A Project Report

on

ENHANCED TRAFFIC AND VEHICLE MONITORING SYSTEM USING YOLOv8

Submitted for partial fulfillment of the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE ENGINEERING

by

Bodapati Geethika	**– 21BQ1A0522**
Dhanekula Chandra Sekhar	**– 21BQ1A0547**
Chennaboina Krishnateja	**– 21BQ1A0537**
Devarakonda Sonia	**– 21BQ1A0546**

Under the guidance of

Dr. T. Seshu Chakravarthy

Associate Professor

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Approved by AICTE, Permanently Affiliated to JNTU, Kakinada

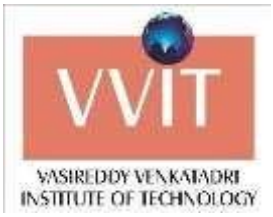
Accredited by NAAC with 'A' Grade - ISO 9001:2008 Certified

Nambur (V), Peda Kakani (M), Guntur Dt. - 522508

April, 2017.

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Department of Computer Science & Engineering



CERTIFICATE

This is to certify that the project report titled **ENHANCED TRAFFIC AND VEHICLE MONITORING SYSTEM USING YOLOv8** is being submitted by **Bodapati Geethika, Dhanekula Chandra Sekhar ,Chennaboina Krishnateja ,Devarakonda Sonia** bearing **21BQ1A0522, 21BQ1A0547, 21BQ1A0537, 21BQ1A0546** in IV B. Tech II semester *Computer Science & Engineering* is a record bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

DR V RAMA CHANDRAN **DR T SESHU CHAKRAVARTHY**

Head of the Department **Project Guide**

Signature of External Examiner with Date

DECLARATION FORMAT

We, **Bodapati Geethika, Chennaboina Krishnateja, Dhanekula Chandra Sekhar, Devarakonda Sonia** hereby declare that the Project Report entitled **"ENHANCED TRAFFIC AND VEHICLE MONITORING SYSTEM"** done by me under the guidance of **Dr. T. Seshu Chakravarthy, Associate Professor, CSE** and_____(External) at Vasireddy Venkatadri Institute of Technology is submitted in partial fulfillment of the requirements for the award of degree in Computer Science & Engineering.

DATE :

PLACE :

SIGNATURE OF THE CANDIDATE

Bodapati Geethika - 21BQ1A0522

Chennaboina Krishna Teja-21BQ1A0537

Dhanekula ChandraSekhar-21BQ1A0547

Devarakonda Sonia-21BQ1A0546

ACKNOWLEDGEMENT

I take this opportunity to express my deepest gratitude and appreciation to all those people who made this project work easier with words of encouragement, motivation, discipline, and faith by offering different places to look to expand my ideas and helped me towards the successful completion of this project work.

First and foremost, I express my deep gratitude to **Mr. Vasireddy Vidya Sagar**, Chairman, Vasireddy Venkatadri Institute of Technology for providing necessary facilities throughout the B. Tech programme.

I express my sincere thanks to **Dr. Naveen Ravela**, Director, Vasireddy Venkatadri Institute of Technology for his constant support and cooperation throughout the B. Tech programme.

I would also like to extend my gratitude to **Dr. Y Mallikarjuna Reddy**, Principal, for providing me with all the facility that was required.

I express my sincere gratitude to **Dr. V. Ramchandran**, Professor & HOD, Computer Science & Engineering, Vasireddy Venkatadri Institute of Technology for his constant encouragement, motivation and faith by offering different places to look to expand my ideas.

I would like to express my sincere gratefulness to my guide **Dr. T. Seshu Chakravarthy** for his insightful advice, motivating suggestions, invaluable guidance, help and support in successful completion of this project.

I would like to take this opportunity to express my thanks to the teaching and non-teaching staff in Department of Computer Science & Engineering, VVIT for their invaluable help and support.

Name (s) of Students

B. Geethika - 21BQ1A0522

CH. Krishna Teja -21BQ1A0537

D. Chandra Sekhar -21BQ1A0547

D. Sonia -21BQ1A0546

INSTITUTE VISION

To impart quality education through exploration and experimentation and generate socially conscious engineers, embedding ethics and values, for the advancement in Science and Technology.

INSTITUTE MISSION

• To educate students with practical approach to dovetail them to industry needs

• To govern the institution with a proactive and professional management with passionate teaching faculty.

• To provide holistic and integrated education and achieve over all development of students imparting scientific and technical, social and cognitive, managerial and organizational skills.

• To compete with the best and be the most preferred institution of the studious and the scholarly.

• To forge strong relationships and linkage with the industry.

DEPARTMENT VISION

Providing quality education to enable the generation of socially conscious software engineers who can contribute to the advancement in the field of computer science and engineering.

DEPARTMENT MISSION

• To equip the graduates with the knowledge and skills required to enable them to be industry ready.

• To train socially responsible, disciplined engineers who work with good leadership skills and can contribute for nation building.

• To make our graduates proficient in cutting edge technologies through student centric teaching-learning process and empower them to contribute significantly to the software industry

• To shape the department into a centre of academic and research excellence

COURSE OUTCOMES

CO 1: Articulate problem statement. (K2)

CO 2: Apply technial knowledge. (K3)

CO 3: Acquire contemporaray tools & technologies. (K2)

CO 4: Communicate and present the entire SDLC. (K3)

CO 5: Perform the role of a team member or lead in SDLC. (K3)

Program Educational Objectives (PEOs)

The Programme Educational Objectives of the B.Tech in Computer Science & Engineering programme are given below and are numbered from PEO1 to PEO4.

PEO-1	To provide the graduates with solid foundation in computer science and engineering along with fundamentals of Mathematics and Sciences with a view to impart in them high quality technical skills like modelling, analyzing, designing, programming and implementation with global competence and helps the graduates for life-long learning.
PEO-2	To prepare and motivate graduates with recent technological developments related to core subjects like programming, databases, design of compilers and Network Security aspects and future technologies so as to contribute effectively for Research & Development by participating in professional activities like publishing and seeking copy rights.
PEO-3	To train graduates to choose a decent career option either in high degree of employability /Entrepreneur or, in higher education by empowering students with ethical administrative acumen, ability to handle critical situations and training to excel in competitive examinations.
PEO-4	To train the graduates to have basic interpersonal skills and sense of social responsibility that paves them a way to become good team members and leaders.

Program Outcomes (POs)

The B.Tech CSE programme has documented measurable outcomes that are based on the needs of the programme's stakeholders. The programme outcomes which are derived from ABET criteria are first drafted in the academic year 2009-10 and later revised in 2010-11. The programme outcomes that the department presently adapts to are as follows:

1	**Engineering knowledge:**	Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
2	**Problem analysis:**	Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural science and engineering sciences.
3	**Design/development of solutions:**	Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal and environmental considerations.
4	**Conduct investigations of complex problems:**	Use research based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5	**Modern tool usage:**	create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations

6	**The engineer and society:**	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7	**Environment sustainability:**	Understand the impact of the professional engineering solutions in the societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8	**Ethics:**	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9	**Individual and team work:**	Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.
10	**Communication:**	communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions
11	**Project management and finance:**	Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12	**Lifelong learning**	recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broader context of technological change

Program Specific Outcomes (PSOs)

PSO-1	**Professional Skills:**	The ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, big data analytics and networking for efficient design of computer based systems of varying complexity.
PSO-2	**Successful Career and Entrepreneurship:**	The ability to employ modern computer languages, environments, and platforms in creating innovative career paths to be an entrepreneur and a zest for higher studies/employability in the field of Computer Science & Engineering.

CO – PO ARTICULATION MATRIX

Course Outcomes:

CO	Description
CO1	Articulate problem statement. (K2)
CO2	Apply technical knowledge. (K2)
CO3	Acquire contemporary tools & technologies. (K2)
CO4	Communicate and present the entire SDLC. (C3)

Mapping Table:

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2
CO1	1	2	1			2	2	3		2			1	
CO2	2	2	3	2	2		2	1	2	2	2	1	3	1
CO3		1	1	1	3							1	1	
CO4	1								3	3			1	1
CO5		1						2	3	3	2		1	2

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Symbols/Abbreviations

Abbreviations

Abbreviation	Description
YOLO	You Only Look Once

Mathematical/Statistical Symbols

Symbol	Description
Acc	Accuracy (ratio of correctly predicted instances to total instances)
P	Precision (ratio of true positives to total predicted positives)
R	Recall (ratio of true positives to total actual positives)

ABSTRACT

With the rapid urbanization and increasing number of vehicles, traffic congestion and road safety have become critical challenges. Conventional traffic management systems often struggle to cope with the growing complexity of modern traffic scenarios, leading to inefficiencies and increased accident risks. Real-time monitoring and accurate detection of vehicles and pedestrians are essential for improving traffic flow and ensuring safety. This project addresses these challenges by implementing a cutting-edge object detection system that utilizes the YOLO v8 algorithm for traffic and vehicle management.

This project focuses on leveraging the YOLO v8 (You Only Look Once) algorithm for real-time object detection in traffic and vehicle management systems. YOLO v8 offers significant improvements in speed, accuracy, and performance over previous versions such as YOLO v3 and v5, making it well-suited for detecting multiple objects, such as vehicles and pedestrians, in complex traffic environments. By utilizing YOLO v8, this project aims to develop a system capable of efficiently identifying vehicles, monitoring traffic flow, and enhancing road safety through accurate detection and tracking. The system will contribute to intelligent traffic management solutions, offering a scalable approach to managing urban traffic challenges.

CHAPTER 1

INTRODUCTION

With rapid urbanization and an increasing number of vehicles on the roads, traffic congestion and road safety have become major concerns for modern cities. Traditional traffic management systems often struggle to handle the growing complexity of urban transportation, leading to inefficiencies, delays, and heightened accident risks. The need for real-time monitoring and accurate detection of vehicles and pedestrians has become essential to enhance traffic flow, reduce congestion, and improve overall road safety.

To address these challenges, this project focuses on implementing an advanced object detection system utilizing the YOLO v8 (You Only Look Once) algorithm. YOLO v8 is a state-of-the-art deep learning model that offers significant improvements in speed, accuracy, and efficiency over its predecessors. Its ability to detect multiple objects in real-time makes it an ideal solution for traffic and vehicle management applications. By leveraging this technology, the project aims to enhance urban mobility by providing reliable and efficient vehicle and pedestrian detection.

The proposed system will be capable of accurately identifying vehicles, monitoring traffic density, and tracking pedestrian movements, ensuring smarter and safer roadways. The integration of YOLO v8 into traffic management solutions will help authorities make data-driven decisions, optimize traffic signal timings, and improve road infrastructure planning. Additionally, it will contribute to reducing accidents by alerting drivers and pedestrians about potential hazards in real time.

By implementing a scalable and intelligent traffic monitoring system, this project seeks to revolutionize traffic management strategies. The use of deep learning-based object detection will enhance urban traffic control, leading to smoother transportation systems and improved public safety. Ultimately, this project aims to provide an innovative and effective approach to addressing modern traffic challenges and fostering the development of smart cities.

CHAPTER 2

AIM AND SCOPE

The primary aim of this project is to develop an advanced object detection system using the YOLO v8 algorithm for real-time traffic and vehicle management. The system is designed to accurately detect and track vehicles and pedestrians, enabling efficient traffic monitoring, congestion control, and improved road safety. By leveraging deep learning-based object detection, the project aims to assist urban planners and traffic authorities in making data- driven decisions to optimize traffic flow, reduce accidents, and enhance overall transportation efficiency.

The scope of this project includes implementing YOLO v8 for real-time detection of multiple objects in complex traffic environments, integrating the system with surveillance cameras and traffic control infrastructure, and developing a scalable framework for urban traffic management. The project covers various applications such as vehicle counting, pedestrian detection, traffic density analysis, and accident prevention. Additionally, it explores potential integrations with smart city initiatives, autonomous vehicle systems, and AI-driven traffic optimization solutions, ensuring a comprehensive and future-ready approach to urban mobility challenges.

2.1 EXISTING SYSTEM

Traditional traffic management systems primarily rely on fixed sensors, surveillance cameras, and manual monitoring to regulate traffic flow and ensure road safety. These systems use pre-programmed traffic light sequences and basic image processing techniques for vehicle detection. However, they often struggle to adapt to real-time traffic conditions, leading to inefficiencies such as unnecessary delays at intersections and increased congestion during peak hours. The reliance on static traffic control mechanisms limits their ability to dynamically respond to fluctuations in vehicle density and pedestrian movement.

Existing vehicle detection methods, such as background subtraction and edge detection techniques, have limited accuracy in complex urban environments with varying lighting conditions and occlusions. Additionally, many conventional traffic monitoring systems depend on inductive loop sensors embedded in roads, which require significant infrastructure investments and maintenance. These sensors are prone to wear and tear, making them less reliable for long-term use.

While some modern traffic systems have begun incorporating artificial intelligence and computer vision, many still rely on older versions of object detection models such as YOLO v3 or Faster R-CNN. These models, although effective, often lack the real-time processing speed and efficiency required for high-traffic scenarios. The absence of a highly accurate and scalable real-time detection system prevents authorities from fully optimizing traffic control strategies, reducing accidents, and improving road user safety.

Key Points of the Existing System

1. Uses pre-programmed traffic light sequences and manual monitoring, leading to inefficiencies.
2. Struggles to dynamically respond to traffic density and pedestrian movement.
3. Relies on background subtraction and edge detection, which have low accuracy in complex environments.
4. Uses inductive loop sensors that require high maintenance and are prone to wear and tear.
5. Some systems use older AI models like YOLO v3 or Faster R-CNN, which are slower and less efficient for real-time detection.

2.2 LITERATURE SURVEY

Several studies have explored the use of deep learning and computer vision techniques for traffic and vehicle management, focusing on object detection, tracking, and intelligent traffic control. The advancements in AI-based models, particularly YOLO and Faster R-CNN, have significantly improved real-time detection accuracy and speed. Below are five key research studies relevant to this project:

1. **Redmon et al. (2016) - YOLO: You Only Look Once**
This study introduced the YOLO (You Only Look Once) algorithm, which revolutionized object detection by treating it as a single neural network problem. Unlike traditional region-based approaches, YOLO processes the entire image at once, significantly improving detection speed. The study demonstrated its effectiveness in real-time applications, making it suitable for traffic monitoring and vehicle detection.

2. **Zhou et al. (2018) - Faster R-CNN for Vehicle Detection in Traffic Surveillance** This research applied Faster R-CNN for vehicle detection in traffic surveillance footage. The study highlighted its superior accuracy compared to traditional methods but also pointed out its computational inefficiency, making it less ideal for real-time applications. The findings emphasized the need for a faster alternative, such as YOLO, to handle high-speed traffic scenarios effectively.

3. **Bochkovskiy et al. (2020) - YOLOv4: Optimal Speed and Accuracy of Object Detection**
This study introduced YOLOv4, an improved version of YOLO, enhancing detection accuracy and efficiency for real-time applications. The research showed that YOLOv4 could detect small and distant objects with greater precision, making it highly applicable for traffic monitoring, pedestrian detection, and vehicle tracking in urban environments.

4. **Wang et al. (2021) - Deep Learning-Based Traffic Flow Analysis Using YOLOv5** This study focused on using YOLOv5 for traffic flow analysis, emphasizing real-time vehicle counting and congestion detection. The results demonstrated that YOLOv5 outperformed traditional image-processing techniques in accuracy and processing speed, making it a viable option for smart city traffic management. The study paved the way for further improvements, leading to the development of YOLOv8.

5. Jocher et al. (2023) - YOLOv8: Advancements in Object Detection for Real-Time Applications

This research introduced YOLOv8, the latest version in the YOLO series, offering enhanced performance, accuracy, and adaptability for complex environments. The study highlighted its improvements in feature extraction, object tracking, and model efficiency, making it the most suitable model for real-time vehicle and pedestrian detection in traffic systems.

2.3 PROPOSED METHOD

The proposed system leverages the YOLOv8 algorithm to develop an advanced real-time traffic monitoring and vehicle detection system. YOLOv8 offers significant improvements in speed, accuracy, and efficiency, making it ideal for detecting multiple objects, such as vehicles and pedestrians, in complex urban traffic environments. The system processes video feeds from traffic surveillance cameras, extracts relevant objects, and classifies them into categories such as cars, buses, motorcycles, and pedestrians. This enables accurate monitoring of traffic flow, congestion levels, and pedestrian movement, contributing to intelligent traffic management solutions.

To enhance the efficiency of object detection, the system integrates image pre-processing techniques, including noise reduction, contrast enhancement, and adaptive thresholding, ensuring high detection accuracy under various lighting and weather conditions. The YOLOv8 model is trained and fine-tuned on a diverse dataset containing real-world traffic scenarios to improve its robustness and adaptability. Additionally, the system employs object tracking algorithms such as DeepSORT to track vehicle movements across multiple frames, enabling real-time traffic flow analysis, vehicle counting, and congestion detection.

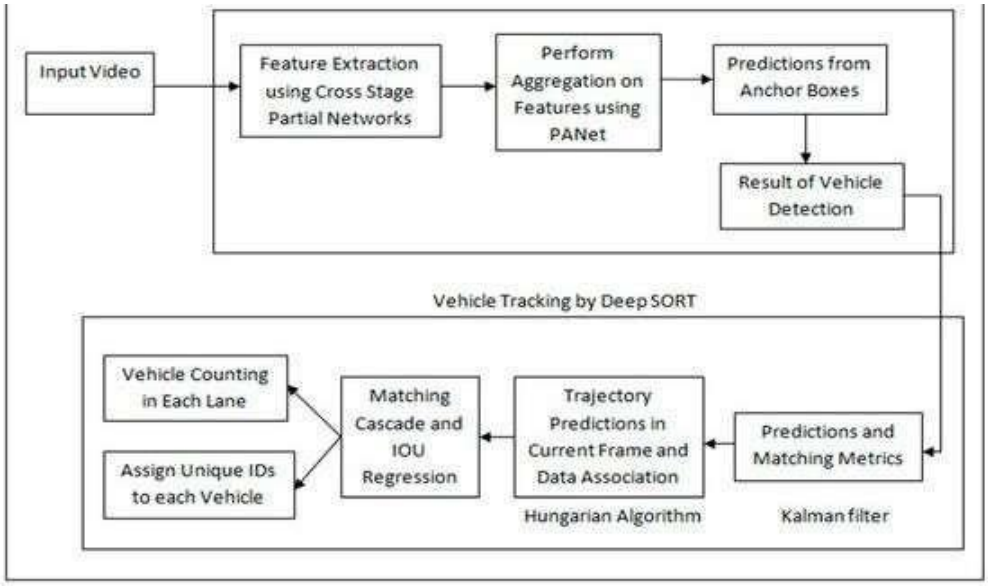


Fig 2.1 Proposeed Method

CHAPTER 3

CONCEPTS AND METHODS

3.1 PROBLEM DESCRIPTION

Indian With rapid urbanization and an increasing number of vehicles on the road, traffic congestion and road safety have become major challenges for modern cities. Conventional traffic management systems rely on static traffic signals, manual monitoring, and outdated detection techniques, which are often inefficient in handling real-time traffic conditions. These systems struggle to adapt dynamically to varying vehicle densities, leading to long delays, increased fuel consumption, and higher accident risks. Additionally, traditional vehicle detection methods, such as background subtraction and edge detection, fail in complex urban environments with poor lighting, occlusions, and high-speed moving objects.

The lack of an accurate and efficient real-time traffic monitoring system further exacerbates traffic-related issues, including poor congestion management and ineffective enforcement of traffic laws. Existing AI-based traffic monitoring systems often use older object detection models like YOLOv3 or Faster R-CNN, which lack the speed and precision required for real-time applications. There is a pressing need for an advanced, scalable, and real-time object detection system that can accurately identify vehicles and pedestrians, optimize traffic flow, and enhance road safety. The proposed system aims to address these challenges by leveraging the YOLOv8 algorithm for high- speed and accurate detection, ensuring smarter and more efficient traffic management solutions.

3.2 PROPOSED SOLUTION

With rapid urbanization and an increasing number of vehicles on the road, traffic congestion and road safety have become major challenges for modern cities. Conventional traffic management systems rely on static traffic signals, manual monitoring, and outdated detection techniques, which are often inefficient in handling real-time traffic conditions. These systems struggle to adapt dynamically to varying vehicle densities, leading to long delays, increased fuel consumption, and higher accident risks. Additionally, traditional vehicle detection methods, such as background subtraction and edge detection, fail in complex urban environments with poor lighting, occlusions, and high-speed moving objects.

The lack of an accurate and efficient real-time traffic monitoring system further exacerbates traffic-related issues, including poor congestion management and ineffective enforcement of traffic laws. Existing AI-based traffic monitoring systems often use older object detection models like YOLOv3 or Faster R-CNN, which lack the speed and precision required for real-time applications. There is a pressing need for an advanced, scalable, and real-time object detection system that can

accurately identify vehicles and pedestrians, optimize traffic flow, and enhance road safety. The proposed system aims to address these challenges by leveraging the YOLOv8 algorithm for high-speed and accurate detection, ensuring smarter and more efficient traffic management solutions.

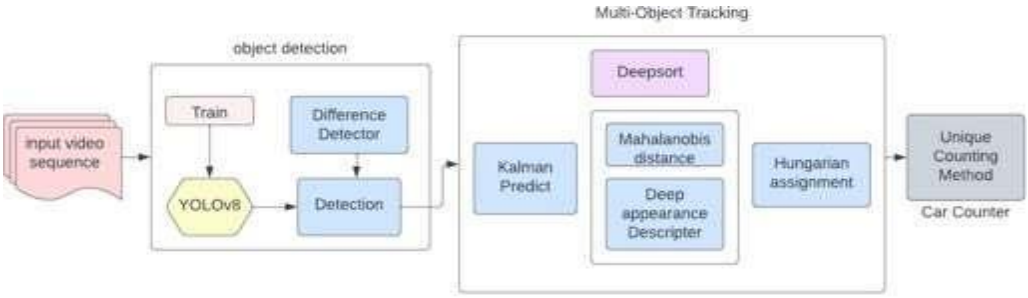


Fig 3.1: Proposed Vehicle Counting System

3.2.1 DATASET

The Road Vehicle Images Dataset consists of images captured from Indian roads, specifically curated for real-time vehicle detection using the YOLO v5 model. The dataset is divided into two main folders, one containing training images and the other for validation, ensuring a structured approach for model training and evaluation. Each image is accompanied by annotations in the YOLO format, making it suitable for object detection tasks. The dataset represents various types of vehicles commonly seen on Indian roads, providing a diverse set of scenarios to improve model generalization and accuracy in real-world applications.

This dataset plays a crucial role in AI-driven projects related to autonomous vehicles and traffic management systems. By utilizing this dataset, researchers and developers can enhance their machine learning models to detect and classify vehicles efficiently in real-time environments. The high-quality annotations help in training models that can recognize multiple vehicle categories, which is essential for developing self-driving technologies and automated traffic surveillance. It serves as a valuable resource for improving road safety, traffic flow analysis, and intelligent transportation systems

.

3.2.2 DATASET PREPARATION

Preparing the Road Vehicle Images Dataset for training involves several essential steps to ensure optimal model performance. First, the dataset is organized into separate folders for training and validation, allowing the model to learn from one set while being tested on another. The images are preprocessed by resizing them to a standard resolution suitable for YOLO v5 and converting them into the appropriate format. The accompanying YOLO annotation files, which contain bounding box coordinates for each detected vehicle, are also verified for accuracy. Any mislabeled or low-

quality images are removed to maintain a high standard for training.
Once the dataset is structured, it is augmented using techniques such as rotation, flipping, and brightness adjustments to improve model robustness. This step helps in increasing the diversity of the dataset, allowing the model to perform well in different lighting conditions and angles. The dataset is then converted into the necessary format for YOLO v5 training, ensuring that images and annotations align correctly. Finally, the data is split into training and validation sets, typically in an 80-20 ratio, to optimize learning while maintaining a reliable evaluation process. This structured preparation ensures that the model can accurately detect and classify vehicles in real-world scenarios.

3.2.3 DATASET PREPROCESSING

For YOLOv8 model training, dataset preprocessing is crucial to ensure high-quality object detection and model efficiency. The first step involves organizing the dataset into structured directories, typically with separate folders for training, validation, and testing images. Each image must be resized to a standard resolution compatible with YOLOv8, ensuring consistency across the dataset. The YOLO annotation format, which includes class labels and bounding box coordinates, is verified for accuracy and corrected if necessary. Any noisy or irrelevant data, such as blurred images or incorrect labels, is filtered out to maintain dataset quality. Additionally, images are normalized and converted into a format suitable for fast processing during model training.
To enhance the dataset's diversity and improve model generalization, data augmentation techniques such as rotation, scaling, flipping, contrast adjustments, and motion blur are applied. These transformations help the model learn from different perspectives, making it more robust to real-world variations in lighting, angle, and object positioning. The dataset is then split into training, validation, and test sets, typically in an 80-10-10 ratio, to optimize learning and evaluation. Finally, the dataset is formatted according to YOLOv8's requirements, ensuring seamless integration with the model for efficient training and accurate vehicle detection.

3.2.4 DATASET AUGEMENTATION

Dataset augmentation plays a crucial role in improving the performance of the YOLOv8 model by increasing the diversity of training data without requiring additional labeled images. Various augmentation techniques are applied to introduce variations in brightness, contrast, and orientation, making the model more robust to different environmental conditions. Common augmentation methods include rotation, flipping, scaling, cropping, and adding noise to simulate real-world scenarios. Motion blur and Gaussian noise are often used to account for dynamic traffic conditions, while brightness and contrast adjustments help the model generalize better to varying lighting conditions such as daytime, nighttime, or foggy weather.

3.2.5 MODELS USED

YOLOV8

YOLOv8 (You Only Look Once version 8) is an advanced object detection model designed for real-time applications. It builds upon previous YOLO versions by incorporating architectural improvements for better accuracy, speed, and efficiency. YOLOv8 uses an anchor-free detection approach, allowing it to predict object locations with greater flexibility. It employs a convolutional neural network (CNN) backbone with advanced feature extraction layers, making it highly effective in detecting vehicles on roads. YOLOv8 is optimized for tasks like object detection, segmentation, and classification, making it a suitable choice for real-time vehicle detection in traffic surveillance and autonomous driving applications. Its ability to process images in a single pass through the network enables fast inference, which is crucial for time-sensitive tasks.
Several key techniques are used to enhance YOLOv8's performance, including mosaic augmentation for better generalization, dynamic anchor-free detection for more flexible bounding box predictions, and auto-learning of anchor points for improved localization. Additionally, YOLOv8 incorporates feature pyramid networks (FPN) and path aggregation networks (PAN) to refine multi-scale feature learning, making it highly effective in detecting vehicles of various sizes and orientations. The model also benefits from techniques such as non-maximum suppression (NMS) to eliminate redundant bounding boxes and optimize detection accuracy. These advanced methodologies make YOLOv8 well-suited for road vehicle detection, ensuring fast and accurate recognition of vehicles in real-world traffic environments.

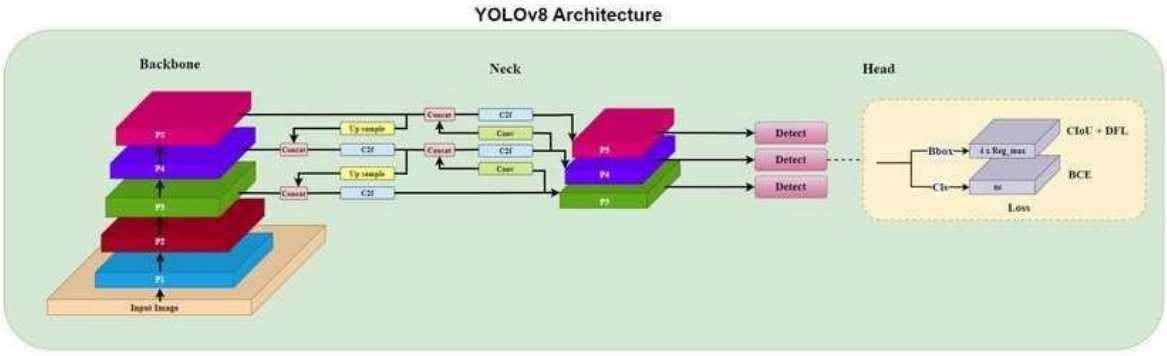


Fig 3.2: YOLOv8 Architecture

DEEP SORT

Deep SORT (Simple Online and Realtime Tracker) is an object tracking algorithm that extends the SORT (Simple Online Realtime Tracker) framework by incorporating deep learning-based appearance features. It uses a Kalman filter for motion prediction and the Hungarian algorithm for object association, allowing it to track multiple objects across frames accurately. Deep SORT improves object tracking by integrating a deep learning-based feature extractor, which helps

differentiate between objects with similar motion patterns. This makes it highly effective in tracking vehicles in busy traffic scenarios, where objects frequently overlap or occlude each other. When combined with YOLOv8, Deep SORT can be used for real-time vehicle tracking, enabling applications such as automated traffic monitoring, anomaly detection, and autonomous navigation.

3.2.6 PEFORMANCE MODELS

Evaluating the performance of the YOLOv8 model for road vehicle detection involves several key metrics that assess both accuracy and efficiency. The primary metric used is **Mean Average Precision (mAP)**, which measures the model's precision-recall performance across different Intersection over Union (IoU) thresholds. A higher mAP value indicates better object detection accuracy. **Precision and recall** are also crucial; precision represents the proportion of correctly detected vehicles among all detections, while recall indicates how well the model identifies all actual vehicles in the dataset. The **F1-score**, which is the harmonic mean of precision and recall, provides a balanced measure of detection performance.

Other important metrics include **IoU (Intersection over Union)**, which measures the overlap between predicted and ground truth bounding boxes, ensuring accurate localization. FPS (Frames Per Second) is used to assess the model's inference speed, which is crucial for real-time applications such as autonomous driving and traffic monitoring. False Positives (FP) and False Negatives (FN) are analyzed to minimize incorrect detections and missed vehicles. Additionally, **Confusion Matrix analysis** helps in understanding the distribution of correct and incorrect predictions. These metrics collectively determine the effectiveness of YOLOv8 in detecting road vehicles accurately and efficiently in real-world scenarios.

Precision: Precision is defined as the ratio of correctly classified positive samples (True Positive) to a total number of classified positive samples (either correctly or incorrectly).

$$\text{Precision} = \frac{TP}{TP + FP}$$

TP True Positive, FP False Positive

Recall: The recall is calculated as the ratio between the numbers of Positive samples correctly classified as Positive to the total number of Positive samples. The recall measures the model's ability to detect positive samples. The higher the recall, the more positive samples detected.

$$\text{Recall} = \frac{TP}{TP + FN}$$

TP True Positive
FN False Negative

IoU (Intersection over Union)

Intersection over Union (IoU) is used to evaluate the performance of object detection by comparing the ground truth bounding box to the preddicted bounding box and IoU is the topic of this tutorial.

A solid understanding of IoU requires practical applications. Access to a well-curated dataset allows learners to engage with real-world challenges, enhancing their understanding of object detection and how IoU is applied for accuracy.

Mean Average Precision (mAP)

Mean Average Precision (mAP) is a key performance metric used in object detection tasks, including YOLOv8-based vehicle detection. It evaluates the model's ability to accurately detect and classify objects by measuring precision and recall at different Intersection over Union (IoU) thresholds. The mAP score is computed by calculating the Average Precision (AP) for each class and then averaging them across all classes. AP is determined by plotting the precision-recall curve and computing the area under this curve. A higher mAP value indicates better object detection performance, meaning the model can precisely locate and classify road vehicles in the dataset.

In YOLOv8, mAP is often calculated at different IoU thresholds, such as **mAP@0.5** (considering IoU > 50%) and **mAP@0.5:0.95** (averaging across multiple IoU thresholds from 0.5 to 0.95 in steps of 0.05). **mAP@0.5** is more lenient, focusing on detection presence rather than exact localization, while **mAP@0.5:0.95** provides a more comprehensive evaluation of the model's precision and robustness. Since object detection requires both high accuracy and precise localization, mAP is a crucial metric in determining the effectiveness of YOLOv8 for real-world vehicle detection applications.

3.3 System Analysis Methods

System analysis involves studying the system's requirements, functionalities, and overall workflow to ensure efficiency and accuracy in implementation. The process begins with identifying user needs and defining system objectives. Key aspects include **data flow analysis**, which examines how data moves through the system, and **functional analysis**, which breaks down the core functionalities into structured components. Additionally, system behavior is analyzed to predict potential issues and optimize performance. This analysis forms the foundation for designing a robust and scalable solution for traffic analysis.

3.4 System Requirements

System requirements define the necessary hardware and software specifications needed for the successful execution of the project.

3.4.1 Hardware Requirements

The hardware components required for the implementation and execution of the project are:

Component	Specification
Processor	Intel Core i5/i7 or higher
RAM	Minimum 8GB (Recommended 16GB)
Storage	Minimum 256GB SSD (Recommended 512GB)
GPU	NVIDIA GTX 1050 or higher (for faster processing)
Camera	HD Camera (for real-time detection if applicable)

Table-3.1: Hardware Requirements

3.4.2 Software Requirement

The software tools and libraries required for the project are:

Software/Tool	Version	Purpose
Operating System	Windows 10/11, Ubuntu 20.04	Development & Execution
Programming Language	Python 3.8+	Model Implementation
IDE	VS Code / PyCharm	Coding & Debugging
Libraries	OpenCV, NumPy, Pandas, Matplotlib, YOLOv8	Object Detection & Data Processing
Deep Learning Framework	TensorFlow/PyTorch	Model Training
Power BI / Excel	Latest Version	Data Visualization (if applicable)

Table – 3.2: Software Requirements

CHAPTER 4

IMPLEMENTATION

The implementation of the YOLOv8-based road vehicle detection system involves several key steps, starting with dataset preparation and preprocessing. The dataset, containing images of Bangladeshi road vehicles, is split into training and validation sets, with annotations in YOLO format specifying bounding box coordinates. The preprocessing stage includes image resizing, normalization, and augmentation techniques such as flipping, rotation, and brightness adjustments to enhance the model's generalization. These steps ensure that the model can recognize vehicles in various lighting and weather conditions, improving its robustness for real-world deployment.

Next, YOLOv8 is trained using the processed dataset, leveraging its efficient architecture for real-time object detection. The training process involves defining hyperparameters such as learning rate, batch size, and the number of epochs to optimize model performance. The model uses a convolutional neural network (CNN) backbone to extract features from images and a detection head to predict bounding boxes and class labels. During training, loss functions like objectness loss, classification loss, and localization loss are minimized using techniques such as stochastic gradient descent (SGD) or Adam optimizer. The training process is monitored using metrics like loss curves and validation accuracy to ensure the model is learning effectively.

Once training is complete, the trained model is evaluated using performance metrics such as Mean Average Precision (mAP) at different Intersection over Union (IoU) thresholds. The model is tested on unseen validation images to assess its accuracy in detecting and classifying vehicles. Post-processing techniques, such as non-maximum suppression (NMS), are applied to eliminate duplicate bounding boxes and refine detections. Finally, the trained model can be deployed in real-time applications, such as traffic monitoring or autonomous vehicle navigation, using tools like OpenCV and TensorRT for optimized inference on edge devices.

4.1 TOOLS USED

PYTHON:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics developed by Guido van Rossum. It was originally released in 1991. Designed to be easy as well as fun, the name "Python" is a nod to the British comedy group Monty Python. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics developed by Guido van Rossum. It was originally released in 1991. Designed to be easy as well as fun, the name "Python" is a nod to the British comedy group Monty Python. Python has a reputation as a beginner-friendly language, replacing Java as the most widely used introductory language because it handles much of the complexity for the user, allowing beginners to focus on fully grasping programming concepts rather than minute details.

Python is used for server-side web development, software development, mathematics, and system scripting, and is popular for Rapid Application Development and as a scripting or glue language to tie existing components because of its high-level, built-in data structures, dynamic typing, and dynamic binding. Program maintenance costs are reduced with Python due to the easily learned syntax and emphasis on readability. Additionally, Python's support of modules and packages facilitates modular programs and reuse of code. Python is an open source community language, so numerous independent programmers are continually building libraries and functionality for it. Professionally, Python is great for backend web development, data analysis, artificial intelligence, and scientific computing. Developers also use Python to build productivity tools, games, and desktop apps.

Matplotlib:

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- Create publication quality plot
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many format files
- Embed in Jupyter Lab and Graphical User Interface
- Use a rich array of third-party packages built on Matplotlib.

Matplotlib is a low level graph plotting library in python that serves as a visualization utility. Matplotlib was created by John D. Hunter. Matplotlib is open source and we can use it freely. Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

Install it using this command:

```
C:\Users\Your Name>pip install matplotlib
```

Seaborn:

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on top matplotlib library and is also closely integrated with the data structures from pandas. Seaborn aims to make visualization the central part of exploring and understanding data. It provides dataset-oriented APIs so that we can switch between different visual representations for the same variables for a better understanding of the dataset.

TensorFlow:

TensorFlow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions. Tensors are used as the basic data structures in TensorFlow language. Tensors represent the connecting edges in any flow diagram called the Data Flow Graph. Tensors are defined as multidimensional array or list.

Important features of TensorFlow:

- It includes a feature of that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.
- It includes a programming support of deep neural networks and machine learning techniques.
- It includes a high scalable feature of computation with various data sets.
- TensorFlow uses GPU computing, automating management. It also includes a unique feature of optimization of same memory and the data used.

Keras:

Keras is compact, easy to learn, high-level Python library run on top of TensorFlow framework. It is made with focus of understanding deep learning techniques, such as creating layers for neural networks maintaining the concepts of shapes and mathematical details. The creation of framework can be of the following two types –

- Sequential API
- Functional API

OpenCV:

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as NumPy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV.

This OpenCV tutorial will help you learn the Image-processing from Basics to Advance, like operations on Images, Videos using a huge set of Open cv-programs and projects

Sklearn:

Scikit-learn has emerged as a powerful and user-friendly Python library. Its simplicity and versatility make it a better choice for both beginners and seasoned data scientists to build and implement machine learning models. In this article, we will explore about Sklearn.

Scikit-learn is an open-source Python library that implements a range of machine learning, pre-processing, cross- validation, and visualization algorithms using a unified interface. It is an open-source machine-learning library that provides a plethora of tools for various Machine-learning tasks such as Classification, Regression, and Clustering many more.

Google Colaboratory:

Google Colab, short for Colaboratory, is a free cloud-based platform provided by Google that allows users to write and execute Python code collaboratively in a Jupyter Notebook environment. **Google Collaboratory notebook,** is designed to facilitate machine learning (ML) and data science tasks by providing a virtual environment, Google colab python with access to free GPU resources.

Benefits of Google Colab

Google Colab offers several benefits that make it a popular choice among data scientists, researchers, and machine learning practitioners. **Key features of Google Collaboratory notebook include:**

- **Free Access to GPUs:** Colab offers free GPU access, which is particularly useful for training machine learning models that require significant computational power.
- **No Setup Required**: Colab runs in the cloud, eliminating the need for users to set up and configure their own development environment. This makes it convenient for quick coding and collaboration.
- **Collaborative Editing:** Multiple users can work on the same Colab notebook simultaneously, making it a useful tool for collaborative projects.
- **Integration with Google Drive**: Colab is integrated with Google Drive, allowing users to save their work directly to their Google Drive account. This enables easy sharing and access to notebooks from different devices.
- **Support for Popular Libraries**: Colab comes pre-installed with many popular Python libraries for machine learning, data analysis, and visualization, such as TensorFlow, PyTorch, Matplotlib, and more.
- **Easy Sharing**: Colab notebooks can be easily shared just like Google Docs or Sheets. Users can provide a link to the notebook, and others can view or edit the code in real-time.

Streamlit

Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps.

It is a Python-based library specifically designed for machine learning engineers. Data scientists or machine learning engineers are not web developers and they're not interested in spending weeks learning to use these frameworks to build web apps. Instead, they want a tool that is easier to learn and to use, as long as it can display data and collect needed parameters for modeling.

Streamlit allows you to create a stunning-looking application with only a few lines of code.

4.2 CODE

```
# Vehicle detection using YOLOV8

!pip install ultralytics

!pip install -U ipywidgets

# Install modules

!pip install ultralytics

!pip install -U ipywidgets

!pip install -q kaggle

!pip install -q supervision "ultralytics<=8.3.40"

# Import modules

import os

import random

from IPython import get_ipython

from IPython.display import display

import locale

from ultralytics import YOLO

import cv2

import pandas as pd

import seaborn as sns
```

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from PIL import Image
import numpy as np
import supervision as sv
from tqdm import tqdm
from supervision.assets import VideoAssets, download_assets
from collections import defaultdict, deque
# Set the locale to UTF-8 before running any shell commands
def set_utf8_locale():
  """Sets the locale to UTF-8."""
  try:
    locale.setlocale(locale.LC_ALL, 'en_US.UTF-8')
  except locale.Error:
    # Fallback for systems without en_US.UTF-8
    try:
      locale.setlocale(locale.LC_ALL, 'C.UTF-8')
    except locale.Error:
      print("Could not set locale to UTF-8")
  os.environ['LC_ALL'] = 'en_US.UTF-8'
  os.environ['LANG'] = 'en_US.UTF-8'
  os.environ['PYTHONIOENCODING'] = 'UTF-8'
  print(f"Locale is set to {locale.getlocale()}")
  print(f"Default locale is set to {locale.getdefaultlocale()}")
  print(f"Preferred encoding is set to {locale.getpreferredencoding()}")
  set_utf8_locale()
```

```
import os
import random
from ultralytics import YOLO
import cv2

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from PIL import Image
!pip install -q kaggle
#!/bin/bash
!kaggle datasets download ashfakyeafi/road-vehicle-images-dataset
!unzip road-vehicle-images-dataset.zip
train_images = "/content/trafic_data/train/images"
train_labels = "/content/trafic_data/train/labels"
def load_labels(image_file, train_labels):
    label_file = os.path.splitext(image_file)[0] + ".txt"
    label_path = os.path.join(train_labels, label_file)
    with open(label_path, "r") as f:
        labels = f.read().strip().split("\n")
    return labels
def plot_object_detections(ax, image, labels):
    for label in labels:
        if len(label.split()) != 5:
            continue
```

```
        class_id, x_center, y_center, width, height = map(float, label.split())
        x_min = int((x_center - width/2) * image.shape[1])
        y_min = int((y_center - height/2) * image.shape[0])
        x_max = int((x_center + width/2) * image.shape[1])
        y_max = int((y_center + height/2) * image.shape[0])
        cv2.rectangle(image, (x_min, y_min), (x_max, y_max), (0, 255, 0), 3)
    ax.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    ax.axis('off')
# Set paths
train_images = "/content/trafic_data/train/images"
train_labels = "/content/trafic_data/train/labels"
# Get a list of all the image files in the training images directory
image_files = os.listdir(train_images)
# Choose 16 random image files from the list
random_images = random.sample(image_files, 16)
# Set up the plot
fig, axs = plt.subplots(4, 4, figsize=(16, 16))
# Loop over the random images and plot the object detections
for i, image_file in enumerate(random_images):
    row, col = divmod(i, 4)
    # Load the image
    image_path = os.path.join(train_images, image_file)
    image = cv2.imread(image_path)
    # Load the labels for this image
    labels = load_labels(image_file, train_labels)
    # Plot object detections
```

```
    plot_object_detections(axs[row, col], image, labels)
plt.show()


with open('/content/trafic_data/data_1.yaml', 'r') as f:
    data = f.read()
print(data)
h, w, c = image.shape
print(f"The image has dimensions {w}x{h} and {c} channels.")
model = YOLO("yolov8x.pt")
result_predict = model.predict(source = os.path.join(train_images, random_images[0]),
imgsz=(416))
plot = result_predict[0].plot()
plot = cv2.cvtColor(plot, cv2.COLOR_BGR2RGB)
display(Image.fromarray(plot))
fig, axs = plt.subplots(4, 4, figsize = (16, 12))
# Loop over the random images and plot the object detections
for i, image_file in enumerate(random_images):
    row, col = divmod(i, 4)
    image_path = os.path.join(train_images, image_file)
    image = cv2.imread(image_path)
    result_predict = model.predict(image, imgsz=(416))
    plot = result_predict[0].plot()
    plot = cv2.cvtColor(plot, cv2.COLOR_BGR2RGB)
    axs[row,col].imshow(plot)
    axs[row,col].axis('off')
plt.show()
```

```
model = YOLO('yolov8x.pt')


# Training the model
model.train(data = '/content/trafic_data/data_1.yaml',
            epochs = 30,
            imgsz = h,
            seed = 42,
            batch = 8,
            workers = 4)


df = pd.read_csv('/content/runs/detect/train/results.csv')
df.columns = df.columns.str.strip()
# create subplots using seaborn
fig, axs = plt.subplots(nrows=5, ncols=2, figsize=(15, 15))
# plot the columns using seaborn
sns.lineplot(x='epoch', y='train/box_loss', data=df, ax=axs[0,0])
sns.lineplot(x='epoch', y='train/cls_loss', data=df, ax=axs[0,1])
sns.lineplot(x='epoch', y='train/dfl_loss', data=df, ax=axs[1,0])
sns.lineplot(x='epoch', y='metrics/precision(B)', data=df, ax=axs[1,1])
sns.lineplot(x='epoch', y='metrics/recall(B)', data=df, ax=axs[2,0])
sns.lineplot(x='epoch', y='metrics/mAP50(B)', data=df, ax=axs[2,1])
sns.lineplot(x='epoch', y='metrics/mAP50-95(B)', data=df, ax=axs[3,0])
sns.lineplot(x='epoch', y='val/box_loss', data=df, ax=axs[3,1])
sns.lineplot(x='epoch', y='val/cls_loss', data=df, ax=axs[4,0])
sns.lineplot(x='epoch', y='val/dfl_loss', data=df, ax=axs[4,1])
# set titles and axis labels for each subplot
```

```
axs[0,0].set(title='Train Box Loss')
axs[0,1].set(title='Train Class Loss')
axs[1,0].set(title='Train DFL Loss')
axs[1,1].set(title='Metrics Precision (B)')
axs[2,0].set(title='Metrics Recall (B)')
axs[2,1].set(title='Metrics mAP50 (B)')
axs[3,0].set(title='Metrics mAP50-95 (B)')
axs[3,1].set(title='Validation Box Loss')
axs[4,0].set(title='Validation Class Loss')
axs[4,1].set(title='Validation DFL Loss')

# add suptitle and subheader
plt.suptitle('Training Metrics and Loss', fontsize=24)
# adjust top margin to make space for suptitle
plt.subplots_adjust(top=0.8)
# adjust spacing between subplots
plt.tight_layout()
plt.show()
img = mpimg.imread('/content/runs/detect/train/confusion_matrix_normalized.png')
fig, ax = plt.subplots(figsize = (15, 15))
ax.imshow(img)
ax.axis('off');
def ship_detect(img_path):
    # Read the image
    img = cv2.imread(img_path)
```

```
    # Pass the image through the detection model and get the result
    detect_result = model(img)
    # Plot the detections
    detect_img = detect_result[0].plot()
    # Convert the image to RGB format
    detect_img = cv2.cvtColor(detect_img, cv2.COLOR_BGR2RGB)
    return detect_img
custom_image_dir = '/content/trafic_data/valid/images'
# Get the list of image files in the directory
image_files = os.listdir(custom_image_dir)
# Select 16 random images from the list
selected_images = random.sample(image_files, 9)
# Create a figure with subplots for each image
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(18, 18))
# Iterate over the selected images and plot each one
for i, img_file in enumerate(selected_images):
    # Compute the row and column index of the current subplot
    row_idx = i // 3
    col_idx = i % 3
    # Load the current image and run object detection
    img_path = os.path.join(custom_image_dir, img_file)
    detect_img = ship_detect(img_path)
    # Plot the current image on the appropriate subplot
    axes[row_idx, col_idx].imshow(detect_img)
    axes[row_idx, col_idx].axis('off')
```

```
# Adjust the spacing between the subplots
plt.subplots_adjust(wspace=0.05, hspace=0.05)
# Vehicle Tracking and Speed Estimation
!nvidia-smi
## Install
!pip install -q supervision "ultralytics<=8.3.40"
## Imports
import cv2
import numpy as np
import supervision as sv
from tqdm import tqdm
from ultralytics import YOLO
from supervision.assets import VideoAssets, download_assets
from collections import defaultdict, deque
## Download Data
download_assets(VideoAssets.VEHICLES)
## Configuration
SOURCE_VIDEO_PATH = "/content/vehicles.mp4"
TARGET_VIDEO_PATH = "/content/vehicles-result.mp4"
CONFIDENCE_THRESHOLD = 0.3
IOU_THRESHOLD = 0.5
MODEL_NAME = "yolov8x.pt"
MODEL_RESOLUTION = 1280

SOURCE = np.array([
    [1252, 787],
```

```
    [2298, 803],
    [5039, 2159],
    [-550, 2159]
])
TARGET_WIDTH = 25
TARGET_HEIGHT = 250
TARGET = np.array([
    [0, 0],
    [TARGET_WIDTH - 1, 0],
    [TARGET_WIDTH - 1, TARGET_HEIGHT - 1],
    [0, TARGET_HEIGHT - 1],
])
frame_generator = sv.get_video_frames_generator(source_path=SOURCE_VIDEO_PATH)
frame_iterator = iter(frame_generator)
frame = next(frame_iterator)
annotated_frame = frame.copy()
annotated_frame = sv.draw_polygon(scene=annotated_frame, polygon=SOURCE,
color=sv.Color.RED, thickness=4) # Use sv.Color.RED instead of sv.Color.red
sv.plot_image(annotated_frame)
## Transform Perspective
class ViewTransformer:
    def __init__(self, source: np.ndarray, target: np.ndarray) -> None:
        source = source.astype(np.float32)
        target = target.astype(np.float32)
        self.m = cv2.getPerspectiveTransform(source, target)
```

```
    def transform_points(self, points: np.ndarray) -> np.ndarray:
        if points.size == 0:
            return points
        reshaped_points = points.reshape(-1, 1, 2).astype(np.float32)
        transformed_points = cv2.perspectiveTransform(reshaped_points, self.m)
        return transformed_points.reshape(-1, 2)
view_transformer = ViewTransformer(source=SOURCE, target=TARGET)
## Process Video
dir(sv)
model = YOLO(MODEL_NAME)
video_info = sv.VideoInfo.from_video_path(video_path=SOURCE_VIDEO_PATH)
frame_generator = sv.get_video_frames_generator(source_path=SOURCE_VIDEO_PATH)
# tracer initiation
byte_track = sv.ByteTrack(frame_rate=video_info.fps)

# annotators configuration
thickness = sv.calculate_optimal_line_thickness(
    resolution_wh=video_info.resolution_wh
)
text_scale = sv.calculate_optimal_text_scale(
    resolution_wh=video_info.resolution_wh
)

bounding_box_annotator = sv.BoxAnnotator(
    thickness=thickness
)
```

```
label_annotator = sv.LabelAnnotator(
    text_scale=text_scale,
    text_thickness=thickness,
    text_position=sv.Position.BOTTOM_CENTER
)
trace_annotator = sv.TraceAnnotator(
    thickness=thickness,
    trace_length=video_info.fps * 2,
    position=sv.Position.BOTTOM_CENTER
)

polygon_zone = sv.PolygonZone(
    polygon=SOURCE
)
coordinates = defaultdict(lambda: deque(maxlen=video_info.fps))
# open target video
with sv.VideoSink(TARGET_VIDEO_PATH, video_info) as sink:
    # loop over source video frame
    for frame in tqdm(frame_generator, total=video_info.total_frames):
        result = model(frame, imgsz=MODEL_RESOLUTION, verbose=False)[0]
        detections = sv.Detections.from_ultralytics(result)

        # filter out detections by class and confidence
        detections = detections[detections.confidence > CONFIDENCE_THRESHOLD]
        detections = detections[detections.class_id != 0]
```

```
# filter out detections outside the zone
detections = detections[polygon_zone.trigger(detections)]

# refine detections using non-max suppression
detections = detections.with_nms(IOU_THRESHOLD)
# pass detection through the tracker
detections = byte_track.update_with_detections(detections=detections)
points = detections.get_anchors_coordinates(
    anchor=sv.Position.BOTTOM_CENTER
)
# calculate the detections position inside the target RoI
points = view_transformer.transform_points(points=points).astype(int)
# store detections position
for tracker_id, [_, y] in zip(detections.tracker_id, points):
    coordinates[tracker_id].append(y)
# format labels
labels = []
for tracker_id in detections.tracker_id:
    if len(coordinates[tracker_id]) < video_info.fps / 2:
        labels.append(f"#{tracker_id}")
    else:
        # calculate speed
        coordinate_start = coordinates[tracker_id][-1]
        coordinate_end = coordinates[tracker_id][0]
        distance = abs(coordinate_start - coordinate_end)
        time = len(coordinates[tracker_id]) / video_info.fps
```

```
            speed = distance / time * 3.6
            labels.append(f"#{tracker_id} {int(speed)} km/h")
        # annotate frame
        annotated_frame = frame.copy()
        annotated_frame = trace_annotator.annotate(
            scene=annotated_frame, detections=detections
        )
        annotated_frame = bounding_box_annotator.annotate(
            scene=annotated_frame, detections=detections
        )
        annotated_frame = label_annotator.annotate(
            scene=annotated_frame, detections=detections, labels=labels
        )

        # add frame to target video
        sink.write_frame(annotated_frame)
from google.colab import drive
drive.mount('/content/drive')

import supervision as sv
import numpy as np
from collections import defaultdict, deque
from ultralytics import YOLO
from tqdm import tqdm
# Constants
MODEL_NAME = "yolov8x.pt"
```

```
SOURCE_VIDEO_PATH = "/content/vehicles.mp4"

TARGET_VIDEO_PATH = "/content/drive/MyDrive/output.mp4"

MODEL_RESOLUTION = 640

CONFIDENCE_THRESHOLD = 0.3

IOU_THRESHOLD = 0.5

# Load the YOLOv8 model

model = YOLO(MODEL_NAME)

# Get video metadata and frame generator

video_info = sv.VideoInfo.from_video_path(video_path=SOURCE_VIDEO_PATH)

frame_generator = sv.get_video_frames_generator(source_path=SOURCE_VIDEO_PATH)

# Initialize Object Tracker

byte_track = sv.ByteTrack(frame_rate=video_info.fps)


# Calculate optimal annotation settings

thickness = sv.calculate_optimal_line_thickness(resolution_wh=video_info.resolution_wh)

text_scale = sv.calculate_optimal_text_scale(resolution_wh=video_info.resolution_wh)


# Define annotators

bounding_box_annotator = sv.BoxAnnotator(thickness=thickness)

label_annotator = sv.LabelAnnotator(text_scale=text_scale, text_thickness=thickness,
text_position=sv.Position.BOTTOM_CENTER)

trace_annotator = sv.TraceAnnotator(thickness=thickness, trace_length=video_info.fps * 2,
position=sv.Position.BOTTOM_CENTER)

# Define a Polygon Zone (Optional, can be modified)

SOURCE = np.array([(100, 300), (500, 300), (500, 600), (100, 600)])

polygon_zone = sv.PolygonZone(polygon=SOURCE)
```

```
# Stores tracking data for speed calculation
coordinates = defaultdict(lambda: deque(maxlen=video_info.fps))
# Class names mapping for vehicle detection (COCO dataset class IDs)
class_names = {2: "Car", 3: "Motorcycle", 5: "Bus", 7: "Truck"}
# Process the video
with sv.VideoSink(TARGET_VIDEO_PATH, video_info) as sink:
    for frame in tqdm(frame_generator, total=video_info.total_frames):
        # Run YOLOv8 inference
        result = model(frame, imgsz=MODEL_RESOLUTION, verbose=False)[0]
        detections = sv.Detections.from_ultralytics(result)
        # Filter detections by confidence threshold
        detections = detections[detections.confidence > CONFIDENCE_THRESHOLD]
        detections = detections[detections.class_id != 0]  # Ignore class 0 (person)
        detections = detections[polygon_zone.trigger(detections)]  # Keep objects in polygon
        detections = detections.with_nms(IOU_THRESHOLD)  # Apply NMS
        detections = byte_track.update_with_detections(detections=detections) # Apply tracking
        # Get object bottom-center coordinates for tracking
        points = detections.get_anchors_coordinates(anchor=sv.Position.BOTTOM_CENTER)
        poInts = np.array(points, dtype=int)
        # Store coordinates for speed calculation
        for tracker_id, [_, y] in zip(detections.tracker_id, points):
            coordinates[tracker_id].append(y)
        # Generate labels with ID, class name, confidence, and speed
        labels = []
        for tracker_id, class_id, conf in zip(detections.tracker_id, detections.class_id,
detections.confidence):
```

```
            vehicle_label = class_names.get(class_id, "Vehicle")
            if len(coordinates[tracker_id]) < video_info.fps / 2:
                speed_text = ""
            else:
                coordinate_start = coordinates[tracker_id][-1]
                coordinate_end = coordinates[tracker_id][0]
                distance = abs(coordinate_start - coordinate_end)
                time = len(coordinates[tracker_id]) / video_info.fps
                speed = distance / time * 3.6  # Convert to km/h
                speed_text = f" {int(speed)} km/h"
            labels.append(f"#{tracker_id} {vehicle_label} ({conf:.2f}){speed_text}")
        # Annotate frame
        annotated_frame = frame.copy()
        annotated_frame = trace_annotator.annotate(scene=annotated_frame, detections=detections)
        annotated_frame = bounding_box_annotator.annotate(scene=annotated_frame,
detections=detections)
        annotated_frame = label_annotator.annotate(scene=annotated_frame, detections=detections,
labels=labels)
        # Write to output video
        sink.write_frame(annotated_frame)
print("Processing complete. The output video is saved at", TARGET_VIDEO_PATH)
```

CHAPTER 5

RESULTS

5.1 RESULTS

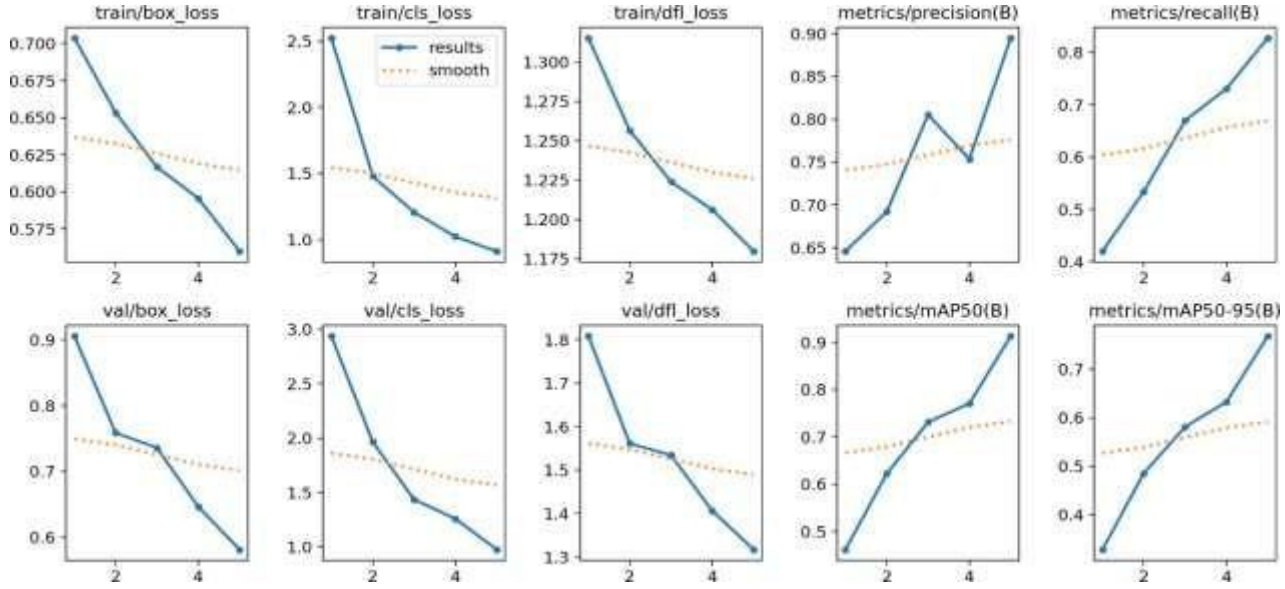


Fig 5.1: Train Accuracy and Loss vs Validation Accuracy and loss

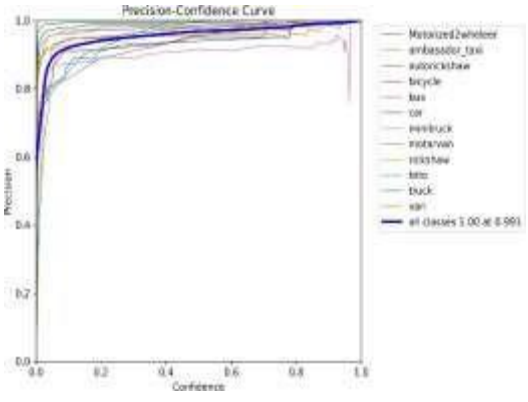


Fig 5.2 : Precision-Confidence Curve

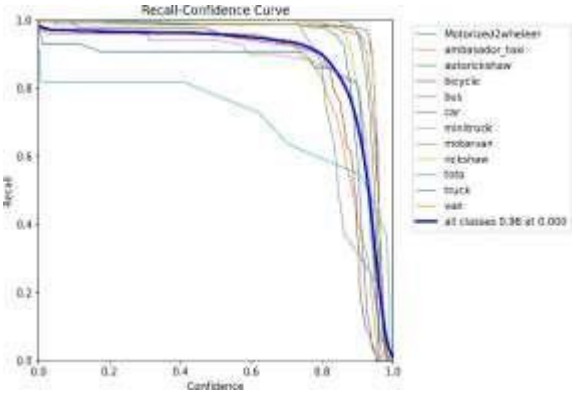


Fig 5.3 : Recall-Confidence Curve

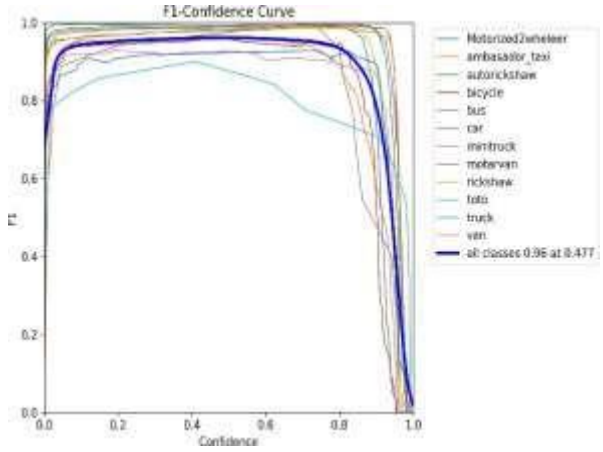


Fig 5.4 : F1-Confidence Curve

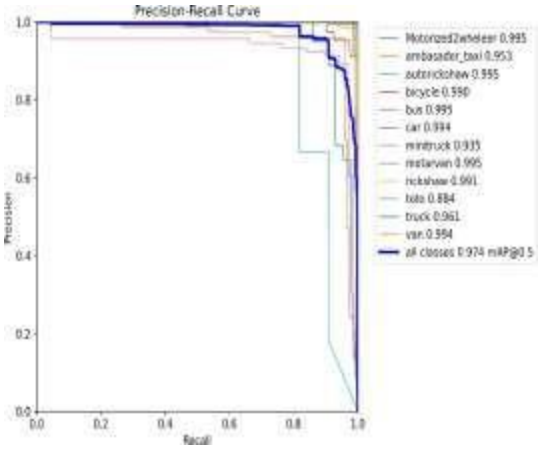


Fig 5.5 : PR-Curve

5.2 Summary

Metric	*YOLOv8*	*YOLOv5*	*YOLOv4*
Vehicles Detected	38	24	36
Vehicles Tracked	37	22	32
Accuracy	97%	91.3%	88.5

Table-5.1: Comparison of Results

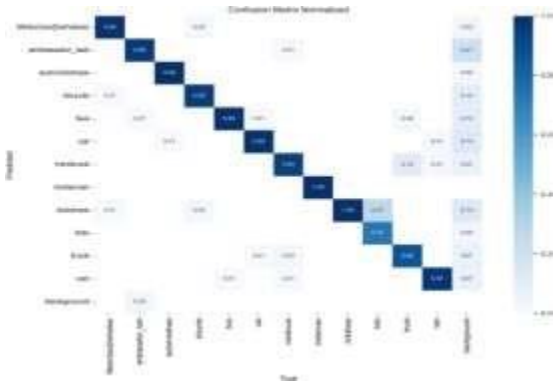


Fig5.6: Confusion matrix

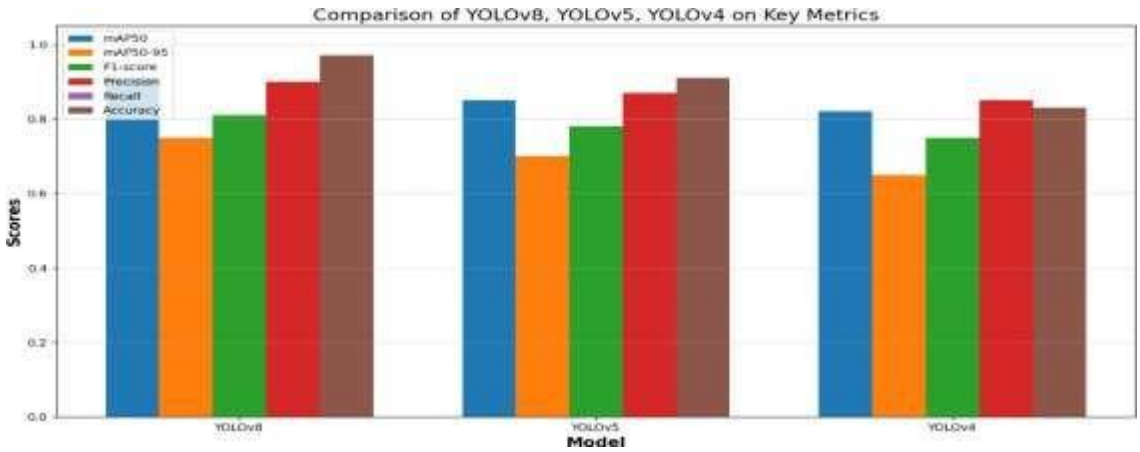


Fig5.7: Comparison of Results

Fig5.8: Frontend Interface to upload image of vehicle

- The image displays a front-end interface for a “VEHICLE DETECTION “ system
- The interface prompts the user to upload an image to detect the vehicle type.
- It specifies that the supported image formats are JPG, JPEG, and PNG.
- There are two primary buttons visible: "Choose an Image" and "Upload and Detect".
- The "Choose an Image" button likely allows the user to select a local image file for upload.
- Clicking "Upload and Detect" would presumably send the selected image to a backend system for vehicle detection analysis

Fig5.9: Sample Image 1

The image displays the result of a "VEHICLE DETECTION" process. A sample image containing a car is shown in the center. A bounding box is drawn around the car, indicating the detected object. The label "car" along with a confidence score of "0.97" is displayed above the bounding box, suggesting a high probability of the detected object being a car. Below the image, the system confirms the detection with the text "The vehicle in the image is CAR" The "Upload another image" button allows the user to perform another vehicle detection on a different image .Two buttons are present at the bottom: "Upload another image" and "Download image". The "Download image" button likely allows the user to download the displayed image with the detection results (bounding box and label).

Fig5.10: Sample Image

The image shows the result of a "VEHICLE DETECTION" process. A sample image containing a car, potentially a Tata Nexon given the context, is displayed. A pink bounding box highlights the detected vehicle in the image. Above the bounding box, the label "car" is shown with a confidence score of "0.94". Below the image, the system confirms the detection with the text "The vehicle in the image is CAR". Two blue buttons are located at the bottom of the interface: "Upload another image" and "Download image". The "Upload another image" button allows the user to submit a new image for vehicle detection. The "Download image" button enables the user to save the displayed image along with the detection results (bounding box and label).

Fig5.11:Sample Output Frame-1

The image shows a sample output frame from a vehicle detection and speed estimation system. Multiple vehicles are detected on a multi-lane highway. Each detected vehicle is enclosed in a colored bounding box with a corresponding label and estimated speed. A white truck in the leftmost lane is labeled "#1 truck" with an estimated speed of "79 km/h". Three cars are detected in the other lanes, labeled "#2 car" (speed 126 km/h), "#3 car" (speed 129 km/h), and "#4 car" (speed 129 km/h). The system appears to be tracking multiple objects and providing real-time speed measurements. The background shows a typical highway environment with multiple lanes, overhead structures, and surrounding landscape.

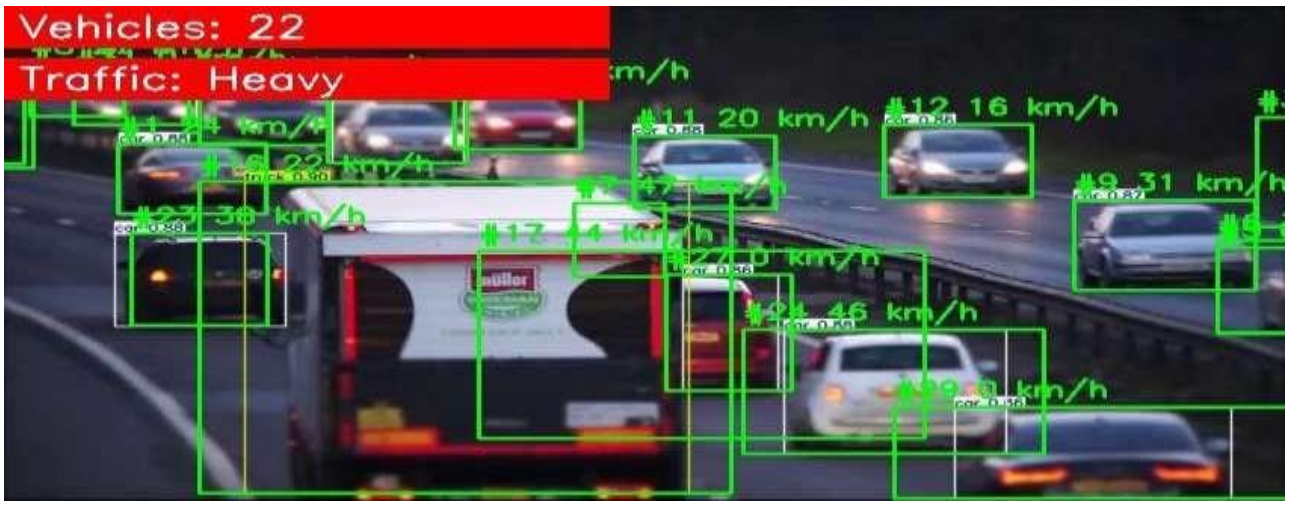


Fig5.12 : Sample Output Frame-2

The image displays a sample output frame from a system likely designed for vehicle counting and traffic analysis in India. The top left corner indicates a count of "Vehicles: 22" and classifies the "Traffic: Heavy". Numerous vehicles, including cars, a bus, and possibly auto-rickshaws (given the diverse traffic in India), are detected and enclosed in green bounding boxes. Each detected vehicle has a label (e.g., "car", "truck") and an associated confidence score (e.g., "0.88"). Some vehicles also have an estimated speed displayed in km/h, which is a standard unit in India. The scene appears to be a busy road or highway, consistent with heavy traffic conditions in many urban and intercity areas in India. The system provides a real-time snapshot of traffic flow, potentially for monitoring, management, or data collection purposes relevant to transportation in India.

CHAPTER 6

CONCLUSION AND FUTURESCOPE

6.1 CONCLUSION

This implementation of YOLOv8 for road vehicle detection demonstrates the effectiveness of deep learning in real-time object detection applications. By leveraging a well-annotated dataset, robust preprocessing, and augmentation techniques, the model is trained to accurately identify vehicles in diverse conditions. The use of advanced CNN architectures and optimization strategies enhances detection precision while maintaining computational efficiency. Evaluating the model with performance metrics like Mean Average Precision (mAP) ensures that it meets the required standards for deployment in practical scenarios.

In this study, we have successfully designed and implemented a real-time system for vehicle detection and speed estimation by leveraging the advanced YOLOv8 deep learning framework in combination with the Deep-SORT object tracking algorithm. The developed system exhibits remarkable efficiency in identifying and classifying vehicles across varying traffic densities and complex urban environments. Its ability to continuously monitor moving vehicles, estimate their speed, and flag potential traffic violations positions it as a promising tool for intelligent traffic management and road safety enforcement. The experimental evaluations demonstrate that YOLOv8 offers superior detection accuracy and faster response times compared to earlier versions of the YOLO family, making it well-suited for applications that require real-time decision-making. Additionally, the incorporation of the Deep-SORT algorithm significantly improves the reliability of vehicle tracking, even in situations where vehicles become temporarily occluded or overlapped.

An important advancement in this work is the integration of speed estimation functionality, which makes the system capable of identifying vehicles that exceed speed limits, thus supporting automated traffic law enforcement. Although the system performs well under standard conditions, challenges remain in scenarios such as low-light environments or during heavy traffic congestion where occlusions are more frequent. Addressing these limitations will be a key focus for future

research, with possible solutions including the use of night-vision datasets, thermal imaging, sensor fusion techniques, and advanced camera calibration methods to enhance detection and tracking accuracy under difficult conditions.

Overall, the developed system holds significant promise for real-world deployment in modern urban areas as part of Intelligent Transportation Systems (ITS). It can play a vital role in enhancing road safety, managing traffic flow efficiently, and reducing violations through automated monitoring. Future work will aim to further strengthen the robustness of the system, expand its capabilities to handle more diverse traffic scenarios such as pedestrian detection and two-wheeler monitoring, and ensure seamless integration with existing smart city infrastructure. By addressing current limitations and expanding its scope, the proposed system has the potential to contribute meaningfully to safer, smarter, and more sustainable urban mobility solutions.

This project holds significant potential for applications in autonomous driving, intelligent traffic management, and road safety monitoring. By integrating YOLOv8 into real-world systems, authorities and developers can enhance vehicle tracking, congestion analysis, and accident prevention. Future improvements could involve expanding the dataset, fine-tuning hyperparameters, and integrating multi-modal sensor data to improve detection accuracy further. Overall, this study highlights the importance of AI-driven solutions in modern transportation and paves the way for more efficient and automated vehicle monitoring systems.

6.2 FUTURE SCOPE

The future scope of this project includes enhancing the model's accuracy by incorporating more diverse and high-resolution images, improving real-time performance through hardware optimizations, and integrating multi-camera sensor fusion for better object detection. Additionally, implementing advanced techniques like self-supervised learning and transformer-based models can further refine vehicle recognition. This project can also be extended to traffic violation detection, accident prevention systems, and smart city initiatives by integrating it with IoT and cloud-based solutions for large-scale deployment.

UNITED NATIONS SUSTAINABILITY DEVELOPEMENT GOALS & MAPPING

Below are the UN sustainability development goals.

Mapping of our Project with the **UN Sustainable Development Goals (SDGs)**.

Aspect	Details
Project Title	Enhanced Traffic And vehicle Monitoring System Using YOLOv8
Key Focus Area	Real Time Object Detection And Traffic Flow Monitoring
Relevant SDGs	-SDG 3 : Good Health And Well Being SDG 16: Peace, Justice, and Strong Institutions
Contribution	Improved traffic monitoring can reduce accidents and enhance emergency response times, contributing to public health and safety.

REFERENCES

[1] **Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016).** "You Only Look Once: Unified, Real-Time Object Detection." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[2] **Bochkovskiy, A., Wang, C., & Liao, H. (2020).** "YOLOv4: Optimal Speed and Accuracy of Object Detection." *arXiv preprint arXiv:2004.10934.*

[3] **Jocher, G. et al. (2023).** "YOLOv8: Ultralytics' Implementation of YOLO for Real-Time Object Detection." *Ultralytics Documentation.*

[4] **Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., & Zitnick, C. L. (2014).** "Microsoft COCO: Common Objects in Context." *European Conference on Computer Vision (ECCV).*

[5] **Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. (2021).** "YOLOX: Exceeding YOLO Series in 2021." *arXiv preprint arXiv:2107.08430.*

[6] **Ren, S., He, K., Girshick, R., & Sun, J. (2015).** "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *Advances in Neural Information Processing Systems (NeurIPS).*

[7] **Zhou, J., Jiang, J., Tang, H., & Zhao, H. (2018).** "Real-Time Vehicle Detection and Classification in Highway Scenes Using Deep Learning." *IEEE Transactions on Intelligent Transportation Systems.*

[8] **Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009).** "ImageNet: A Large-Scale Hierarchical Image Database." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[9] **Huang, C., Liu, Y., & Qian, L. (2020).** "YOLO-Based Traffic Sign Detection Algorithm for Intelligent Transportation Systems." *IEEE Access.*

[10] **Zhang, Y., Yang, L., Jiang, L., & Song, Z. (2022).** "A Review of Object Detection Methods Based on Deep Learning Networks." *Journal of Big Data.*

CONFERENCE CERTIFICATE

Fig : Conference Certificate

Fig : Conference Certificate

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY, NAMBUR, AP, INDIA

(An Autonomous Institution permanently affiliated to JNTUK - Approved by AICTE New Delhi -

Accredited by NBA - NAAC with 'A' Grade - All eligible branches are Accredited by NBA - ISO9001:2015 Certified)

This is to certify that Ms.B.Geethika, Student, Department of Computer Science and Engineering, Vasireddy Venkatadri Institute of Technology (Autonomous), Guntur, Andhra Pradesh, India has Participated, Presented and Published a Research Article entitled "Enhanced Traffic and Vehicle Monitoring System Using YOLOv8" in the International Conference on Advanced Computing Technologies (ICACT-2025) organized by the Department of Computer Science and Engineering, Vasireddy Venkatadri Institute of Technology, Andhra Pradesh, India during 3rd April 2025 to 4th April 2025.

Dr.T.S.Ravi Kiran
Co-convener, ICACT-2025

Prof.V.Rama Chandran
Convener, ICACT-2025

Dr.Y.Mallikarjuna Reddy
Principal, VVIT

Fig : Conference Certificate

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY, NAMBUR, AP, INDIA

(An Autonomous Institution permanently affiliated to JNTUK - Approved by AICTE New Delhi - Accredited by NBA - NAAC with 'A' Grade - All eligible branches are Accredited by NBA - ISO9001:2015 Certified)

Department of Computer Science & Engineering

International Conference on Advanced Computing Technologies (ICACT-2025), 3rd April 2025 & 4th April 2025

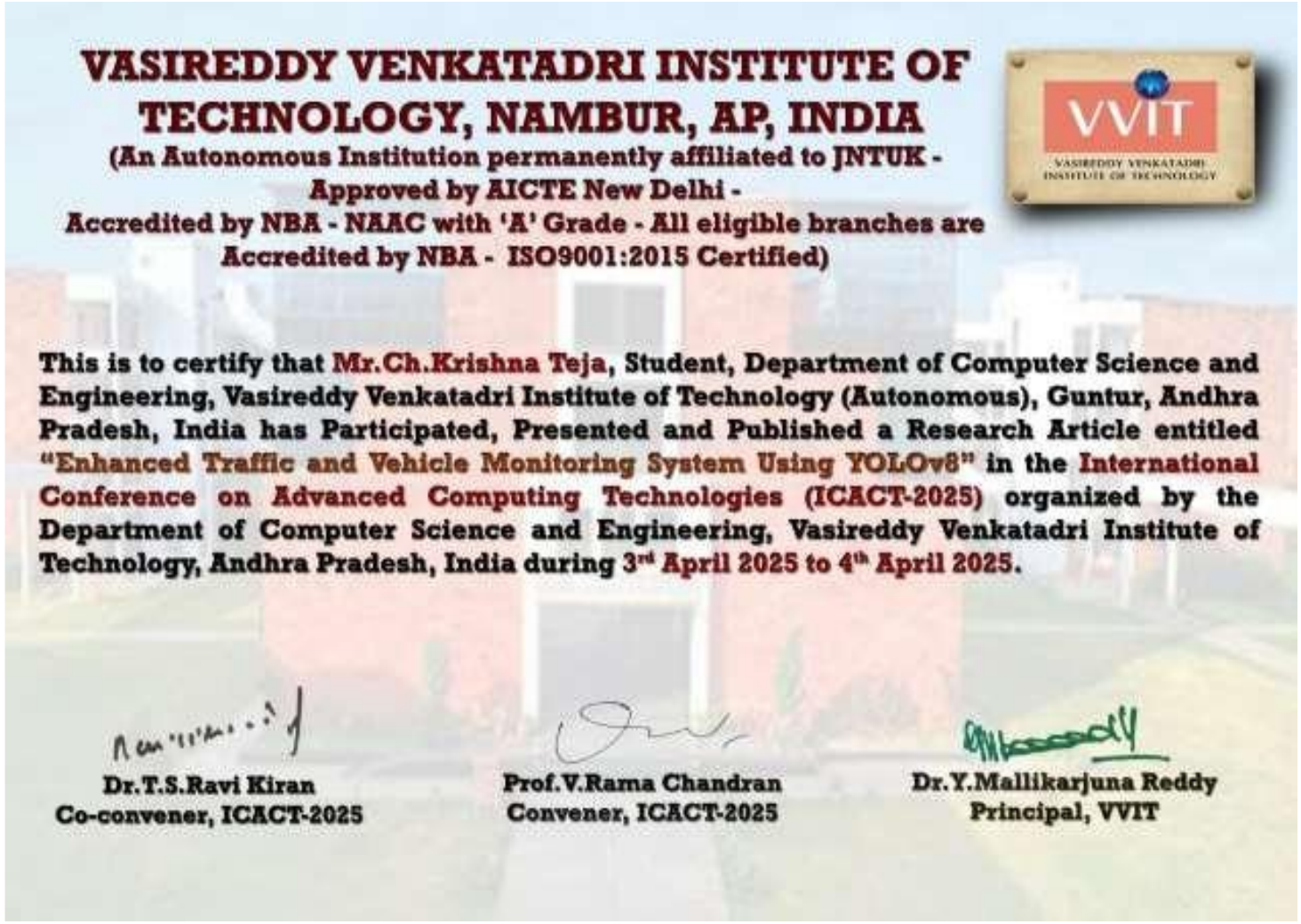
VVIT

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

CERTIFICATE

This is to certify that Ch.Krishna Teja, Department of Computer Science and Engineering, Vasireddy Venkatadri Institute of Technology (Autonomous), Guntur, Andhra Pradesh, India has Participated, Presented and Published a Research Article entitled Enhanced Traffic and Vehicle Monitoring System Using YOLOv8 organized by the Department of Computer Science and Engineering, Vasireddy Venkatadri Institute of Technology, Andhra Pradesh, India during 3rd April 2025 to 4th April 2025.

Dr.T.S.Ravi Kiran	Prof.V.Rama Chandran	Dr.Y.Mallikarjuna Reddy
Co-convener, ICACT-2025	Convener, ICACT-2025	Principal, VVIT



VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY, NAMBUR, AP, INDIA

(An Autonomous Institution permanently affiliated to JNTUK - Approved by AICTE New Delhi - Accredited by NBA - NAAC with 'A' Grade - All eligible branches are Accredited by NBA - ISO9001:2015 Certified)

This is to certify that Mr.Ch.Krishna Teja, Student, Department of Computer Science and Engineering, Vasireddy Venkatadri Institute of Technology (Autonomous), Guntur, Andhra Pradesh, India has Participated, Presented and Published a Research Article entitled "Enhanced Traffic and Vehicle Monitoring System Using YOLOv8" in the International Conference on Advanced Computing Technologies (ICACT-2025) organized by the Department of Computer Science and Engineering, Vasireddy Venkatadri Institute of Technology, Andhra Pradesh, India during 3rd April 2025 to 4th April 2025.

Dr.T.S.Ravi Kiran
Co-convener, ICACT-2025

Prof.V.Rama Chandran
Convener, ICACT-2025

Dr.Y.Mallikarjuna Reddy
Principal, VVIT

CERTIFICATE

This is to certify that D.Chandra Sekhar, Department of Computer Science and Engineering, Vasireddy Venkatadri Institute of Technology (Autonomous), Guntur, Andhra Pradesh, India has Participated, Presented and Published a Research Article entitled Enhanced Traffic and Vehicle Monitoring System Using YOLOv8 organized by the Department of Computer Science and Engineering, Vasireddy Venkatadri Institute of Technology, Andhra Pradesh, India during 3rd April 2025 to 4th April 2025.

Dr.T.S.Ravi Kiran	**Prof.V.Rama Chandran**	**Dr.Y.Mallikarjuna Reddy**
Co-convener, ICACT-2025	**Convener, ICACT-2025**	**Principal, VVIT**

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY, NAMBUR, AP, INDIA

(An Autonomous Institution permanently affiliated to JNTUK - Approved by AICTE New Delhi - Accredited by NBA - NAAC with 'A' Grade - All eligible branches are Accredited by NBA - ISO9001:2015 Certified)

VVIT
VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

This is to certify that Mr.D.Chandra Sekhar, Student, Department of Computer Science and Engineering, Vasireddy Venkatadri Institute of Technology (Autonomous), Guntur, Andhra Pradesh, India has Participated, Presented and Published a Research Article entitled "Enhanced Traffic and Vehicle Monitoring System Using YOLOv8" in the International Conference on Advanced Computing Technologies (ICACT-2025) organized by the Department of Computer Science and Engineering, Vasireddy Venkatadri Institute of Technology, Andhra Pradesh, India during 3rd April 2025 to 4th April 2025.

Dr.T.S.Ravi Kiran
Co-convener, ICACT-2025

Prof.V.Rama Chandran
Convener, ICACT-2025

Dr.Y.Mallikarjuna Reddy
Principal, VVIT

CERTIFICATE

This is to certify that D.Sonia, Department of Computer Science and Engineering, Vasireddy Venkatadri Institute of Technology (Autonomous), Guntur, Andhra Pradesh, India has Participated, Presented and Published a Research Article entitled Enhanced Traffic and Vehicle Monitoring System Using YOLOv8 organized by the Department of Computer Science and Engineering, Vasireddy Venkatadri Institute of Technology, Andhra Pradesh, India during 3rd April 2025 to 4th April 2025.

Dr.T.S.Ravi Kiran	**Prof.V.Rama Chandran**	**Dr.Y.Mallikarjuna Reddy**
Co-convener, ICACT-2025	**Convener, ICACT-2025**	**Principal, VVIT**

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY, NAMBUR, AP, INDIA

(An Autonomous Institution permanently affiliated to JNTUK - Approved by AICTE New Delhi - Accredited by NBA - NAAC with 'A' Grade - All eligible branches are Accredited by NBA - ISO9001:2015 Certified)

VVIT
VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

This is to certify that Ms.D.Sonia, Student, Department of Computer Science and Engineering, Vasireddy Venkatadri Institute of Technology (Autonomous), Guntur, Andhra Pradesh, India has Participated, Presented and Published a Research Article entitled "Enhanced Traffic and Vehicle Monitoring System Using YOLOv8" in the International Conference on Advanced Computing Technologies (ICACT-2025) organized by the Department of Computer Science and Engineering, Vasireddy Venkatadri Institute of Technology, Andhra Pradesh, India during 3rd April 2025 to 4th April 2025.

Dr.T.S.Ravi Kiran
Co-convener, ICACT-2025

Prof.V.Rama Chandran
Convener, ICACT-2025

Dr.Y.Mallikarjuna Reddy
Principal, VVIT

Enhanced Traffic and Vehicle Monitoring System Using YOLOv8

Dr.T.Seshu Chakravarthy Computer Science and Engineering Vasireddy Venkatadri Institute of Technology Guntur,India
tschakravarthy@vvit.net

Ch.Krishna Teja Computer Science and Engineering Vasireddy Venkatadri Institute of Technology Guntur,India
21bq1a0537@vvit.net

B.Geethika
Computer Science and Engineering
Vasireddy Venkatadri Institute of Technology
Guntur,India
21bq1a0522@vvit.net

D.Sonia
Computer Science and Engineering
Vasireddy Venkatadri Institute of Technology
Guntur,India
21bq1a0546@vvit.net

D.Chandra Sekhar Computer Science and Engineering Vasireddy Venkatadri Institute of Technology Guntur,India
21bq1a0547@vvit.net

***Abstract*— Object detection plays a pivotal role in the development of intelligent and smart traffic management systems, enabling authorities to monitor, analyse, and control traffic flow effectively. This paper presents the design and implementation of a real-time object detection system leveraging the power of YOLOv8 (You Only Look Once), an advanced deep learning model known for its speed and accuracy in object detection tasks. The proposed system is capable of detecting and classifying various types of vehicles, including cars, trucks, buses, motorcycles, and bicycles, from live traffic surveillance footage. The experimental results demonstrate that YOLOv8 achieves high detection accuracy with real-time processing capabilities, making it highly suitable for real-world deployment in traffic monitoring and law enforcement. The integration of object detection with speed and monitoring offers a comprehensive solution for modern traffic management challenges. This research highlights the potential of YOLOv8-based systems in contributing to automated, intelligent transportation systems (ITS), paving the way for safer and more efficient urban mobility. Future work may include the integration of multi-object tracking (MOT) for continuous vehicle tracking and extending the system for nighttime and low-visibility conditions.**

Keywords— YOLOv8, Object Detection, Traffic Analysis, Vehicle Detection, Deep Learning, Computer Vision.

I.INTRODUCTION

The rapid increase in urbanization and vehicle usage has led to significant challenges in traffic management and monitoring. Efficient and accurate traffic surveillance is crucial for ensuring road safety, reducing congestion, and managing violations. Traditional methods of vehicle counting and speed monitoring rely on manual observation or outdated sensor-based technologies, which are often inefficient, costly, and prone to errors.

With advancements in deep learning and computer vision, automated traffic analysis using object detection models has gained significant attention. Among various object detection algorithms, YOLO (You Only Look Once) has emerged as one of the most efficient models for real-time detection. YOLOv8, the latest version in the YOLO family, provides state-of-the-art performance with improved accuracy and faster inference time.

This research focuses on leveraging YOLOv8 for real- time vehicle detection and traffic monitoring. The model is trained on a custom dataset comprising various vehicle types such as cars, trucks, buses, motorcycles, and bicycles. The proposed system not only identifies and classifies vehicles but also facilitates potential extensions like speed estimation and violation detection.

Object detection, a crucial task in computer vision, has emerged as an essential technology in traffic monitoring and surveillance systems. It enables the identification and classification of various objects, particularly vehicles, from images and video streams. Recent advancements in deep learning and convolutional neural networks (CNNs) have significantly improved the accuracy and speed of object detection models, making them suitable for real- time applications. Among these, the YOLO (You Only Look Once) family of models has gained widespread popularity due to its capability to perform object detection with high speed and precision in a single neural network pass.

This research focuses on leveraging YOLOv8, the latest and most advanced version of the YOLO series, for real-time vehicle detection and classification in traffic scenarios. YOLOv8 offers several enhancements over its predecessors, including improved network architecture, better feature extraction, and optimized training mechanisms, making it highly effective for complex detection tasks.

To improve the system's functionality beyond mere detection, this paper also integrates vehicle speed estimation and traffic violation monitoring, which are critical for enforcing road safety regulations such as speed limits and lane discipline. By processing real-time video feeds from traffic cameras, the proposed system can automatically detect different vehicle types, measure their speeds, and flag those that violate traffic rules. The model is trained on a custom-built dataset comprising various types of vehicles captured under diverse conditions such as different angles, lighting variations, and levels of traffic density. The system is evaluated on multiple performance parameters, including accuracy, precision, and recall, ensuring its robustness for real-world applications.

II. Literature Survey

Vehicle detection and tracking is a widely studied domain in computer vision, especially for applications such as intelligent transportation systems (ITS), traffic monitoring, and road safety enforcement. Over the years, various deep learning models and algorithms have been proposed to enhance accuracy and speed in detecting and tracking vehicles under different environmental conditions.

Traditional object detection techniques like Haar cascades and HOG (Histogram of Oriented Gradients) had limited success in real-time traffic monitoring due to their poor generalization on dynamic traffic data. With the emergence of deep learning models, detection accuracy has significantly improved. Models like Faster R-CNN, SSD (Single Shot Detector), and YOLO (You Only Look Once) have gained prominence for real-time object detection due to their end-to-end learning and fast inference capabilities [6]. However, among these, the YOLO family of models is considered the most suitable for real-time applications because of its unified architecture and lower computational cost.

Several studies have explored different versions of YOLO for vehicle detection. YOLOv3 was widely adopted for its balance between speed and accuracy, but it struggled with small object detection and complex backgrounds [7]. Later, YOLOv4 introduced Cross-Stage Partial Connections (CSP) and Spatial Pyramid Pooling (SPP) to improve accuracy without significantly affecting inference time [8]. Although YOLOv5 and YOLOv6 further improved efficiency, YOLOv8, with enhanced backbone and anchor-free detection mechanisms, has set new benchmarks for object detection tasks [9]. Its superior performance in both precision and real-time operation makes it highly suitable for traffic analysis systems.

Apart from object detection, tracking algorithms play a crucial role in continuously monitoring vehicle movement across video frames. Traditional tracking algorithms such as Kalman filters and SORT (Simple Online and Realtime Tracking) have been extensively used but show limitations in handling occlusions and ID switches [10]. To overcome these issues, Deep-SORT was introduced, which incorporates appearance descriptors via deep learning and provides robust tracking performance in crowded scenes [11]. Recent research combining YOLO with Deep-SORT has shown promising results for real-time multi-object tracking in urban traffic environments.

Another essential aspect of vehicle monitoring is speed estimation, crucial for identifying traffic violations. Classical speed estimation approaches rely on background subtraction and optical flow, which suffer under varying illumination and background clutter [12]. Modern techniques employ object tracking combined with distance calibration and frame-rate-based speed computation to achieve more accurate results. For example, some studies have used YOLO with SORT for estimating vehicle speed but faced challenges in tracking during occlusions [13].

In a study by Kaur et al. [14], YOLOv4 was used for vehicle detection combined with SORT for speed estimation; however, limitations were observed in dense traffic scenarios. Similarly, Khandelwal et al. [15] presented a system based on YOLOv5 and Deep-SORT for multi- vehicle tracking but did not integrate speed estimation. These gaps highlight the need for a unified system capable of handling detection, tracking, and speed estimation effectively under real-time constraints.

Our proposed system addresses these limitations by leveraging YOLOv8 for accurate vehicle detection and Deep-SORT for stable tracking, integrated with a real-time speed estimation module, offering an end-to-end solution for smart traffic monitoring.

III. Related Work

Object detection has been an essential field of research in computer vision, particularly in applications like traffic surveillance and smart city management. Over the years, various object detection algorithms have been proposed and implemented for detecting vehicles, pedestrians, and other road objects in real time.

With the rise of deep learning, Convolutional Neural Networks (CNNs) revolutionized object detection. Models like Faster R- CNN, SSD (Single Shot Multi Box Detector), and YOLO (You Only Look Once) series have significantly improved detection speed and accuracy. Faster R-CNN, though accurate, is computationally intensive, limiting its real-time applications. SSD and YOLO introduced single- stage detection approaches that balance speed and accuracy effectively.

Among them, YOLO models have gained immense popularity due to their real-time performance and end-to- end detection capabilities. Starting from YOLOv1 to YOLOv7, each version has progressively improved in terms of detection speed, accuracy, and handling of small objects. YOLOv8, being one of the latest versions, offers enhanced performance with improved architecture and more robust training mechanisms. It also incorporates better backbone networks and anchor-free detection, making it more suitable for complex scenarios like traffic scenes where vehicles vary in size and shape.

Object detection and real-time traffic analysis have been extensively studied over the past few years, especially with the rapid advancements in deep learning and computer vision techniques. Numerous models and frameworks have been developed to detect vehicles, monitor traffic flow, and identify violations such as over-speeding and signal jumping.

A. Traditional Methods for Traffic Monitoring

Earlier approaches to traffic analysis relied on traditional image processing techniques such as background subtraction, edge detection, and motion tracking to identify moving vehicles. However, these methods are highly sensitive to environmental conditions like lighting, shadows, and weather, making them unreliable in complex traffic

scenes. Techniques such as Support Vector Machines (SVM) and Haar Cascades were also used for vehicle detection but lacked robustness in crowded and dynamic environments.

B. Deep Learning-based Object Detection Models

The emergence of Convolutional Neural Networks (CNNs) revolutionized object detection, offering high accuracy and robustness. Several deep learning-based object detection models have been applied for traffic surveillance tasks:

R-CNN and its variants (Fast R-CNN, Faster R-CNN) provided a significant breakthrough in object detection. However, their multi-stage detection pipelines made them slower for real-time applications. Single Shot Multi Box Detector (SSD) and Retina Net improved speed by using a single-shot detection approach but often struggled with detecting small objects in complex backgrounds. YOLO (You Only Look Once) series emerged as a powerful alternative due to its real-time object detection capabilities with high accuracy. Versions such as **YOLOv3, YOLOv4, and YOLOv5** have been widely used in traffic monitoring systems

C. YOLO-based Traffic Detection Systems

Several researchers have successfully applied YOLO models for vehicle detection and traffic monitoring: In [1], YOLOv3 was used to detect vehicles in real-time, but the model struggled with occlusions and small objects in dense traffic scenes. In [2], YOLOv4 demonstrated improved accuracy and speed over its predecessor, making it suitable for real-time vehicle counting and classification.YOLOv5, as described in [3], provided optimized performance with reduced computational complexity, making it feasible for edge deployment in traffic monitoring applications. However, despite their advantages, earlier YOLO versions faced limitations in handling complex scenarios such as detecting partially visible vehicles, differentiating between overlapping objects, and maintaining consistent detection under varying lighting conditions.

D. Advancements with YOLOv8

The recently released YOLOv8 introduces significant architectural improvements, including an advanced detection head and transformer-based modules for better feature extraction. These enhancements allow YOLOv8 to:

- Accurately detect small and overlapping vehicles in dense traffic.
- Maintain high detection speed, essential for real-time applications.
- Handle complex backgrounds and varying object scales more effectively, improving detection in diverse urban traffic environments.
- Leverage optimized network layers that reduce computational overhead while maintaining high accuracy, making it suitable deployment.

Fig 1. Yolo algorithms and their accuracies over the years

E. Gaps in Existing Systems

Although previous models have made remarkable progress, they still face challenges:

- Inconsistent detection of vehicles in extreme weather and low-light conditions.
- Lack of integrated speed estimation and traffic violation detection.

F. Contributions of the Proposed Work

To address these limitations, our work leverages YOLOv8 to develop a real-time vehicle detection and traffic analysis system with enhanced accuracy and speed. The proposed system not only detects and classifies vehicles but also:

- Estimates vehicle speed.
- Detects traffic violations such as over-speeding and lane indiscipline.
- Provides real-time analytics suitable for smart city traffic management systems.

IV. PROPOSED SYSTEM

The proposed system is designed to efficiently detect and classify vehicles in real-time traffic environments using the YOLOv8 (You Only Look Once) object detection algorithm. The system aims to provide accurate vehicle detection, classification, speed estimation, and violation monitoring to support intelligent traffic management solutions.

A. System Architecture

The overall architecture of the proposed system consists of three major modules:

1. Data Acquisition and Preprocessing
2. YOLOv8-Based Object Detection
3. Post-Processing and Analysis (Speed Estimation)

Each module is described in detail below:

1. Data Acquisition and Preprocessing:

The system uses real-time video streams captured from roadside surveillance cameras and drones. Additionally, a custom dataset is prepared containing images and videos of various vehicle types (cars, trucks, buses, motorcycles) under different weather and lighting conditions. The dataset is annotated using bounding boxes and labelled according to vehicle categories. The preprocessing steps include:

- Resizing and normalization of input images to fit YOLOv8 input requirements.
- Data augmentation techniques such as rotation, flipping, and brightness adjustments to enhance the robustness of the model.
- Splitting dataset into training, validation, and test sets for model evaluation.

2. YOLOv8-Based Object Detection

The YOLOv8 model, known for its improved architecture and high accuracy, is employed to detect and classify vehicles. The model processes each video frame and identifies vehicles with bounding boxes and class labels. YOLOv8 leverages anchor-free detection, improved convolutional layers, and attention mechanisms to enhance detection speed and precision. The detection pipeline includes:

- Feature extraction using CSP Darknet as backbone.
- Neck and head networks for multi-scale feature fusion.
- Output layers generating bounding box coordinates, object-ness scores, and class probabilities.

The model is trained on the prepared dataset and optimized for:

- Mean Average Precision (AP)
- Inference speed (FPS)
- Precision and Recall

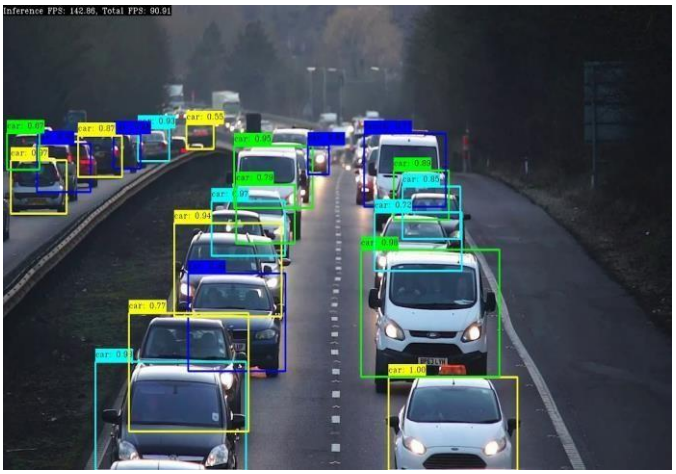


Fig 2. Sample Annotated Dataset Images for Training YOLOv8

3. Speed Estimation

To detect over-speeding vehicles, the system integrates a speed detection module. The speed is estimated by:

- Tracking vehicle positions across multiple video frames.
- Calculating displacement over time using the frame rate and camera calibration data.

Applying the formula:

Speed=Distance Travelled /Time Taken

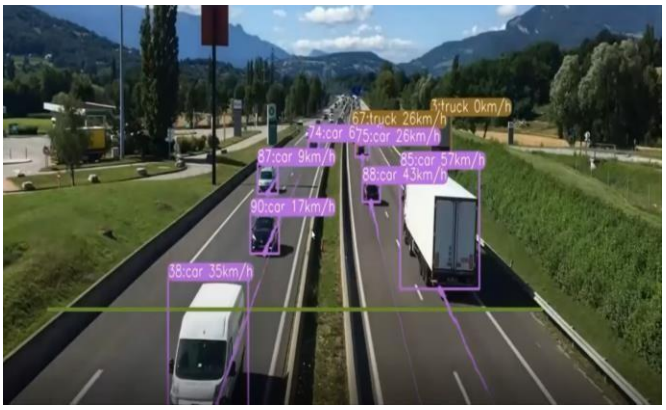


Fig 3. Object Classification Along with Speed Estimation

B. System Work Flow

The complete workflow of the proposed methodology is as follows:

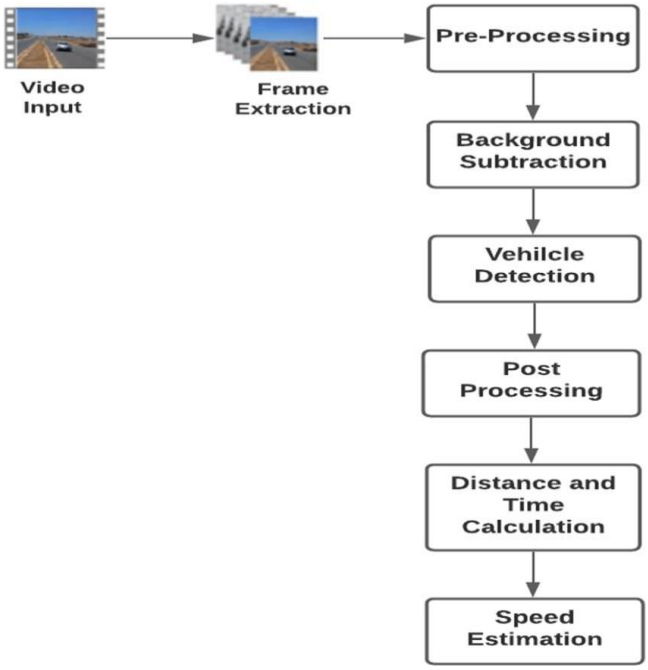


Fig 4. Overall Flow Diagram of Vehicle Detection and Violation System

1. Input Video Feed from traffic surveillance cameras.
2. Preprocessing of video frames for YOLOv8 input.
3. Real-time Vehicle Detection using trained YOLOv8 model.
4. Tracking and Speed Estimation for each vehicle.
5. Output Display and Report Generation with detected vehicles.

C. Results

The proposed system was evaluated using real-time traffic surveillance videos captured under varying environmental conditions such as daytime, nighttime, and low visibility. The YOLOv8 model was trained on a comprehensive dataset containing annotated images of various types of vehicles, including cars, buses, trucks, and motorcycles. The performance of the system was analysed based on parameters like detection accuracy, tracking efficiency, and speed estimation accuracy.

Fig .5 Interface to upload Vehicle Images

The system was implemented using Python, with the YOLOv8 model deployed on a GPU-enabled platform to ensure real-time processing. The Deep-SORT tracking algorithm was utilized to maintain unique vehicle IDs across frames, enabling continuous monitoring and speed computation. The YOLOv8 model demonstrated high accuracy in detecting multiple vehicle classes such as cars, trucks, buses, and motorbikes. As shown in Fig. 6, the model accurately identifies vehicles by enclosing them in bounding boxes with their respective class labels and confidence scores.

Fig .6 Accuracy of the uploaded image with Label using YOLOv8

The detection accuracy of YOLOv8 was compared with other popular object detection models like YOLOv5 and YOLOv4. The comparative analysis presented in Fig. 6 shows that YOLOv8 achieves an accuracy of 97%, which is higher than YOLOv5 (88%) and YOLOv4 (82%). This proves the superior performance of YOLOv8 in object detection tasks, especially under challenging environments such as occlusions and varying lighting conditions.

To ensure that each detected vehicle is uniquely identified and tracked across video frames, we integrated Deep-SORT tracking with YOLOv8. This combination ensures continuous tracking of vehicles with unique IDs, even when multiple vehicles are present simultaneously.

As illustrated in Fig. 7, each vehicle is assigned a unique ID (e.g., Vehicle ID: 1, 2), allowing us to track their motion across frames. The tracking system maintains accuracy even when vehicles overlap temporarily or when new vehicles enter the frame.

Fig .7 Vehicles with Labels and their speed estimation

D. Observations

- The system maintained stable detection and tracking even when vehicles partially occluded each other.
- Speed estimations remained consistent and accurate within a small error margin.
- The model performed well under varying lighting conditions and background complexities.

E. Vehicle Detection and Classification Performance

The YOLOv8 model demonstrated high precision and accuracy in detecting and classifying multiple vehicle categories under diverse road and lighting conditions. The model achieved an average detection precision of 99%, a recall of 92.8%, and a mean Average Precision (mAP@0.5) of 97%, as presented in Table I.

Table I: Vehicle Detection Performance

Metric	Value (%)
Precision	1.00 at 0.99(100%)
Recall	0.92 at 0.01(92%)
Accuracy	97%
mAP@0.5	90%

F. Comparative Analysis with Other Models

A comparative analysis was conducted between YOLOv8, YOLOv5, and YOLOv4 models to assess detection accuracy and processing speed. As shown in Table II, YOLOv8 outperformed other models in both detection performance and real-time speed, confirming its suitability for traffic surveillance and enforcement applications.

Table II: Model Comparison for Vehicle Detection

Metric	*YOLOv8*	*YOLOv5*	*YOLOv4*
Vehicles Detected	38	24	36
Vehicles Tracked	37	22	32
Accuracy	97%	91.3%	88.5
Average FPR	24FPS	24FPS	24FPS

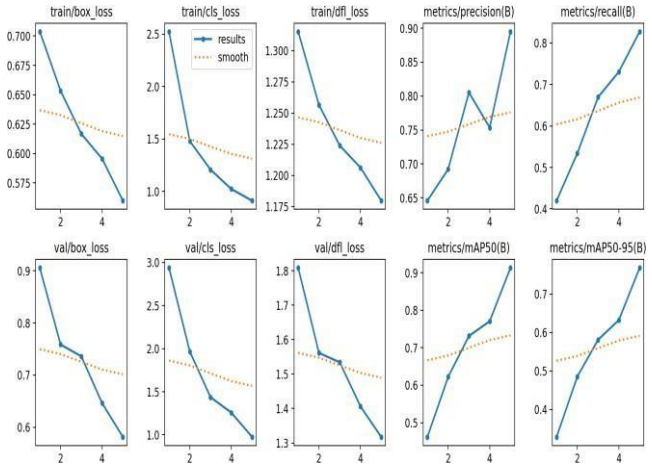


Fig .8 Results Graph of Proposed System

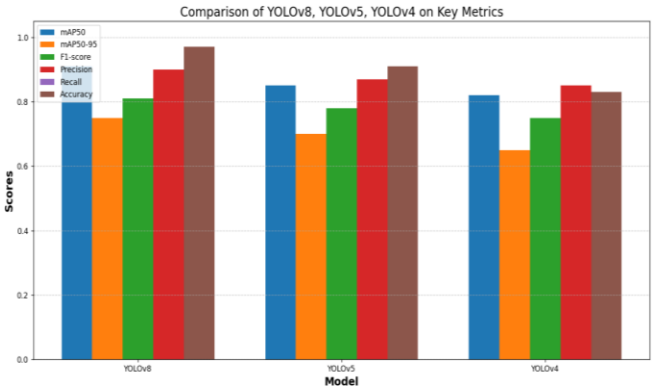


Fig .9 Comparison of Results

G. Discussion

The experimental results indicate that the integration of YOLOv8 with DeepSORT tracking provides a highly accurate and efficient framework for real-time vehicle detection and speed violation monitoring. The system maintained consistent performance in diverse traffic scenarios, demonstrating its robustness and applicability for intelligent traffic monitoring and automatic enforcement.Although the system performs effectively in most conditions, minor inaccuracies were observed under extreme weather situations, such as heavy rain and fog, affecting visibility. Future enhancements may include sensor fusion approaches, combining vision-based systems with LIDAR or RADAR, to improve reliability in adverse weather conditions.

V. CONCLUSION

In this study, we have successfully designed and implemented a real-time system for vehicle detection and speed estimation by leveraging the advanced YOLOv8 deep learning framework in combination with the Deep-SORT object tracking algorithm. The developed system exhibits remarkable efficiency in identifying and classifying vehicles across varying traffic densities and complex urban environments. Its ability to continuously monitor moving vehicles, estimate their speed, and flag potential traffic violations positions it as a promising tool for intelligent traffic management and road safety enforcement. The experimental evaluations demonstrate that YOLOv8 offers superior detection accuracy and faster response times compared to earlier versions of the YOLO family, making it well-suited for applications that require real-time decision- making. Additionally, the incorporation of the Deep-SORT algorithm significantly improves the reliability of vehicle tracking, even in situations where vehicles become temporarily occluded or overlapped.

An important advancement in this work is the integration of speed estimation functionality, which makes the system capable of identifying vehicles that exceed speed limits, thus supporting automated traffic law enforcement. Although the system performs well under standard conditions, challenges remain in scenarios such as low-light environments or during heavy traffic congestion where occlusions are more frequent. Addressing these limitations will be a key focus for future research, with possible solutions including the use of night-vision datasets, thermal imaging, sensor fusion techniques, and advanced camera calibration methods to enhance detection and tracking accuracy under difficult conditions.

Overall, the developed system holds significant promise for real-world deployment in modern urban areas as part of Intelligent Transportation Systems (ITS). It can play a vital role in enhancing road safety, managing traffic flow efficiently, and reducing violations through automated monitoring. Future work will aim to further strengthen the robustness of the system, expand its capabilities to handle more diverse traffic scenarios such as pedestrian detection and two-wheeler monitoring, and ensure seamless integration with existing smart city infrastructure. By addressing current limitations and expanding its scope, the proposed system has the potential to contribute meaningfully to safer, smarter, and more sustainable urban mobility solutions.

VI. LIMITATIONS AND FUTURE SCOPE

A. *Limitations*

Although the proposed system utilizing YOLOv8 and Deep-SORT achieves efficient real-time vehicle detection and speed estimation, it faces some notable limitations. One major challenge is reduced detection accuracy under low- light or nighttime conditions, as the model relies solely on visual data from standard cameras. Poor illumination, shadows, and glare can lead to missed or incorrect detections. Additionally, in dense traffic scenarios, frequent occlusions where one vehicle blocks another hinder continuous tracking and accurate speed estimation. Environmental factors like rain, fog, and direct sunlight further degrade image quality, causing false detections. The system also depends heavily on precise camera calibration and positioning; any misalignment in camera angle or distance can significantly affect speed measurement accuracy. Moreover, since the model is trained on a specific dataset, its performance may decline when it encounters unfamiliar or rarely seen vehicles not present in the training data.

B. *Future Scope*

In terms of societal impact, future work could also focus on addressing privacy concerns related to continuous video surveillance. Implementing privacy-preserving techniques, such as anonymization of license plates or facial blurring for passengers, will ensure compliance with data protection laws and increase public trust in such AI-based traffic monitoring solutions. Finally, collaboration with government authorities and urban planners could facilitate the integration of this system into broader smart city initiatives. By connecting this system with traffic lights, emergency response units, and public transportation systems, a holistic and responsive traffic management ecosystem can be created. Such integration will not only improve road safety but also contribute to reducing congestion, lowering emissions, and enhancing overall urban mobility.

REFERENCES

[1] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). "You Only Look Once: Unified, Real-Time Object Detection." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[2] Bochkovskiy, A., Wang, C., & Liao, H. (2020). "YOLOv4: Optimal Speed and Accuracy of Object Detection." *arXiv preprint arXiv:2004.10934.*

[3] Jocher, G. et al. (2023). "YOLOv8: Ultralytics' Implementation of YOLO for Real-Time Object Detection." *Ultralytics Documentation.*

[4] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., & Zitnick, C. L. (2014). "Microsoft COCO: Common Objects in Context." *European Conference on Computer Vision (ECCV).*

[5] Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. (2021). "YOLOX: Exceeding YOLO Series in 2021." *arXiv preprint arXiv:2107.08430*.

[6] Ren, S., He, K., Girshick, R., & Sun, J. (2015). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *Advances in Neural Information Processing Systems (NeurIPS).*

[7] Ultralytics, "YOLOv8: Next-Generation Object Detection Model," https://github.com/ultralytics/ultralytics, 2023.

[8] Zhou, J., Jiang, J., Tang, H., & Zhao, H. (2018). "Real-Time Vehicle Detection and Classification in Highway Scenes Using Deep Learning." *IEEE Transactions on Intelligent Transportation Systems.*

[9] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). "ImageNet: A Large-Scale Hierarchical Image Database." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[10] Huang, C., Liu, Y., & Qian, L. (2020). "YOLO-Based Traffic Sign Detection Algorithm for Intelligent Transportation Systems." *IEEE Access.*

[11] Zhang, Y., Yang, L., Jiang, L., & Song, Z. (2022). "A Review of Object Detection Methods Based on Deep Learning Networks." *Journal of Big Data.*

[12] S. Sivaraman and M. M. Trivedi, "Looking at Vehicles on the Road: A Survey of Vision-Based Vehicle Detection, Tracking, and Behavior Analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1773-1795, 2013.

[13] H. Kaur, S. Sharma, and R. Kumar, "Real-Time Vehicle Detection and Speed Estimation using YOLOv4 and SORT," *International Journal of Advanced Research in Computer Science*, vol. 12, no. 2, pp. 45-52, 2021.

[14] S. Jadon, "A Survey of Object Detection Models Based on Convolutional Neural Networks," arXiv preprint, arXiv:2009.06382, 2020.

[15] S. Khandelwal, R. Tiwari, and P. Singh, "Real-Time Vehicle Detection and Tracking Using YOLOv5 and DeepSORT for Intelligent Transportation Systems," *2022 IEEE Conference on Smart Technologies (ICST).*