *
image_w)
        boxes[i].ymin = int((boxes[i].ymin - y_offset) / y_scale *
image_h)
        boxes[i].ymax = int((boxes[i].ymax - y_offset) / y_scale *
image_h)

def do_nms(boxes, nms_thresh):
    if len(boxes) > 0:
        nb_class = len(boxes[0].classes)
    else:
        return

    for c in range(nb_class):
        sorted_indices = np.argsort([-box.classes[c] for box in boxes])

        for i in range(len(sorted_indices)):
            index_i = sorted_indices[i]

            if boxes[index_i].classes[c] == 0: continue

            for j in range(i+1, len(sorted_indices)):
                index_j = sorted_indices[j]

                if bbox_iou(boxes[index_i], boxes[index_j]) >=
nms_thresh:
                    boxes[index_j].classes[c] = 0

def draw_boxes(image, boxes, line, labels, obj_thresh, dcnt):
    print(line)

```

```

for box in boxes:
    label_str = ''
    label = -1

    for i in range(len(labels)):
        if box.classes[i] > obj_thresh:
            label_str += labels[i]
            label = i
            print(labels[i] + ': ' + str(box.classes[i]*100) + '%')
            print('line: (' + str(line[0][0]) + ', ' +
str(line[0][1]) + ') (' + str(line[1][0]) + ', ' + str(line[1][1]) +
''))
            print('Box: (' + str(box.xmin) + ', ' + str(box.ymin) +
') (' + str(box.xmax) + ', ' + str(box.ymax) + '))')
            print()

    if label >= 0:
        tf = False

        (rxmin, rymin) = (box.xmin, box.ymin)
        (rxmax, rymax) = (box.xmax, box.ymax)

        tf = False
        tf |= intersection(line[0], line[1], (rxmin, rymin),
(rxmin, rymax))
        tf |= intersection(line[0], line[1], (rxmax, rymin),
(rxmax, rymax))
        tf |= intersection(line[0], line[1], (rxmin, rymin),
(rxmax, rymin))
        tf |= intersection(line[0], line[1], (rxmin, rymax),
(rxmax, rymax))

        print(tf)

        cv2.line(image, line[0], line[1], (255, 0, 0), 3)

        if tf:
            cv2.rectangle(image, (box.xmin,box.ymin),
(box.xmax,box.ymax), (255,0,0), 3)
            cimg = image[box.ymin:box.ymax, box.xmin:box.xmax]
            cv2.imshow("violation", cimg)
            cv2.waitKey(5)
            cv2.imwrite("C:/Users/HP/Desktop/Traffic-Signal-
Violation-Detection-System/Detected
Images/violation_"+str(dcnt)+".jpg", cimg)
            dcnt = dcnt+1
        else:

```

```

        cv2.rectangle(image, (box.xmin,box.ymin),
(box.xmax,box.ymax), (0,255,0), 3)

        cv2.putText(image,
                    label_str + ' ' + str(round(box.get_score(),
2)),
                    (box.xmin, box.ymin - 13),
                    cv2.FONT_HERSHEY_SIMPLEX,
                    1e-3 * image.shape[0],
                    (0,255,0), 2)

    return image

weights_path = "C:/Users/HP/Desktop/Traffic-Signal-Violation-Detection-System/yolov3.weights"
# set some parameters
net_h, net_w = 416, 416
obj_thresh, nms_thresh = 0.5, 0.45
anchors = [[116,90, 156,198, 373,326], [30,61, 62,45, 59,119],
[10,13, 16,30, 33,23]]
labels = ["person", "bicycle", "car", "motorbike", "aeroplane", "bus",
"train", "truck", \
        "boat", "traffic light", "fire hydrant", "stop sign",
"parking meter", "bench", \
        "bird", "cat", "dog", "horse", "sheep", "cow", "elephant",
"bear", "zebra", "giraffe", \
        "backpack", "umbrella", "handbag", "tie", "suitcase",
"frisbee", "skis", "snowboard", \
        "sports ball", "kite", "baseball bat", "baseball glove",
"skateboard", "surfboard", \
        "tennis racket", "bottle", "wine glass", "cup", "fork",
"knife", "spoon", "bowl", "banana", \
        "apple", "sandwich", "orange", "broccoli", "carrot", "hot
dog", "pizza", "donut", "cake", \
        "chair", "sofa", "pottedplant", "bed", "diningtable",
"toilet", "tvmonitor", "laptop", "mouse", \
        "remote", "keyboard", "cell phone", "microwave", "oven",
"toaster", "sink", "refrigerator", \
        "book", "clock", "vase", "scissors", "teddy bear", "hair
drier", "toothbrush"]

# make the yolov3 model to predict 80 classes on COCO
yolov3 = make_yolov3_model()

# load the weights trained on COCO into the model
weight_reader = WeightReader(weights_path)
weight_reader.load_weights(yolov3)

```

```

# my defined functions
def intersection(p, q, r, t):
    print(p, q, r, t)
    (x1, y1) = p
    (x2, y2) = q

    (x3, y3) = r
    (x4, y4) = t

    a1 = y1-y2
    b1 = x2-x1
    c1 = x1*y2-x2*y1

    a2 = y3-y4
    b2 = x4-x3
    c2 = x3*y4-x4*y3

    if(a1*b2-a2*b1 == 0):
        return False
    print((a1, b1, c1), (a2, b2, c2))
    x = (b1*c2 - b2*c1) / (a1*b2 - a2*b1)
    y = (a2*c1 - a1*c2) / (a1*b2 - a2*b1)
    print((x, y))

    if x1 > x2:
        tmp = x1
        x1 = x2
        x2 = tmp
    if y1 > y2:
        tmp = y1
        y1 = y2
        y2 = tmp
    if x3 > x4:
        tmp = x3
        x3 = x4
        x4 = tmp
    if y3 > y4:
        tmp = y3
        y3 = y4
        y4 = tmp

    if x >= x1 and x <= x2 and y >= y1 and y <= y2 and x >= x3 and x <=
x4 and y >= y3 and y <= y4:
        return True
    else:
        return False

```



## *Project GUI.py*

```

from tkinter import *
from PIL import Image, ImageTk
from tkinter import filedialog
import object_detection as od
import numpy as np
import imageio
import cv2

class Window(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)

        self.master = master
        self.pos = []
        self.line = []
        self.rect = []
        self.master.title("GUI")
        self.pack(fill=BOTH, expand=1)

        self.counter = 0

        menu = Menu(self.master)
        self.master.config(menu=menu)

        file = Menu(menu)
        file.add_command(label="Open", command=self.open_file)
        file.add_command(label="Exit", command=self.client_exit)
        menu.add_cascade(label="File", menu=file)

        analyze = Menu(menu)
        analyze.add_command(label="Region of Interest",
command=self.regionOfInterest)
        menu.add_cascade(label="Analyze", menu=analyze)

        self.filename = "Images/home.jpg"
        self.imgSize = Image.open(self.filename)
        self.tkimage = ImageTk.PhotoImage(self.imgSize)
        self.w, self.h = (1366, 768)

        self.canvas = Canvas(master = root, width = self.w, height =
self.h)
        self.canvas.create_image(20, 20, image=self.tkimage,
anchor='nw')
        self.canvas.pack()

```

```

def open_file(self):
    self.filename = filedialog.askopenfilename()

    cap = cv2.VideoCapture(self.filename)

    reader = imageio.get_reader(self.filename)
    fps = reader.get_meta_data()['fps']

    ret, image = cap.read()
    cv2.imwrite('C:/Users/HP/Desktop/Traffic-Signal-Violation-
Detection-System/Images/preview.jpg', image)

    self.show_image('C:/Users/HP/Desktop/Traffic-Signal-Violation-
Detection-System/Images/preview.jpg')

def show_image(self, frame):
    self.imgSize = Image.open(frame)
    self.tkimage = ImageTk.PhotoImage(self.imgSize)
    self.w, self.h = (1366, 768)

    self.canvas.destroy()

    self.canvas = Canvas(master = root, width = self.w, height =
self.h)
    self.canvas.create_image(0, 0, image=self.tkimage, anchor='nw')
    self.canvas.pack()

def regionOfInterest(self):
    root.config(cursor="plus")
    self.canvas.bind("<Button-1>", self.imgClick)

def client_exit(self):
    exit()

def imgClick(self, event):

    if self.counter < 2:
        x = int(self.canvas.canvasx(event.x))
        y = int(self.canvas.canvasy(event.y))
        self.line.append((x, y))
        self.pos.append(self.canvas.create_line(x - 5, y, x + 5, y,
fill="red", tags="crosshair"))
        self.pos.append(self.canvas.create_line(x, y - 5, x, y + 5,
fill="red", tags="crosshair"))
        self.counter += 1

    # elif self.counter < 4:

```

```

        # x = int(self.canvas.canvasx(event.x))
        # y = int(self.canvas.canvasy(event.y))
        # self.rect.append((x, y))
        # self.pos.append(self.canvas.create_line(x - 5, y, x + 5,
y, fill="red", tags="crosshair"))
        # self.pos.append(self.canvas.create_line(x, y - 5, x, y +
5, fill="red", tags="crosshair"))
        # self.counter += 1

    if self.counter == 2:
        #unbinding action with mouse-click
        self.canvas.unbind("<Button-1>")
        root.config(cursor="arrow")
        self.counter = 0

        #show created virtual line
        print(self.line)
        print(self.rect)
        img = cv2.imread('C:/Users/HP/Desktop/Traffic-Signal-
Violation-Detection-System/Images/preview.jpg')
        cv2.line(img, self.line[0], self.line[1], (0, 255, 0), 3)
        cv2.imwrite('C:/Users/HP/Desktop/Traffic-Signal-Violation-
Detection-System/Images/copy.jpg', img)
        self.show_image('C:/Users/HP/Desktop/Traffic-Signal-
Violation-Detection-System/Images/copy.jpg')

        ## for demonstration
        # (rxmin, rymin) = self.rect[0]
        # (rxmax, rymax) = self.rect[1]

        # tf = False
        # tf |= self.intersection(self.line[0], self.line[1],
(rxmin, rymin), (rxmin, rymax))
        # print(tf)
        # tf |= self.intersection(self.line[0], self.line[1],
(rxmax, rymin), (rxmax, rymax))
        # print(tf)
        # tf |= self.intersection(self.line[0], self.line[1],
(rxmin, rymin), (rxmax, rymin))
        # print(tf)
        # tf |= self.intersection(self.line[0], self.line[1],
(rxmin, rymax), (rxmax, rymax))
        # print(tf)

        # cv2.line(img, self.line[0], self.line[1], (0, 255, 0), 3)

        # if tf:

```

```

        # cv2.rectangle(img, (rxmin,rymin), (rxmax,rymax),
(255,0,0), 3)
        # else:
        # cv2.rectangle(img, (rxmin,rymin), (rxmax,rymax),
(0,255,0), 3)

        # cv2.imshow('traffic violation', img)

        #image processing
        self.main_process()
        print("Executed Successfully!!!")

        #clearing things
        self.line.clear()
        self.rect.clear()
        for i in self.pos:
            self.canvas.delete(i)

def intersection(self, p, q, r, t):
    print(p, q, r, t)
    (x1, y1) = p
    (x2, y2) = q

    (x3, y3) = r
    (x4, y4) = t

    a1 = y1-y2
    b1 = x2-x1
    c1 = x1*y2-x2*y1

    a2 = y3-y4
    b2 = x4-x3
    c2 = x3*y4-x4*y3

    if(a1*b2-a2*b1 == 0):
        return False
    print((a1, b1, c1), (a2, b2, c2))
    x = (b1*c2 - b2*c1) / (a1*b2 - a2*b1)
    y = (a2*c1 - a1*c2) / (a1*b2 - a2*b1)
    print((x, y))

    if x1 > x2:
        tmp = x1
        x1 = x2
        x2 = tmp
    if y1 > y2:
        tmp = y1
        y1 = y2

```

```

        y2 = tmp
    if x3 > x4:
        tmp = x3
        x3 = x4
        x4 = tmp
    if y3 > y4:
        tmp = y3
        y3 = y4
        y4 = tmp

    if x >= x1 and x <= x2 and y >= y1 and y <= y2 and x >= x3 and
x <= x4 and y >= y3 and y <= y4:
        return True
    else:
        return False

def main_process(self):

    video_src = self.filename

    cap = cv2.VideoCapture(video_src)

    reader = imageio.get_reader(video_src)
    fps = reader.get_meta_data()['fps']
    writer = imageio.get_writer('C:/Users/HP/Desktop/Traffic-
Signal-Violation-Detection-System/Resources/output/output.mp4', fps =
fps)

    j = 1
    while True:
        ret, image = cap.read()

        if (type(image) == type(None)):
            writer.close()
            break

        image_h, image_w, _ = image.shape
        new_image = od.preprocess_input(image, od.net_h, od.net_w)

        # run the prediction
        yolos = od.yolov3.predict(new_image)
        boxes = []

        for i in range(len(yolos)):
            # decode the output of the network
            boxes += od.decode_netout(yolos[i][0], od.anchors[i],
od.obj_thresh, od.nms_thresh, od.net_h, od.net_w)

```

```

        # correct the sizes of the bounding boxes
        od.correct_yolo_boxes(boxes, image_h, image_w, od.net_h,
od.net_w)

        # suppress non-maximal boxes
        od.do_nms(boxes, od.nms_thresh)

        # draw bounding boxes on the image using labels
        image2 = od.draw_boxes(image, boxes, self.line, od.labels,
od.obj_thresh, j)

        writer.append_data(image2)

        # cv2.imwrite('C:/Users/HP/Desktop/Traffic-Signal-
Violation-Detection-System/Images/frame'+str(j)+'.jpg', image2)
        # self.show_image('C:/Users/HP/Desktop/Traffic-Signal-
Violation-Detection-System/Images/frame'+str(j)+'.jpg')

        cv2.imshow('Traffic Violation', image2)

        print(j)

        if cv2.waitKey(10) & 0xFF == ord('q'):
            writer.close()
            break

        j = j+1

    cv2.destroyAllWindows()

root = Tk()
app = Window(root)
root.geometry("%dx%d"%(535, 380))
root.title("Traffic Violation")

root.mainloop()

```

## Module 2: Pedestrian Detection

### *DetectPedestrian.py*

```
import numpy as np
import cv2

image = cv2.imread('people1.jpg')

# initialize the HOG descriptor
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

# detect humans in input image
(humans, _) = hog.detectMultiScale(image, winStride=(10, 10),
padding=(32, 32), scale=1.1)

print('Pedestrians Detected : ', len(humans))
countperweek=len(humans)*720*7
print('Approximate Pedestrian Count per Week: ',countperweek)
if (countperweek<=50000):
    print ("Low Pedestrian Density! RECOMMENDATION: IMPOSE SPEED LIMIT RULES")
elif (countperweek>50000 and countperweek<75000):
    print ("Crowded Zone! RECOMMENDATION: BUILD A FOOTBRIDGE")
elif (countperweek>75000):
    print ("Heavily Crowded Zone! RECOMMENDATION: BUILD A SUBWAY")

for (x, y, w, h) in humans:
    pad_w, pad_h = int(0.15 * w), int(0.01 * h)
    cv2.rectangle(image, (x + pad_w, y + pad_h), (x + w - pad_w, y + h - pad_h), (0, 255, 0), 2)

cv2.imshow("Image", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## Module 3: Helmet and Number Plate Detection

### *Detect.py*

```
import cv2
import numpy as np
import os
import imutils
from tensorflow.keras.models import load_model

os.environ['TF_FORCE_GPU_ALLOW_GROWTH'] = 'true'
net = cv2.dnn.readNet("yolov3-custom_7000.weights", "yolov3-
custom.cfg")
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)

model = load_model('helmet-nonhelmet_cnn.h5')
print('model loaded!!!')

cap = cv2.VideoCapture('video.mp4')
COLORS = [(0,255,0),(0,0,255)]

layer_names = net.getLayerNames()
output_layers = [layer_names[i - 1] for i in
net.getUnconnectedOutLayers()]

fourcc = cv2.VideoWriter_fourcc(*"XVID")
writer = cv2.VideoWriter('output.avi', fourcc, 5,(888,500))

def helmet_or_nohelmet(helmet_roi):
    try:
        helmet_roi = cv2.resize(helmet_roi, (224, 224))
        helmet_roi = np.array(helmet_roi,dtype='float32')
        helmet_roi = helmet_roi.reshape(1, 224, 224, 3)
        helmet_roi = helmet_roi/255.0
        return int(model.predict(helmet_roi)[0][0])
    except:
        pass

ret = True

while ret:

    ret, img = cap.read()
    img = imutils.resize(img,height=500)
    # img = cv2.imread('test.png')
```



```

height, width = img.shape[:2]

blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0),
True, crop=False)

net.setInput(blob)
outs = net.forward(output_layers)

confidences = []
boxes = []
classIds = []

for out in outs:
    for detection in out:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.3:
            center_x = int(detection[0] * width)
            center_y = int(detection[1] * height)

            w = int(detection[2] * width)
            h = int(detection[3] * height)
            x = int(center_x - w / 2)
            y = int(center_y - h / 2)

            boxes.append([x, y, w, h])
            confidences.append(float(confidence))
            classIds.append(class_id)

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

for i in range(len(boxes)):
    if i in indexes:
        x,y,w,h = boxes[i]
        color = [int(c) for c in COLORS[classIds[i]]]
        # green --> bike
        # red --> number plate
        if classIds[i]==0: #bike
            helmet_roi =
img[max(0,y):max(0,y)+max(0,h)//4,max(0,x):max(0,x)+max(0,w)]
        else: #number plate
            x_h = x-60
            y_h = y-350
            w_h = w+100
            h_h = h+100
            cv2.rectangle(img, (x, y), (x + w, y + h), color, 7)

```

```

        # h_r = img[max(0,(y-330)):max(0,(y-330 + h+100)) ,
max(0,(x-80)):max(0,(x-80 + w+130))]
        if y_h>0 and x_h>0:
            h_r = img[y_h:y_h+h_h , x_h:x_h+w_h]
            c = helmet_or_nohelmet(h_r)
            cv2.putText(img,['helmet','no-helmet'][c],(x,y-
100),cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0),2)
            cv2.rectangle(img, (x_h, y_h), (x_h + w_h, y_h +
h_h),(255,0,0), 10)

writer.write(img)
cv2.imshow("Image", img)

if cv2.waitKey(1) == 27:
    break
writer.release()
cap.release()
cv2.waitKey(0)
cv2.destroyAllWindows()

```

# PLAGIARISM REPORT

## Plagchecktraffic

### ORIGINALITY REPORT

<b>3%</b>	<b>1%</b>	<b>1%</b>	<b>3%</b>
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

### PRIMARY SOURCES

<b>1</b>	<b>towardsdatascience.com</b> Internet Source	<b>1%</b>
<b>2</b>	<b>Submitted to University of East London</b> Student Paper	<b>&lt;1%</b>
<b>3</b>	<b>Submitted to Dr. B R Ambedkar National Institute of Technology, Jalandhar</b> Student Paper	<b>&lt;1%</b>
<b>4</b>	<b>Submitted to University of Lugano</b> Student Paper	<b>&lt;1%</b>
<b>5</b>	<b>Submitted to Coventry University</b> Student Paper	<b>&lt;1%</b>
<b>6</b>	<b>Submitted to University of Stirling</b> Student Paper	<b>&lt;1%</b>
<b>7</b>	<b>Submitted to University of Perpetual Help Las Pinas System Dalta</b> Student Paper	<b>&lt;1%</b>
<b>8</b>	<b>cse.anits.edu.in</b> Internet Source	<b>&lt;1%</b>
<b>9</b>	<b>Submitted to Nottingham Trent University</b>	

	Student Paper	<1 %
10	Submitted to Southampton Solent University Student Paper	<1 %
11	Submitted to University of Derby Student Paper	<1 %
12	Submitted to University of Hertfordshire Student Paper	<1 %
13	<a href="http://www.researchgate.net">www.researchgate.net</a> Internet Source	<1 %
14	"Smart Societies, Infrastructure, Technologies and Applications", Springer Science and Business Media LLC, 2018 Publication	<1 %
15	Di Mauro, E.C., T.F. Cootes, G.J. Page, and C.B. Jackson. "Check! A generic and specific industrial inspection tool", IEE Proceedings - Vision Image and Signal Processing, 1996. Publication	<1 %
16	<a href="http://www.arxiv-vanity.com">www.arxiv-vanity.com</a> Internet Source	<1 %

Exclude quotes    On  
Exclude bibliography    On

Exclude matches    Off

## JOURNAL PUBLICATION



## Format – I

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY		
(Deemed to be University u/s 3 of UGC Act, 1956)		
Office of Controller of Examinations		
REPORT FOR PLAGIARISM CHECK ON THE DISSERTATION/PROJECT REPORTS FOR UG/PG PROGRAMMES (To be attached in the dissertation/ project report)		
1	Name of the Candidate (IN BLOCK LETTERS)	PRATIKSHA GHOSH
2	Address of the Candidate	98A/17, DR. S.C. Banerjee Road, Surya Residency, Belegghata, Kolkata- 700010, West Bengal. <b>Mobile Number:</b> 8334097485
3	Registration Number	RA1911028010142
4	Date of Birth	13th November 2000
5	Department	Network and Communications (NwC)
6	Faculty	Engineering and Technology, School of Computing
7	Title of the Dissertation/Project	Urban Traffic Administration System
8	Whether the above project /dissertation is done by	<p>Individual or group: (Strike whichever is not applicable )</p> <p>If the project/ dissertation is done in group, then how many students together completed the project : 2 (Two)</p> <p>Mention the Name &amp; Register number of other candidates</p> <p>Gevariya Meet B, RA1911028010131</p>
9	Name and address of the Supervisor /Guide	<p>Dr. Murugaanandam S, Associate Professor</p> <p>Department of Network and Communications SRM Institute of Science and Technology Kattankulatur - 603 203</p> <p><b>Mail ID:</b> <a href="mailto:murugaas@srmist.edu.in">murugaas@srmist.edu.in</a></p> <p><b>Mobile Number:</b> 9994140019</p>
10	Name and address of Co-Supervisor /Co-Guide (if any)	<p>NIL</p> <p><b>Mail ID: Mobile Number:</b></p>

11	Software Used	Turnitin		
12	Date of Verification			
13	<b>Plagiarism Details: (to attach the final report from the software)</b>			
Chapter	Title of the Chapter	Percentage of similarity index (including self citation)	Percentage of similarity index (Excluding self citation)	% of plagiarismafter excluding Quotes, Bibliography, etc.,
1	INTRODUCTION	0%	0%	2%
2	LITERATURE SURVEY	1%	1%	1%
3	TECHNICAL SPECIFICATIONS	1%	1%	0%
4	ARCHITECTURE	1%	0%	0%
5	MODULES	0%	0%	0%
6	DEMONSTRATION AND RESULTS	0%	1%	0%
7	CONCLUSION	0%	0%	0%
8	FUTURE ENHANCEMENTS	0%	0%	0%
9				
10				
<b>Appendices</b>		3%	3%	3%
I / We declare that the above information have been verified and found true to the best of my / our knowledge.				
<b>Signature of the Candidate</b>		<b>Name &amp; Signature of the Staff (Who uses the plagiarism check software)</b>		
<b>Name &amp; Signature of the Supervisor/ Guide</b>		<b>Name &amp; Signature of the Co-Supervisor/Co-Guide</b>		
<b>Name &amp; Signature of the HOD</b>				