OBJECT TRACKING BASED TRAFFIC ANALYSIS USING ML

Project Report Submitted

In Partial Fulfillment of the Requirements

For the Degree Of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

SOHAIL UDDIN SYED	**(1604-19-733-016)**
MUTTAHAR KHALID	**(1604-19-733-022)**
RAHMAN MEHMOOD	**(1604-19-733-039)**

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

MUFFAKHAM JAH COLLEGE OF ENGINEERING & TECHNOLOGY

(Affiliated to Osmania University)

Mount Pleasant, 8-2-249, Road No. 3, Banjara Hills, Hyderabad-34

2023

DECLARATION

This is to certify that the work reported in the major project entitled "**OBJECT TRACKING BASED TRAFFIC ANALYSIS USING ML***"* is a record of the bonafide work done by us in the Department of Computer Science and Engineering, Muffakham Jah College of Engineering and Technology, Osmania University. The results embodied in this report are based on the project work done entirely by us and not copied from any other source.

1. **SOHAIL UDDIN SYED (1604-19-733-016) -**
2. **MUTTAHAR KHALID (1604-19-733-022) -**
3. **RAHMAN MEHMOOD (1604-19-733-039) -**

ACKNOWLEDGEMENT

At this moment of accomplishment, we are presenting our work with great pride and pleasure, we would like to express our sincere gratitude to all those who helped us in the successful completion of our venture.

We are very much indebted and express our deep sense of gratitude to **Dr. Syed Shabbeer Ahmad (Professor CSED, MJCET),** our project supervisor, who took keen interest throughout the course of the project. Our zeal was kept alive throughout the project and he provided the much needed guidance and unflinching support to complete the project. Without his constant encouragement and guidance this dissertation work could not have taken the present shape.

We sincerely thank the Project Coordinator: Dr. Shabbeer Ahmad, Professor, CSE Department, MJCET, for the in-depth and tough appraisals conducted by them and for their timely and qualitative suggestions that truly helped in improving the quality of the project work. More importantly the thesis writing guidelines provided by them was a blessing in disguise to write the thesis in proper structure and format overcoming major drawbacks.

We would like to thank **Dr. Syed Shabbeer Ahmad**, Professor and Head, Department of Computer Science and Engineering, for his endless support and guidance throughout the course. All his valuable suggestions and feedback has helped us a lot to mould our self. He has also been a great source of inspiration in terms of work and studies.

SOHAIL UDDIN SYED

MUTTAHAR KHALID

RAHMAN MEHMOOD

Object Tracking Based Traffic Analysis Using ML

ABSTRACT

Intelligent vehicle detection is becoming increasingly important in the field of highway management. However, due to the different sizes of vehicles, their detection remains a challenge that directly affects the accuracy of vehicle counts. To address this issue, this paper proposes a vision-based vehicle detection and counting system using You Only Look Once (YOLO-V7) based Byte Track model for real time vehicle detection and tracking from video sequences. Byte Track algorithm is added, which will track actual presence of vehicles from video frame predicted by YOLO-V7 so the false prediction perform by YOLOV7 can be avoid by using Byte Track algorithm. The video will be converted into multiple frames and give as input to YOLO-V7 for vehicle detection. The detected vehicle frame will be further analysed by Byte Track algorithm to track vehicle and if vehicle tracked then Byte Track will put bounding box across tracked vehicle and increment the tracking count. The proposed model is trained with three different datasets such as public and custom collected dataset

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVATIONS

CV	Computer Vision
OpenCV	Open-Source Computer Vision Library
YOLO	You Only Look Once
ANPR	Automatic Number Plate Recognition
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DSC	Digital Signal Controller
GDP	Gross Domestic Product
OCR	Optical Character Recognition
IVS	Intelligent Visual Surveillance
ITS	Intelligent Transport System
PDF	Portable Document Format
PERCLOS	Percentage of Eye Closure
GTI	Gran Turismo Injection
HOG	Histogram of oriented gradients
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute
AUROC	Area under the ROC Curve
LinearSVC	Linear Support Vector Classification
CCTV	Closed Circuit Television
DVR	Digital Video Recorder
NVR	Network Video Recorder

FPS	Frames per Second
COCOMO	Constructive Cost Model
LOC	Lines of Code
CPU	Central Processing Unit
GPU	Graphical Processing Unit
YOLOV7	You Only Look Once Version 7
UML	Unified Modelling Language
MARS	Motion Analysis and Re-identification
IoU	Intersection over Union
AI	Artificial Intelligence
ML	Machine Learning
ANPR	Automatic Number Plate Recognition

CHAPTER 1
INTRODUCTION

1.1 Introduction

Road accidents in India are a major cause of decreasing life expectancy with road accidents contributing to over 148,000 deaths out of 467,000 deaths in 2016. Indian Economy has a hit of 3 percent of GDP growth due to road accidents as per the United Nations with an estimated loss of $58,000 in terms of value every year. The metropolitan cities such as Chennai, Mumbai and New Delhi have been increasingly highlighted for lack of road safety and rash driving cases. The recent trends show that there has been an increase in the global number of road accidents even in developed countries.

However, under-developed and developing countries suffer a more significant impact due to life and economic losses. These accidents occur due to violation of traffic safety rules, careless rash driving, driver drowsiness and lack of good quality roads. The problem becomes more adverse for highways and hilly areas where accidents are unavoidable. Road accidents are characterized by high death rates due to delay in arrival of help and inefficient systems of mitigation to alert the concerned authorities. Road accidents on the highways are typically caused by natural reasons such as extreme weather conditions such as fog and consecutive collision of vehicles are common on Indian. highways due to lack of visibility. The states of Maharashtra, Tamil Nadu and Uttar Pradesh account for the highest number of road accidents in India.

The problem can be handled by making use of computer vision and low-cost sensor networks. The current solutions involved heavy dependency on sensor networks and area coverage. This can be substantially replaced by making use of object detection and image segmentation for accident classification. The system identifies the accident-prone areas which are the target stakeholders for the deployment and sets it apart from other implementations since it provides a feasibility factor associated with it. Furthermore, the system provides enhanced mitigation alert to the concerned authorities which helps in preventing any consecutive collisions that could possibly lead to greater loss of lives.

1.2 Computer Vision

Computer vision is the field of computer science that focuses on replicating parts of the complexity of the human vision system and enabling computers to identify and process objects in images and videos in the same way that humans do. Until recently, computer vision only worked in limited capacity. Thanks to advances in artificial intelligence and innovations in deep learning and neural networks, the field has been able to take great leaps in recent years and has been able to surpass humans in some tasks related to detecting and labeling objects. One of the driving factors behind the growth of 3 computer vision is the amount of data we generate today that is then used to train and make computer vision better. Along with a tremendous amount of visual data (more than 3 billion images are shared online every day), the computing power required to analyze the data is now accessible. As the field of computer vision has grown with new hardware and algorithms so has the accuracy rates for object identification. The goal of computer vision is to derive descriptive information about a scene by computer analysis of images of the scene. Vision algorithms can often serve as computational models for biological visual processes, and they also have many practical uses; but vision problems are often ill-defined, ill-posed, or computationally intractable; nevertheless, successes have been achieved in many specific areas like document processing and industrial inspection, for example. By limiting the domain of application, carefully choosing the task, using redundant data (multi-sensor, multi-frame), and applying adequate computing power, useful solutions to many vision problems can be obtained. Methods of designing such solutions are the subject of the emerging discipline of vision engineering. With projected advances in sensor and computing technologies, the domains of applicability and ranges of problems that can be solved will gradually expand.

1.3 YOLO

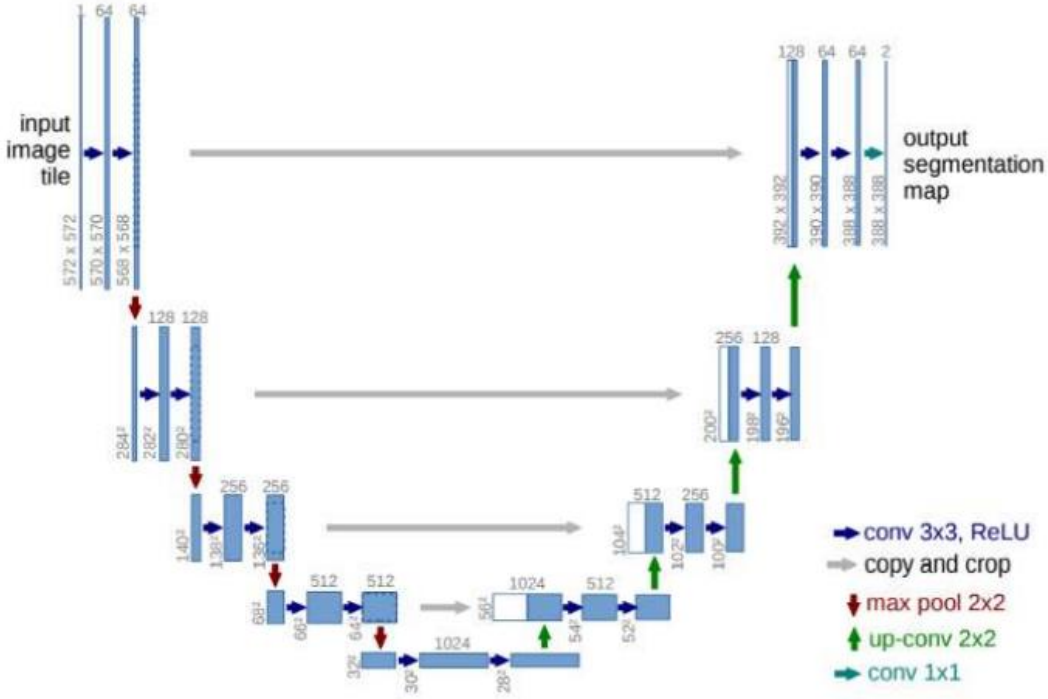


Figure 1.1: Field of View

YOLO stands for you only look once. Previously, the popular method for object detection was to reuse classifiers to classify local regions of an image and use a sliding window approach to check if each region of an image has an object. YOLO shifted the paradigm by proposing object detection as a regression problem, where they only use a single network for the entire pipeline and process the whole image at once rather than in regions. YOLO divides the input image into an SxS grid. And for each grid predicts if the center of an object is present within the grid. If the center of the object is in the grid, the grid will then predict a bounding box with 5 values, x, y, w, h, c. (x, y) are the coordinates of the center of 7 the object relative to the grid, (w, h) is the width and height of the object relative to the whole image and (c) is the class of the object. YOLO is extremely fast. Since the paper frames detection as a regression problem, it doesn't need a complex pipeline. It reasons globally about the image when making predictions. Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their

3

appearance. YOLO learns generalizable representations of objects. Since YOLO is highly generalizable it is less likely to break down when applied to new domains or unexpected inputs.

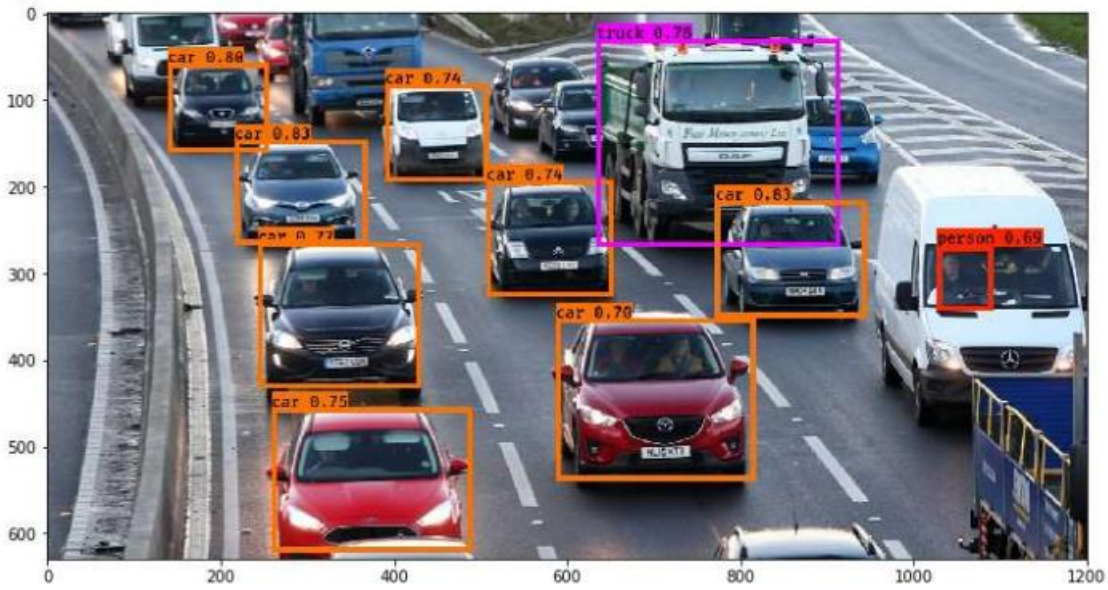


Figure 1.2: YOLO SEGMENTATION

4

1.4 Optical Character Recognition

A process called Optical Character Recognition (OCR) converts printed texts into digital image files. It is a digital copier that uses automation to convert scanned documents into editable, shareable PDFs that are machine-readable. OCR may be seen in action when you use your computer to scan a receipt. The scan is then saved as a picture on your computer. The words in the image cannot be searched, edited, or counted, but you may use OCR to convert the image to a text document with the content stored as text. OCR software can extract data from scanned documents, camera photos, and image-only PDFs. It makes static material editable and does away with the necessity for human data entry.[1]

Widely used as a form of data entry from printed paper data records – whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printouts of static-data, or any suitable documentation – it is a common method of digitizing printed texts so that they can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as cognitive computing, machine translation, (extracted) text-to-speech, key data and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision.

Early versions needed to be trained with images of each character, and worked on one font at a time. Advanced systems capable of producing a high degree of recognition accuracy for most fonts are now common, and with support for a variety of digital image file format inputs.[2] Some systems are capable of reproducing formatted output that closely approximates the original page including images, columns, and other non-textual components.

5

1.5 Neural Networks in Computer Vision

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria. Neural networks contains layers of interconnected nodes. Each node is a perceptron and is similar to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear. In a multi-layered perceptron (MLP), perceptron's are arranged in interconnected layers. The input layer collects input patterns. The output layer has classifications or output signals to which input patterns may map.

1.6 Convolutional Neural Network (CNN)

CNN is a type of deep learning model for processing data that has a grid pattern, such as images, which is inspired by the organization of animal visual cortex and designed to learn spatial hierarchies of features automatically and adaptively, from low- to high-level patterns. CNN is a mathematical construct that is typically composed of three types of layers (or building blocks): convolution, pooling, and fully connected layers [3]. The first two, convolution and pooling layers, perform feature extraction, whereas the third, a fully connected layer, maps the extracted features into final output, such as classification. A convolution layer plays a key role in CNN, which is composed of a stack of mathematical operations, such as convolution, a specialized type of linear operation. In digital images, pixel values are stored in a two-dimensional (2D) grid, i.e., an array of numbers (Fig. 1.3.1), and a small grid of parameters called kernel, an optimizable feature extractor, is applied at each image position, which makes CNNs highly efficient for image processing, since a feature may occur anywhere in the image. As one layer feeds its output into the next layer, extracted features can hierarchically and progressively become more complex. The process of optimizing parameters such as kernels is called training, which is performed so as to minimize the difference between outputs and ground truth labels through an optimization algorithm called backpropagation and gradient descent, among others.

6

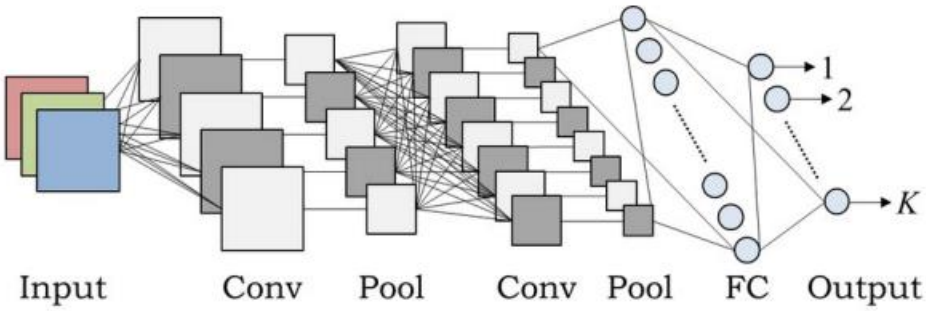


Figure 1.3 : CNN Architecture

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

7

CHAPTER 2

LITERATURE SURVEY

2.1 Vehicle Tracking

Vangala P. Kumar, K. Rajesh, Motike Ganesh, India Educators' Conference (TIIEC), 2014 Texas Instruments: This research paper consists of two 3D-analogoutput accelerometers, signal conditioning circuit, Digital Signal Controller and transmitter section. Signal conditioning circuit fills the need of filtering and amplification of these signals taken from the accelerometer. On-chip analog-to-digital converter on DSC (TMS320F28027) converts conditioned analog data from signal conditioning circuit to digital data. These digital signals are further processed in DSC to perform cross correlation operation with signals which are already stored in the flash memory. The flash memory is firstly loaded with the signal patterns generated when the vehicle is rashly driven. When the correlated is not as much as specific threshold value, the message is transmitted to the concerned authority about the vehicle being over speed, or rashly driven.

T. Shyam Ramanath, A. Sudharsan, U. Felix Udhayaraj, Mechanical and Electrical Technology (ICMET), 2016 4th International Conference: In this research paper, in order to monitor the driver conduct advanced cell sensors are utilized. The rash driving is recognition utilizing increasing speed and orientation sensors. Number of mischances caused by weakness of sharpness in vehicle drivers represents a genuine peril to individuals, not just the jumpers who are driving their vehicle yet in addition to the overall population represent a genuine danger because of hazardous driving. Driving style can typically obliging and perceive driving occasions that diminish into classifications can support in medium security frameworks. Here an electronic unit with two sensors, to be specified GPS and accelerometer, and hypothetical models, which incorporate both increasing speed and speed information, are utilized to distinguish and report whimsical driving of a minibus taxis. As of smartphone’s affordable low price, a set of embedded sensors and small sizes, these devices are gaining popularity for building driver intelligent assistance systems at a large scale. Firstly, it should be highlighted that there is a certain

scientific and technical groundwork in the area of smartphone camerabased solutions focused on recognizing dangerous driving behavior in the real-time scenarios. There is a certain set of mobile applications utilizing smartphone's front-facing camera and applying data image processing algorithms to provide driving safety while driving. The increasing popularity of smartphone camera-based approaches is due to its ability to measure contact free. On the other hand, the study demonstrates that detection of blinks can be affected by the driver state, level of automation, the measurement frequency, and the algorithms used. It proposes the evaluation of the performance of an electrooculogram- and camera-based blink detection algorithms in both manually and conditionally automated driving conditions under various constraints. During the experiment, the participants were requested to rate their subjective drowsiness level with the Karolinska Sleepiness Scale every 15 minutes. The main goal of the existing smartphone-based research studies and solutions is to early warn driver about recognized dangerous state and eliminate the risk of drowsy or distracted driving. Let's consider driver's drowsiness determination related studies. This study demonstrates a monitoring system developed to detect and alert the vehicle driver about the presence of the drowsiness state. To recognize whether the driver is drowsy, the visual indicators that reflect the driver's condition, comprising the state of the eyes, the head pose and the yawning, were assessed. The number of tests were proposed to assess the driver's state, including yawning, front nodding, blink detection, etc. Although the proposed recognition method gets 93% of total drowsiness detections, its unclear which dataset was utilized to evaluate the system and whether the detection method was tested under different light conditions. In this study the Android-based smartphone was utilized to assess the driver's state. Another study presents the developed smartphone mobile application "Drowsy Driver Scleral-Area" related to driver's drowsiness detection. The proposed mobile application includes a Haar cascade classifier, provided by the computer vision framework OpenCV for driver's face and eyes detection; and a module written in Java and responsible for image processing and alerting driver about potential hazards while driving. The developed application is configured to detect prolonged eyelid closure exceeding three seconds indicating drowsiness state. Also, it was tested on a static photo sequence, person in a laboratory and in a vehicle. The paper highlights that the pixel density analysis method was used that eliminates the need to manually count pixels and determine a threshold for drowsiness. It involves the calculation of the ratio of white pixels to

9

maximum white pixels (corresponding to full eye opening) in the region of detection. The authors of the study consider that additional tests need to be conducted under more dynamic motion and reduced light conditions. One more paper proposes the three-stage drowsiness detection framework for vehicle drivers, developed for Android based smartphones. The first stage uses the PERCLOS obtained through images captured by the front-facing camera with an eye state classification method. The system uses near infrared lighting for illuminating the face of the driver while night-driving. The next step uses the voiced to the unvoiced ratio calculated based on the speech data taken from the built-in smartphone microphone, in the event PERCLOS crosses the threshold. A final stage is used as a touch response within a specified time to declare the driver as drowsy and subsequently alert with an audible alarm. According to the received results of the study the developed framework for smartphones demonstrates 93% drowsiness state classification. The final measurement indicators used in this study include PERCLOS, the voiced-unvoiced ratio and a reaction test response of the driver on the smartphone screen.

10

2.2 Object Detection

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class in digital images and videos. Machine learning can be used to detect and classify objects in images and videos. Vehicle detection, also known as computer vision object recognition, is basically the scientific methods and ways of how machines see rather than human eyes. Vehicle detection is one of the widely used features by companies and organizations these days. We can use computer vision to detect different types of vehicles in a video or real-time via a camera. Vehicle detection and tracking finds its applications in traffic control, car tracking, creating parking sensors and many more. We have used the methods discussed in Dimililer et al. (2020) paper of Vehicle detection and tracking to built a system using OpenCV and Python for both images and videos that is able to detect and track the vehicles automatically. We used a dataset consisting of 8792 vehicle images and 8968 non-vehicle images from a combination of the GTI vehicle image database, and the KITTI vision benchmark suite. 5568 features were extracted for training using three feature extraction techniques: Spatial Binning, Color histogram, and Histogram of oriented gradients(HOG). We have used LinearSVC, which is one of the classifiers in SVMs, to predict vehicles in this project. Then we used stratified 10-fold cross validation to assess model performance in terms of the area under the ROC curve (AUROC). LinearSVC accuracy was around 99.37% on the test split of the dataset. Hog sub-sampling was used for implementing the sliding window approach with starting position of the window search from 350px to 656px and cells_per_step to 1. The pipeline video was created by processing each frame of the video images with hog sub-sampling and combining overlapping detections. It is done in video sequences like security cameras and CCTV surveillance feed; the objective is to track the path followed, speed of an object. The rate of real time detection can be increased by employing object tracking and running classification in few frames captured in a fixed interval of time. Object detection can run on a slow frame rates looking for objects to lock onto and once those objects are detected and locked, then object tracking, can run in faster frame speed.

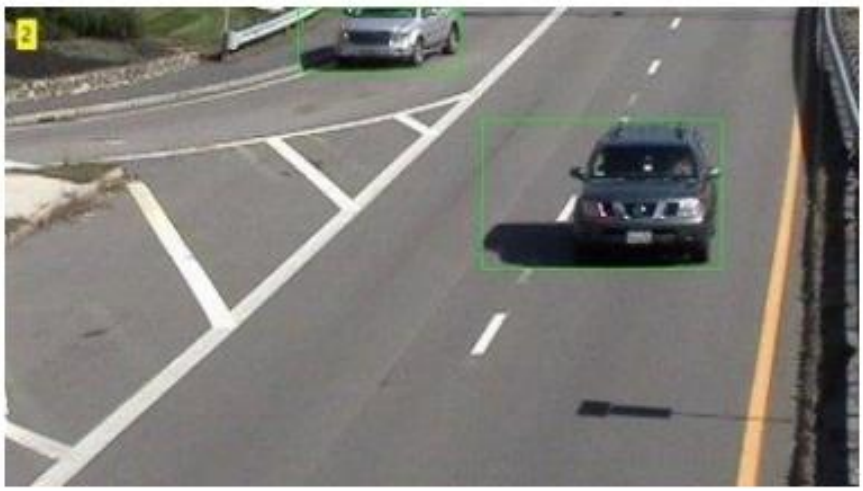


Figure 2.2: Tracking of Car

Fig. 2.2 shows the tracking of car. Two ways in which the object can be tracked in the above example are: (1)-Tracking in a sequence of detection. In this method a CCTV video sequence of a traffic which is in motion takes place. Suppose someone wants to track a car or person's movement here, he will take different images or frames at different interval of time. With the help of these images one can target the object like a car or person. Then, by checking how my object has moved in different frames of the video, I can track it. Velocity of the object can be calculated by verifying the object's displacement with the help of different frames taken at different interval of time. This method is actually a flaw where one is not tracking but detecting the object at different intervals of time. Improved method is "detection with dynamics". In this method estimation of car's trajectory or movement takes place. By checking it's position at a particular time 't' and estimating its position at another time interval let's say 't+10'. From this actual image of car at 't+10' time can be proposed with the help of estimation

2.3 Object Tracking

The detection and counting of vehicles have significant role in an intelligent transportation system, especially for traffic management. Traffic issues has become a biggest problem where city planners facing for years. The detection of moving vehicles more accurately, several computer vision techniques, vehicle counting is done by using a virtual detection zone. Traffic analysis will calculate for the number of vehicles in an area per some arbitrary period and classify the vehicles. But the moving vehicles and their detection, tracking, and counting are very critical for flow of traffic in monitoring, planning, and controlling. By analyzing the recorded video of traffic flow sequence from a camera, a video- based solution technology is applied in combination with virtual detector and blob tracking technologies and YOLO are necessary. With this technology we applied Open CV for real time video applications. Such methods help us to detect, track, count and classify the moving vehicles.

The main goal of Vehicle detection and counting in traffic video projects is to develop and implement methodology for automatic detection and counting of moving vehicles on highways. Intelligent visual surveillance (IVS) for road vehicles is a key feature to intelligent transportation systems (ITS). Segmentation with initial background subtraction method is used to detect and count vehicles and same method using morphological operators to determine salient regions in surveillance video sequential frames. Edges are being counted to show how many areas of particular size which have particular vehicles like car locate the points and count the vehicles in the traffic domain and monitoring over it on highways.

The necessary to manage and monitor the traffic on roads has grown in last few years as there is increase in the number of vehicles, road lanes and networks and most importantly the size of vehicles. It is better to understand and analyze traffic flow, an increasing reliance in surveillance of traffic is in the requirement of vehicle detection in bett er way at a wide -area. Computer vision which includes some significant practical applications, such as traffic analysis and security faces challenging problem in automatic vehicle detection and counting. YOLO for object detection, OPENCV, Coco Dataset, SORT.PY are the methodology used to detect and count the vehicles. The proposed system involves a Convolutional neural

network which is a type of artificial neural network.

The system is used to detect, recognize, and track the vehicles in the sequence of video frames, after that classification of vehicles is done which are detected in accordance with their size in different classes. YOLO with convolutional neural network (CNN) for detecting moving objects in real-time and object tracking with OpenCV. The input video is served to process, the frames from the video, frames are then converted into the blobs, the pixels are then read and the object detection model is called for each blob the trained model detects the object and creates the bonding boxes for each detected object. Once the tracking of vehicles is completed, the detection line method is used to achieve vehicle counting. The proposed system can be used for traffic surveillance, with cameras on traffic lights further improvements can be done on the proposed system to make it real time detection and tracking, Our Approach has developed an insight to use object detection models for tracking and counting further speed and accuracy of the system can also be improved with improvements in object detection models.

CHAPTER 3
SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In the security world, there are two types of cameras, the first one is Analog surveillance camera and another one is Network surveillance camera or IP camera. These cameras send video signals to DVR or NVR. If it is a DVR, it converts the analog signal to digital and stores it in the DVR. If you are using NVR, it just stores the video in NVR.

The main component of a surveillance system is the DVR/NVR, which records live video feed and gives the ability to play recorded video anytime when you want. DVR/NVR uses a hard drive to store the videos coming from cameras. They are connected to a Monitor to view the video feed of each camera. DVR/NVR can also be connected to a network or modem so as to make the video feed accessible over the internet. These systems have all the basic features of recording, storing and viewing the video stream from the cameras.

Fig. 3.1 Existing System

3.1.1 Problems with Existing System

3.1.1.1 Lack of object detection

Surveillance systems are not equipped with any object recognition software which can help keep an eye on goods, personal items or people in real-time.

3.1.1.2 Adaptation to system changes

The flexibility to extend or update the necessary services is not provided in the current system.
These systems are rigid and only give access to basic things or services provided at the time of buying. The systems that are currently used resist updates or changes to given services.

3.1.1.3 Absence of object tracking service

The existing surveillance systems cannot perform the task of taking an initial set of object detections, creating a unique ID for each of the initial detections, and then tracking each of the objects as they move around frames in a video, maintaining the ID assignment.

3.1.1.4 Movement tracing for existing entities

Existing systems fail to record or capture movement of the objects that are present in the camera frames. The inability of the surveillance system results in failure to provide a path taken by any particular object.

3.2 Proposed System

The aim is to detect and identify rash drivers and factors leading to traffic using Machine Learning algorithms and Computer Vision.

We have utilized Python and OpenCV for object detection and tracking. further, we would be implementing Yolo algorithms for object detection and byte-track algorithms for object tracking.

Features of Proposed System

- Object detection
- Object tracking
- Traces the path of the cars to detect patterns of reckless driving.

3.2.1 Advantages

 Unique Object detection using car attributes.
 Tracking of each car using their unique number plate.
 Getting the path taken by each car.

3.2.2 Limitations

The main problem is that there is a lot of computation required to process the frames which involve the detection of all the objects present in the frame, storing each of these objects and tracking these objects in the next frames with the knowledge of previous frames. Due to heavy computation the FPS (frames per second) of the output stream may decrease to an extent. With better processors this issue can be resolved.

3.3 Feasibility Study

Feasibility study is the test of a system proposal according to its workability, impact on the organization, ability to meet user needs, and effective use of resources. It focuses on the evaluation of existing systems and procedures analysis of alternative candidate system cost estimates. Feasibility analysis was done to determine whether the system would be feasible.

The development of a computer-based system or a product is more likely plagued by resources and delivery dates. Feasibility study helps the analyst to decide whether or not to proceed, amend, postpone or cancel the project, particularly important when the project is large, complex and costly. Once the analysis of the user requirement is complete, the system has to check for the compatibility and feasibility of the software package that is aimed at. An important outcome of the preliminary investigation is the determination that the system requested is feasible.

3.3.1 Types of Feasibility

➢ Technical Feasibility

➢ Operational Feasibility

➢ Economical Feasibility

3.3.1.1 Technical Feasibility

The application can be developed with the current equipment and has the technical capacity to hold the data required by the system.

 This technology supports the modern trends of technology.

 Easily accessible, more secure technologies.

Technical feasibility on the existing system and to what extent it can support the proposed addition.
We can add new modules easily without affecting the Core Program. Most of the parts are running in the server using the concept of stored procedures.

3.3.1.2 Operational Feasibility

In Operational Feasibility degree of providing service to requirements is analysed along
with how much easy product will be to operate and maintenance after deployment. Along with these other operational scopes are determining usability of product, determining suggested solution by software development team is acceptable or not etc.

3.3.1.3 Economical Feasibility

If benefits outweigh costs, then the decision is made to design and implement the system. An entrepreneur must accurately weigh the cost versus benefits before taking an action. This system is more economically feasible and budget friendly. So, it is economically a good project.

3.4 Effort and Cost Estimation

COCOMO (Constructive Cost Model) is a regression model based on LOC, i.e. number of Lines of Code. It is a procedural cost estimate model for software projects and often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time and quality. It was proposed by Barry Boehm in 1970 and is based on the study of 63 projects, which make it one of the best documented models.
The key parameters which define the quality of any software products, which are also an outcome of the COCOMO are primarily Effort & Schedule:

 Effort: Amount of labour that will be required to complete a task. It is measured in person-months units.

 Schedule: Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put. It is measured in the units of time such as weeks, months.

Different models of COCOMO have been proposed to predict the cost estimation at different levels, based on the amount of accuracy and correctness required. All of these models can be applied to a variety of projects, whose characteristics determine the value of constant to be used in subsequent calculations. These characteristics pertaining to different system types are mentioned below.

Boehm's definition of organic, semidetached, and embedded systems:

1. Organic – A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also, the team members have a nominal experience regarding the problem.

2. Semi-detached – A software project is said to be a Semi-detached type if the vital characteristics such as team-size, experience, knowledge of the various programming environment lie in between that of organic and embedded. The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience and better guidance and creativity. Eg: Compilers or different Embedded Systems can be considered of Semi-Detached type.

3. Embedded – A software project with requiring the highest level of complexity, creativity, and experience requirement fall under this category. Such software requires a larger team size than the other two models and the developers need to be sufficiently experienced and creative to develop such complex models.

3.4.1 Basic COCOMO Model

E=a*(KLOC)^b

D=c*€^d

Where E=Effort applied in person-months, D is the development time in chronological months and KLOC is estimated number of delivered lines of code for project expressed in per thousand. The coefficients a, b, c, d come from the given table considering our project to be organic.

Software project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Table 3.4.1: Coefficients for Basic COCOMO model

The KLOC for our project are as follows:

E=2.4*(0.493)^1.05

=1.14 person-months

D=2.5*(1.14)^0.38

= 4 months

The number of people required to complete this project were 3.

3.5 Software Requirements and Specifications

3.5.1 Purpose

In our proposed solution we focus on developing a system that dynamically tracks movements in each area using camera and triggering necessary actions, this solution implements object tracking in computer vision over a network of cameras generating geofences with occupancy map allowing the system to track movements of individuals, traffic and objects across any area with attention to features such as direction of motion, velocity, physical appearance etc. This would allow us to leverage the best of multivariate data.

3.5.2 Scope

In our proposed system we have developed a solution which detects the cars and tracks the movement of these cars. The tracked cars which give us traces of irregular patterns can be monitored to check any abnormal activity or violation of rules. It can be used by traffic authorities to detect rash driving and can be fined accordingly.

3.5.3 Hardware Requirements

- Processor: Intel® Core TM i5; Recommended i7 or more
- Speed: Minimum 1 GHz; Recommended 2GHz or more
- Hard Disk: Minimum 32 GB; Recommended 64 GB or more
- Memory (RAM): Minimum 1 GB; Recommended 8 GB or above

3.5.4 Software Requirements

- Operating System: Windows XP/2000/vista/7/8/10/11
- Coding Language: Python
- IDE: VS Code/Google Collab/PyCharm
- Development Kit: Python 3.10.4
- Version Control: GitHub

3.6 Technologies Used

a. Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on.

The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

b. Visual Studio Code

Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging. First and foremost, it is an editor that gets out of your way. The delightfully frictionless edit-build-debug cycle means less time fiddling with your environment, and more time executing on your ideas. Visual Studio Code features a lightning-fast source code editor, perfect for day-to-day use. With support for hundreds of languages, VS Code helps you be instantly productive with syntax highlighting, bracket-matching, auto-indentation, box-selection, snippets, and more. Intuitive keyboard shortcuts, easy customization and community contributed keyboard shortcut mappings let you navigate your code with ease. For serious coding, you will often benefit from tools with more code understanding than just blocks of text. Visual Studio Code includes built-in support for IntelliSense code completion, rich semantic code understanding and navigation, and code

refactoring. And when the coding gets tough, the tough gets debugging. Debugging is often the one feature that developers miss most in a leaner coding experience, so we made it happen.

Visual Studio Code includes an interactive debugger, so you can step through source code, inspect variables, view call stacks, and execute commands in the console.

c. NumPy

NumPy is a library in Python that allows for efficient numerical computing in Python. This library is highly optimized to do mathematical tasks. In the project workflow NumPy is heavily used in data pre-processing and preparation One of the main features about NumPy is, it is highly efficient n-dimensional array (ndarray). Compared to a list in Python a NumPy array can be n-dimensions and has more features associated with the ndarray. NumPy can also perform more efficient mathematical operations compared to the math library in Python.

d. Matplotlib

Matplotlib is a Python plotting library that allows programmers to create a wide variety of graphs and visualizations with ease of use. The great feature about Matplotlib is that it integrates very well with Jupyter Notebook and creating visualizations is simplified. Matplotlib also works very well with pandas and NumPy.

e. OpenCV

OpenCV (Open-Source Computer Vision) is a well-established computer vision library which is written in C/C++ and has been abstracted to interface with C++, Python and Java.

This is a powerful tool when working with images and has a myriad of tools regarding image data manipulation, feature extraction etc.

f. TensorFlow

Tensorflow is an open-source deep learning library by Google. It was originally developed by Google's engineers who were working on Google Brain and has been used for research on machine learning and deep learning. Tensorflow at its core is about computations of multidimensional arrays called tensors but what makes Tensorflow great is its ability to be flexible to deploy computations on different devices such as CPU's and GPU's.

4. SYSTEM DESIGN

4.1 Block Diagram

A block diagram is a visual representation of a system that uses simple, labelled blocks that represent single or multiple items, entities, or concepts, connected by lines to show relationships between them. Block diagrams are used heavily in engineering and design of diagrams for electronics, hardware, software, and processes. Most commonly, they represent concepts and systems in a higher level, less detailed overview. The diagrams are useful for troubleshooting technical issues.

Block diagrams are a generalized representation of a concept and are not intended to display complete information in regards to design or manufacture. Unlike schematics, blueprints and layout diagrams, block diagrams do not portray the necessary detail for physical construction. Block diagrams are made simple so as not to cloud concepts.

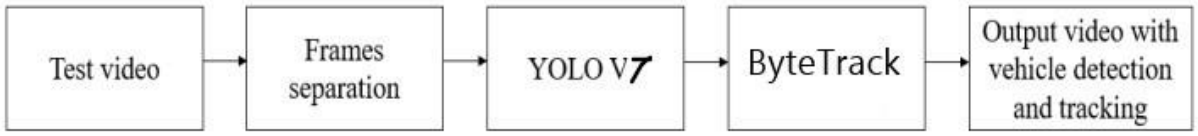


Figure 4.1: Block Diagram

As the vehicle passes the frame counter is incremented to keep the count. All these will be accessed by a web app so that authorities can monitor the vehicles and take appropriate decisions. For the above system, there is a need for a camera to be placed at the entry part, and the whole detection and counting process will be automatically done by the system and algorithm. For tracking the vehicles which are already detected using YOLO there is the requirement of a robust tracking algorithm like ByteTrack. All these will be accessed by a web app so that authorities can monitor the vehicles and take appropriate decisions. For the above system, there is a need for a camera to be placed at the entry part, and the whole detection and counting process will be automatically done by the system and algorithm.

4.2 UML Diagrams

4.2.1 Activity Diagram

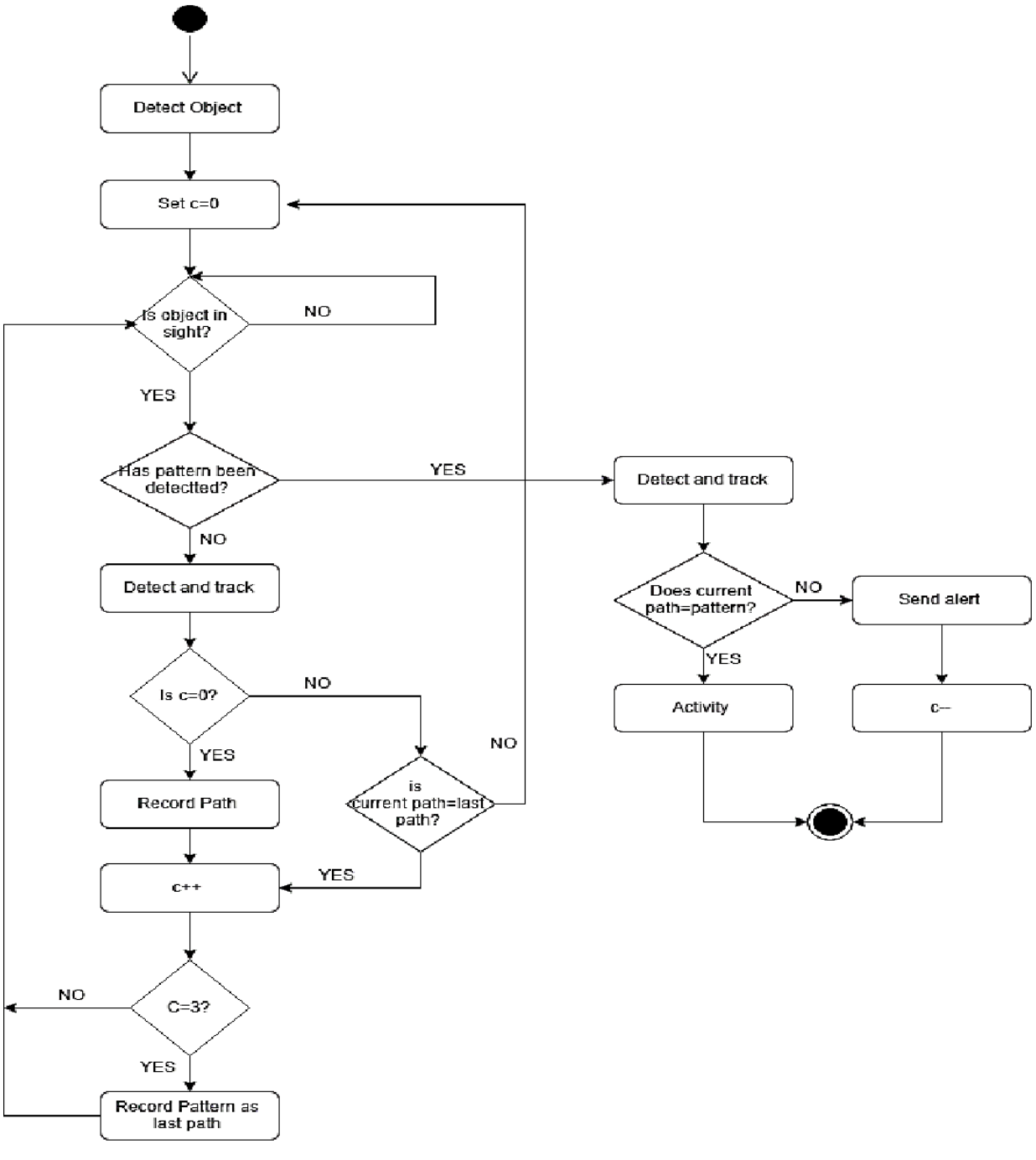


Figure 4.2.1: Activity Diagram

Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some
operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another,
in particular, use cases where activities may overlap and require coordination. It is also suitable for modelling how a collection of use cases coordinates to represent business workflows.

4.2.2 Sequence Diagram

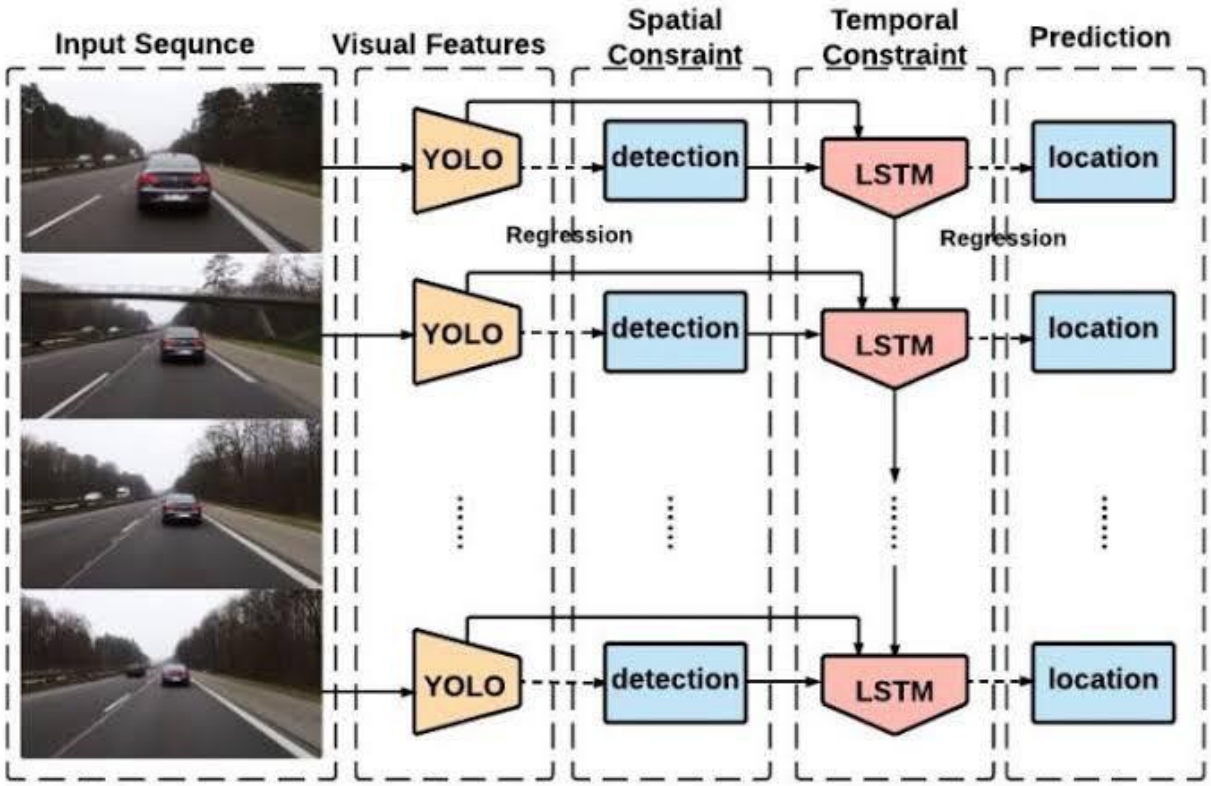


Figure 4.2.2: Sequence Diagram

The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time. In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching. The purpose of the class diagram can be summarized as –

• To model high-level interaction among active objects within a system.

• To model interaction among objects inside a collaboration realizing a use case.

• It either model's generic interactions or some certain instances of interaction.

5. IMPLEMENTATION

5.1 Data Acquisition and processing

5.1.1 Data Collection

It is the process of gathering data relevant to your AI project's goals and objectives. You eventually obtain a dataset, which is essentially your collection of data, all set to be trained and fed into an ML model. Data collection is the first and most fundamental step in the machine learning pipeline. It's part of the complex data processing phase within an ML lifecycle.

A Dataset contains a lot of separate pieces of data but can be used to train an algorithm with the goal of finding predictable patterns inside the whole dataset. The common types of data include:

- Text data
- Image data
- Audio data
- Video data
- Numeric data

The data is usually first labeled/annotated in order for the algorithm to understand what the outcome needs to be.

Our dataset entirely consists of Image data. We have acquired our data from various sources online and have covered a large variety of classes due to which we were able to obtain ~80% precision.

5.1.2 Data Pre-processing

As our data has been collected from multiple resources we starting by assuming that the entire dataset is flawed and thus needed formatting. The images are then pre-processed i.e. dataset is re-appropriated to fit our specific goals. All unnecessary images that don't meet our requirements are discarded.

The entire process is continued until we are left with high quality data which is capable of further use and training.

5.1.3 Data Splitting

A dataset is used not only for training purposes. A single training dataset that has already been processed is usually split into several parts, which is needed to check how well the training of the model went. For this purpose, a testing dataset is usually separated from the data. Next, a validation dataset, while not strictly crucial, is quite helpful to avoid training your algorithm on the same type of data and making biased predictions.

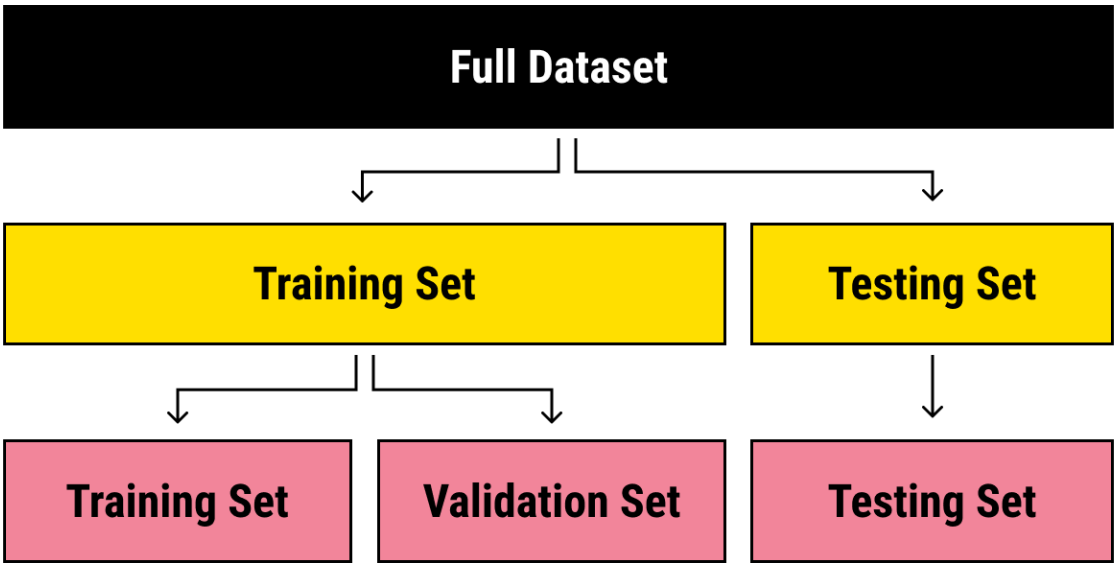


Figure 5.1.3(a): Splitting of a dataset

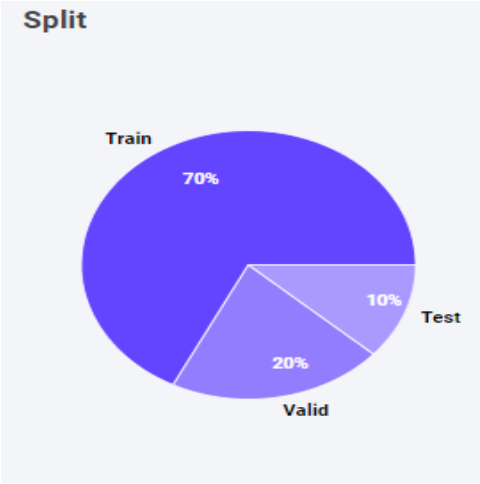


Figure 5.1.3(b): Split ratio of our dataset

5.1.4 Data Labelling

Data preparation for machine learning doesn’t stop at the preprocessing stage. There’s one more task we have to accomplish, which is annotating data. Building and training a versatile and high-performing ML algorithm requires a significant amount of labeled data. Depending on the task at hand and the individual dataset in question, labels can vary for each specific dataset.

After We've ensured that our data is clean and relevant, we also need to make sure it's understandable for a computer to process. Machines do not understand the data the same way as we humans do i.e. they aren't able to assign the same meaning to the images or words as we.

Data annotation involves a lot of time and manual labor. We need to label each and every image in our dataset and specify the classes that we need our model to identify. In our dataset we have specified a few basic classes of vehicles such as Cars, Trucks, Bikes etc.

Figure 5.1.4: An example label from our vehicle’s dataset.

5.2 Model Training

Training a model simply means learning (determining) good values for all the weights and the bias from labeled examples. In supervised learning, a machine learning algorithm builds a model by examining many examples and attempting to find a model that minimizes loss.

Loss is the penalty for a bad prediction. That is, **loss** is a number indicating how bad the model's prediction was on a single example. If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater. The goal of training a model is to find a set of weights and biases that have *low* loss, on average, across all examples.

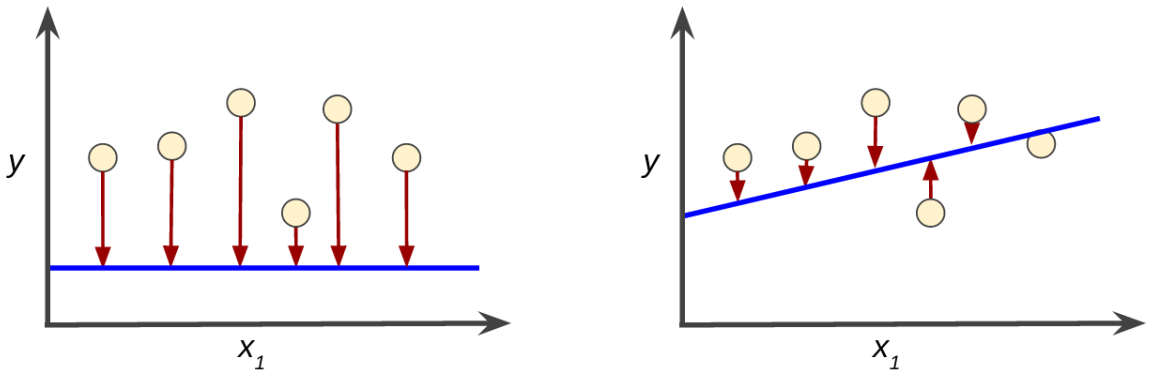


Figure 5.2(a): shows a high loss model on the left and a low loss model on the right.

We chose Yolov7 as our algorithm of choice for model training. YOLOv7 is the fastest and most accurate real-time object detection model for computer vision tasks.

A good fit is the goal of the learning algorithm and exists between an overfit and underfit model. A good fit is identified by a curve that decreases to a point of stability with a minimal gap between the two final loss values.

The learning curves that were obtained during the training of our model are as follows:

Figure 5.2(b): Learning curve of our model

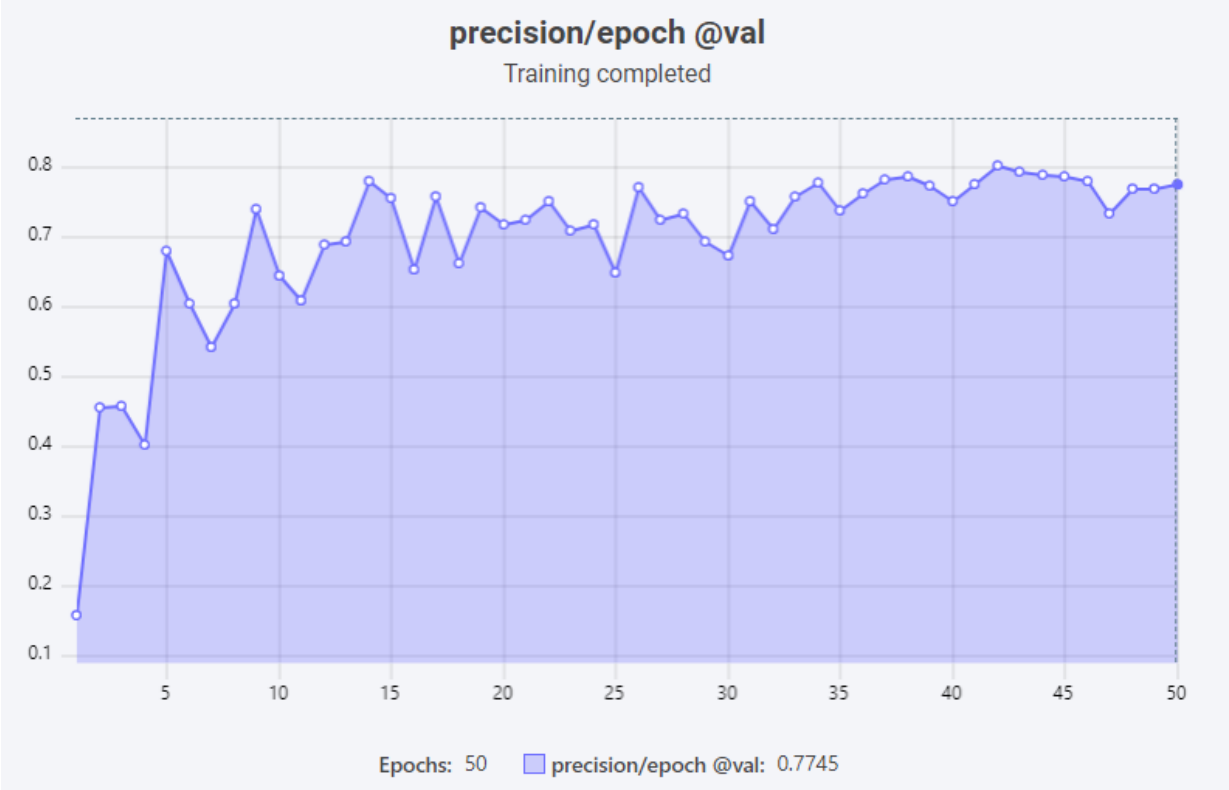


Figure 5.2(c): Precision obtained while training

5.3 Object detection and tracking

5.3.1 Vehicle Detection Model

The content of this section is the process of building vehicle detection models and using them to classify and localize vehicles for vehicle tracking and statistics. It is notable that the objective of vehicle detection model is to generate bounding boxes accurately and efficiently with confidence value, which is the input for vehicle tracking and statistics.

YOLO is a Convolutional Neural Network object detection system, that handles object detection as one regression problem, from image pixels to bounding boxes with their class probabilities. Its performance is much better than other traditional methods of object detection, since it trains directly on full images. YOLO is formed of 27 CNN layers, with 24 convolutional layers, two fully connected layers, and a final detection layer.

The YOLO network breaks the picture into a defined number of grids. Each grid is accountable for estimating objects within the grid whose central points are. Then after several years the YOLO framework has been developed it algorithm to version 7 which improve speed and accuracy of object detection. YOLO V7 extracts the residual network part of the future entreats each region in the entire future map equally considering that each region contributes the same to the final detection however in real life scenes complex and rich contextual information often exist around the object to be detected in the image and each region in the feature map is treated equally resulting in a lack of network feature expression ability inaccurate bounding box position poor robustness and other problems. To solve these problems a channel attention mechanism module is introduced into the YOLO V7 object detection algorithm.

The advantages of the YOLOv7 detection model are the speed of detection and the ability to recognize, in order to make it accurately locating small objects. YOLOv7 surpasses all previous object detectors in terms of both speed and accuracy, ranging from 5 FPS to as much as 160 FPS. The YOLO v7 algorithm achieves the highest accuracy among all other real-time object detection models.

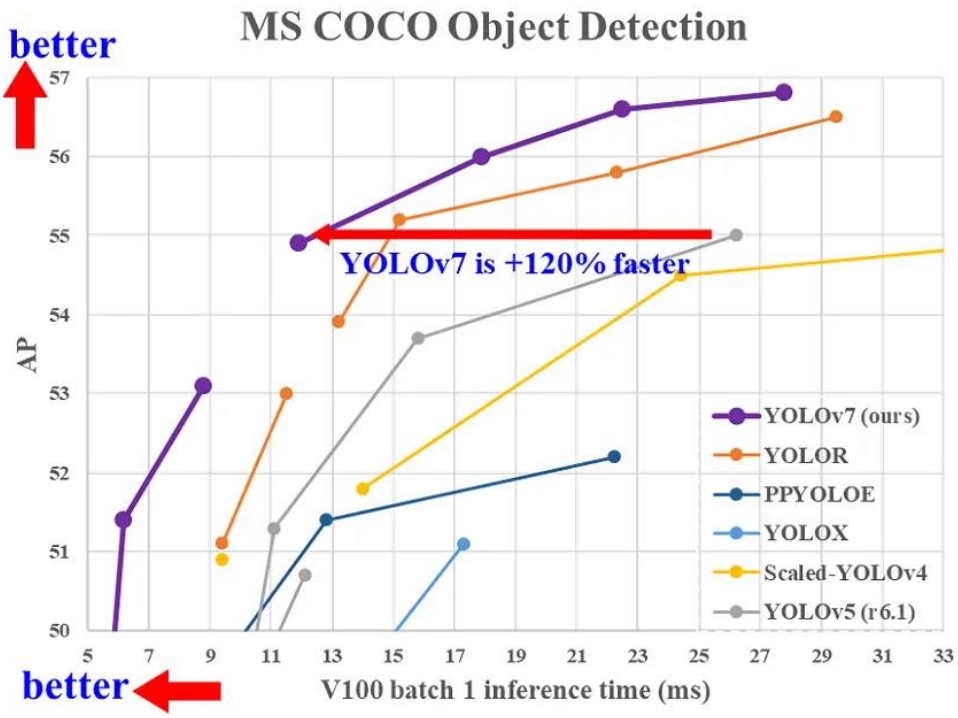


Fig. 5.3.1 YOLOv7 statistics compared to other YOLO algorithms.

A test video is taken to apply the algorithms and split into different frames. The spitted frames are applied to YOLOV7 to detect the vehicles and to track the vehicles, frames are applied to ByteTrack. Finally, the output video is generated with vehicle detection and tracking.

5.3.2 Vehicle Tracking using ByteTrack

The goal of Multi-Object Tracking(MOT) is to draw the bounding boxes around objects by detecting and identifying them in a video and then maintaining their trajectories with high accuracy. MOT takes a single continuous video as an input and splits it into discrete frames at a specific frame rate. The output of the MOT is:

- Detection: what objects are present in each frame
- Localization: where objects are in each frame
- Association: whether objects in different frames belong to the same or different objects

Most MOT methods retain only the high score detection boxes above a certain threshold, i.e., 0.5, and use these high score detection boxes as the input for data association.

MOT is a challenging task in complex and crowded conditions as some tracklets get unmatched because they do not match to an appropriate high score detection box. Some challenges with existing MOT techniques are:

- Occlusions can cause missed object detection.
- Motion Blur can cause missed trajectories with objects frequently entering and leaving the frame generating fragmented trajectories and ID switching.
- Size changes can cause missed trajectories and ID switching

BYTE is a simple and effective association method for MOT. This MOT technique is named BYTE as it considers each detection box as a basic unit of the tracklist like a byte in a computer program and the tracking method values every detection box and not just the ones with high scores.

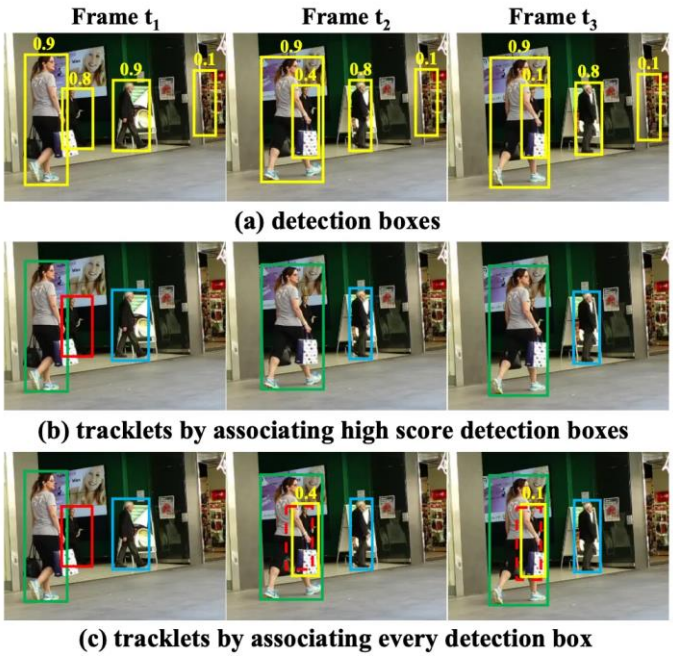


Figure 5.3.2: An example of ByteSort association.

In the figure above,

a) Displays the detection boxes identified using object detection.

b) Associate tracklets with high score detection boxes where the threshold≥ 0.5. The same box color represents the same identity.

c) Implements BYTE where all the detection boxes are valued, even the low score ones. As a result, the occluded person with low detection scores is matched correctly to the previous tracklet, and the background in the right part of the image is removed.

ByteTrack performs MOT on a video using the high-performance detector YOLOX and performs association between the detection boxes and the tracks using BYTE.

5.3.3 Code Overview

- We first start by creating an instance of YOLOv7

```
yolov7 = YOLOv7()
yolov7.load('coco.weights', classes='coco.yaml', device='cpu')
```

Here ‘best.weights’ and ‘classes.yaml’are the weights and classes obtained as output after the YOLO model has been trained.

- Reading the frames of the video

```
ret, frame = video.read()
        if ret == True:
            detections = yolov7.detect(frame, track=True)
            detected_frame = frame
```

If the frame is okay the we pass the frame to YOLOv7 for it to detect the vehicle. ‘track = true’ is used to enable tracking of ByteTrack.

- Assigning unique colors to every vehicle which is detected

```
color = (np.random.randint(0,255), np.random.randint(0,255),
np.random.randint(0,255))
```

- We have a dictionary called ‘lines = {}’ in which store all the information for drawing the trailing lines of each vehicle.
- Each vehicle has its own unique id and series of points i.e. trail it took throughout the frames of the video. We use the trail of the vehicle to calculate its direction for the generation of vectors.

- New car appears in the video

```
if 'id' in detection:
   detection_id = detection['id']

  if detection_id not in lines:
     detection['color'] = color
     lines[detection_id] = {'points':[], 'arrows':[], 'color':color}
  else:
     detection['color'] = lines[detection_id]['color']
```

A new color will be created for the vehicle and its 'id' will be added to the dictionary and it will also have a list of (points, arrows, color) created for its position across the frames.

If it is an exiting car then assign the same color which was assigned initially to its bounding box.

- To calculate the center of the bounding box:

```
lines[detection_id]['points'].append(np.array([detection['x'] +
detection['width']/2, detection['y'] + detection['height']/2],
np.int32))
points = lines[detection_id]['points']
```

We use the top left co-ordinate (x, y) to calculate the center of the bounding box. This will be used to calculate the vector of the car.

- Calculating the start and end-points of the vector trail

```
if len(points) >= 2:
   arrow_lines = lines[detection_id]['arrows']
   if len(arrow_lines) > 0:
      distance = np.linalg.norm(points[-1] - arrow_lines[-1]['end'])
      if distance >= arrow_line_length:
            start = np.rint(arrow_lines[-1]['end'] - ((arrow_lines[-
1]['end'] - points[-1])/distance)*10).astype(int)
            arrow_lines.append({'start':start, 'end':points[-1]})
```

```
   else:
       distance = 0
       arrow_lines.append({'start':points[-2], 'end':points[-1]})
```

- Drawing the vector trail

```
 for line in lines.values():
      arrow_lines = line['arrows']
     for arrow_line in arrow_lines:
           detected_frame = cv2.arrowedLine(detected_frame,
arrow_line['start'], arrow_line['end'], line['color'], 2,
line_type=cv2.LINE_AA)
```

We look through all values of 'lines' and for each value of line we generate an 'arrow line'.

- Calculating the total vehicle count

We create a set 'counted_vehicles' to keep track of the detection IDs of vehicles that have already been counted.

```
if detection_id not in counted_vehicles:
      counted_vehicles.add(detection_id)
      vehicle_count += 1
```

We iterate through all detection IDs, if any vehicle has not been counted then we add it to the 'counted_vehicles' set and then increment 'vehicle_count'.

6. TESTING

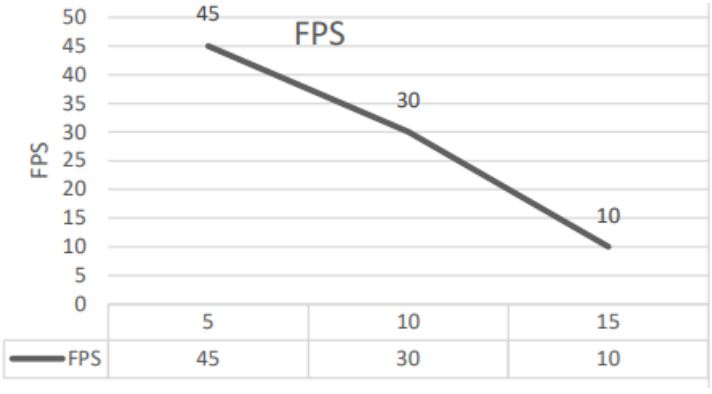
The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and /or a finished product.

6.1 Types of Testing

6.1.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests endure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Test Cases: A test case is a set of conditions or variables under which a tester will determine whether an application, software system or one of its features is working as it was originally established for it to do.



Vehicle encodings detected

Figure 6.1.1: Unit Testing on FPS

6.1.2 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner working, structure or knowledge of the module being tested. These tests can be functional or nonfunctional, though usually functional. This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories.

Test Results:

Table 6.1.2 Test Results

Test case Id	**Test Case Name**	**Test Case Desc.**	**Test Steps**		
			Input	**Output**	**Requirement verified**
01	Vehicle detection and recognition	To check whether the system correctly recognizes labelled cars.	Vehicle encodings from frames	Detected cars	yes
02	Tracking	To check if the system can track individuals correctly.	Camera Feed	Occupancy map	yes

6.1.3 Comparison of Results

The following table displays the FPS and run times for various iterations of the models on the three videos as a result of the YOLOv7 test.

Models	First Video		Second Video		Third Video	
	Inference time	FPS	Inferenc e time	FPS	Inferenc e time	FPS
Yolov4		14.8		14.8		14.8
Yolov5	7.0	142	7.61	126	8.20	122
Yolov5 large	29.1	35	28	35	28.81	35
Yolov7	51.2	18.75	49.5	20.2	50	20

Table 6.1.3: FPS Comparison of different Detection Models

The following evaluation metrics were employed in the experiment:

Tracking Accuracy (TA) - False positives, missed targets, and identity shifts are the three error factors combined in this measurement.

Tracking Precision (TP) - A summary of total tracking precision measured by the amount of ground-truth and reported position bounding boxes overlap. IDF1 Score - the proportion of accurately identified detections to the usual number of computed and ground truth detections.

Target Tracked (TT) – it is a proportion of ground-truth trajectories that a track hypothesis covers for at least 80%.

Model	TA	TP	IDF1	TT
Yolov5s	39.60	80.85	52.39	15.45%
Yolov5m	39.01	81.87	51.56	17.41%
Yolov5i	40.77	81.56	52.43	20.70%
Yolov7	40.92	82.08	53.65	20.92%

Table 6.1.3.1: Tracking performance is compared

7. SCREENSHOTS/OUTPUTS/RESULTS

After running our model on various testing videos the outputs obtained are as follows:

Test video #1 –

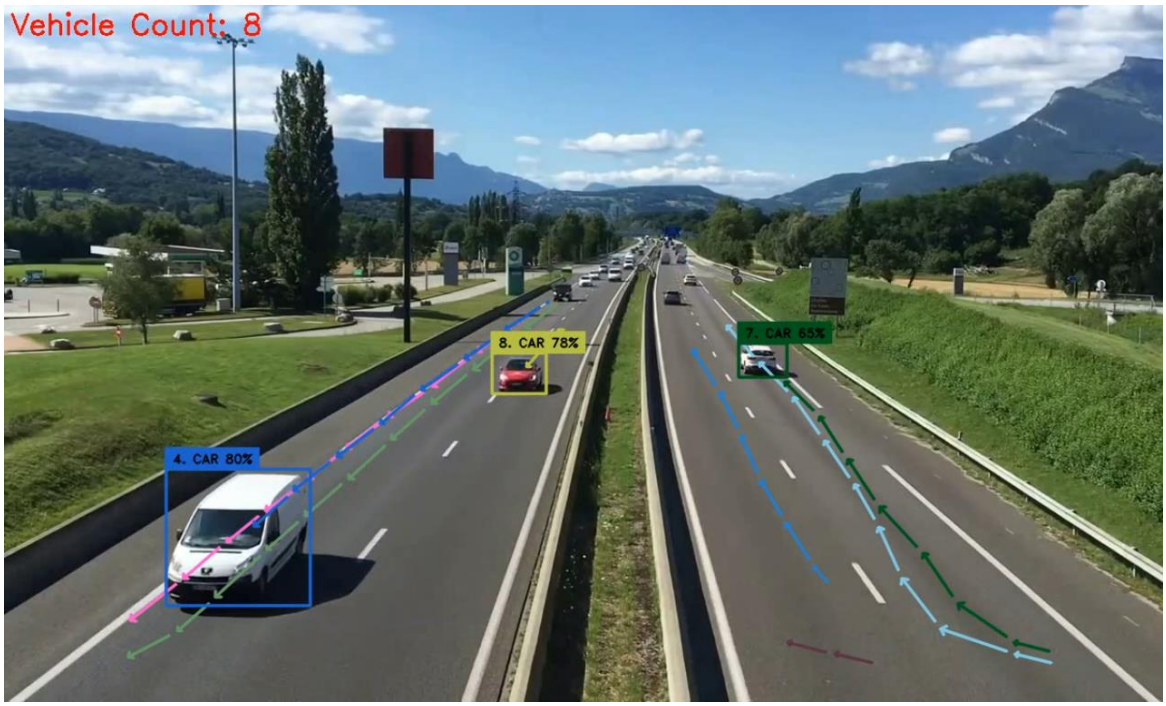


Figure 6(a): Output of Test video #1.

Test video #2 –

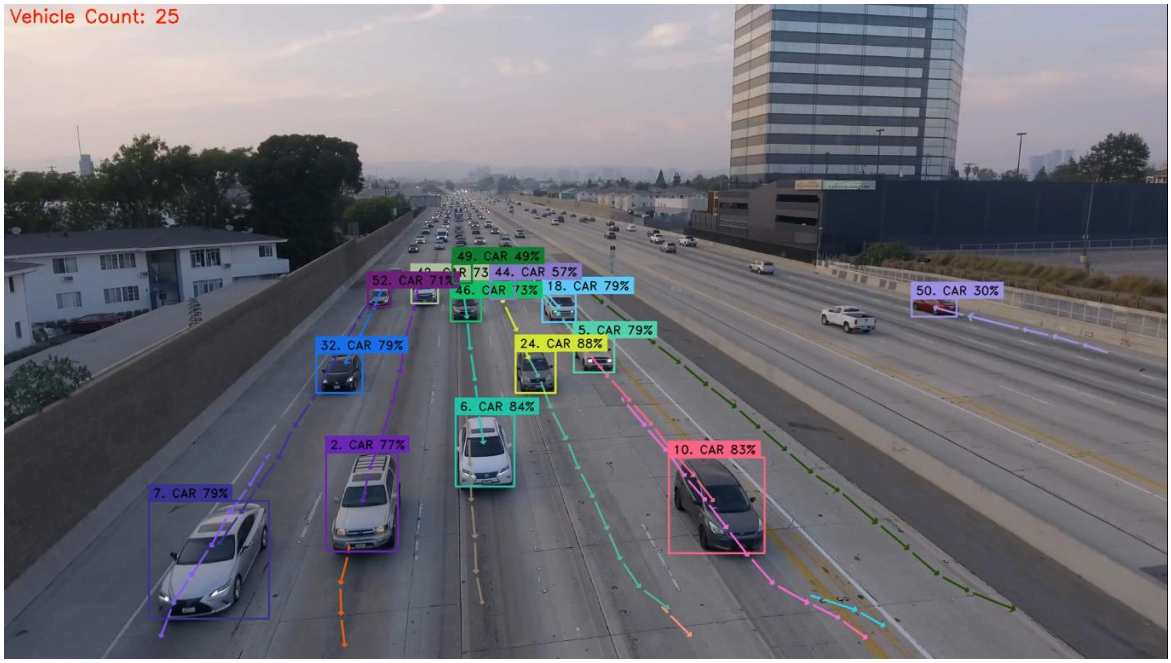


Figure 6(b): Output of Test video #2

Test video #3 –

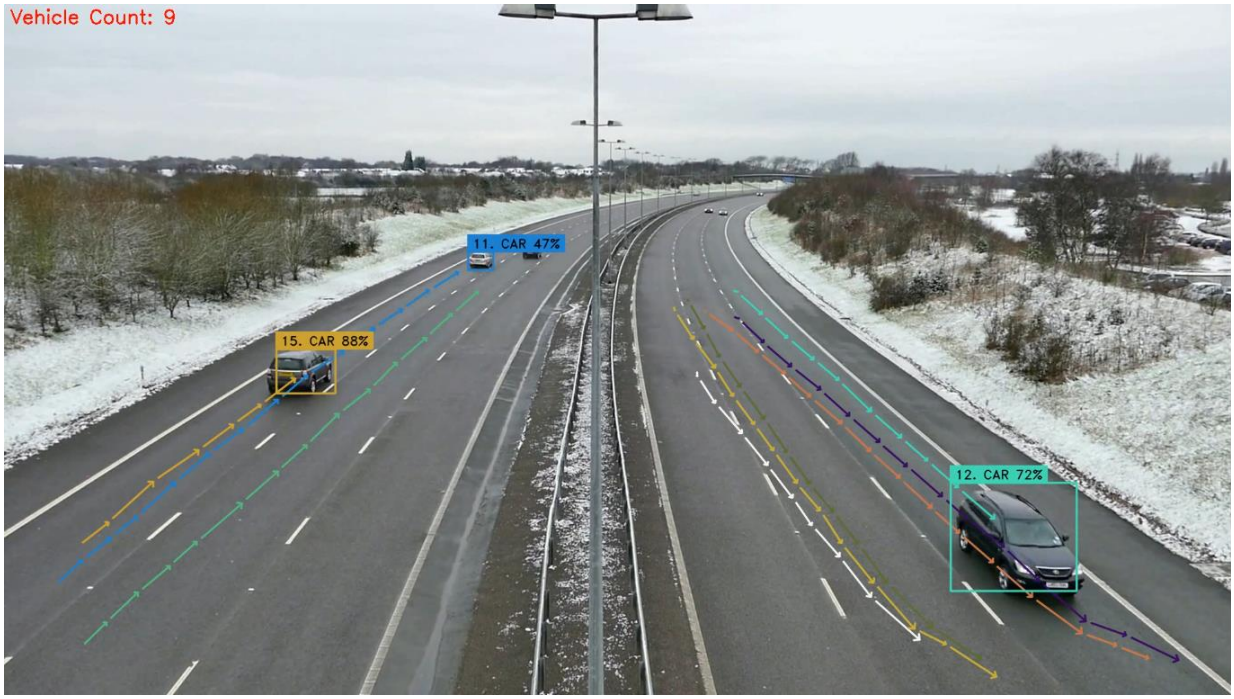


Figure 6(c): Output of Test video #3.

8. CONCLUSION

In conclusion, the vehicle detection and tracking method presented uses ByteTrack algorithm based on YOLOv7 model. It can be proven that using YOLOv7 is acceptable and a faster model than its predecessors.

It can be used in Realtime surveillance camera in the highway or recording video to evaluate the number of vehicles pass by according to what time it started recorded to last recorded. This data then can be used for traffic management by implementing answer if the place proven a lot of congestion or not.

Performance of ByteTrack totally depends upon the performance of the detector because it could be a tracker which follows a tracking by detection approach. For Future work, the system is trained for more vehicles i.e. larger dataset so that the accuracy can be increased and it can detect vehicles in extreme conditions. For that, there is a requirement of GPU so that the process will be smooth which not only increases accuracy but also decreases the time required for training.

This system can be improved to be more adaptable for vehicle detection. The first idea is using raspberry pi as the platform to put it in small places at the surveillance camera that detect vehicle passing such as in highway. This will make it more suitable to put than big machine at that place. Using the cloud computation to use as video tracking could be better and faster by using expensive GPU for computing this system. Also, it will make the computer use for running this system use lesser and small component as no GPU need in the system like as raspberry pi.

9. FUTURE ENHANCEMENTS

In the future versions we would integrate some more features into this system such as:

- Automatic Number Plate Recognition
- Accident Detection
- Rash Driving Detection based on Vector Trails

Automatic Number Plate Recognition

Traffic control and vehicle owner identification has become major problem in every country. Sometimes it becomes difficult to identify vehicle owner who violates traffic rules and drives too fast. Therefore, it is not possible to catch and punish those kinds of people because the traffic personal might not be able to retrieve vehicle number from the moving vehicle because of the speed of the vehicle. Therefore, there is a need to develop Automatic Number Plate Recognition (ANPR) system as a one of the solutions to this problem. There are numerous ANPR systems available today. These systems are based on different methodologies but still it is really challenging task as some of the factors like high speed of vehicle, non-uniform vehicle number plate, language of vehicle number and different lighting conditions can affect a lot in the overall recognition rate. Most of the systems work under these limitations.

Accident Detection

Every year, about 1.35 million humans are reduce off due to severa crashes withinside the occasion of a street visitors coincidence. According to statistics, 20 to 50 million people are injured due to it. People lose their lives due to such street injuries. These conditions are the result of a loss of coordination a number of the entities involved. In addition, failing to absolutely exercise the regulations and strategies to be accompanied amplifies the graph upwards. Risk elements consist of speeding, drinking, and driving, distracted driving, bad infrastructure, risky automobiles, breaching regulations, and lots of others. As a end result, a device this

is preferably able to coordinating the various steps that must be performed for a rapid reaction on the coincidence vicinity is required.

According to the research, such detection structures hire numerous technologies including Deep Learning methodologies and system learning approaches, amongst others. All automobiles are protected with the aid of using those detection structures, and extra technology are being examined.

Rash Driving Detection based on Vector Trails

The purpose is to detect rash driving on highway and to alert traffic authorities wirelessly the speed details and any speed violation. Accidents due to rash driving on highways are on the rise and people are losing their lives because of others mistakes. In the present system, to detect rash driving the police must use a handheld radar gun and then aim at the vehicle to record its speed. If the speed of the vehicle exceeds the speed limit, nearest police station is informed to stop the speeding vehicle. This is an ineffective process as after detecting one must inform the same and a lot of time is wasted.

Based on the path tracked by our project, we can further extend this to detect rash drivers based on their driving patterns. If a driver is driving irregular paths instead of moving along with other cars in a straight line, then the authorities can be alerted for such rash drivers.

10. REFERENCES

[1] C. Wang, A. Musaev, P. Sheinidashtegol, and T. Atkison, “Towards Detection of Abnormal Vehicle Behavior Using Traffic Cameras,” in Big Data -- BigData 2019, 2019, pp. 125–136.

[2] Kumar, A.; Jiang, M.; Fang, Y. Where not to go? In Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval; ACM: New York, NY, USA, 2014; Volume 2609550, pp. 1223–1226.

[3] H. Song, H. Liang, H. Li, Z. Dai, and X. Yun, “Visionbased vehicle detection and counting system using deep learning in highway scenes,” Eur. Transp. Res. Rev., vol. 11, no. 1, p. 51, Dec. 2019.

[4] Tejaswin, P.; Kumar, R.; Gupta, S. Tweeting Traffic: Analyzing Twitter for generating realtime city traffic insights and predictions. In Proceedings of the 2nd IKDD Conference on Data Sciences; ACM: New York, NY, USA, 2015; pp. 1–4.

[5] Suma, S.; Mehmood, R.; Albeshri, A. Automatic Event Detection in Smart Cities Using Big Data Analytics. In Proceedings of the Communications and Networking; Metzler, J.B., Ed.; Springer: Cham, Switzerland, 2018; Volume 224, pp. 111–122.

[6] X. Chen, K. Kundu, Y. Zhu, H. Ma, S. Fidler, and R. Urtasun, “3D Object Proposals Using Stereo Imagery for Accurate Object Class Detection,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 5, pp. 1259–1272, 2018.

[7] D. Sudha and J. Priyadarshini, “An intelligent multiple vehicle detection and tracking using modified vibe algorithm and deep learning algorithm,” Soft Comput., vol. 0123456789, 2020.

[8] Sang, J.; Wu, Z.; Guo, P.; Hu, H.; Xiang, H.; Zhang, Q.; Cai, B. An improved YOLOv2 for vehicle detection. Sensors 2018, 18, 4272.

[9] Du, L.; Chen, W.; Fu, S.; Kong, H.; Li, C.; Pei, Z. Real-time detection of vehicle and traffic light for intelligent and connected vehicles based on YOLOv3 network. In Proceedings of the 5th International Conference on Transportation Information and Safety (ICTIS), Liverpool, UK, 14–17 July 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 388–392.

[10] Mahto, P.; Garg, P.; Seth, P.; Panda, J. Refining YOLOv4 for Vehicle Detection. Int. J. Adv. Res. Eng. Technol. (IJARET) 2020, 11, 409–419.

[11] Liu, T.; Liu, Y. Deformable model-based vehicle tracking and recognition using 3-D constrained multiple-Kernels and Kalman filter. IEEE Access 2021, 9, 90346–90357

[12] S. Sheik Mohammed Ali, B. George, L. Vanajakshi, and J. Venkatraman, “A multiple inductive loop vehicle detection system for heterogeneous and lane-less traffic,” IEEE Trans. Instrum. Meas., vol. 61, no. 5, pp. 1353– 1360, 2012.

[13] Neupane, Bipul, Teerayut Horanont, and Jagannath Aryal. "Real-Time Vehicle Classification and Tracking Using a Transfer Learning-Improved Deep Learning Network." Sensors 22.10 (2022): 3813.

[14] A. H. Abdel-Gawad, A. Khamis, L. A. Said and A. G. Radwan, "Vulnerable Road Users Detection and Tracking using YOLOv4 and Deep SORT," 2021 9th International Japan-Africa Conference on Electronics, Communications, and Computations (JAC-ECC), 2021, pp. 140-145, doi: 10.1109/JAC-ECC54461.2021.9691441.

[15] Tao J, Wang H, Zhang X, Li X, Yang H (2018) An object detection system based on YOLO in traffic scene. In: 6th International conference on computer science and network technology, pp 315