**TITLE OF THE PROJECT**

**SENIOR DESIGN PROJECT REPORT**

*Submitted in partial fulfillment of the*

*requirement for the award of the*

*Degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

*by*

**S. PRANEETH SAI (21BEC8483)**

**N. MANISH (21BEC8226)**

**S. MONIKA SIRI (21BEC8899)**

*Under the Guidance of*

**Prof. Hemanth Kumar Reddy**

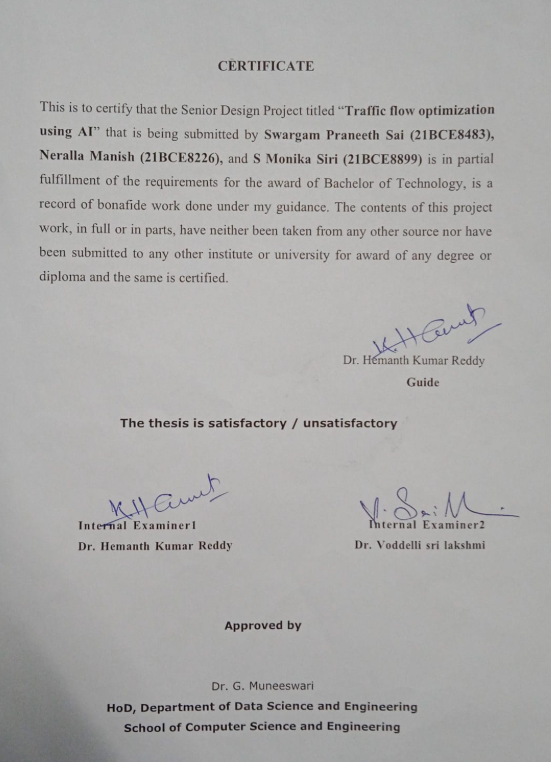


SCHOOL OF COMPUTER ENGINEERING

VIT-AP UNIVERSITY

AMARAVATI- 522237

*MAY 2025*

**

**ACKNOWLEDGEMENTS**

We would like to express our deepest gratitude to all those who have supported and guided us throughout the completion of our project, *"Optimization of Traffic Using AI."*

First and foremost, we would like to thank our respected guide, **Prof. Hemanth Kumar reddy sir**, for their constant support, valuable insights, and encouragement throughout the development of this project. Their guidance played a crucial role in shaping our work and helping us stay focused on our objectives.

We sincerely thank our peers and teammates for their collaboration, enthusiasm, and dedication, which made this project both productive and enjoyable.

Last but not least, we are grateful to our families and friends for their continuous motivation, patience, and understanding throughout this journey.

This project has been a great learning experience for us, and we hope it contributes meaningfully to the field of intelligent transportation systems and real-world traffic optimization.

**ABSTRACT**

In urban areas, efficient traffic management is essential for minimizing delays and enhancing emergency response times. This project introduces a machine learning-based approach for intelligent traffic management at four-way signals, focusing on emergency vehicle identification and prioritization. The methodology involves several steps: identifying emergency vehicles using YOLOv8 and RCNN techniques on video data, counting and categorizing all vehicles at the intersection, and analyzing the videos to predict the optimal time required to clear traffic. Utilizing a video database from COCO, the system processes inputs to detect emergency vehicles and provides real-time alerts to prioritize their passage. If an emergency vehicle is identified, a message displays the vehicle's location and clears the route promptly. When no emergency vehicles are present, the system estimates the time needed to clear the intersection based on the total vehicle count, optimizing traffic flow. This approach balances routine traffic management with emergency response needs, aiming to reduce congestion and enhance overall traffic efficiency. The results indicate that intelligent traffic management using advanced techniques can significantly improve urban mobility and emergency response capabilities.

**Keywords:** Intelligent Traffic Management, Machine Learning, YOLOv8, RCNN, Emergency Vehicle Identification, Video Analysis, COCO Database, Traffic Optimization, Urban Mobility, Real-time Alerts.

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.No.** | **Chapter** | **Title** | **Page Number** |
| **1.** |  | **Acknowledgement** | **3** |
| **2.** |  | **Abstract** | **4** |
| **3.** |  | **List of Figures and Table** | **6** |
| **4.** | **1**  **1.1**  **1.2**  **1.3** | **Introduction**  **Objectives**  **Problem Statement**  **Scope** | **7**  **7**  **7**  **8** |
| **5.** | **2.1** | **Project Introduction** | **9** |
| **6.** | **3.1** | **System Analysis** | **10** |
| **7.** | **4**  **4.1**  **4.2** | **Hardware Analysis**  **Software requirements**  **Hardware requirements** | **13**  **13** |
| **8.** | **5** | **Methodologies** | **15** |
| **9.** | **6** | **Result** | **21** |
| **10.** | **7** | **Appendix** | **23** |
| **11.** | **8** | **References** | **29** |

**List of Tables**

**Table No. Title Page No.**

- - -

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page No.** |
| 1 | Project Flow | 12 |
| 2 | System Architecture | 14 |
|  |  |  |

**1. INTRODUCTION**

**1.1 OBJECTIVE OF PROJECT:**

The project's objective is to create an intelligent traffic management system using machine learning to improve traffic control at four-way intersections, with a focus on emergency vehicle prioritization. It aims to accurately detect emergency vehicles using YOLOv8 and RCNN from video data, provide real-time alerts for their prioritization, and optimize traffic flow by predicting the time needed to clear intersections based on vehicle counts. The system integrates a COCO video database for robust model training and validation. Ultimately, the project seeks to balance routine traffic management with emergency needs, enhancing urban mobility and response efficiency.

**1.2 PROBLEM STATEMENT:**

Urban traffic management systems often fail to dynamically adapt to real-time conditions, leading to inefficiencies such as delays for emergency vehicles and increased congestion. Existing systems typically use fixed-time or actuated controls, which lack the ability to prioritize emergency vehicles or optimize traffic flow based on current traffic patterns. This results in suboptimal response times for emergencies and increased congestion during peak hours. The problem is exacerbated by minimal use of real-time data analysis and limited adaptability, necessitating a more advanced solution that integrates machine learning to enhance both traffic efficiency and emergency vehicle prioritization.

**1.3 MOTIVATION:**

The motivation for this project stems from the critical need to improve urban traffic management systems to better handle both routine traffic and emergency situations. Traditional systems often fall short in dynamically adapting to real-time traffic conditions, leading to increased congestion and delays for emergency vehicles. Enhancing traffic management with machine learning techniques offers a promising solution to these issues. By leveraging advanced algorithms for real-time vehicle detection and prioritization, this project aims to significantly reduce emergency response times and optimize traffic flow, ultimately improving overall urban mobility and safety. This approach not only addresses current inefficiencies but also aligns with the growing demand for smarter, more adaptive city infrastructure.

**1.4 SCOPE:**

The scope of this project encompasses:

1. Development of Machine Learning Models: Implementing YOLOv8 and RCNN techniques for real-time identification and classification of vehicles from video data.

2. Real-Time Traffic Management: Creating a system to prioritize emergency vehicles by providing real-time alerts and optimizing traffic signal timings based on detected vehicle counts.

3. Integration with COCO Database: Utilizing the COCO video database for training and validating the machine learning models to ensure high accuracy and reliability.

4. Traffic Flow Optimization: Analyzing traffic patterns to estimate the optimal time needed for clearing intersections and managing routine traffic effectively.

5. System Testing and Evaluation: Assessing the system's performance in simulated and real-world scenarios to ensure it meets objectives and improves traffic efficiency and emergency response.

6. User Interface: Developing a user-friendly interface for traffic management authorities to monitor and control the system.

The project aims to enhance urban traffic management, reduce congestion, and improve emergency vehicle response times.

**2. INTRODUCTION**

**2.1 PROJECT INTRODUCTION:**

Effective traffic management is crucial for optimizing urban mobility and ensuring timely emergency responses. Traditional traffic control systems often rely on fixed-time or actuated signal controls that do not adjust dynamically to real-time traffic conditions or prioritize emergency vehicles effectively. This can result in increased congestion, delays, and inefficient traffic flow.

This project proposes an innovative solution using machine learning techniques to address these challenges. By leveraging YOLOv8 and RCNN for real-time vehicle detection and classification, the system aims to enhance traffic management at four-way intersections. The approach involves analyzing video data to identify and prioritize emergency vehicles, providing real-time alerts for route clearance, and estimating the optimal time needed to clear the intersection based on current traffic patterns. Utilizing the COCO video database for model training ensures robust performance and accuracy.

The goal is to develop a system that not only improves traffic flow and reduces congestion but also enhances emergency vehicle response times, ultimately leading to a more efficient and responsive urban traffic management system.

**3. SYSTEM ANALYSIS**

**3.1 Existing System**

Existing urban traffic management systems often rely on fixed-time or actuated traffic signals. Fixed-time signals follow predetermined schedules, leading to inefficient traffic flow and congestion. Actuated signals adjust based on vehicle detection but lack real-time adaptability and do not prioritize emergency vehicles effectively. These systems are limited by their static nature, requiring manual intervention for adjustments and failing to utilize real-time data for optimization. Consequently, they struggle with emergency vehicle delays and congestion during peak hours. This underscores the need for a more dynamic and intelligent approach to enhance traffic efficiency and emergency response.

**3.2 Disadvantages of the Existing System**

1. Inefficient Traffic Flow: Fixed-time signals operate on predetermined schedules, leading to suboptimal traffic management and increased congestion.

2. Limited Adaptability: Actuated signals react only to vehicle presence without adjusting to real-time traffic conditions or prioritizing emergency vehicles.

3. Manual Intervention Required: Adjustments to traffic signal timings often require manual changes, making the system reactive rather than proactive.

4. Poor Emergency Vehicle Prioritization: Existing systems typically do not effectively prioritize emergency vehicles, causing potential delays in response times.

5. Minimal Real-Time Data Utilization: There is limited use of real-time data for optimizing traffic flow and predicting congestion patterns.

6. Safety and Efficiency Challenges: Problems include increased delays during peak hours and inadequate support for efficient urban mobility and sustainable practices.

**3.3 Proposed System**

The proposed system introduces a machine learning-based approach to enhance traffic management at four-way intersections. Utilizing YOLOv8 and RCNN techniques, the system detects and classifies vehicles in real-time, with a focus on prioritizing emergency vehicles. It processes video data to provide real-time alerts for clearing emergency vehicles and estimates optimal traffic signal timings based on current traffic conditions. By integrating the COCO video database for model training, the system aims to dynamically adapt to traffic patterns, reduce congestion, and improve emergency response times, offering a more efficient and responsive solution compared to existing traffic management systems.

**3.4 Advantages of the Proposed System**

1. Dynamic Adaptability: The system uses machine learning to adjust traffic signal timings in real-time based on current traffic conditions, improving traffic flow and reducing congestion.

2. Emergency Vehicle Prioritization: YOLOv8 and RCNN techniques enable accurate detection and prioritization of emergency vehicles, ensuring timely response and route clearance.

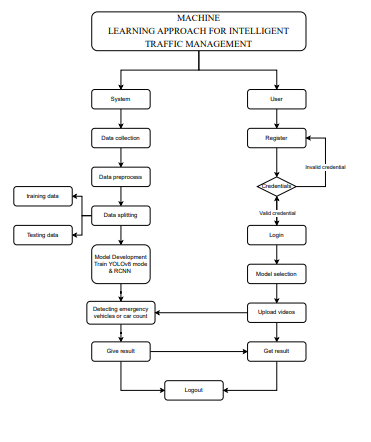
3. Real-Time Data Processing: The system leverages real-time video data to provide immediate alerts and make dynamic adjustments, enhancing responsiveness compared to traditional systems.

4. Optimized Traffic Management: Accurate vehicle classification and counting allow for precise estimation of optimal signal timings, improving overall traffic efficiency.

5. Integration with COCO Database: Utilizes a robust dataset for training, ensuring high accuracy and reliability in vehicle detection and classification.

6. Enhanced Urban Mobility: Balances routine traffic management with emergency response needs, leading to improved overall traffic flow and safety.

* 1. **PROJECT FLOW**



**4. HARDWARE & SOFTWARE REQUIREMENTS**

**4.1 SOFTWARE REQUIREMENS**

Operating System : Windows 7/8/10

Server side Script : HTML, CSS, Bootstrap & JS

Programming Language : Python

Libraries :Flask, Torch, Tensorflow, Pandas, Mysql.connector

IDE/Workbench : VSCode

Server Deployment : Xampp Server

Database : MySQL

**4.2 HARDWARE REQUIREMENTS**

Processor - I3/Intel Processor

RAM - 8GB (min)

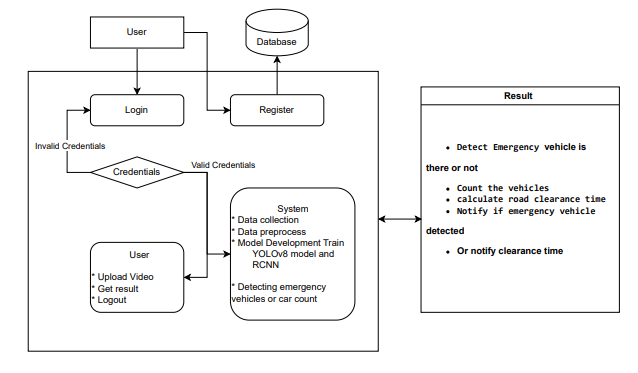
Hard Disk - 128 GB

Key Board - Standard Windows Keyboard

Mouse - Two or Three Button Mouse

Monitor - Any

**4.3 ARCHITECTURE**:



1. **METHODOLOGIES**

**YOLOv8 for Traffic Management**

The methodology section outlines the application of YOLOv8 in developing an advanced traffic management system. YOLOv8, or "You Only Look Once version 8," is a state-of-the-art object detection model that significantly enhances real-time vehicle detection and classification. This model builds on its predecessors with improved accuracy, speed, and efficiency, making it well-suited for traffic management applications. Here, we explore how YOLOv8 is employed to address key challenges in urban traffic management, focusing on its integration into the proposed system.

**1. Introduction to YOLOv8:**

YOLOv8 is the latest iteration in the YOLO (You Only Look Once) series, renowned for its capability to detect and classify objects within images in real-time. YOLOv8 offers advancements in both architecture and training methodologies compared to earlier versions. Its core strength lies in its ability to perform object detection with high speed and accuracy, which is crucial for dynamic environments such as urban traffic management. YOLOv8 integrates a more efficient backbone network and improved detection heads that contribute to enhanced performance, especially in handling diverse and complex traffic scenarios.

**2. Data Preparation and Annotation:**

For YOLOv8 to effectively detect and classify vehicles, it requires a well-annotated dataset. In this project, the COCO (Common Objects in Context) video database is used, providing a rich source of video data with comprehensive annotations for various vehicle types and traffic scenarios. The dataset includes diverse urban environments, lighting conditions, and vehicle types, which helps train the YOLOv8 model to generalize well across different conditions.

Data preparation involves several steps: collecting and organizing video data, annotating vehicles with bounding boxes, and converting these annotations into a format compatible with YOLOv8. Annotation tools like LabelImg or VoTT are used to mark vehicles in the video frames, ensuring that the model learns to identify and classify vehicles accurately. The annotated data is then split into training, validation, and test sets to train and evaluate the model's performance.

**3. Model Training:**

Training YOLOv8 involves several stages, including model configuration, data loading, and hyperparameter tuning. The YOLOv8 architecture consists of a backbone network for feature extraction, a neck for feature fusion, and detection heads for predicting bounding boxes and class labels. The model is configured with specific parameters, such as the number of classes (e.g., cars, trucks, emergency vehicles) and anchor boxes, to tailor it to the traffic management application.

The training process uses the prepared COCO video dataset. The model is trained using a combination of loss functions, including classification loss and localization loss, which help the model accurately predict vehicle positions and types. During training, techniques like data augmentation, learning rate scheduling, and early stopping are employed to improve model robustness and prevent overfitting. The training process is computationally intensive and requires powerful hardware, such as GPUs, to handle large volumes of video data efficiently.

**4. Real-Time Vehicle Detection:**

Once trained, YOLOv8 is integrated into the traffic management system for real-time vehicle detection. The model processes live video feeds from traffic cameras, detecting and classifying vehicles as they enter the intersection. YOLOv8's ability to process frames at high speeds ensures that the system can handle the dynamic nature of urban traffic, providing timely and accurate information for traffic signal control.

The detection output includes bounding boxes around vehicles, along with their class labels and confidence scores. This information is used to identify the type of vehicles present at the intersection, including emergency vehicles, which are crucial for prioritization. YOLOv8's high accuracy and low latency make it an ideal choice for real-time applications where timely decision-making is essential.

**5. Vehicle Classification and Prioritization:**

In addition to detecting vehicles, YOLOv8 enables the classification of different vehicle types. For traffic management, accurate classification is important for differentiating between regular vehicles and emergency vehicles. YOLOv8's advanced detection capabilities allow the system to classify vehicles into categories such as cars, trucks, and ambulances with high precision.

Emergency vehicle prioritization is a key feature of the proposed system. When an emergency vehicle is detected, the system triggers real-time alerts to adjust traffic signal timings, ensuring that the vehicle can pass through the intersection without delay. YOLOv8's reliable classification helps ensure that emergency vehicles are promptly identified and given priority, enhancing response times and improving overall traffic management.

**6. System Integration and Testing:**

Integrating YOLOv8 into the traffic management system involves several technical considerations, including system architecture, data flow, and real-time processing. The model is deployed on servers or edge devices that interface with traffic cameras, processing video feeds and generating detection results.

Testing is a critical phase, involving both simulated and real-world scenarios. In simulations, the system's performance is evaluated using test data to measure accuracy, latency, and robustness. Real-world testing involves deploying the system in actual traffic environments to assess its effectiveness in dynamic conditions. Performance metrics such as detection accuracy, processing speed, and response time are monitored to ensure that the system meets the desired objectives.

1. **Performance Evaluation and Optimization:**

Evaluating the performance of YOLOv8 in the traffic management system involves analyzing key metrics, such as precision, recall, and F1-score, to assess the model's accuracy in vehicle detection and classification. Additionally, real-time processing speed and latency are evaluated to ensure the system can handle live video feeds effectively.

Optimization may involve fine-tuning the model, adjusting hyperparameters, or employing techniques like model pruning to improve efficiency. Continuous monitoring and iterative improvements help maintain the system's performance over time, adapting to changes in traffic patterns or environmental conditions.

**8. Conclusion:**

YOLOv8 plays a pivotal role in the proposed traffic management system, offering advanced capabilities for real-time vehicle detection and classification. Its integration into the system enhances the ability to manage traffic efficiently, prioritize emergency vehicles, and optimize signal timings. By leveraging YOLOv8's strengths, the project aims to address the limitations of traditional traffic management systems and improve overall urban mobility and safety.

**RCNN for Traffic Management:**

1. Introduction to RCNN:

Region-based Convolutional Neural Networks (RCNN) revolutionized object detection with their approach of combining region proposals with deep learning for high accuracy. RCNNs segment an image into regions, apply convolutional neural networks (CNNs) to each region, and then classify these regions. Despite being one of the earliest methods in object detection, RCNNs paved the way for more advanced models by demonstrating the effectiveness of region-based object detection. In the context of traffic management, RCNN can be leveraged to enhance vehicle detection and classification.

2. Data Preparation and Annotation:

For RCNN to perform optimally, it requires a well-annotated dataset. The process begins with collecting a diverse set of video frames or images from traffic scenes. The COCO (Common Objects in Context) dataset is particularly useful as it provides a wide range of annotated images with various vehicle types and urban scenarios.

Annotation involves marking vehicles within images using bounding boxes. Tools like LabelImg or VGG Image Annotator (VIA) are used for this purpose. Accurate annotations are crucial, as they directly impact the model's performance. Once annotated, the data is divided into training, validation, and test sets to ensure robust model evaluation.

3. Model Training:

Training an RCNN involves several steps: region proposal generation, feature extraction, and classification. The process begins with generating region proposals using a method like Selective Search. These proposals suggest potential areas in an image where objects might be located.

Each proposed region is then processed through a CNN to extract features. The CNN is typically pre-trained on a large dataset (e.g., ImageNet) and fine-tuned on the traffic data. These extracted features are fed into a classifier, such as a support vector machine (SVM), to determine the presence and type of vehicle in each region.

The model's performance is influenced by several factors, including the quality of the region proposals, the CNN architecture, and the choice of classifier. Hyperparameter tuning, such as adjusting learning rates and training epochs, is performed to optimize the model.

4. Model Evaluation and Fine-Tuning:

Evaluating the RCNN model involves assessing its accuracy in detecting and classifying vehicles. Metrics such as Precision, Recall, and Intersection over Union (IoU) are used to quantify performance. Precision measures the proportion of correctly identified vehicles among all detected instances, while Recall assesses the proportion of actual vehicles that were correctly identified.

IoU measures the overlap between the predicted bounding boxes and the ground truth annotations, providing insight into how accurately the model delineates object boundaries. Fine-tuning involves adjusting model parameters and re-training to improve these metrics.

5. Integration into Traffic Management System:

Once trained, the RCNN model is integrated into the traffic management system. It processes video feeds to identify and classify vehicles in real-time. The model's output is used to inform traffic signal adjustments and prioritize emergency vehicles.

The RCNN model's ability to detect and classify vehicles accurately contributes to optimizing traffic flow and improving response times. It complements other components of the traffic management system, such as YOLOv8, by providing additional insights into vehicle presence and behavior.

6. Challenges and Considerations:

While RCNN offers high accuracy, it is computationally intensive, requiring significant processing power and memory. This can be a challenge for real-time applications. Additionally, RCNN's reliance on region proposals may result in slower processing speeds compared to more recent models like YOLOv8. Balancing accuracy and performance is crucial for integrating RCNN into a real-time traffic management system.

In summary, RCNN plays a vital role in enhancing traffic management by providing accurate vehicle detection and classification. Its integration with other models and techniques ensures a comprehensive approach to managing urban traffic, improving overall efficiency and response times.

1. **IMPLEMENTATION AND RESULTS**

**6.1 Modules**

**1. System:**

**1.1 Data Collection:** Gather a comprehensive dataset of traffic video footage, capturing various urban traffic conditions, including emergency and non-emergency vehicles. The dataset is divided into training and testing subsets, typically with an 80% to 20% split.

**1.2 Data Preprocessing:** This step includes video frame extraction, annotation of vehicle types and emergency vehicles, and data augmentation to increase variability. Preprocessed video frames are then ready for model training and testing.

**1.3 Model Training with YOLOv8:** Train the YOLOv8 model using 80% of the dataset. This involves configuring the network, setting hyperparameters, and optimizing performance to accurately detect and classify vehicles in real-time.

**1.4 Model Training with RCNN:** Train the RCNN model using the same 80% subset of the dataset. This includes generating region proposals, feature extraction using CNNs, and classifying these regions to detect and identify vehicles and emergencies.

**1.5 Hybrid Model Integration:** Integrate YOLOv8 and RCNN models to leverage the strengths of both approaches. This module focuses on combining their outputs to improve detection accuracy and prioritize emergency vehicles effectively.

**1.6 Model Testing:** Evaluate the performance of the trained models using the remaining 20% of the dataset. Metrics such as precision, recall, F1 score, and detection accuracy are computed to assess the effectiveness of the models.

**1.7 Model Saving:** Save the trained models in formats such as .h5 or .pkl to preserve their parameters and weights for future use. This ensures that the models can be easily reloaded and used for real-time traffic management.

**1.8 Model Prediction and Real-Time Processing**: Implement the models for real-time video analysis. This module handles the input of live traffic video, applies the trained models to detect and classify vehicles, and prioritizes emergency vehicles. It outputs real-time alerts and traffic signal adjustments based on detected emergencies.

**2. User:**

**2.1 Register:** Users register an account in the system with their credentials to gain access to traffic management features.

**2.2 Login:** Users log in with their registered credentials to access the system and its functionalities.

**2.3 Upload Data:** Users can upload video footage for analysis. The system processes these videos to detect and classify vehicles and provide insights.

**2.4 View Results:** Users receive and view real-time predictions and alerts from the model, including information about emergency vehicle prioritization and traffic signal adjustments.

**2.5 Logout:** Users can log out of the system to ensure their session and personal data are secure.

**7. APPENDIX**

import cv2

import torch

import tempfile

from ultralytics import YOLO

import streamlit as st

from norfair import Detection, Tracker

import numpy as np

import torch.amp

# Load the YOLOv8 model for emergency vehicles

yolo\_v8\_emergency = YOLO('best\_emergency\_vehicle\_model.pt')  # Adjust path if needed

# Load the YOLOv5 model for non-emergency vehicles

yolo\_v5\_non\_emergency = torch.hub.load('ultralytics/yolov5', 'yolov5s')  # Pre-trained YOLOv5 model

# Define labels for emergency and non-emergency vehicles

emergency\_labels = ['Police Car', 'Police Van', 'Fire Truck', 'Ambulance']

non\_emergency\_labels = ['car', 'bus', 'truck', 'motorcycle']

st.title("ADVANCED TRAFFIC FLOW OPTIMIZATION FOR INTELLIGENT TRAFFIC SYSTEM - Emergancy Vehicle Detection")

# File uploader for multiple videos

uploaded\_files = st.file\_uploader(

    "Upload up to 4 Videos", type=["mp4", "mov", "avi", "mkv"], accept\_multiple\_files=True

)

def create\_detections(results, labels, model\_type="yolov8"):

    """Convert YOLO detection results to Norfair detections for tracking."""

    detections = []

    if model\_type == "yolov8":

        if isinstance(results, list):

            results = results[0]

        if hasattr(results, 'boxes'):

            for box in results.boxes:

                x1, y1, x2, y2 = map(int, box.xyxy[0])

                centroid = np.array([[(x1 + x2) / 2, (y1 + y2) / 2]])

                label = labels[int(box.cls)]

                conf = box.conf[0].item()

                # Check if the label matches any emergency or non-emergency vehicle

                if label in emergency\_labels + non\_emergency\_labels:

                    detections.append(

                        Detection(

                            centroid,

                            data={"label": label, "conf": conf, "box": (x1, y1, x2, y2)}

                        )

                    )

    elif model\_type == "yolov5":

        if hasattr(results, 'xyxy'):

            for result in results.xyxy[0]:

                if len(result) >= 6:

                    x1, y1, x2, y2, conf, cls = result[:6]

                    label = labels[int(cls)]

                    centroid = np.array([[(x1 + x2) / 2, (y1 + y2) / 2]])

                    if label in emergency\_labels + non\_emergency\_labels:

                        detections.append(

                            Detection(

                                centroid,

                                data={"label": label, "conf": conf, "box": (int(x1), int(y1), int(x2), int(y2))}

                            )

                        )

    return detections

if uploaded\_files:

    for idx, uploaded\_file in enumerate(uploaded\_files[:4]):  # Process up to 4 videos

        st.write(f"### Processing Video {idx + 1}: {uploaded\_file.name}")

        tfile = tempfile.NamedTemporaryFile(delete=False)

        tfile.write(uploaded\_file.read())

        cap = cv2.VideoCapture(tfile.name)

        stframe = st.empty()

        emergency\_detected = False  # Reset for each video

        # Unique IDs for each video

        unique\_emergency\_ids = set()

        unique\_non\_emergency\_ids = set()

        # Initialize ByteTrack tracker

        tracker = Tracker(distance\_function="euclidean", distance\_threshold=30)

        # Process each frame of the video

        while cap.isOpened():

            ret, frame = cap.read()

            if not ret:

                break

            # Run YOLOv8 model for emergency vehicle detection

            emergency\_results = yolo\_v8\_emergency(frame)

            # Run YOLOv5 model for non-emergency vehicle detection

            non\_emergency\_results = yolo\_v5\_non\_emergency(frame)

            # Generate detections from both models

            detections = create\_detections(emergency\_results, yolo\_v8\_emergency.names, model\_type="yolov8") + \

                         create\_detections(non\_emergency\_results, yolo\_v5\_non\_emergency.names, model\_type="yolov5")

            # Update tracked objects

            tracked\_objects = tracker.update(detections)

            # Draw bounding boxes on the frame

            for obj in tracked\_objects:

                label = obj.last\_detection.data["label"]

                x1, y1, x2, y2 = obj.last\_detection.data["box"]

                # If it’s an emergency vehicle

                if label in emergency\_labels:

                    if obj.id not in unique\_emergency\_ids:

                        unique\_emergency\_ids.add(obj.id)

                        emergency\_detected = True  # Trigger notification if emergency detected

                    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 2)

                    cv2.putText(frame, f'{label} {obj.id}', (x1, y1 - 10),

                                cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, (0, 0, 255), 2)

                # If it’s a non-emergency vehicle

                elif label in non\_emergency\_labels:

                    if obj.id not in unique\_non\_emergency\_ids:

                        unique\_non\_emergency\_ids.add(obj.id)

                    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

                    cv2.putText(frame, f'{label} {obj.id}', (x1, y1 - 10),

                                cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, (0, 255, 0), 2)

            # Update the frame in Streamlit

            # stframe.image(frame, channels="BGR", use\_container\_width=True)

            if frame is not None and isinstance(frame, np.ndarray):

                frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)  # Convert BGR to RGB

                stframe.image(frame, channels="RGB", use\_container\_width=True)

            else:

                st.warning("Frame is None or has an invalid format.")

        cap.release()

        # After the video ends, display the results

        if emergency\_detected:

            st.warning(f"🚨 Emergency vehicle detected in Video {idx + 1}. Please clear the road!")

        # Calculate and display road clearance time for this video

        non\_emergency\_count = len(unique\_non\_emergency\_ids)

        emergency\_count = len(unique\_emergency\_ids)

        clearance\_time = max(0, (non\_emergency\_count - emergency\_count) \* 3)

        st.write(f"### Results for Video {idx + 1}")

        st.write(f"Final Non-Emergency Vehicles: {non\_emergency\_count}")

        st.write(f"Estimated Road Clearance Time: {clearance\_time} seconds")

**8. REFERENCES**

[1] Jr-Jen Huang, Yi-Yu Chu, and Yen-Jen Chen, “The System Design and Implementation of Vehicle Management”, *Journal of Advances in Computer Networks*, Vol. 1, No. 1, March 2013.

[2] M. Abdoos, N. Mozayani, and A. L. Bazzan, “Traffic light control in non-stationary environments based on multi agent Q-learning,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1494–1507, Dec. 2011.

[3] M. Wiering, “Intelligent traffic light control,” *Institute of Information and Computing Sciences*, Utrecht University, Technical Report UU-CS-2000-29, 2000.

[4] M. L. Puterman, “Markov Decision Processes: Discrete Stochastic Dynamic Programming,” *Wiley Series in Probability and Mathematical Statistics*, 1994.

[5] C. G. Panayiotou, M. M. Matthaiou, and M. M. Polycarpou, “A multi-agent system for intelligent traffic light control,” *European Control Conference (ECC)*, 2002.

[6] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, “Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC),” *Transportation Research Record*, no. 2259, pp. 1–10, 2011.

[7] A. Prashanth and M. B. Sridharan, “Reinforcement learning based local traffic signal control,” *International Conference on Machine Learning and Applications (ICMLA)*, 2010.

[8] L. Li, Y. Lv, and F.-Y. Wang, “Traffic signal timing via deep reinforcement learning,” *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 3, pp. 247–254, 2016.

[9] Y. Wei, Y. Zheng, and Q. Yang, “Traffic signal control with adaptive reinforcement learning agents,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 4, pp. 1–24, 2019.

[10] T. Chu, J. Wang, L. Codecà, and Z. Li, “Multi-agent deep reinforcement learning for large-scale traffic signal control,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 1086–1095, 2020.

[11] H. A. Rakha and Y. Ding, “Impact of stops on vehicle fuel consumption and emissions,” *Journal of Transportation Engineering*, vol. 129, no. 1, pp. 23–32, 2003.

[12] S. C. Wong and N. H. C. Yung, “Modeling driver car-following behavior using fuzzy logic,” *Transportation Research Part B: Methodological*, vol. 33, no. 6, pp. 381–392, 1999.

[13] R. S. Sutton and A. G. Barto, “Reinforcement Learning: An Introduction,” *MIT Press*, 2nd Edition, 2018.

[14] P. Mannion, J. Duggan, and E. Howley, “An experimental review of reinforcement learning algorithms for adaptive traffic signal control,” *Autonomic Road Transport Support Systems (ARTS)*, 2016.

[15] J. K. Suhr and H. G. Jung, “Sensor fusion-based vehicle detection algorithm for autonomous driving under challenging lighting conditions,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 2, pp. 613–623, Feb. 2018.

[16] M. S. Hossain and G. Muhammad, “Cloud-assisted industrial internet of things (IIoT) – enabled framework for health monitoring,” *Computer Networks*, vol. 101, pp. 192–202, 2016.

[17] J. Tang, Y. Zhang, Y. Liu, and W. Wang, “A deep learning approach for traffic prediction using graph-based spatial features and long-term temporal features,” *Neural Networks*, vol. 113, pp. 1–9, 2019.

[18] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang, “Traffic Flow Prediction With Big Data: A Deep Learning Approach,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, April 2015.

[19] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, “A survey on deep transfer learning,” *International Conference on Artificial Neural Networks (ICANN)*, Springer, Cham, 2018.

[20] H. Ye, Y. Zhang, J. Tang, and Y. Wang, “A unified model for lane-level traffic speed prediction based on graph convolutional networks,” *Transportation Research Part C: Emerging Technologies*, vol. 104, pp. 1–15, 2019.

**BIODATA**



Name : S. Praneeth Sai

Mobile Number : 7036527689

E-mail : praneeth.21bce8483@vitapstudent.ac.in

Permanaent Address : 8-39/2, FF2 Vikas school road, Vijayawada



Name : N. Manish

Mobile Number : 7989753027

E-mail : manish.21bce8226@vitapstudent.ac.in

Permanaent Address : SN Puram, Manepalli street, Vijayawada



Name : S. Monika Siri

Mobile Number : 8555096372

E-mail : monika.21bce8899@vitapstudent.ac.in

Permanaent Address : 4-53, Satya SaI Nagar, Tirupathi