A Project Report

on

AUTONOMOUS TRAFFIC MANAGEMENT SYSTEM

submitted for partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

by

P. RAGHAVENDRA	**-**	**21BQ1A05G8**
N. TEJA VISHNUVARDHAN REDDY	**-**	**21BQ1A05F7**
P. SATYA SAI SREE RAMA KOUSHIK	**-**	**21BQ1A05H7**
SK. BASHA	**-**	**22BQ5A0521**

Under the guidance of

Mr. M. JEEVAN BABU

Assistant Professor

(Autonomous)

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, 2025.

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY
(Autonomous)
Permanently Affiliated to JNTU, Kakinada, Approved by AICTE
Accredited by NAAC with 'A' Grade, ISO 9001:2015 Certified
Nambur, Pedakakani (M), Guntur (Dt) - 522508
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
B.Tech Programme Accredited by NBA

CERTIFICATE

This is to certify that the project report titled **"AUTONOMOUS TRAFFIC MANAGEMENT SYSTEM"** is being submitted by, **Mr. P. RAGHAVENDRA, Mr. N. TEJA VISHNUVARDHAN REDDY, Mr. P. SATYA SAI SREE RAMA KOUSHIK, and Mr. SK. BASHA**, bearing Registered Numbers **21BQ1A05G8, 21BQ1A05F7, 21BQ1A05H7, and 22BQ5A0521** respectively of 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.

Project Guide
(Mr. M. Jeevan Babu, Asst. Professor)

Head of Department
(Dr. V. Ramachandran, Professor)

Submitted for Viva-voice Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We, **Mr. P. RAGHAVENDRA, Mr. N. TEJA VISHNUVARDHAN REDDY, Mr. P. SATYA SAI SREE RAMA KOUSHIK, and Mr. SK. BASHA**, hereby declare that the Project Report entitled "AUTONOMOUS TRAFFIC MANAGEMENT SYSTEM" done by us under the guidance of **Mr. M. JEEVAN BABU, Assistant Professor** at **Vasireddy Venkatadri Institute of Technology** is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science & Engineering. The results embodied in this report have not been submitted to any other university for the award of any degree.

DATE :

PLACE :

SIGNATURE OF THE CANDIDATE(s)

1.

2.

3.

4.

ACKNOWLEDGEMENT

We 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 our ideas and helped us towards the successful completion of this project.

First and foremost, we express our deep gratitude to **Sri. Vasireddy Vidya Sagar**, Chairman, Vasireddy Venkatadri Institute of Technology for providing necessary facilities throughout the B. Tech program.

We express our sincere thanks to **Dr. Y. Mallikarjuna Reddy**, Principal, Vasireddy Venkatadri Institute of Technology for his constant support and cooperation throughout the B. Tech program.

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

We would like to express our sincere gratefulness to our Guide **Mr. M. Jeevan Babu**, Assistant Professor, CSE for his insightful advice, motivating suggestions, invaluable guidance, help and support in the successful completion of this project.

We would like to express our sincere heartfelt thanks to our Project Coordinator **Dr. N. Sri Hari**, Professor, CSE for his valuable advices, motivating suggestions, moral support, help and coordination among us in successful completion of this project.

We would like to take this opportunity to express our thanks to all the **Teaching and Non-Teaching** Staff in the Department of Computer Science & Engineering, VVIT for their invaluable help and support.

Name(s) of Students

P. RAGHAVENDRA	-	21BQ1A05G8
N. TEJA VISHNUVARDHAN REDDY	-	21BQ1A05F7
P. SATYA SAI SREE RAMA KOUSHIK	-	21BQ1A05H7
SK. BASHA	-	22BQ5A0521

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)
CO5	Perform the role of a team member or lead in SDLC. (K3)

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

INDEX

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Traffic management is a growing problem in many cities leading to accidents, high delays, and increasing air pollution. Generally, traffic is managed manually by traffic policemen or with fixed traffic light times. This research will provide an autonomous method by selecting the efficient method among RCNN, YOLO V7 & YOLO V8. The detection speed by using the mentioned methods is 20 FPS, 40 FPS & 45 FPS respectively. Based on the detection speeds, YOLO V8 is the most efficient method among them. The ambulance custom dataset is used to train YOLO V8 for ambulance detection. The trained model attained an average precision of 99.1% and an accuracy of 97.7%. Overall, the research will manage the traffic based on the vehicle count of each lane. It will also prioritize the lane for ambulances.

Keywords: YOLO, Ambulance detection, Smart traffic management system, Vehicle count detection, COCO dataset.

CHAPTER – 1

INTRODUCTION

The traditional traffic management systems are manually controlled by traffic policemen and by fixed traffic light cycles. Both of them have several limitations. Fixed signal timings will not work out for real-time traffic flow, it may cause unnecessary delays and an increase in accident count. On the other hand, manual traffic control will require a lot of manpower. To overcome these drawbacks this research will provide a way to manage traffic autonomously by utilizing the efficient method.

With the help of several deep learning methods like Region-Based Convolutional Neural Networks (RCNN), You Only Look Once (YOLO) V7, and YOLO V8 vehicle detection will be achieved. This deep-learning vehicle detection will detect the vehicles with different class labels and draw the vehicle count. The main advantage of this approach is to improve traffic management in real time and reduce accidents.

This research will demonstrate the efficient model among RCNN, YOLO V7 & YOLO V8 for vehicle detection. Our study evaluates these models based on the detection speed which is an important aspect of traffic management. The detection speed for the RCNN model is 20 FPS, the YOLO V7 model is 40 FPS and the YOLO V8 model is 45 FPS. Based on this output, YOLO V8 is the efficient model for vehicle detection.

Additionally, this research implemented ambulance detection by further training the YOLO V8 model with a custom ambulance dataset. The performance metrics of ambulance detection are 99.1% average precision and 97.7% accuracy. This detection will help to prioritize the ambulance-detected lane.

The proposed system will allow the green time based on the vehicle count. If an ambulance is detected in any lane, then the signals will pause and prioritize the ambulance lane after 30 seconds the regular signal will resume. By integrating this autonomous traffic management system, we can ensure the safety and minimize the delays.

CHAPTER – 2

AIM AND SCOPE

The aim of the project “Autonomous Traffic Management System” is to develop an autonomous traffic management system that improves real-time vehicle detection and ambulance prioritization using deep learning models. This study evaluates RCNN, YOLO V7, and YOLO V8 based on detection speeds to identify the efficient model for vehicle detection. Additionally, a custom ambulance dataset is used to train YOLO V8 for ambulance detection to enable automatic traffic signal adjustments and ensure emergency vehicles receive priority. The scope of this project is to develop an autonomous traffic management system using deep learning models to improve real-time vehicle detection and ambulance prioritization by dynamically adjusting traffic signals based on vehicle density and emergency detection.

2.1 EXISTING SYSTEMS

Existing traffic management systems primarily rely on fixed-time traffic signals and manual interventions by traffic police, which often lack adaptability to real-time traffic conditions. Traditional systems use pre-defined signal cycles that do not consider actual vehicle density, leading to unnecessary delays and congestion. Manual traffic control requires significant human resources and is prone to inefficiencies, especially during peak hours. While some cities have adopted vehicle sensors and basic automation, these systems still struggle to dynamically adjust traffic signals based on real-time conditions, making them ineffective in optimizing road efficiency.

To improve traffic flow, some urban areas have implemented smart traffic management systems that utilize camera-based surveillance and loop detectors to monitor congestion. These systems offer better traffic regulation but remain expensive to deploy and maintain. Moreover, their performance is often affected by environmental factors such as poor lighting or adverse weather conditions. While AI-driven traffic monitoring solutions have been explored, their implementation is still limited due to computational demands, infrastructure constraints, and integration challenges with traditional traffic control systems.

Although research initiatives have demonstrated the potential of deep learning models for traffic management, widespread adoption remains a challenge. Many existing solutions do not prioritize emergency vehicles such as ambulances, leading to critical delays in response times. A fully autonomous traffic management system powered by advanced AI models could effectively address these shortcomings by optimizing traffic signals in real time and ensuring priority for emergency vehicles. The development of such a system would significantly reduce congestion, improve road safety, and enhance overall urban mobility.

An autonomous traffic system for emergency vehicles is presented by Mamoona Humayun et al. (2022)[1]. This project proposed an Emergency Vehicle Management Solution (EVMS) that prioritizes emergency vehicles by dynamically creating space in adjacent lanes. Mathematical modeling results show that EVMS significantly reduces emergency vehicle travel times without disrupting normal traffic performance.

Pankaj Kunekar et al. (2024)[2] utilized the YOLO algorithm for the traffic management system. This study integrates computer vision and machine learning using the YOLO object detection system to optimize traffic signal phases based on queue density and vehicle waiting time.

AI-based autonomous traffic regulation by Sreelatha R et al. (2023)[3] utilizes sensors, cameras, and communication networks to collect real-time traffic data and then analyzed it by AI algorithms to optimize traffic flow. By leveraging machine learning, deep learning, and reinforcement learning, the system can adapt dynamically to changing traffic conditions and contribute to the development of intelligent and sustainable transportation systems.

Nitin Sakhare et al. (2024)[4] proposed an IoT-driven adaptive traffic management system using YOLOv3 and image processing to detect vehicles and optimize signal light patterns based on lane-specific traffic density.

Christofel Rio Goenawan (2024)[6] presents an Autonomous Smart Traffic Management (ASTM) system that leverages AI, utilizing the YOLO V5 Convolutional Neural Network for vehicle detection and an RNN-LSTM model to predict traffic for the next 12 hours. By optimizing traffic cycle length based on these predictions, the system significantly improves traffic flow, achieving a 50% increase in vehicle throughput and a 70% reduction in vehicle pass delays, as demonstrated in the CARLA simulation environment.

2.2 PROPOSED SYSTEMS

The proposed methods for this project are various advanced object detection methods, such as RCNN (Region-based Convolutional Neural Networks), YOLOv7, and YOLOv8. RCNN is well known for its high accuracy in object detection, but its main drawback is its slow processing speed. Since RCNN processes regions of interest one at a time, it becomes inefficient for real-time applications like traffic management systems. Due to this limitation, alternative methods that balance both speed and accuracy are preferred for traffic monitoring.

YOLO (You Only Look Once) is a widely recognized object detection algorithm known for its exceptional speed and accuracy, making it an ideal choice for real-time applications such as traffic surveillance and autonomous driving. Unlike RCNN, YOLO processes an entire image in a single pass through the neural network, allowing it to detect multiple objects almost instantaneously. This efficiency makes YOLO highly suitable for dynamic environments where quick responses are necessary, such as adjusting traffic signals based on vehicle density or detecting congestion in real time.

Among the various YOLO versions, YOLOv7 is particularly notable for its balance between robustness and ease of implementation. YOLOv8 further enhances speed and precision, making it even more efficient for vehicle detection in traffic systems. The continuous advancements in YOLO models help improve the accuracy of object detection while maintaining high processing speeds, ultimately leading to smarter and more responsive traffic management solutions. Exploring these models can help determine the most suitable approach for optimizing traffic flow and enhancing road safety.

After selecting the most efficient model among the three, the next step involves training it with a custom ambulance dataset to enhance its ability to detect emergency vehicles accurately. This specialized training ensures that the model can distinguish ambulances from other vehicles in real-time traffic scenarios by allowing for automatic lane prioritization. By leveraging this customized detection system, traffic management systems can dynamically adjust signals and clear paths for ambulances and significantly reduce response times during emergencies. This approach not only improves the efficiency of emergency vehicle movement but also enhances overall road safety by minimizing disruptions to regular traffic flow.

Table 2.1: Comparison of Accuracies

Base paper	Model and accuracies	Our models and accuracies
Pankaj Kunekar, Yogita Narule, Richa Mahajan, Shantanu Mandlapure, Eshan Mehendale and Yashashri Meshram[2]	YOLOv7(40 FPS)	YOLOv8(45 FPS) YOLOv7(40 FPS) RCNN(20 FPS)
Dr. Sreelatha R, Mahalakshmi B S, Riya Yadav, Shreyam Pandey, Vandit Agarwal[3]	YOLOv2(25 FPS)	
Nitin Sakhare, Mrunal Hedau, Gokul B., Omkar Malpure, Trupti Shah, Anup Ingle[4]	YOLOv3(30 FPS)	

CHAPTER – 3

CONCEPT AND METHODS

3.1 PROBLEM DESCRIPTION

Deep learning algorithms like YOLO have been extensively used for vehicle detection in complex traffic environments. Most existing research primarily focuses on improving accuracy or identifying specific vehicle types, but little attention has been given to optimizing these systems for live traffic management. Urban areas experience variable traffic conditions that pose significant challenges for current detection models. Additionally, integrating these deep learning models with existing traffic infrastructure has not been fully explored.

Traditional traffic management systems rely on fixed-timing signals or manual control, making them inefficient in adapting to real-time traffic conditions. This results in frequent congestion, increased travel time, and delays in emergency response. The inability of these systems to dynamically adjust traffic lights based on actual vehicle volume further exacerbates these problems.

To address these challenges, this project proposes a real-time autonomous traffic management system using YOLOv8, designed to optimize traffic signal control based on live traffic data. By dynamically adjusting signal timings according to real-time traffic flow, this system aims to reduce congestion, improve road efficiency, and enhance emergency response times. The integration of deep learning with traffic management infrastructure has the potential to revolutionize urban mobility and make roads safer and more efficient while laying the groundwork for future smart city advancements.

3.2 PROPOSED SOLUTION

To develop an efficient Autonomous Traffic Management System, we applied various deep learning models to optimize traffic control. Using COCO dataset, we explored advanced object detection models such as YOLOv7, YOLOv8, and RCNN to identify and count vehicles in real-time and selected the most accurate model for deployment. For ambulance detection and

lane prioritization, we trained YOLOv8 model with a custom ambulance dataset to ensure emergency vehicles receive priority access.

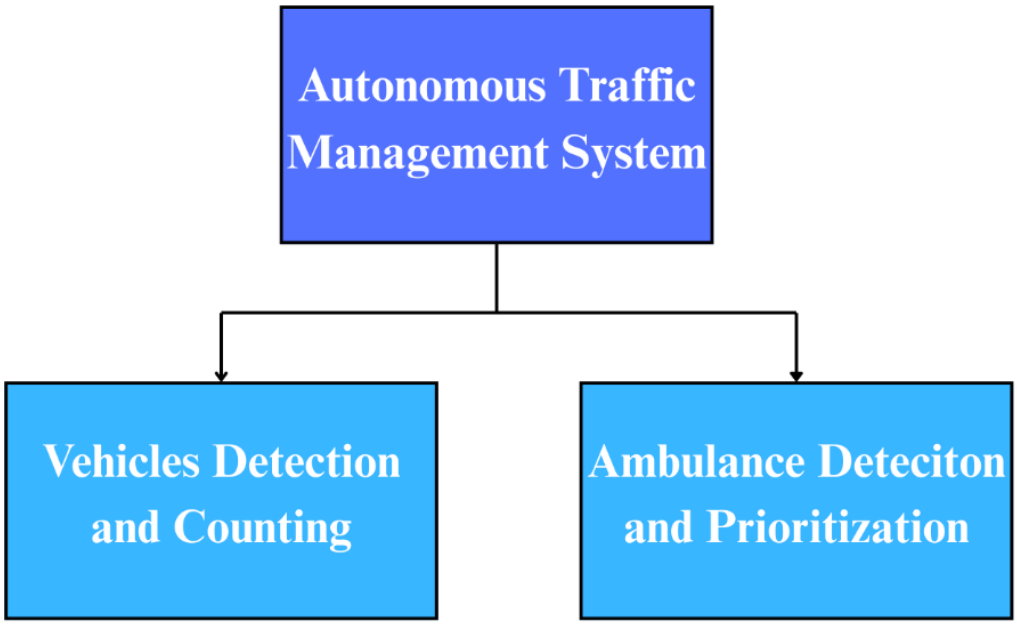

Fig 3.1: Project structure

The system continuously monitors live CCTV feeds from all lanes to analyze real-time traffic conditions and optimize signal timings accordingly. It prioritizes traffic signals based on the vehicle count in each lane and ensures that lanes with higher congestion receive longer green light durations to improve traffic flow. Additionally, the system is equipped with ambulance detection capabilities that allow it to identify emergency vehicles in any lane using a custom-trained deep-learning model. Whenever an ambulance is detected, the system immediately prioritizes that lane by adjusting the traffic signals and ensures the ambulance can pass through with minimal delay.

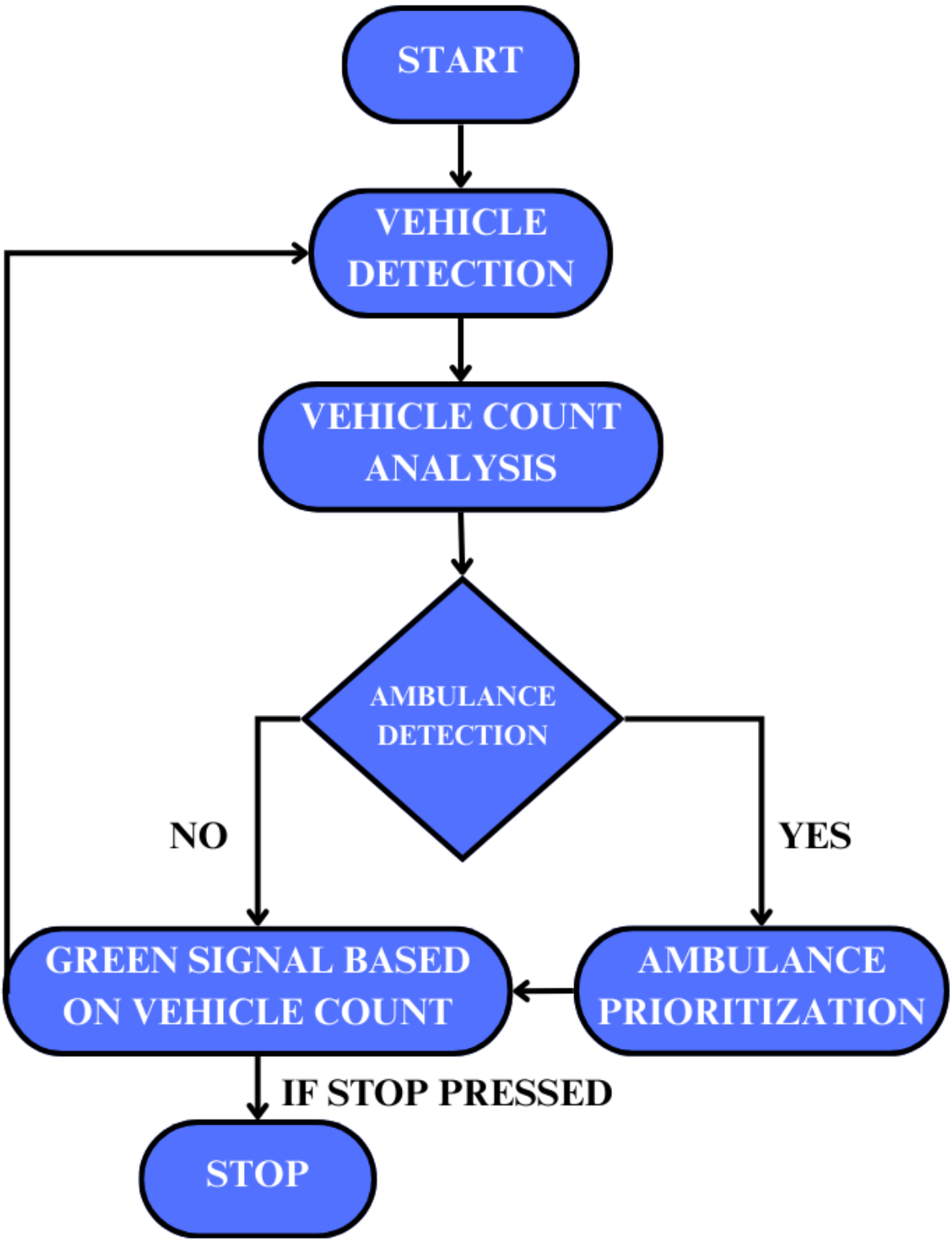

Fig 3.2: Working of Autonomous Traffic Management System

3.2.1 DATASET

The datasets used for the Autonomous Traffic Management System consists of two primary sources: the COCO dataset for general vehicle detection and a custom ambulance dataset for emergency vehicle identification. The COCO dataset is a widely used large-scale object detection dataset that includes various vehicle types such as cars, buses, trucks, and motorcycles, making it an ideal choice for training a robust traffic monitoring system. Additionally, a custom ambulance dataset was curated, containing images of ambulances in diverse traffic scenarios to enhance the system's ability to detect emergency vehicles accurately.

Table 3.1: Approximate number of images in each category of vehicles in COCO dataset

Cars	Buses	Trucks	Motorcycles	Bicycles	Total
27,000	4,500	6,000	5,000	4,000	46,500

To train the model for ambulance detection, a custom ambulance dataset was created using 807 images collected from various online sources with different lighting conditions, angles, and traffic environments. The dataset was split into 600 images for training, 137 images for validation, and 70 images for testing, following an approximate 75-17-8% split.

Table 3.2: Custom ambulance dataset after splitting

Set	No. of images
Training set	600
Validation set	137
Testing set	70
Total	807

Fig 3.3: Sample of custom ambulance dataset

3.2.2 DATA PREPARATION

In the context of developing an Autonomous Traffic Management System, data preparation is a critical step to ensure the accuracy and efficiency of AI-driven traffic control models. One of the key components of this system is vehicle detection, which requires a diverse and well-annotated dataset. For this purpose, we utilize the COCO dataset that contains a wide range of images labelled with various vehicle types, including cars, buses, trucks, motorcycles, and bicycles. This dataset provides a comprehensive foundation for training deep learning models to accurately detect different types of vehicles in traffic scenarios. However, raw datasets often contain redundant or irrelevant images, which may negatively impact model performance. To address this, pre-processing techniques such as duplicate removal and irrelevant object filtering are applied.

For ambulance detection and lane prioritization, a specialized custom ambulance dataset has been created to enhance the system's ability to detect emergency vehicles in real time. This dataset consists of 806 images, sourced from various online repositories and traffic surveillance datasets. To facilitate structured model training and evaluation, the dataset is split into 600 images for training, 137 for validation, and 70 for testing, following a 75%-15%-10% ratio. This division allows the model to be trained on a substantial dataset while being validated and tested on unseen data, ensuring its generalization capability in real-world applications. Since ambulance detection is a high-priority task in traffic management, further refinements are applied to improve accuracy. Advanced augmentation techniques, such as contrast enhancement, brightness adjustment, and noise reduction, are implemented to help the model recognize ambulances under varied lighting conditions, weather changes, and different angles of surveillance cameras.

To maximize the real-time detection accuracy, manual annotation is performed on each image to ensure precise bounding box placements, helping the model distinguish ambulances from other vehicles effectively. The dataset is then used to train a YOLOv8 model, known for its high-speed processing and superior object detection accuracy, making it ideal for real-time traffic applications. By incorporating this ambulance detection module, the traffic management system can prioritize lanes dynamically, allowing ambulances to move through traffic efficiently. This ensures a faster emergency response, reducing delays caused by congestion and ultimately enhancing road safety and public health outcomes.

3.2.3 DATA PREPROCESSING

To apply data pre-processing for the Autonomous Traffic Management System, we need to follow a systematic approach to clean, transform, and prepare the dataset for training. First, the COCO dataset for vehicle detection and the custom ambulance dataset for emergency vehicle prioritization is loaded and examined for missing values, duplicates, and inconsistencies. Any irrelevant images that do not contribute to the model's learning should is removed to ensure high-quality data. Image metadata, such as resolution, file format, and annotation accuracy, is verified to maintain consistency across the dataset.

Since the dataset consists of image data, pre-processing involves several computer vision techniques. This includes image resizing to standard dimensions, ensuring uniform input size across models. Noise reduction techniques are employed to enhance image clarity, particularly in low-light scenarios. Data augmentation, including rotation, flipping, brightness adjustment, and contrast enhancement, is applied to improve generalization and robustness to varying environmental conditions. In the case of the ambulance dataset, additional transformations such as colour space conversion are performed to highlight unique features like sirens and emergency lights.

For deep learning models such as YOLOv7, YOLOv8, and RCNN, image data needs to be transformed into a format suitable for real-time object detection. To ensure that the ambulance dataset is properly labelled, manual annotation was performed using Roboflow. This includes bounding box annotation verification, ensuring that labelled vehicles and ambulances are accurately detected. To handle class imbalance in the ambulance dataset, data oversampling and synthetic augmentation are used to increase the number of ambulance samples. Finally, the dataset is split into training (75%), validation (15%), and testing (10%) to ensure the models generalize well to unseen data. These pre-processing steps enhance the efficiency and accuracy of the Autonomous Traffic Management System, making it robust and effective for real-world deployment.

3.2.4 DATA AUGMENTATION

In image-based tasks such as vehicle detection and ambulance recognition for the Autonomous Traffic Management System, data augmentation involves applying a range of transformations to images, including rotation, flipping, zooming, brightness adjustment, and contrast enhancement. These transformations create new variations of the original images, simulating

different viewing angles, lighting conditions, weather scenarios, and occlusions. By introducing these variations, the augmented dataset becomes more representative of real-world traffic conditions, enabling the model to better generalize to unseen data and accurately detect vehicles and ambulances in diverse environments.

Furthermore, data augmentation helps address issues like class imbalance and overfitting by providing the model with a more extensive and diverse set of training examples. Instead of relying solely on the limited number of original images. The augmented dataset expands the scope of training data by allowing the model to learn more robust and invariant features. This is particularly beneficial in ambulance detection, where emergency vehicles appear less frequently in regular traffic footage. By augmenting ambulance images with variations such as hue shifts (to highlight sirens), and motion blur (to simulate speed), the model becomes more reliable in identifying ambulances under challenging conditions.

Moreover, data augmentation can be tailored to specific tasks and domain knowledge. For example, in vehicle detection, augmentations such as geometric transformations (scaling, cropping) and shadow simulations help the model recognize vehicles in different lane positions and under varying shadow conditions. Similarly, in ambulance prioritization, augmentations like brightness adjustments and synthetic reflections help the model detect ambulances even at night or in low-visibility conditions. These tailored augmentations ensure that the Autonomous Traffic Management System remains highly accurate and adaptable, making it effective in real-world deployment.

3.2.5 DEEP LEARNING MODELS

For the Autonomous Traffic Management System, we used YOLOv7, YOLOv8, and RCNN to detect and classify vehicles in real-time. Deep learning-based object detection models were evaluated to determine the most efficient approach for real-time deployment. The models were trained on the COCO dataset for vehicle detection and a custom ambulance dataset to ensure robust performance across various conditions. The final model was selected based on its detection speed measured in frames per second (FPS). YOLOv8 was chosen as the best model due to its high FPS with balancing accuracy and speed for efficient traffic management. Additionally, we trained the YOLOv8 model for lane prioritization by detecting emergency vehicles such as ambulances.

Region-based Convolutional Neural Networks (RCNN)

Region-based Convolutional Neural Networks (RCNN) are a powerful deep learning approach for object detection and have been applied in the Autonomous Traffic Management System to identify and classify vehicles in real-time. RCNN follows a two-stage process: it first generates region proposals using Selective Search to identify potential objects and then applies a Convolutional Neural Network (CNN) to classify and refine these detections. This method ensures accurate detection of various vehicle types such as cars, buses, trucks, motorcycles, and bicycles.

RCNN individually processes each proposed region, it is highly effective at recognizing vehicles with minimal false positives. A major limitation of RCNN for autonomous traffic management is its slow processing speed. Traditional RCNN processes each detected region separately and making it computationally expensive and inefficient for real-time applications. The detection speed of RCNN is approximately 20 frames per second (FPS), which is significantly lower than modern real-time object detection models such as YOLO. This slow speed poses a critical drawback in fast-moving traffic environments.

To overcome this limitation, more optimized object detection models like YOLO can be integrated into the system to enhance real-time performance. However, for high-speed traffic monitoring where real-time adjustments are crucial, YOLO-based models may be a more suitable alternative due to their single-shot detection architecture.

Despite its drawbacks in real-time processing, RCNN remains a valuable model for applications where detection accuracy is prioritized over speed. It can be used in scenarios where traffic analysis is performed offline, such as post-event traffic pattern analysis or historical congestion studies. Additionally, advancements like Fast RCNN and Faster RCNN have improved the efficiency of the original RCNN by integrating region proposal generation and classification into a unified framework, reducing computation time. While RCNN may not be the optimal choice for real-time autonomous traffic management, its precision and adaptability make it useful for certain aspects of traffic surveillance and planning.

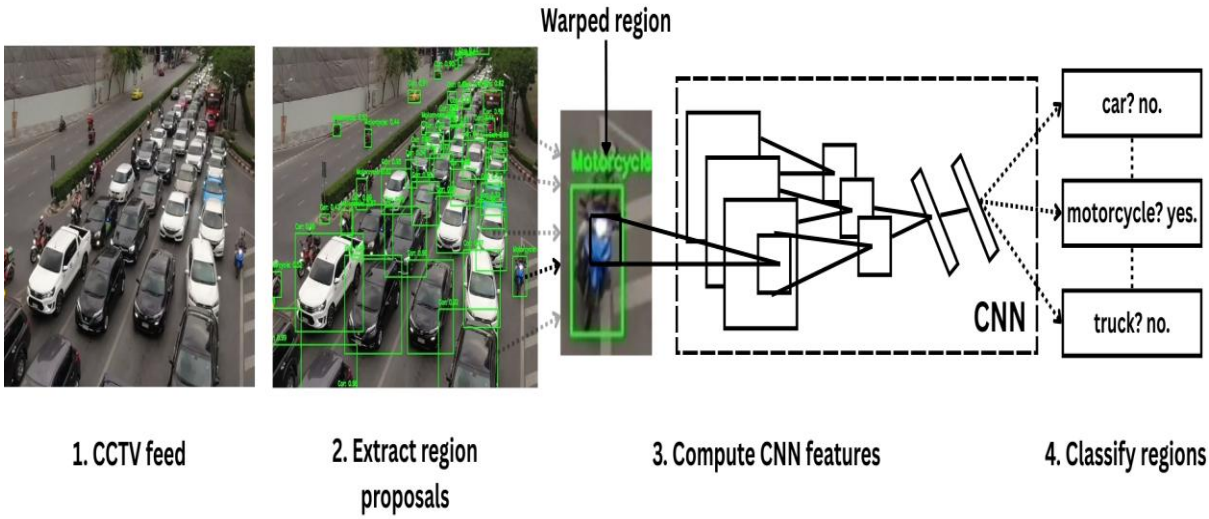

Fig 3.4: RCNN architecture for autonomous traffic management system

You Only Look Once version 7 (YOLOv7)

YOLOv7 (You Only Look Once version 7) is an advanced object detection model known for its speed and accuracy, making it highly suitable for real-time applications like the Autonomous Traffic Management System. Unlike traditional two-stage models like RCNN, YOLOv7 follows a single-stage detection approach by enabling it to process an entire image in one pass and predict bounding boxes with high precision. This architecture makes it ideal for identifying and classifying various vehicles such as cars, buses, trucks, motorcycles, and bicycles.

One of the key advantages of YOLOv7 is its optimized computational efficiency, which allows it to run smoothly on edge devices and embedded systems used in smart traffic management. The model is designed with an improved backbone network and extended receptive fields, enabling it to capture both small and large objects more effectively. In an autonomous traffic control setup, YOLOv7 can accurately detect vehicles across multiple lanes by ensuring that traffic signals are dynamically adjusted based on real-time vehicle density. Additionally, its ability to detect objects under varying lighting and weather conditions makes it robust for deployment in real-world traffic environments.

A major benefit of YOLOv7 for this project is its high detection speed of approximately 40 frames per second (FPS), which is twice as fast as RCNN-based models. This rapid processing capability is essential for real-time traffic monitoring, as it allows for instant decision-making and adaptive signal control. With its speed advantage, YOLOv7 ensures that emergency vehicle detection, congestion analysis, and lane prioritization happen seamlessly without

delays. This makes it a preferred choice for smart city traffic management solutions where efficiency and accuracy must go hand in hand.

However, while YOLOv7 is much faster than RCNN-based models but it is still slower than YOLOv8. YOLOv8 introduces further optimizations in network architecture, model compression, and inference speed, allowing it to achieve even lower latency in high-speed traffic scenarios.

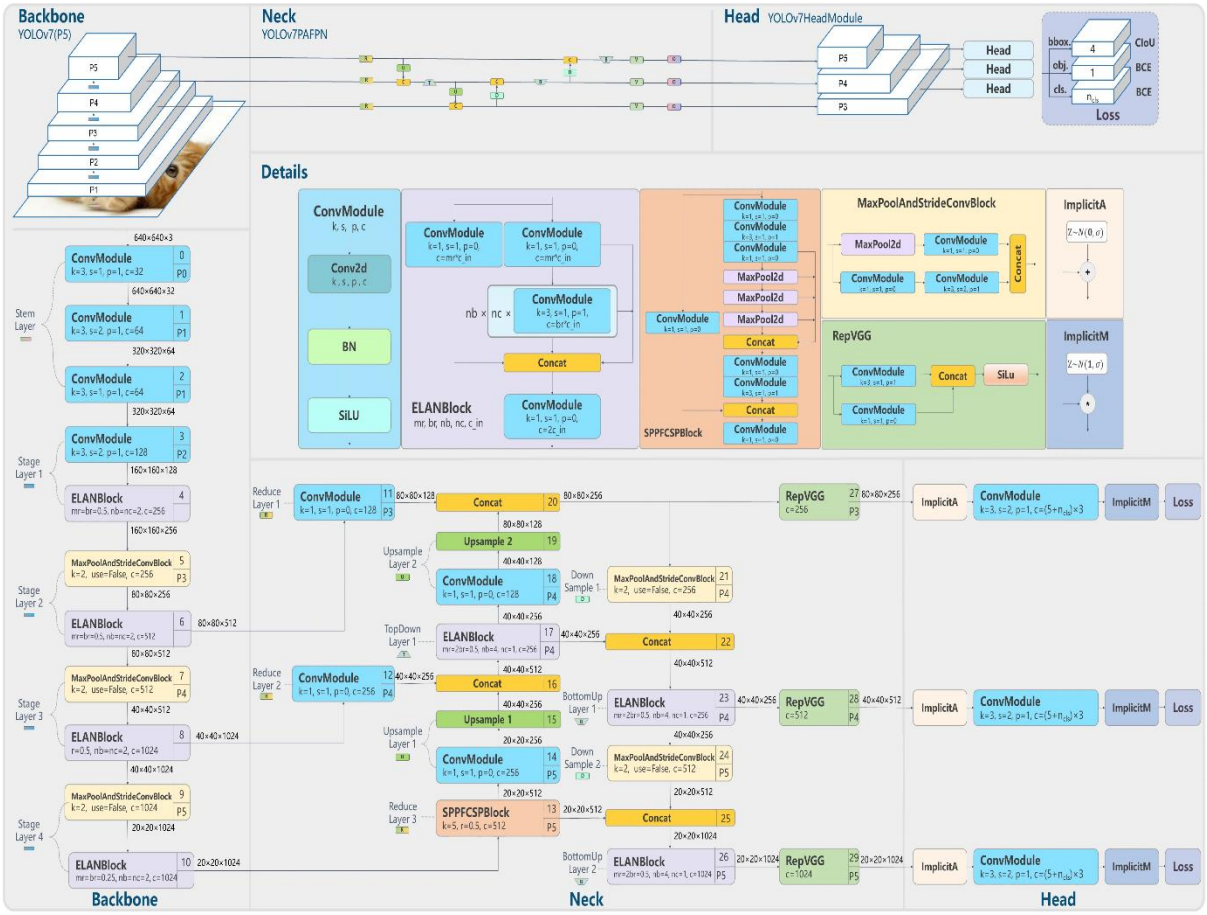

Fig 3.5: YOLOv7 architecture

You Only Look Once version 8 (YOLOv8)

YOLOv8 (You Only Look Once version 8) is the most advanced real-time object detection model that offers significant improvements over previous YOLO versions. It is designed for high-speed, high-accuracy detection and has been integrated into our Autonomous Traffic Management System to identify and classify vehicles in real-time. It's one-stage detection pipeline enables it to process images in a single pass, ensuring faster inference times compared to two-stage detectors like RCNN.

A major advantage of YOLOv8 is its exceptional detection speed of over 50 FPS, which is the highest among the models evaluated for this project. This speed allows the system to analyze real-time CCTV footage efficiently and make dynamic adjustments to traffic signals based on vehicle density. The improved convolutional backbone and anchor-free detection strategy enhance speed and accuracy by making YOLOv8 highly suitable for real-world autonomous traffic control applications. Unlike traditional object detection models that struggle with processing delays, YOLOv8 ensures minimal latency that is crucial for fast-moving traffic environments.

In addition to general vehicle detection, YOLOv8 has been custom-trained for ambulance detection using a dataset of 806 ambulance images collected from various online sources. The dataset was manually annotated using Roboflow, ensuring precise bounding boxes for better detection performance. The dataset was split into 600 images for training, 137 for validation, and 70 for testing to ensure the model generalizes well. Advanced augmentation techniques, including brightness adjustments, rotation, and contrast enhancement, were applied to improve model robustness across different lighting and weather conditions. This enables the system to prioritize emergency vehicles and dynamically adjust traffic signals to provide clear passage for ambulances.

While YOLOv8 delivers the highest speed and efficiency, it is also more computationally intensive than simpler models like YOLOv7. The model's high speed and reliability in detecting both regular vehicles and emergency vehicles make it the most suitable choice for deployment in this project.

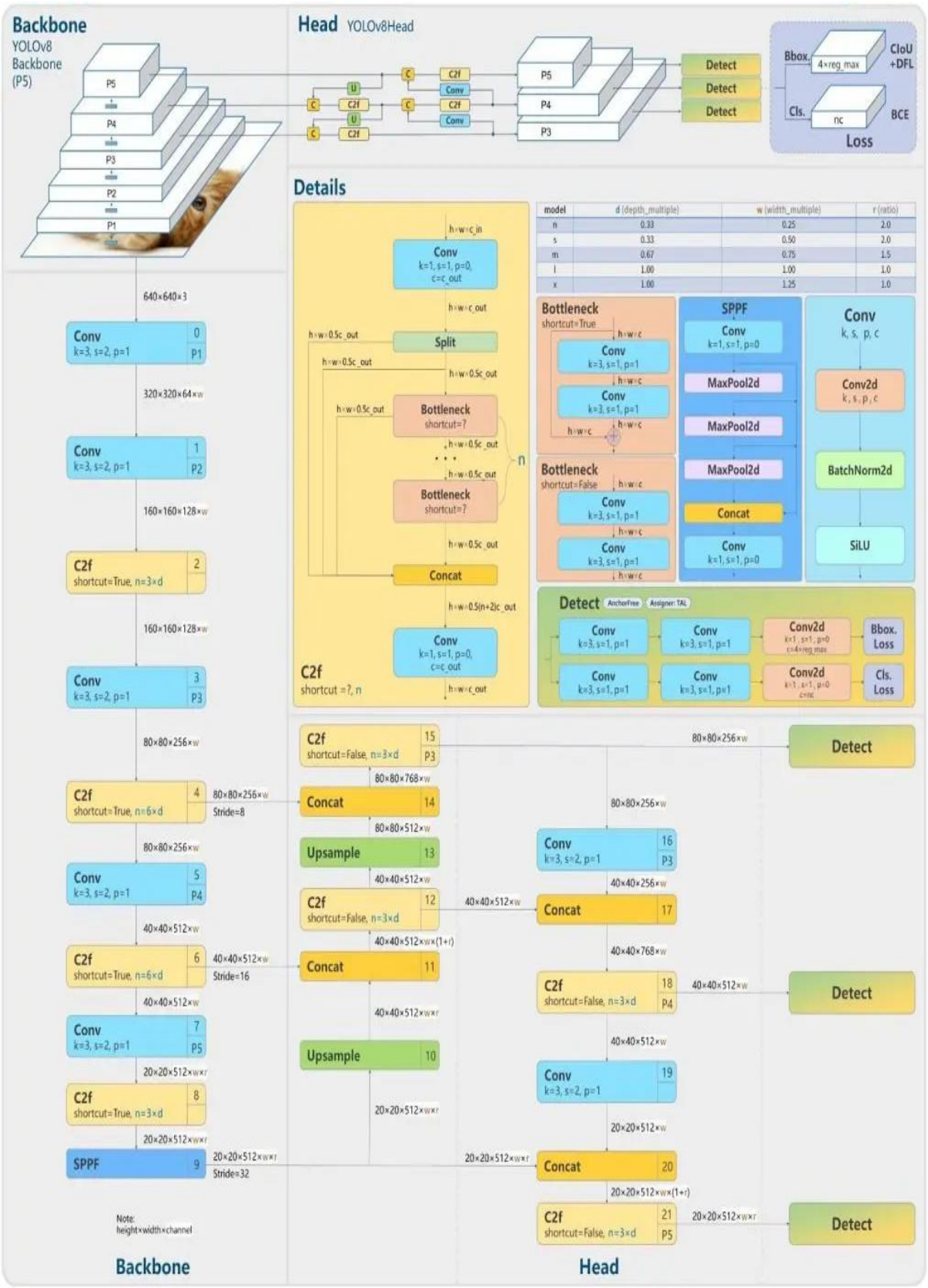

Fig 3.6: YOLOv8 architecture

3.2.6 PERFORMANCE METRICS

Accuracy

Accuracy measures how well the model correctly detects ambulances in the traffic environment. It is calculated as:

$$Accuracy = \frac{TruePositives + TrueNegatives}{TruePositives + TrueNegatives + FalsePositives + FalseNegatives}$$ Equation-1

A high accuracy score indicates that the model correctly identifies ambulances with minimal misclassifications. However, accuracy alone may not be sufficient when dealing with imbalanced data (e.g., fewer ambulances than regular vehicles), so other metrics like precision and recall are needed for a more comprehensive evaluation.

Precision

Precision evaluates how many of the detected vehicles are correct, making it critical in avoiding false detections. In ambulance detection, high precision ensures that other vehicles are not misclassified as ambulances, which could lead to incorrect lane prioritization.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$ Equation-2

Recall

Recall measures the model's ability to correctly detect all actual instances of a given object, particularly ambulances in emergency scenarios. A high recall value is essential for detecting ambulances promptly to avoid missing emergency cases. A low recall means the system is failing to recognize some ambulances, which could lead to delays in emergency response.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$ Equation-3

F1-Score

The F1-score provides a balance between precision and recall, which is especially useful when there is an uneven distribution of vehicle types. In high-traffic environments, models need both high precision and high recall to detect vehicles effectively without missing or falsely classifying objects. A high F1-score ensures the system is both accurate and sensitive to detecting emergency vehicles while minimizing unnecessary interventions.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$ Equation-4

Inference Speed (Frames Per Second - FPS)

The inference speed measures how fast the model processes video frames in real-time. The FPS rate directly impacts the system's ability to monitor traffic continuously and respond to live conditions.

RCNN: 20 FPS (Too slow for real-time applications)

YOLOv7: 40 FPS (Faster but still not the best)

YOLOv8: 45 FPS (Highest FPS, selected for this project due to real-time efficiency)

CHAPTER – 4

IMPLEMENTATION

4.1 TOOLS USED

Python

Python is the backbone of this Autonomous Traffic Management System by providing a powerful and flexible environment for implementing deep learning-based object detection. Python's extensive ecosystem of libraries, such as OpenCV for real-time image processing, NumPy for numerical computations, and Ultralytics YOLO for object detection, enables seamless integration of various functionalities.

One of Python's key advantages in this project is its support for multi-threading and multiprocessing, which is essential for handling real-time video streams efficiently. By using the threading module, the system can capture video frames, perform YOLOv8-based ambulance and vehicle detection, and display results simultaneously without lag. Furthermore, Python's extensive support for APIs, such as the Roboflow API, allows seamless dataset retrieval and model training, streamlining the entire machine-learning workflow. Python also facilitates the deployment of the trained YOLOv8 model in real-world scenarios, enabling integration with traffic surveillance systems.

Jupyter Notebook

Jupyter Notebook plays a crucial role in the development and experimentation phase of the Autonomous Traffic Management System, providing an interactive computing environment for executing Python code. Its cell-based execution model allows developers to break down complex workflows into smaller ones. Jupyter supports inline visualizations with OpenCV, enabling real-time analysis of vehicle detection outputs. This is particularly useful when evaluating YOLOv8's performance on the custom ambulance dataset, as results such as bounding boxes, confidence scores, and misclassifications can be visualized and interpreted directly within the notebook.

In this project, Jupyter Notebook facilitates dataset pre-processing and YOLO model training in a structured manner. The notebook format also enables easy documentation of the model's training progress by making it an excellent tool for collaborative research and development. Additionally, Jupyter's support for multiple kernels enables seamless switching between different Python environments, ensuring compatibility with various deep learning frameworks such as Ultralytics YOLO. This flexibility makes Jupyter Notebook an indispensable tool for developing, debugging, and optimizing the Autonomous Traffic Management System.

NumPy

NumPy (Numerical Python) is a key tool in the Autonomous Traffic Management System, helping with fast calculations and image processing. In this project, NumPy is used to handle images and video frames efficiently. Since OpenCV reads images as NumPy arrays, it allows quick modifications like resizing, normalizing, and changing colour formats, which are necessary for preparing data for the YOLOv8 model. These steps ensure that vehicle and ambulance detection happen smoothly. NumPy is also used for working with bounding boxes, which are the rectangles drawn around detected vehicles. It helps in adjusting these boxes, removing unnecessary detections, and improving accuracy.

NumPy is also helpful in evaluating model performance by calculating important metrics like Precision, Recall, and F1-score. Since it performs calculations quickly, it makes the entire process more efficient. Additionally, NumPy is used in motion detection techniques and compares video frames to track moving vehicles. This helps in real-time traffic monitoring. Because of its speed and ability to handle large datasets, NumPy is an essential part of this project.

Install it using this command:

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

Time

The time module in Python is used to handle time-related functions, making it crucial for real-time applications like the Autonomous Traffic Management System. This project time module helps in displaying signal timers and tracking how long each traffic light remains green, yellow, or red. The time.sleep() function is used to introduce delays, allowing signal changes to happen at proper intervals, just like in real-world traffic management systems. It also helps in synchronizing multiple tasks, such as detecting ambulances and changing traffic signals accordingly.

Copy

The copy module is used to duplicate image frames before applying object detection algorithms. Instead of modifying the original frame, copy.deepcopy() allows the system to store multiple versions of an image, which can be used for debugging or alternative processing.

This feature is particularly useful in multi-threaded applications, where different processes handle different image copies simultaneously. It ensures that one part of the system can apply YOLO detection while another part is used for ambulance detection.

Threading

Threading enables the system to execute multiple tasks in parallel, improving the efficiency of real-time object detection. Without threading, the system would have to capture frames, process YOLO detection, and display results sequentially, leading to delays. By using separate threads for video object detection and displaying results, the system achieves smooth real-time performance without bottlenecks. This is especially important in traffic management, where continuous and fast detection of ambulances is required.

The code below is the syntax for creating, starting, and waiting thread:

```
C:\Users\Your Name>threadname = threading.Thread(target=functionname)

C:\Users\Your Name>threadname.start()

C:\Users\Your Name>threadname.join()
```

Roboflow

Roboflow was used for annotating and preparing the ambulance dataset. The annotation process involved manual labelling ambulance objects in images, defining bounding boxes, and generating a YOLO-compatible dataset. Roboflow also provides tools for dataset augmentation, such as rotation, flipping, and brightness adjustments, to improve model robustness.

Additionally, Roboflow's cloud-based dataset management allows easy collaboration and version control. The custom ambulance dataset was uploaded and processed through Roboflow before being used to train the YOLOv8 model, ensuring high detection accuracy.

Install it using this command:

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

Ultralytics

Ultralytics is a deep learning framework primarily used for object detection, specifically for implementing YOLO (You Only Look Once) models. In this project, Ultralytics YOLOv8 has been used for vehicle and ambulance detection with high-speed and high-accuracy object detection. The Ultralytics package provides a simple interface for training, validating, and deploying YOLO models, making it easier to handle custom datasets. The YOLOv8 model was trained on a custom ambulance dataset to detect ambulances in real-time and prioritize them in traffic management. The framework's optimized architecture enables fast inference, which is essential for autonomous traffic systems where real-time decisions are required.

Additionally, Ultralytics provides pre-trained models that can be fine-tuned for specific applications, reducing the training time required. By using Ultralytics YOLOv8, this project achieved high-speed object detection, with a detection speed surpassing previous YOLO versions, making it the most suitable choice for real-time autonomous traffic management.

Install it using this command:

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

OpenCV

OpenCV (Open Source Computer Vision Library) is a powerful tool used in this project for image processing, real-time video analysis, and object detection. In the Autonomous Traffic Management System, OpenCV helps in pre-processing video frames, applying image transformations, and integrating YOLOv8 for vehicle and ambulance detection. It enables functionalities such as capturing live video feeds from cameras, converting images into suitable formats for deep learning models, and performing real-time bounding box visualization for detected objects. OpenCV also assists in resizing, normalizing, and augmenting input frames, ensuring better model performance.

Additionally, OpenCV plays a crucial role in displaying traffic signals, tracking vehicle movement, and visualizing real-time detections on the road. For instance, when an ambulance is detected, OpenCV can highlight its location and trigger lane prioritization actions. Its efficiency in handling frame-by-frame processing makes it an essential tool for this autonomous traffic management system.

Install it using this command:

```
C:\Users\Your Name>pip install opencv-python
```

4.2 CODE

Model training with a custom ambulance dataset code

```
pip install roboflow

from roboflow import Roboflow

rf = Roboflow(api_key="IxpmXEdTb0yDhuxk2fcy")

project = rf.workspace("ambulance-oxmvh").project("ambulance-2ks9z")

version = project.version(3)

dataset = version.download("yolov8")

from ultralytics import YOLO

# Loading the YOLOv8 model

model = YOLO("yolov8n.pt")

# Training the model on your custom dataset

model.train(

    data="Ambulance-3/data.yaml",

    epochs=50,

    imgsz=640

)
```

Autonomous Traffic Management System code

```
pip install ultralytics

pip install opencv-python

pip install ultralytics deep-sort-realtime opencv-python

import cv2

from ultralytics import YOLO

import numpy as np

import time

import copy
```

```
import threading


# Loading YOLOv8 model
model = YOLO("yolov8n.pt")
#Loading ambulance trained model
ambulance_model = YOLO("C:/Users/ragha/runs/detect/train5/weights/best.pt")
# Defining vehicle classes (COCO dataset IDs for vehicles)
vehicle_classes = [2, 3, 5, 7]  # Car, Bus, Truck, Motorcycle
# Initializing variables for lane monitoring
lanes = {
    "lane_1": {"cctv": "Lane1.mp4", "count": 0, "ambulance_detected": False},
    "lane_2": {"cctv": "Lane2.mp4", "count": 0, "ambulance_detected": False},
    "lane_3": {"cctv": "Lane3.mp4", "count": 0, "ambulance_detected": False},
    "lane_4": {"cctv": "Lane4.mp4", "count": 0, "ambulance_detected": False}
}
# Signal states and timers
signal_states = {"lane_1": "red", "lane_2": "red", "lane_3": "red", "lane_4": "red"}
signal_timers = {"lane_1": 0, "lane_2": 0, "lane_3": 0, "lane_4": 0}
# Flag to stop the process
stop_process = False
lock = threading.Lock()
caps = {lane_key: cv2.VideoCapture(lanes[lane_key]["cctv"]) for lane_key in lanes.keys()}
for cap in caps.values():
        cap.set(cv2.CAP_PROP_FPS, 1200)
```

```
def count_vehicles_and_draw(frame, lane_key):
    """Detect and count vehicles in the frame using YOLO, and draw bounding boxes."""
    results = model(frame, stream=False, verbose=False)
    vehicle_count = 0
    for result in results:
        for box in result.boxes:
            cls_id = int(box.cls.cpu().numpy().item())
            if cls_id in vehicle_classes:  # Checking if the detected object is a vehicle
                vehicle_count += 1
                # Drawing bounding box
                x1, y1, x2, y2 = map(int, box.xyxy.cpu().numpy()[0])
                cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
                cv2.putText(
                    frame,
                    model.names[cls_id],
                    (x1, y1 - 10),
                    cv2.FONT_HERSHEY_SIMPLEX,
                    0.5,
                    (0, 255, 0),
                    2,
                )
    with lock:
        lanes[lane_key]["count"] = vehicle_count
    return frame
```

```
def process_lane():
    """Process each lane's CCTV feed and update vehicle counts with detection frames."""
    for lane_key in ['lane_1','lane_2','lane_3', 'lane_4']:
        cap =cv2.VideoCapture(lanes[lane_key]["cctv"])
        ret, frame = cap.read()
        if ret:
            frame = count_vehicles_and_draw(frame, lane_key)
        else:
            frame = np.zeros((300, 400, 3), dtype=np.uint8)
            cv2.putText(
                frame,
                f"No feed for {lane_key.upper()}",
                (50, 150),
                cv2.FONT_HERSHEY_SIMPLEX,
                0.7,
                (0, 0, 255),
                2,
            )
        cap.release()
        return frame


def display_signals():
    """Display the current traffic signal status for all lanes along with timers."""
    global next_signal, reason
    signal_image = np.zeros((300, 650, 3), dtype=np.uint8)
    positions = {
```

```
        "lane_1": (50, 30),
        "lane_2": (50, 70),
        "lane_3": (50, 110),
        "lane_4": (50, 150),
    }
    colors = {
        "red": (0, 0, 255),
        "green": (0, 255, 0),
        "yellow": (0, 255, 255),
    }
    # Determining the next lane with the highest vehicle count
    next_lane = next_signal
    next_note = f"Next GREEN: {next_lane.upper()} {reason}"
    for lane, state in signal_states.items():
        pos = positions[lane]
        color = colors[state]
        timer = signal_timers[lane]
        text = f"{lane.upper()}: {state.upper()} - {timer}s"
        cv2.putText(signal_image, text, (pos[0], pos[1]),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, color, 2)
    # Adding the note for the next green signal
    cv2.putText(signal_image, next_note, (50, 230), cv2.FONT_HERSHEY_SIMPLEX, 1,
(255, 255, 255), 2)
    cv2.imshow("Traffic Signals", signal_image)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        global stop_process
```

```
        stop_process = True


def transition_to_yellow(lane_key):
    """Transition a lane's signal to yellow for 5 seconds."""
    global signal_states, signal_timers
    signal_states[lane_key] = "yellow"
    signal_timers[lane_key] = 5
    print(f" 🚦 {lane_key.upper()} YELLOW for 5 seconds.")
    while signal_timers[lane_key] > 0:
        display_signals()
        time.sleep(1)
        signal_timers[lane_key] -= 1


def manage_signals():
    """Manage traffic signals based on vehicle counts."""
    global lanes, signal_states, signal_timers, stop_process, next_signal, current_green, reason
    while not stop_process:
        # Sorting lanes by vehicle count and manage signals
        reason="More Vehicles"
        flag=0
        lanes_data=copy.deepcopy(lanes)
        data=list(lanes_data)
        while flag<4:
            vehicount={}
            flag=flag+1
            for lane_key in lanes_data.keys():
```

```
            vehicount[lane_key]=lanes[lane_key]["count"]
        lane= max(vehicount, key=vehicount.get)
        next_signal=lane
        reason="More Vehicles"
        vehicle_count = vehicount[lane]
        del lanes_data[lane]
        if vehicle_count > 0:
            green_time = min(max(10, vehicle_count * 2), 60)  # Green time limits
            # Turning other signals to red
            for other_lane in signal_states:
                if other_lane != lane and signal_states[other_lane] != "red":
                    transition_to_yellow(other_lane)
                    signal_states[other_lane] = "red"
            # Transition from red to green
            if signal_states[lane] == "red":
                transition_to_yellow(lane)
                signal_states[lane] = "green"
                reason=""
                next_signal=""
                signal_timers[lane] = green_time
            current_green=lane
            print(f" 🚦 {lane.upper()} GREEN for {green_time} seconds. Vehicles:
{vehicle_count}")
            while signal_timers[lane] > 0:
                display_signals()
                time.sleep(1)
```

```
            signal_timers[lane] -= 1
            prev_time=signal_timers[lane]
            for lane_key in lanes.keys():
                if lanes[lane_key]["ambulance_detected"]:
                    next_signal=lane_key
                    reason="Ambulance"
                    prioritize_ambulane_lane(lane_key)
                    if stop_process:
                        return
                    next_signal=lane
                    reason="More Vehicles"
                    transition_to_yellow(lane)
                    current_green=lane
            signal_timers[lane]=prev_time
            signal_states[lane] = "green"
            next_signal=""
            reason=""
            if stop_process:
                return
        if stop_process:
            return
    else:
        print(f" 🚦 {lane.upper()} YELLOW for 5 seconds (No vehicles detected).")
        transition_to_yellow(lane)
        signal_states[lane] = "red"
        if stop_process:
```

```
                return
        print("Cycle complete. Starting over...")


def prioritize_ambulane_lane(lane):
    global signal_states, signal_timers, next_signal, current_green, lanes, reason
    if current_green==lane:
        signal_timers[lane]=signal_timers[lane]+30
        lanes[lane]["ambulance_detected"]=False
        return
    else:
        # Turning other signals to red
        for other_lane in signal_states:
            if other_lane != lane and signal_states[other_lane] != "red":
                transition_to_yellow(other_lane)
                signal_states[other_lane] = "red"
        # Transition from red to green
        if signal_states[lane] == "red":
            transition_to_yellow(lane)
            print(f" 🚦 {lane.upper()} GREEN for ambulance")
            current_green=lane
            signal_states[lane] = "green"
            signal_timers[lane] = 30
            reason=""
            next_signal=""
        while signal_timers[lane] > 0:
            display_signals()
```

```
            time.sleep(1)
            signal_timers[lane] -= 1
            if stop_process:
                return
        lanes[lane]["ambulance_detected"]=False
        transition_to_yellow(lane)
        signal_states[lane]="red"
        signal_timers[lane]=0
        return

def detect_ambulance(frame, lane_key, confidence_threshold=0.7):
    if lanes[lane_key]["ambulance_detected"]:
        results = ambulance_model(frame,verbose=False)
        for result in results:
            for box in result.boxes:
                cls_id = int(box.cls.cpu().numpy().item())
                class_name = ambulance_model.names[cls_id]
                if class_name.lower() == "ambulance" and box.conf.cpu().numpy() > confidence_threshold:
                    # Drawing the bounding box on the frame for ambulance
                    x1, y1, x2, y2 = map(int, box.xyxy.cpu().numpy()[0])
                    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 2)
                    cv2.putText(
                        frame,
                        "Ambulance",
                        (x1, y1 - 10),
```

```
                cv2.FONT_HERSHEY_SIMPLEX,
                0.9,
                (0, 0, 255),
                2,
            )
    return frame
else:
    results = ambulance_model(frame,verbose=False)
    for result in results:
        for box in result.boxes:
            cls_id = int(box.cls.cpu().numpy().item())
            class_name = ambulance_model.names[cls_id]
            if class_name.lower() == "ambulance" and box.conf.cpu().numpy() > confidence_threshold:
                lanes[lane_key]["ambulance_detected"] = True
                print(f"🚑 Ambulance detected in {lane_key}! Prioritizing this lane.")
                # Drawing the bounding box on the frame for ambulance
                x1, y1, x2, y2 = map(int, box.xyxy.cpu().numpy()[0])
                cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 2)
                cv2.putText(
                    frame,
                    "Ambulance",
                    (x1, y1 - 10),
                    cv2.FONT_HERSHEY_SIMPLEX,
                    0.9,
                    (0, 0, 255),
```

```
                2,
            )
        return frame


def display_cameras():
    """Display all 4 CCTV feeds in one frame."""
    global stop_process, caps
    lane_labels = {
        "lane_1": "Lane 1",
        "lane_2": "Lane 2",
        "lane_3": "Lane 3",
        "lane_4": "Lane 4",
    }
    while not stop_process:
        frames = []
        for lane_key, cap in caps.items():
            for _ in range(10):
                cap.grab()
            ret, frame = cap.read()
            if ret:
                # Adding lane label to the frame
                label = lane_labels[lane_key]
                cv2.putText(
                    frame,
                    label,
                    (10, 55),
```

```
            cv2.FONT_HERSHEY_SIMPLEX,
            2,
            (0, 255, 255),
            4,
        )
        frame = count_vehicles_and_draw(frame, lane_key)
        frame= detect_ambulance(frame, lane_key)
    else:
        # If video ends or fails, display a placeholder
        frame = np.zeros((300, 400, 3), dtype=np.uint8)
        cv2.putText(
            frame,
            "No feed",
            (50, 150),
            cv2.FONT_HERSHEY_SIMPLEX,
            2,
            (0, 0, 255),
            4,
        )
        stop_process = True
        break
    frames.append(frame)
# Resizing frames to fit in a single window
resized_frames = [cv2.resize(f, (400, 300)) for f in frames]
row1 = np.hstack(resized_frames[:2])
row2 = np.hstack(resized_frames[2:])
```

```
            combined_frame = np.vstack([row1, row2])
            cv2.imshow("All CCTV Feeds", combined_frame)
            if cv2.waitKey(1) & 0xFF == ord("q"):
                stop_process = True
                break
        # Release video captures and close windows
        for cap in caps.values():
            cap.release()
        cv2.destroyAllWindows()

def main():
    """Main loop to monitor traffic and manage signals."""
    global stop_process
    process_lane()
    # Start threads for camera display and signal management
    camera_thread = threading.Thread(target=display_cameras)
    signal_thread = threading.Thread(target=manage_signals)
    camera_thread.start()
    signal_thread.start()
    camera_thread.join()
    signal_thread.join()
    print("Process stopped.")

if __name__ == "__main__":
    main()
```

4.3 RESULTS AND SCREENSHOTS

Table 4.1: Inference Speed of proposed models

Module	Models(Inference Speed)
Vehicle Detection	RCNN(20 FPS) YOLOv7(40 FPS) YOLOv8(45 FPS)

Table 4.2: Ambulance dataset training results

Module	Precision	Recall	Mean Average Precision (mAP@50)	Mean Average Precision (mAP@50-95)
Ambulance dataset training	0.991	0.935	0.977	0.729

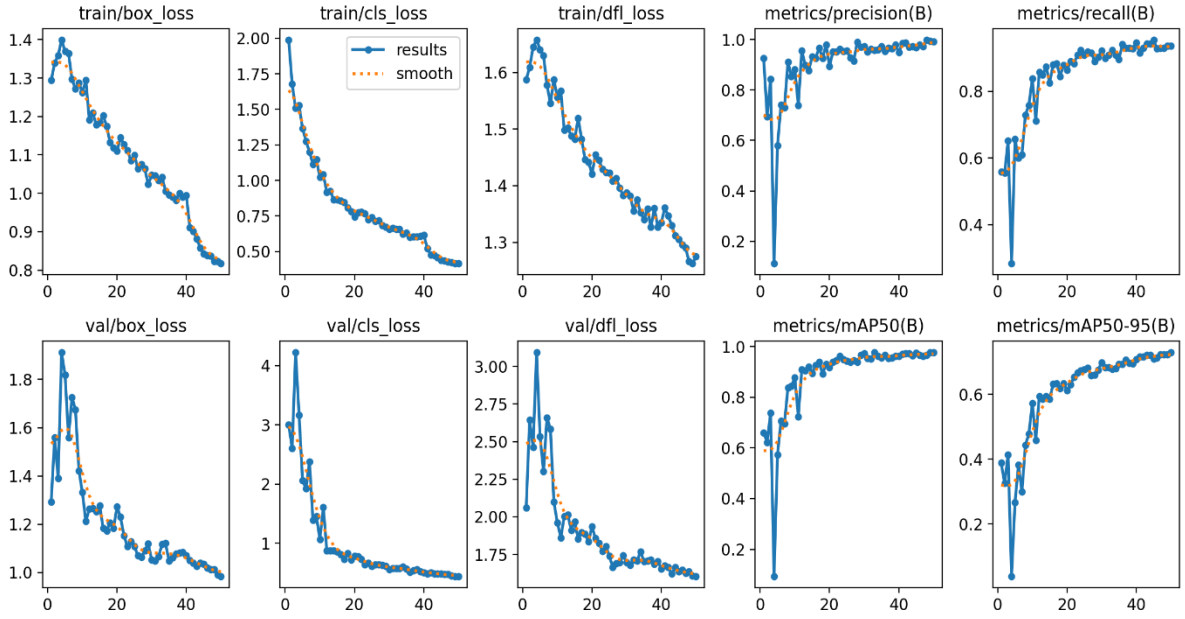

Fig 4.1: Ambulance dataset training result graphs

Fig 4.2: Ambulance prediction along with confidence

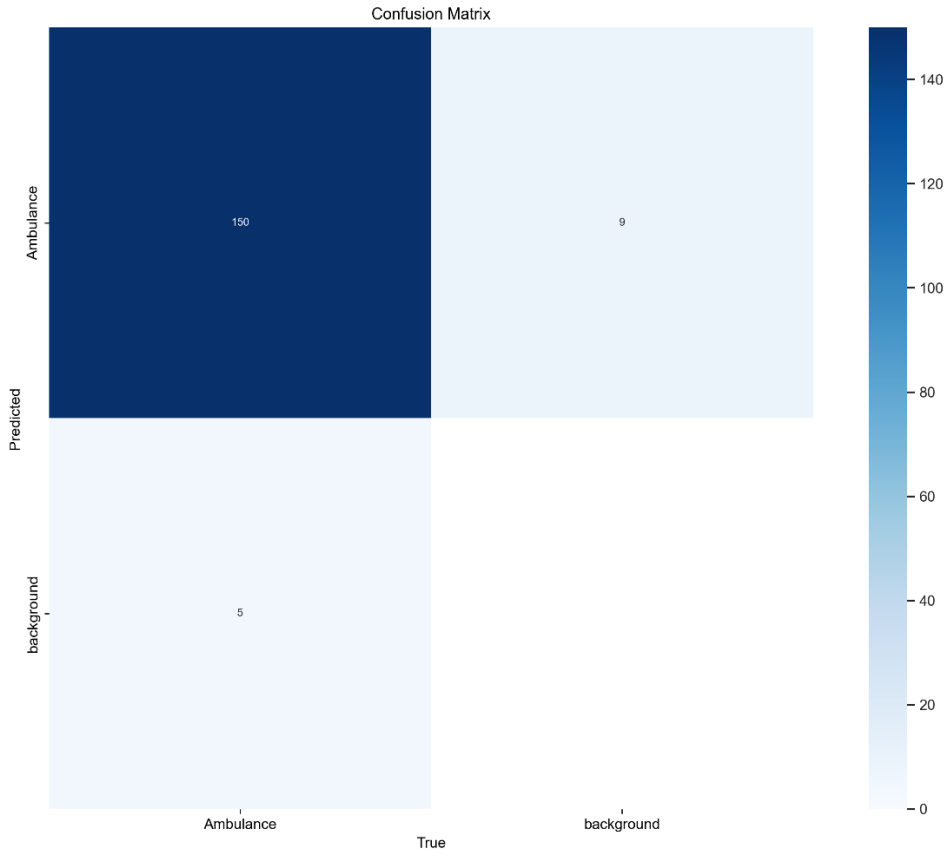

Fig 4.3: Ambulance detection confusion matrix

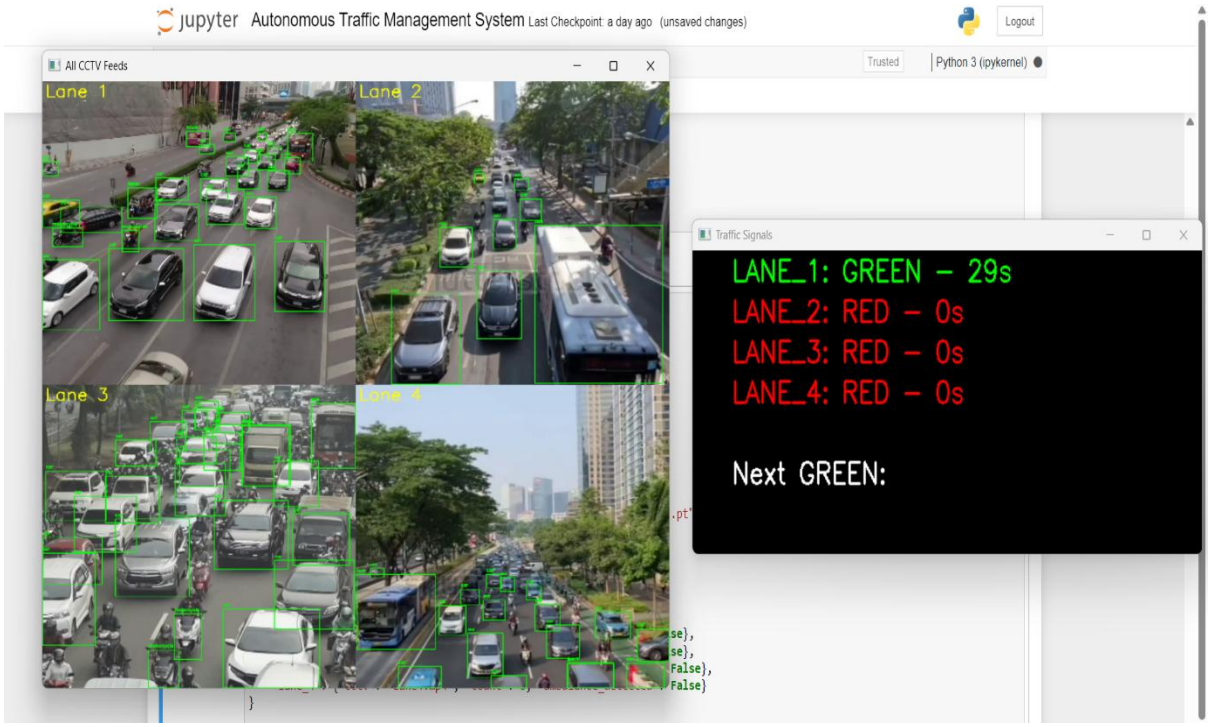

Fig 4.4: Autonomous Traffic Management System displaying CCTV feeds and signals

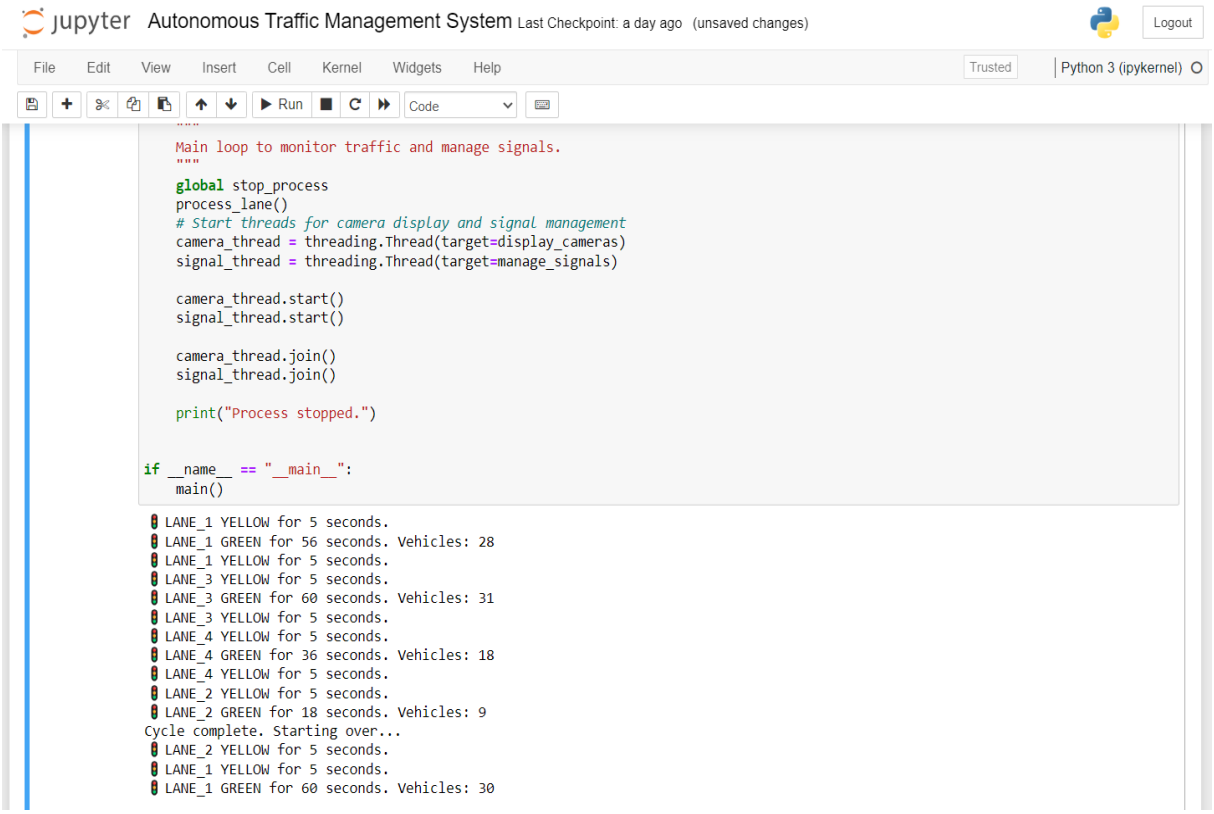

Fig 4.5: Autonomous Traffic Management System's output in console

Fig 4.6: Prioritizing lane 2 as the ambulance is detected

CHAPTER – 5

CONCLUSION AND FUTURE SCOPE

CONCLUSION

This project successfully developed an autonomous traffic management system that uses deep learning-based object detection models to improve traffic flow and prioritize emergency vehicles. By evaluating different object detection models, including RCNN, YOLOv7, and YOLOv8. The study identified YOLOv8 as the most efficient due to its superior detection speed of 45 FPS and high accuracy. Integrating deep learning with traffic management ensures rapid and precise vehicle identification, which is crucial for real-time decision-making in urban traffic scenarios.

To enhance emergency response efficiency, a custom dataset for ambulance detection was created and trained using YOLOv8, achieving an average precision of 99.1% and an accuracy of 97.7%. The system effectively detects ambulances in real-time and dynamically adjusts traffic signals to provide uninterrupted movement for emergency vehicles. By automating traffic signal control based on detected vehicles, the system minimizes delays and ensures faster response times for ambulances in critical situations.

The experimental results demonstrate that deep learning-based object detection can significantly improve intelligent traffic control systems by enabling real-time detection and adaptive signal adjustments. The high detection accuracy and speed of YOLOv8 confirm its potential for real-world deployment in smart city traffic management. The system represents a major advancement in utilizing artificial intelligence for public safety and traffic optimization, reducing congestion and enhancing the overall efficiency of urban transportation networks.

FUTURE SCOPE

In the future, the system can be enhanced by implementing an intelligent traffic signal timing mechanism that dynamically adjusts the signal duration based on the distance and speed of vehicles approaching the intersection. This would ensure that each type of vehicle is allotted an optimal amount of time to cross the signal smoothly by reducing unnecessary delays and improving traffic flow. By incorporating real-time traffic density analysis, the system can further optimize signal timing to minimize congestion and enhance road efficiency.

Another significant improvement is the expansion of the model to detect other emergency vehicles, such as fire engines and police cars, ensuring a comprehensive emergency response system. Training the model on a diverse dataset containing multiple types of emergency vehicles will allow the system to prioritize them effectively and adjust traffic signals accordingly. Additionally, integrating vehicle-to-infrastructure (V2I) communication technology can further enhance real-time decision-making by allowing emergency vehicles to send signals to the traffic management system for faster clearance.

Future advancements can also include integration with smart city infrastructure, where the system connects with IoT-enabled traffic cameras and sensors to gather real-time data on road conditions, weather patterns, and traffic congestion. This data can be used to make intelligent decisions for traffic management and emergency handling. Additionally, incorporating edge computing can enhance the system’s efficiency by processing data locally at intersections, reducing latency, and improving real-time performance. These advancements will contribute to building a fully autonomous and adaptive traffic management system that ensures seamless and safe transportation in smart cities.

REFERENCES

[1] Mamoona Humayun, Maram Fahhad Almufareh and Noor Zaman Jhanjhi, “Autonomous Traffic System for Emergency Vehicles,” *Electronics*, vol. 11, pp. 510-528, February 2022.

[2] Pankaj Kunekar, Yogita Narule, Richa Mahajan, Shantanu Mandlapure, Eshan Mehendale, and Yashashri Meshram, “Traffic Management System Using YOLO Algorithm,” in Proceedings of *Engineering*, Dubai, United Arab Emirates, 2024.

[3] Dr. Sreelatha R, Mahalakshmi B S, Riya Yadav, Shreyam Pandey, and Vandit Agarwal, “Artificial Intelligence Based Autonomous Traffic Regulator,” in Proceedings of *CEUR Workshop*, New Delhi, India, 2023.

[4] Nitin Sakhare, Mrunal Hedau, Gokul B., Omkar Malpure, Trupti Shah, and Anup Ingle, “Smart Traffic: Integrating Machine Learning, and YOLO for Adaptive Traffic Management System,” *International Journal of INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING*, vol. 12, no. 12, pp. 347-355, January 2024.

[5] Izhak Rubin, Andrea Baiocchi, Yulia Sunyoto, and Ion Turcanu, “Traffic management and networking for autonomous vehicular highway systems,” *Ad Hoc Networks*, vol. 83, pp. 125-148, February 2019.

[6] Christofel Rio Goenawan, “ASTM: Autonomous Smart Traffic Management System Using Artificial Intelligence CNN and LSTM,” *arXiv preprint arXiv:2410.10929*, October 2024.

[7] S. Sharma, A. Kumar, and P. Gupta, "Smart Traffic Management of Vehicles using Faster R-CNN Based Deep Learning Method," *Scientific Reports*, vol. 13, no. 1, pp. 1-15, May 2023.

[8] J. Oliveira, M. Silva, and R. Fernandes, "Adaptive and Dynamic Smart Traffic Light System for Efficient Traffic Management," *IET Smart Cities*, vol. 5, no. 2, pp. 89-98, August 2023.

[9] A. Smith and B. Jones, "Deep Learning for Emergency Vehicle Identification: A YOLOv8-Based Approach for Smart City Solutions," *Journal of Engineering Science*, vol. 10, no. 4, pp. 345-356, December 2023.

[10] L. Wang and H. Zhao, "Optimization Control of Adaptive Traffic Signal with Deep Reinforcement Learning," *Electronics*, vol. 13, no. 1, pp. 198-210, January 2024.

[11] M. Brown and K. Taylor, "Adaptive Traffic Signal Control for Large-Scale Scenarios with Multi-Agent Deep Reinforcement Learning," *Transportation Research Part C: Emerging Technologies*, vol. 130, pp. 103-120, June 2023.

[12] R. Kumar and S. Patel, "Analyzing Real-Time Object Detection with YOLO Algorithm in Autonomous Traffic Systems," *Journal of Transportation Safety & Security*, vol. 15, no. 2, pp. 220-235, March 2024.

[13] H. Su, Y. D. Zhong, J. Y. J. Chow, B. Dey, and L. Jin, "EMVLight: A Multi-Agent Reinforcement Learning Framework for an Emergency Vehicle Decentralized Routing and Traffic Signal Control System," *arXiv preprint arXiv:2206.13441*, June 2022.

[14] A. Johnson and M. Lee, "Enhancing Emergency Vehicle Detection: A Deep Learning Approach Integrating Acoustic and Visual Information," *Mathematics*, vol. 12, no. 10, pp. 1514-1530, May 2024.

[15] D. Wilson and E. Martinez, "AI-Driven Adaptive Traffic Signal Optimization System with Emergency Vehicle Prioritization," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 3, pp. 45-56, March 2024.

[16] P. Singh and R. Kaur, "Using a YOLO Deep Learning Algorithm to Improve the Accuracy of Traffic Data Collection," *World Electric Vehicle Journal*, vol. 16, no. 1, pp. 9-20, January 2024.

[17] M. Davis and L. Rodriguez, "Smart Traffic Monitoring System Using YOLO and Deep Learning Techniques," *International Journal of Computer Applications*, vol. 182, no. 30, pp. 1-7, August 2023.

[18] J. Park, T. Kim, and H. Lee, "Real-Time Emergency Vehicle Detection and Traffic Signal Control Using Deep Learning-Based Computer Vision," *IEEE Access*, vol. 12, pp. 23845-23857, March 2024.

UNITED NATIONS SUSTAINABILITY DEVELOPMENT GOALS & MAPPING

Below are the UN sustainability development goals.

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

Aspect	Details
Project Title	Autonomous Traffic Management System
Key Focus Area	Traffic
Relevant SDGs	SDG 11 – Sustainable Cities and Communities SDG 13 – Climate Action
Contribution	Contributes to the development of smart and sustainable urban environments by promoting efficient traffic flow and safer roads. Helps lower vehicle emissions by reducing traffic congestion and idle time.

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)

CERTIFICATE

This is to certify that P. Raghavendra, 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 Autonomous Traffic Management System 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

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)

CERTIFICATE

This is to certify that N. Teja Vishnuvardhan Reddy, 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 Autonomous Traffic Management System 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)

CERTIFICATE

This is to certify that P. Satya Sai Sree Rama Koushik, 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 Autonomous Traffic Management System 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)

CERTIFICATE

This is to certify that Sk. Basha, 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 Autonomous Traffic Management System 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

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
Proceedings of International Conference on Advanced Computing Technologies (ICACT-2025), 3rd April 2025 & 4th April 2025, ISBN: 978-81-977391-1-8

Autonomous Traffic Management System

Jeevan Babu Maddala[1], P. Raghavendra[2], N. Teja Vishnuvardhan Reddy[3], P. Satya Sai Sree Rama Koushik[4], Sk. Basha[5]
[1]Assistant Professor, Department of Computer Science and Engineering, Vasireddy Venkatadri Institute of Technology, (Autonomous), Guntur, Andhra Pradesh, India, [1]jeevan.projects@gmail.com
[2,3,4,5] Students, Department of Computer Science and Engineering, Vasireddy Venkatadri Institute of Technology, (Autonomous), Guntur, Andhra Pradesh, India,
raghavendramadhu0@gmail.com, tejavishnuvardhan.nagireddy@gmail.com, koushikpenumarti2004@gmail.com, sb2508886@gmail.com

Abstract- Traffic management is a growing problem in many cities leading to accidents, high delays, and increasing air pollution. Generally, traffic is managed manually by traffic policemen or with fixed traffic light times. This research will provide an autonomous method by selecting the efficient method among RCNN, YOLO V7 & YOLO V8. The detection speed by using the mentioned methods is 20 FPS, 40 FPS & 45 FPS respectively. Based on the detection speeds, YOLO V8 is the most efficient method among them. The ambulance custom dataset is used to train YOLO V8 for ambulance detection. The trained model attained an average precision of 99.1% and an accuracy of 97.7%. Overall, the research will manage the traffic based on the vehicle count of each lane. It will also prioritize the lane for ambulances.

Keywords: YOLO, Ambulance detection, Smart traffic management system, Vehicle count detection, COCO dataset.

I. INTRODUCTION

The traditional traffic management systems are manually controlled by traffic policemen and by fixed traffic light cycles. Both of them have several limitations. Fixed signal timings will not work out for real-time traffic flow, it may cause unnecessary delays and an increase in accident count. On the other hand, manual traffic control will require a lot of manpower. To overcome these drawbacks this research will provide a way to manage traffic autonomously by utilizing the efficient method.

With the help of several deep learning methods like Region-Based Convolutional Neural Networks (RCNN), You Only Look Once (YOLO) V7, and YOLO V8 vehicle detection will be achieved. This deep-learning vehicle detection will detect the vehicles with different class labels and draw the vehicle count. The main advantage of this approach is to improve traffic management in real time and reduce accidents.

This research will demonstrate the efficient model among RCNN, YOLO V7 & YOLO V8 for vehicle detection. Our study evaluates these models based on the detection speed which is an important aspect of traffic management. The detection speed for the RCNN model is 20 FPS, the YOLO V7 model is 40 FPS and the YOLO V8 model is 45 FPS. Based on this output, YOLO V8 is the efficient model for vehicle detection.

Additionally, this research implemented ambulance detection by further training the YOLO V8 model with a custom ambulance dataset. The performance metrics of ambulance detection are 99.1% average precision and 97.7% accuracy. This detection will help to prioritize the ambulance-detected lane.

The proposed system will allow the green time based on the vehicle count. If an ambulance is detected in any lane, then the signals will pause and prioritize the ambulance lane after 30 seconds the regular signal will resume. By integrating this autonomous traffic management system, we can ensure the safety and minimize the delays.

II. RELATED WORKS

In the past years, researchers have researched several ways to overcome the drawbacks of the 2 traditional ways. For example, few researchers utilized YOLO V3, V2, and V7 for autonomous traffic management. These methods have illustrated the improvement over traditional ways. In a few cases, it has some limitations.

One of the main drawbacks of the previous studies is the lower detection speed of the models. Models like YOLO V2, V3, and V7 have slower detection speeds. This delay in detection may reduce the efficiency of adjusting traffic signals dynamically. When the real-time detection is not fast enough then vehicles are not counted accurately and cause accidents.

Another issue with the previous research is the lack of ambulance detection and prioritization. Many studies have focused on traffic management by detecting and counting vehicles. There is no previous study about the emergency response mechanism in their systems. This means during emergencies previous systems are unable to prioritize the lane with the ambulance arrival. In the absence of this feature

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
Proceedings of International Conference on Advanced Computing Technologies (ICACT-2025), 3rd April 2025 & 4th April 2025, ISBN: 978-81-977391-1-8

traffic management systems are not efficient in emergencies. Additionally, there is no system that combines both traffic management and emergency vehicle prioritization into a single system. Existing research works have either traffic management or ambulance prioritization. In real-time emergencies are very important so that traffic management will be effective.

By solving these limitations, this research develops an autonomous traffic management system using YOLO V8. This model has a 45 FPS detection speed and also detects ambulances and prioritizes the lane. This system will set the signal timers dynamically.

III. METHODOLOGY

This research about autonomous traffic management system is designed to manage traffic dynamically in real-time and prioritize emergency vehicles, particularly ambulances. The methodology consists of mainly five phases: Data Collection, Model Selection for Vehicle Detection and Training for Ambulance Detection, Traffic Signal Management, Emergency Vehicle Prioritization, and Real-time Traffic Detection.

A. Data Collection

Traffic management mainly depends on vehicle detection. This research uses the COCO dataset (Common Objects in Context) for general vehicle detection. The COCO dataset is generally used for object detection. This dataset contains around 45k vehicle images of different vehicle types. One major drawback that this dataset has is that it does not contain a category for ambulances.

Several ambulance images were collected from multiple sources. These images are carefully annotated with drawing boxes and labelled ambulance with the help of the Roboflow platform. By collecting and annotating the custom ambulance dataset, the research ensures efficient detection.

B. Model Selection for Vehicle Detection and Training for Ambulance Detection

Initially, RCNN, YOLO V7, YOLO V8, and YOLO V9 are the proposed models. These models were selected based on their best performance in object detection. However, the YOLO V9 model has not yet been officially released this research will proceed with the other three models. This research evaluated the RCNN, YOLO V7, and YOLO V8 models for vehicle detection.

The primary objective of this evaluation is to select an efficient model that has a high detection speed for real-time vehicle detection. To compare these models the performance metric is the detection speed which is calculated in Frames Per Second (FPS). The evaluation results are as follows: RCNN- 20 FPS, YOLO V7- 40 FPS, YOLO V8- 45 FPS

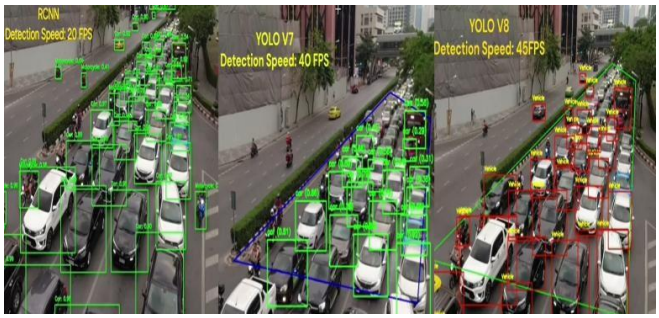

Figure 1. Detection Speed Comparison

From these results, YOLO V8 is the efficient model for vehicle detection and it is the selected model for this research. By using this model, the autonomous traffic management system can achieve faster and more accurate vehicle detection.

The YOLO V8 model is further trained with the custom ambulance dataset and attained an average precision of 99.1% for ambulance detection. This ambulance detection system will detect the ambulance and prioritize the lane to ensure the passage of an ambulance in emergencies.

C. Traffic Signal Management

Initially, all lanes will have a red signal. The YOLO V8 model will detect and count the number of vehicles in each lane with high accuracy. This research uses an algorithm that allows a green signal based on the vehicle count. The lane with the most vehicles gets the first green signal. This approach will improve the traffic flow and reduce delays.

The signal timing is dynamically adjusted based on vehicle count. Each vehicle is allotted 2 seconds to pass through the intersection. The minimum green signal that the system allows is 10 seconds and the maximum is 60 seconds for a lane. If no vehicles are detected in the lane during an active green signal, then the system automatically switches the signal to yellow before turning back to red.

Additionally, this approach is particularly important during off-peak hours, when few lanes may have little to no traffic. Instead of following a fixed cycle, this system skips empty lanes and reduces unnecessary delays for vehicles in other lanes.

D. Emergency Vehicle Prioritization

The model continuously scans for ambulances within the traffic lanes. Whenever it detects an ambulance, the system overrides the standard signals to prioritize the ambulance. The lane with the ambulance detected gets the green signal and all other lanes get red to ensure the passage of the ambulance.

To prevent unnecessary traffic delays the ambulance lane is allotted a green signal duration of 30 seconds. During this duration, the ambulance will pass the intersection. Once this 30-second duration completes then the standard traffic

signals resume from where it paused, dynamically adjusting the green signal based on the vehicle count. This emergency vehicle prioritization in this system will provide a smooth flow for the ambulance in emergencies.

E. Real-time Traffic Detection

The real-time traffic detection is the main phase of this research. The YOLO V8 continuously monitors the feed from CCTV cameras at the intersections. This model has high detection speed and accuracy in detecting and counting the vehicles. By implementing this in real-time the drawbacks of traditional methods can be solved by reducing delays and dynamically adjusting the signal times.

By implementing this system, the drawbacks of traditional traffic management methods such as manual control and pre-set signal durations can be overcome. The system ensures that lanes with higher vehicle counts receive longer green signals reducing delays. Additionally, if a lane has no detected vehicles, the system automatically transitions the signal from green to yellow and then red.

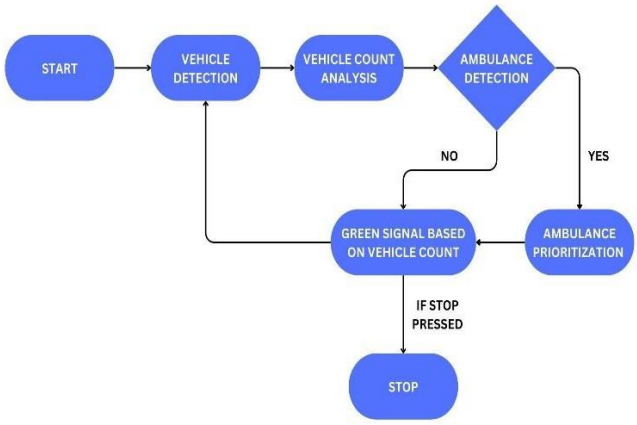

Figure 2. Autonomous Traffic Management System flowchart

The fig. 2. is the flowchart of this research. The YOLO V8 model is used for Vehicle detection and counting. It has also been trained with a custom ambulance dataset. So, that it can undergo ambulance detection. If no ambulance is detected then the general algorithm will be considered.

IV. RESULTS AND DISCUSSIONS

The COCO dataset is used to find the efficient model among the RCNN, YOLO V7 & YOLO V8. The efficiency of the models is calculated in detection speed in terms of Frames Per Second (FPS). The output of the models is mentioned in table 1.

MODEL	DETECTION SPEED (In FPS)
RCNN	20 FPS
YOLO V7	40 FPS
YOLO V8	45 FPS

Table 1. Comparison between models

This research utilizes YOLO V8 because of its high detection speed. The COCO dataset doesn't contain the ambulance category. So, a custom ambulance dataset is been created and used to train the model to detect the ambulances in the lanes. The training results are shown in fig. 3.

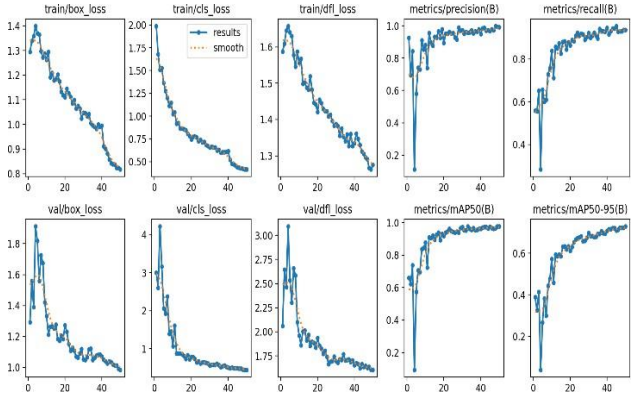

Figure 3. Training results of ambulance detection

All the CCTV feeds will undergo the YOLO V8 model which will draw the boxes and label all the vehicles including ambulances. The output of this system is as follows

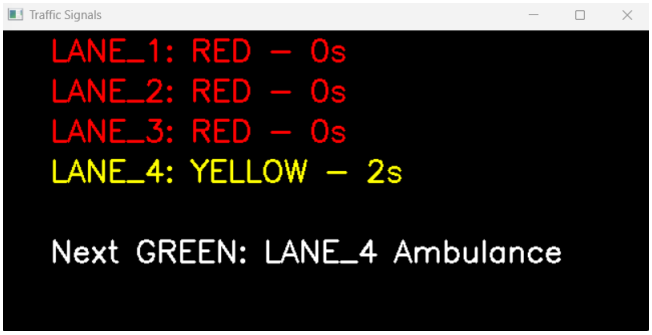

Figure 4. Traffic Signals with ambulance detected

Figure 5. Ambulance detection in lane 4 from CCTV feeds

V. CONCLUSION AND FUTURE WORK

This research presents an autonomous traffic management system utilizing deep learning-based object detection models to manage traffic flow and prioritize emergency vehicles. This study compared RCNN, YOLO V7, and YOLO V8 in terms of detection speed and accuracy. It determined that YOLO V8 is the most efficient model by achieving 45 FPS with high detection speed.

A custom dataset for ambulance detection was created and trained using YOLO V8 achieving an average precision of 99.1% and an accuracy of 97.7%. The system successfully detected ambulances in real-time and dynamically adjusted traffic signals to ensure uninterrupted movement for emergency vehicles.

In the future, a custom time should be calculated and allotted for each type of vehicle to cross the intersection based on the distance from the signal to cross. Another feature is to train the model with other emergency vehicles like fire engines, police cars, etc.

VI. REFERENCES

[1] Mamoona Humayun, Maram Fahhad Almufareh and Noor Zaman Jhanjhi, "Autonomous Traffic System for Emergency Vehicles," Electronics, vol. 11, pp. 510-528, February 2022.

[2] Pankaj Kunekar, Yogita Narule, Richa Mahajan, Shantanu Mandlapure, Eshan Mehendale, and Yashashri Meshram, "Traffic Management System Using YOLO Algorithm," in Proceedings of Engineering, Dubai, United Arab Emirates, 2024.

[3] Dr. Sreelatha R, Mahalakshmi B S, Riya Yadav, Shreyam Pandey, and Vandit Agarwal, "Artificial Intelligence Based Autonomous Traffic Regulator," in Proceedings of CEUR Workshop, New Delhi, India, 2023.

[4] Nitin Sakhare, Mrunal Hedau, Gokul B., Omkar Malpure, Trupti Shah, and Anup Ingle, "Smart Traffic: Integrating Machine Learning, and YOLO for Adaptive Traffic Management System," International Journal of INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING, vol. 12, no. 12, pp. 347-355, January2024.

[5] Izhak Rubin, Andrea Baiocchi, Yulia Sunyoto, and Ion Turcanu, "Traffic management and networking for autonomous vehicular highway systems," Ad Hoc Networks, vol. 83, pp. 125-148, February 2019.

[6] Christofel Rio Goenawan, "ASTM: Autonomous Smart Traffic Management System Using Artificial Intelligence CNN and LSTM," arXiv preprint arXiv:2410.10929, October 2024.