CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

With the increasing number of vehicles on roads, urban traffic congestion has become a major challenge, affecting both daily commuters and emergency response teams. Traffic lights in most cities operate on predefined timing sequences, which fail to account for dynamic traffic conditions. This inefficiency often results in significant delays, particularly for emergency vehicles such as ambulances, which require uninterrupted passage to hospitals. In many cases, ambulances are stuck in traffic, causing life-threatening delays for critical patients. The inability of conventional traffic management systems to detect and respond to ambulances in real time highlights the urgent need for an intelligent traffic control mechanism.To address these challenges, this project proposes an AI-driven Traffic Light Optimization System that employs computer vision and deep learning techniques to detect emergency vehicles using both image and audio inputs. The system utilizes YOLO (You Only Look Once) for vehicle detection and a CNN-based model for siren sound recognition, ensuring accurate and real-time ambulance identification. Upon detection, the system automatically adjusts traffic signals, allowing emergency vehicles to pass without unnecessary delays. By integrating modern AI techniques with real-time traffic management, this project aims to reduce emergency response times, enhance road safety, and improve overall traffic flow efficiency in urban areas..

1.2 ABOUT THE PROJECT

The Intelligent Traffic Light Control System for Emergency Vehicles is designed to prioritize the movement of ambulances and other emergency vehicles through urban traffic by dynamically adjusting traffic signals. This system leverages computer vision, sound recognition, and deep learning algorithms, particularly Convolutional Neural Networks (CNNs) and YOLO (You Only Look Once), to detect and classify emergency vehicles based on their visual features and siren sounds. Once an emergency vehicle is detected, the system communicates with traffic signal controllers to alter the traffic light sequence, ensuring a clear and uninterrupted passage for the ambulance. This real-time adjustment reduces delays caused by traffic congestion and significantly improves emergency response times.

The system consists of four core components:

1. Emergency Vehicle Detection – This module utilizes YOLO (You Only Look Once), a state-of-the-art object detection algorithm, to recognize ambulances based on their physical appearance, such as sirens, emergency lights, and vehicle shape. YOLO is well-suited for real-time detection, offering high-speed and accurate recognition of objects within video streams.

2. Communication and Control – Once an ambulance is detected, the system sends signals to traffic light controllers via wireless communication modules. The system dynamically adjusts the traffic signal timing, turning the traffic light green for the approaching ambulance while stopping other lanes as required. This ensures seamless coordination between road infrastructure and emergency response units, reducing the need for manual intervention by traffic police.

3. YOLO-Based Real-Time Optimization and Traffic Flow Management – The system continuously monitors traffic conditions using real-time video feeds and sensor data. By applying YOLO-based detection and advanced machine learning algorithms, it minimizes disruptions to regular traffic while ensuring emergency vehicles receive priority clearance.

4. Scalability and Smart City Integration – The proposed system is designed to be scalable and adaptable, allowing integration with existing traffic infrastructure in smart cities. The YOLO model’s versatility makes it suitable for deployment in both fixed and mobile surveillance systems, enhancing the overall efficiency of urban mobility.

1.3 PROJECT DESCRIPTION

The proposed project focuses on developing an intelligent traffic management system that leverages deep learning techniques, particularly the YOLO (You Only Look Once) model and CNN (Convolutional Neural Network), to detect emergency vehicles such as ambulances in real-time. The system captures vehicle images through the surveillance cameras installed at traffic intersections and processes them using the model YOLO for image-based recognition. Simultaneously, it utilizes CNN models to analyze audio signals from sirens, ensuring accurate emergency vehicle identification even in visually obstructed conditions such as heavy traffic, fog, or night-time. By integrating both image and audio-based detection, the system enhances the reliability of ambulance detection and improves traffic signal response, allowing priority clearance for emergency vehicles. To implement this, the project consists of multiple modules, including dataset collection, vehicle image processing, audio analysis, model construction, real-time interface integration, and vehicle classification. The dataset module gathers diverse emergency vehicle images from sources like Roboflow and siren audio datasets from Kaggle, ensuring robust training data. The YOLO model is employed for real-time image recognition, distinguishing ambulances from other vehicles, while CNN processes audio signals for additional verification.

1.4 PROBLEM STATEMENT

Traffic congestion at intersections is a major challenge, especially when emergency vehicles such as ambulances need to pass through crowded roads. Traditional traffic management systems are static and do not dynamically adjust to prioritize emergency vehicles, leading to delays that can be life-threatening. In many cases, emergency responders struggle to navigate through heavy traffic, often relying on manual intervention or drivers' willingness to yield, which is inconsistent and inefficient. Additionally, adverse conditions such as low visibility, high noise levels, or traffic density further hinder the quick identification of ambulances. Without an automated system to detect and clear the path for emergency vehicles, valuable time is lost, negatively impacting emergency response outcomes.

The absence of an intelligent system that integrates both image and audio-based detection of emergency vehicles poses a critical gap in modern traffic management. Current approaches predominantly rely on image processing alone, which can fail in obstructed or low-light conditions. Similarly, audio-based systems may struggle in noisy environments. Therefore, a robust, real-time system that combines YOLO-based image recognition and CNN-based siren detection is needed to enhance the accuracy of emergency vehicle identification. By automating traffic light control to prioritize ambulances, this project aims to minimize delays, optimize traffic flow, and ensure that emergency responders can reach their destinations as quickly and safely as possible.

CHAPTER 2

LITERATURE SURVEY

2.1 TITLE: IOT-BASED EMERGENCY VEHICLE SERVICES IN INTELLIGENT TRANSPORTATION SYSTEM

AUTHOR: CHOWDHURY, ABDULLAHI

DESCRIPTION:

This study presents an IoT-based approach to optimizing emergency vehicle services in intelligent transportation systems. The authors propose a real-time monitoring system where IoT sensors are integrated into traffic infrastructure to detect and prioritize emergency vehicles, reducing delays in critical situations. The system utilizes vehicle-to-infrastructure (V2I) communication to transmit real-time location and traffic congestion data to a central control system. Based on this information, the system dynamically adjusts traffic signals and suggests alternative routes to improve response times. The study highlights the importance of edge computing and cloud integration for processing large volumes of sensor data efficiently. The proposed system also integrates GPS and RFID technologies to enhance vehicle tracking accuracy. A key challenge discussed is data latency, which may affect decision-making speed, particularly in highly congested areas. Additionally, cybersecurity concerns, such as unauthorized access to IoT networks, must be addressed to prevent system failures. The authors validate their approach using simulations that demonstrate significant reductions in emergency vehicle travel times. They conclude that implementing IoT-based emergency response systems in urban areas can save lives by improving the efficiency of medical, fire, and police response units. Future improvements include integrating artificial intelligence to predict traffic conditions and optimize emergency routes in real time.

2.2 TITLE: V2X-BASED HIGHLY RELIABLE WARNING SYSTEM FOR EMERGENCY VEHICLES

AUTHOR: ARIKUMAR, KOCHUPILLAI SELVARAJ

DESCRIPTION:

This paper introduces a vehicle-to-everything (V2X)-based communication system designed to enhance road safety by providing reliable warnings for emergency vehicles. The authors propose a system where emergency vehicles broadcast their location and intent to surrounding vehicles and infrastructure, allowing traffic participants to react proactively. The approach ensures minimal delays in high-density urban environments, where emergency vehicles often struggle to navigate congestion. A key feature of this system is its ability to operate in both line-of-sight and non-line-of-sight conditions, improving the reliability of emergency alerts. The system is built upon 5G-enabled V2X communication protocols, ensuring real-time data transmission with minimal latency. The study also discusses the integration of machine learning models to predict potential traffic conflicts and optimize vehicle response strategies. One challenge noted is the adoption of V2X technology, as many vehicles on the road do not yet support such communication. Furthermore, cybersecurity risks associated with data transmission require robust encryption methods to prevent interference. The experimental results demonstrate a significant reduction in response times for emergency vehicles, emphasizing the potential of V2X systems in improving traffic safety. The authors recommend further testing in real-world conditions to validate scalability and reliability across different urban infrastructures.

2.3 TITLE: ACOUSTIC-BASED EMERGENCY VEHICLE DETECTION USING ENSEMBLE OF DEEP LEARNING MODELS

AUTHOR: MITTAL, USHA, AND PRIYANKA CHAWLA

DESCRIPTION:

This research focuses on detecting emergency vehicles through an acoustic-based approach using deep learning techniques. The authors develop a system that recognizes siren sounds in real time using an ensemble of convolutional neural networks (CNNs) and recurrent neural networks (RNNs). The motivation behind this approach is to overcome limitations in visual detection methods, which may struggle under poor visibility conditions such as fog, rain, or night-time traffic. The system extracts audio features such as Mel-Frequency Cepstral Coefficients (MFCCs) to differentiate emergency sirens from background noise. The model is trained on a diverse dataset containing sirens from different vehicle types, ensuring robustness across various real-world scenarios. A key advantage of this system is its adaptability to environments where visual sensors might fail, such as tunnels or highly congested areas. However, the study acknowledges challenges such as distinguishing sirens from other loud noises (e.g., honking or construction sounds), which can lead to false positives. The authors mitigate this issue by integrating additional contextual features, such as Doppler effect analysis and multiple microphone arrays for improved sound localization. Results demonstrate high accuracy in emergency vehicle detection, making the approach suitable for smart city implementations. Future work involves integrating the system with traffic management centers to automate real-time signal adjustments upon emergency detection.

2.4 TITLE: LIBSIGNAL: AN OPEN LIBRARY FOR TRAFFIC SIGNAL CONTROL

AUTHOR: MEI, HAO

DESCRIPTION:

This paper presents LibSignal, an open-source software library designed to optimize traffic signal control using machine learning algorithms. The system is built to support adaptive traffic management by dynamically adjusting signal timings based on real-time traffic flow data. Unlike traditional fixed-timing traffic signals, LibSignal utilizes deep reinforcement learning techniques to minimize congestion and improve overall traffic efficiency. The library is designed for flexibility, allowing researchers and city planners to integrate custom algorithms suited to specific urban environments. The study highlights the importance of real-time data collection, employing sensors, cameras, and vehicle-to-infrastructure (V2I) communication for accurate traffic monitoring. A key feature of LibSignal is its modular framework, enabling easy integration with existing traffic control systems. The study demonstrates significant reductions in vehicle waiting times, fuel consumption, and carbon emissions when deploying adaptive signal controls. However, the research also points out challenges such as computational demands and the need for high-quality training data to ensure effective decision-making. The authors suggest further improvements by incorporating predictive modeling to anticipate traffic patterns and proactively adjust signals. They conclude that LibSignal has the potential to revolutionize modern traffic management, improving both urban mobility and environmental sustainability.

2.5 TITLE: INTELLIGENT ROAD MANAGEMENT SYSTEM FOR AUTONOMOUS, NON-AUTONOMOUS, AND VIP VEHICLES

AUTHOR: NAEEM, AWAD BIN

DESCRIPTION:

This study introduces an intelligent road management system designed to handle mixed traffic scenarios involving autonomous, non-autonomous, and VIP vehicles. The system leverages artificial intelligence and Internet of Things (IoT) technologies to optimize road usage and prioritize emergency and VIP vehicle movement. The proposed framework consists of three key modules: (1) Vehicle Classification, which differentiates between autonomous, manual, and VIP vehicles using RFID and AI-based detection; (2) Dynamic Traffic Control, which adjusts signal timings and lane assignments based on real-time traffic density and vehicle prioritization; and (3) Communication Framework, which enables vehicles to exchange real-time traffic data with control centers for efficient route planning. A major advantage of this system is its ability to adapt dynamically to changing traffic conditions, reducing congestion and improving emergency response times. The study also discusses the ethical and regulatory challenges of prioritizing VIP vehicles, as well as the need for robust cybersecurity measures to prevent misuse. Simulation results show a marked improvement in travel efficiency, especially for emergency vehicles that require immediate clearance. The authors recommend future research on integrating blockchain for secure data transactions and expanding the system to include predictive analytics for congestion forecasting.

2.6 TITLE: A SURVEY OF VEHICULAR NETWORK SYSTEMS FOR ROAD TRAFFIC MANAGEMENT

AUTHOR: JURCZENIA, KAROL, AND JACEK RAK

DESCRIPTION:

This study provides a comprehensive review of vehicular network systems and their role in modern road traffic management. The authors analyze various communication models, including vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) frameworks, which play a crucial role in real-time traffic monitoring and congestion control. The paper highlights the evolution of Intelligent Transportation Systems (ITS), emphasizing how vehicular ad hoc networks (VANETs) improve traffic flow and safety. The study discusses multiple architectures, including centralized, decentralized, and hybrid models, and compares their performance in urban and highway environments. The authors focus on the impact of 5G and edge computing in reducing latency and enhancing decision-making capabilities. Key challenges identified include cybersecurity threats, interoperability between different vehicular communication standards, and infrastructure deployment costs. The paper also explores the role of artificial intelligence (AI) in predictive traffic management, showcasing how machine learning models optimize traffic signals and detect congestion patterns. The study concludes that while vehicular network systems significantly improve transportation efficiency, further research is needed to address data privacy and network scalability concerns. Future work suggests integrating blockchain for secure data transactions and refining AI-driven models for enhanced road safety.

2.7 TITLE: A COMPREHENSIVE SURVEY ON COOPERATIVE INTERSECTION MANAGEMENT FOR HETEROGENEOUS CONNECTED VEHICLES

AUTHOR: GHOLAMHOSSEINIAN, ASHKAN, AND JOCHEN SEITZ

DESCRIPTION:

This survey examines cooperative intersection management (CIM) strategies that enable seamless traffic flow for heterogeneous connected vehicles, including autonomous and manually driven cars. The authors review various communication protocols and decision-making algorithms designed to prevent accidents and improve intersection efficiency. The study emphasizes the role of connected vehicle technology, particularly V2V and V2I communication, in enabling real-time coordination among vehicles. The paper categorizes CIM approaches into centralized, decentralized, and hybrid systems, analyzing their strengths and limitations. A major focus is given to AI-based intersection control, where deep reinforcement learning models are used to dynamically adjust traffic signal timings based on real-time congestion levels. The authors highlight key challenges, including computational complexity, security risks, and sensor inaccuracies, which can impact the reliability of CIM systems. The study also discusses experimental results from real-world deployments, demonstrating reduced wait times and fuel consumption in smart intersections. Future research directions suggest integrating multi-agent reinforcement learning for decentralized decision-making and improving the reliability of vehicle-to-everything (V2X) communication. The authors conclude that cooperative intersection management is a crucial component of future smart cities, but successful implementation requires overcoming current technological and regulatory challenges.

2.8 TITLE: INTELLIGENT TRANSPORTATION SYSTEMS USING ROADSIDE INFRASTRUCTURE: A LITERATURE SURVEY

AUTHOR: CREß, CHRISTIAN, ZHENSHAN BING, AND ALOIS C. KNOLL

DESCRIPTION:

This paper presents a detailed survey of Intelligent Transportation Systems (ITS) that rely on roadside infrastructure to enhance traffic efficiency and safety. The study explores various sensor-based systems, including cameras, LiDAR, and radar, deployed along roads to monitor vehicle movement and optimize traffic control. The authors examine the role of roadside units (RSUs) in facilitating V2I communication and enabling real-time data exchange between vehicles and traffic management centers. The research discusses multiple ITS applications, such as adaptive traffic signalling, automated toll collection, and emergency vehicle prioritization. The authors highlight the growing importance of edge computing and artificial intelligence in analyzing traffic patterns and predicting congestion before it occurs. A key challenge identified is the high cost of infrastructure deployment and maintenance, particularly in developing regions. The study also addresses cybersecurity threats, as roadside units can be vulnerable to cyberattacks that disrupt traffic management. Experimental evaluations show that ITS systems integrated with AI and cloud computing can significantly reduce congestion and improve road safety. The authors recommend future research on integrating blockchain for secure data transmission and enhancing AI models for more accurate traffic predictions.

2.9 TITLE: SAFE INTERSECTION MANAGEMENT WITH COOPERATIVE PERCEPTION FOR MIXED TRAFFIC OF HUMAN-DRIVEN AND AUTONOMOUS VEHICLES

AUTHOR: AOKI, SHUNSUKE, AND RAGUNATHAN RAJKUMAR

DESCRIPTION:

This research focuses on improving intersection safety by leveraging cooperative perception techniques to manage mixed traffic consisting of human-driven and autonomous vehicles. The authors propose a system where vehicles and roadside infrastructure share sensory data to enhance situational awareness and prevent collisions. The study explores various cooperative perception models, including sensor fusion techniques that combine LiDAR, cameras, and radar data for comprehensive traffic monitoring. The authors highlight the importance of latency reduction in data transmission, emphasizing the role of edge computing in real-time decision-making. A key challenge discussed is the varying reaction times of human drivers versus autonomous systems, which can lead to unpredictable behavior at intersections. The research also addresses security concerns, as cyberattacks on cooperative perception systems could lead to manipulated traffic data and accidents. The proposed system is validated through simulations, demonstrating improved intersection throughput and reduced accident risks. Future recommendations include developing more robust AI algorithms to predict driver behavior and refining V2X communication protocols for enhanced reliability. The authors conclude that cooperative perception is a crucial step toward safer urban mobility but requires significant advancements in cybersecurity and standardization for large-scale deployment.

2.10 TITLE: ON THE INTEGRATION OF ENABLING WIRELESS TECHNOLOGIES AND SENSOR FUSION FOR NEXT-GENERATION CONNECTED AND AUTONOMOUS VEHICLES

AUTHOR: BUTT, FARAN AWAIS.

DESCRIPTION:

This paper explores the integration of wireless communication technologies and sensor fusion techniques to enhance the capabilities of next-generation connected and autonomous vehicles (CAVs). The authors review various wireless standards, including 5G, Dedicated Short-Range Communications (DSRC), and millimeter-wave (mmWave) communication, discussing their implications for vehicle connectivity and road safety. The study also delves into sensor fusion methodologies, where data from LiDAR, cameras, radar, and ultrasonic sensors are combined to improve environmental perception and decision-making. The authors analyze the benefits and challenges of real-time data processing, emphasizing the need for low-latency communication to support advanced driver-assistance systems (ADAS) and autonomous navigation. The paper highlights cybersecurity as a major concern, noting that hacking or signal interference can compromise vehicle control and safety. The authors also address power consumption issues, as high-bandwidth communication and sensor processing require significant energy resources. The study concludes that while integrating wireless technologies with sensor fusion can greatly enhance vehicle automation, future research should focus on optimizing energy efficiency and developing secure, fail-safe communication protocols. The authors propose using federated learning for collaborative model training among connected vehicles to improve real-time decision-making.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In the current traffic management systems, emergency vehicles, such as ambulances, often struggle to navigate through congested urban roads, leading to significant delays in reaching hospitals or accident sites. Traditional traffic light control mechanisms operate on fixed-time cycles or sensor-based systems that do not dynamically adjust for emergency vehicles. In some cases, traffic police manually intervene to clear the way for ambulances, but this method is inefficient, time-consuming, and prone to human error. Furthermore, existing traffic monitoring solutions mainly focus on general traffic regulation and do not provide a dedicated mechanism for real-time emergency vehicle prioritization. Some modern traffic systems have introduced RFID-based or GPS-based emergency vehicle tracking to allow selective signal overrides. However, these approaches have limitations, such as signal interference, high implementation costs, and dependency on specialized hardware in both vehicles and traffic signals. Additionally, sound-based emergency detection is rarely implemented, making it difficult to recognize ambulances in low-visibility conditions (such as during fog, heavy rain, or night-time).

3.1.1 DISADVANTAGES

- Fixed traffic light cycles do not adapt to emergency vehicle needs, causing delays.
- Manual traffic management for ambulances is inefficient .
- RFID and GPS-based systems require specialized hardware, increasing costs and complexity.
- Lack of sound recognition prevents the system from identifying ambulances using sirens.

3.2 PROPOSED SYSTEM

The proposed system introduces an intelligent traffic light optimization framework that enhances road traffic management by prioritizing emergency vehicles using deep learning-based detection techniques. Unlike traditional traffic control systems that rely on fixed timing mechanisms, this system dynamically adjusts traffic signals in real time based on the presence of ambulances. To achieve this, YOLO (You Only Look Once) is employed for vehicle image recognition, ensuring rapid and accurate identification of ambulances from real-time camera feeds. Additionally, a Convolutional Neural Network (CNN) model is used for audio classification, enabling the system to detect the characteristic siren sounds of emergency vehicles. By combining these two modalities image and audio the system significantly improves detection accuracy, even in complex traffic conditions where visibility might be low due to environmental factors like fog, rain, or night-time conditions. Once an ambulance is detected, the system processes the data and immediately communicates with the traffic signal controller, modifying the light sequence to allow smooth passage for the emergency vehicle. This is done by prioritizing the green signal for the detected ambulance's route while temporarily pausing traffic in other directions.

3.2.1 ADVANTAGES

- Enhances emergency response times.
- Reduces congestion and improves overall traffic management efficiency.
- Operates in real-time, ensuring quick and adaptive decision-making for emergency situations.
- Provides higher accuracy compared to traditional machine learning-based traffic systems.

3.3 ARCHITECTURE DIAGRAM

The architecture diagram illustrates an intelligent traffic light control system designed to prioritize ambulance passage using computer vision and deep learning techniques. It integrates YOLO-based object detection for identifying ambulances from real-time vehicle images and a CNN model for recognizing ambulance siren sounds from collected audio datasets. Image datasets are sourced from Roboflow, and sound datasets from Kaggle, which are then used to train respective models. A camera and microphone capture live traffic images and sounds, converting them into vector datasets for classification. Once an ambulance is detected, the system communicates with traffic signals to optimize their timing, ensuring an uninterrupted passage. This framework significantly enhances emergency response times by leveraging AI-based real-time ambulance recognition and traffic light adjustments, effectively reducing congestion delays in urban environments.

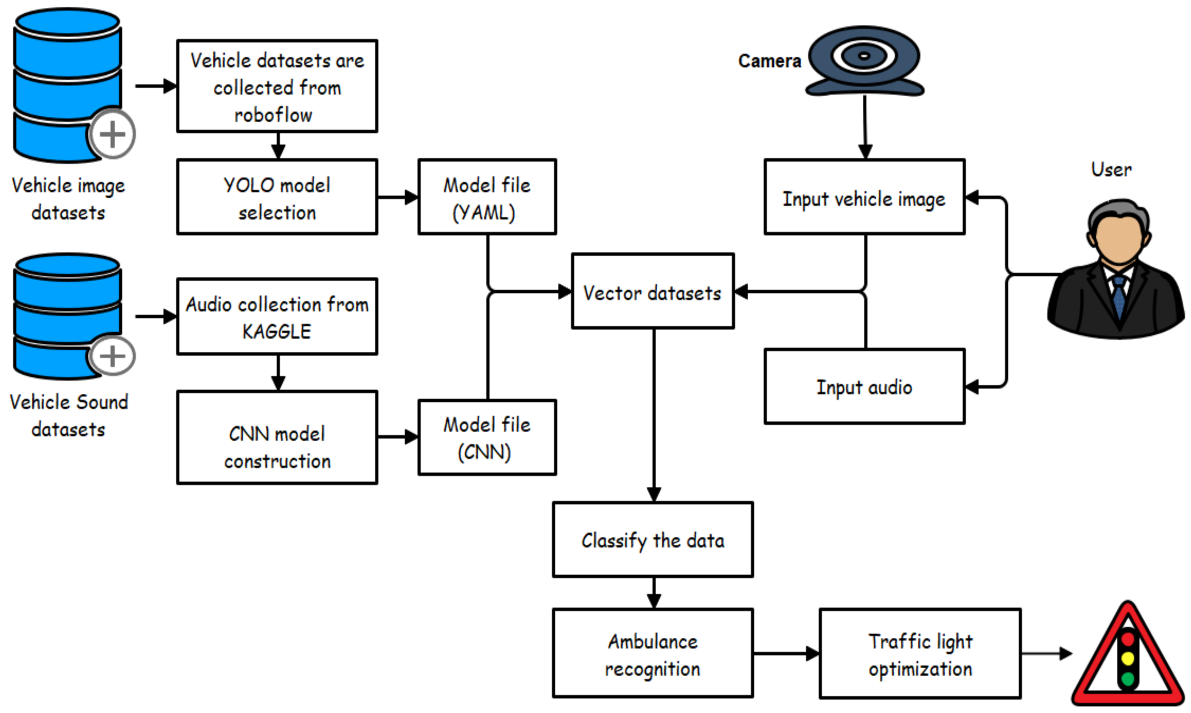


Figure 3.3: System architecture

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

▪ Processor	: Intel processor
▪ Hard disk	: 160 GB
▪ Keyboard	: Standard keyboard
▪ Monitor	: 15 inch color monitor
▪ RAM	: 8GB or higher
▪ Processor	: i5 processor or higher
▪ Camera with min 16 MP	

4.2 SOFTWARE REQUIREMENTS

▪ Operating system	: Windows OS
▪ Front End	: Python
▪ IDE	: PyCharm
▪ Libraries	: Open CV, Tensorflow, KERAS

4.3 SOFTWARE DESCRIPTION

4.3.1 FRONT END: PYTHON

Python is a high-level, interpreted programming language that is widely used in various domains such as web development, scientific computing, data analysis, artificial intelligence, machine learning, and more. It was first released in 1991 by Guido van Rossum and has since become one of the most popular programming languages due to its simplicity, readability, and versatility. One of the key features of Python is its easy-to-learn syntax, which makes it accessible to both novice and experienced programmers. It has a large standard library that provides a wide range of modules for tasks such as file I/O, networking, regular expressions, and more. Python also has a large and active community of developers who contribute to open-source libraries and packages that extend its capabilities. Python is an interpreted language, which means that it is executed line-by-line by an interpreter rather than compiled into machine code like C or C++. This allows for rapid development and testing, as well as easier debugging and maintenance of code. Python is used for a variety of applications, including web development frameworks such as Django and Flask, scientific computing libraries such as NumPy and Pandas, and machine learning libraries such as TensorFlow and PyTorch.

4.3.2 TENSORFLOW LIBARIES IN PYTHON

TensorFlow is an open-source machine learning framework developed by Google Brain Team. It is one of the most popular libraries for building and training machine learning models, especially deep neural networks. TensorFlow allows developers to build complex models with ease, including image and speech recognition, natural language processing, and more. One of the key features of TensorFlow is its ability to handle large-scale datasets and complex computations, making it suitable for training deep neural networks. It allows for parallelization of computations across multiple CPUs or GPUs, allowing for faster training times. TensorFlow also provides a high-level API called Keras that simplifies the process of building and training models.

4.2.3 PYCHARM

PyCharm is an integrated development environment (IDE) for Python programming language, developed by JetBrains. PyCharm provides features such as code completion, debugging, code analysis, refactoring, version control integration, and more to help developers write, test, and debug their Python code efficiently. PyCharm is available in two editions: Community Edition (CE) and Professional Edition (PE). The Community Edition is a free, open-source version of the IDE that provides basic functionality for Python development.

Features:

- Intelligent code completion
- Syntax highlighting
- Code inspection
- Code navigation and search
- Debugging
- Testing
- Version control integration
- Web development support

4.2.4 MY SQL

MySQL is the world's most used open source relational database management system (RDBMS) as of 2008 that run as a server providing multi-user access to a number of databases. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.

CHAPTER 5

MODULES DESCRIPTION

5.1 MODULE: DATASETS COLLECTION

The dataset collection module is responsible for gathering relevant vehicle images and audio data required for training the ambulance detection system. The vehicle images are collected from Roboflow, which provides a wide range of annotated datasets for training deep learning models. Similarly, emergency vehicle siren audio datasets are sourced from Kaggle, ensuring diverse sound variations are considered for accurate classification. The collected datasets undergo preprocessing, such as noise reduction, data augmentation, and normalization, to improve model performance. This module ensures the availability of high-quality datasets for YOLO-based image recognition and CNN-based audio classification, forming the foundation for real-time ambulance detection.

In this module, we collect vehicle images and audio to train the database, enabling classification in subsequent modules.

- **VEHICLE IMAGE**
  - Collect images of different types of emergency vehicles, including ambulances, fire trucks, police cars, and rescue vehicles.
  - These images should showcase vehicles from various angles, in different lighting conditions, and with varying levels of background clutter.
- **AUDIO DATASETS**
  - Record audio samples of ambulance sirens at different distances, volumes, and frequencies.
  - Include variations such as continuous sirens, intermittent sirens, and modulated sirens.

5.2 MODULE: VEHICLE IMAGE

The vehicle image module focuses on processing and analyzing vehicle images captured in real-time through cameras installed at traffic intersections. The images undergo preprocessing, including normalization and resizing techniques, before being fed into a YOLO (You Only Look Once) model for detection. YOLO is a real-time object detection system that identifies ambulances among various vehicles by recognizing their unique features, such as shape, size, and color patterns. The YOLOv5 architecture is employed due to its speed and accuracy in vehicle detection. Once the system classifies a vehicle as an ambulance, the information is sent to the traffic light control system, which optimizes signals to allow the emergency vehicle to pass without obstruction. This module ensures accurate ambulance detection in complex traffic scenarios, enhancing emergency response efficiency.

5.3 MODULE: AUDIO DATASETS

The audio dataset module deals with the collection, preprocessing, and classification of emergency vehicle siren sounds. CNN (Convolutional Neural Network) is used to extract key audio features such as frequency, amplitude, and pitch, allowing the system to distinguish ambulance sirens from general background noise. Kaggle datasets provide various emergency sirens, including those from fire trucks and police cars, enabling a diverse learning dataset. Spectrogram conversion is applied to the audio data, transforming it into a visual representation that the CNN model can analyze effectively. This module enhances the system’s capability to detect ambulances in visually obstructed conditions, such as dense traffic, fog, or night-time, thereby improving accuracy in emergency vehicle identification.

5.4 MODULE: MODEL CONSTRUCTION

The model construction module focuses on building deep learning architectures for accurate vehicle detection and classification. Two models are developed:

- YOLOv5 for image-based ambulance detection
- CNN for audio-based siren recognition

The YOLOv5 model uses convolutional layers with Darknet-based architecture, detecting vehicles in real-time with high precision. The model is trained on annotated datasets containing emergency vehicles and background objects to minimize false detections.

YOLO (You Only Look Once) is a real-time object detection algorithm used in the image-based process for ambulance detection. Unlike traditional methods that use region proposal-based approaches, YOLO treats object detection as a single regression problem, predicting bounding boxes and class probabilities directly from full images in one evaluation. The model divides the input image into a grid and assigns each cell the responsibility of detecting objects within it.

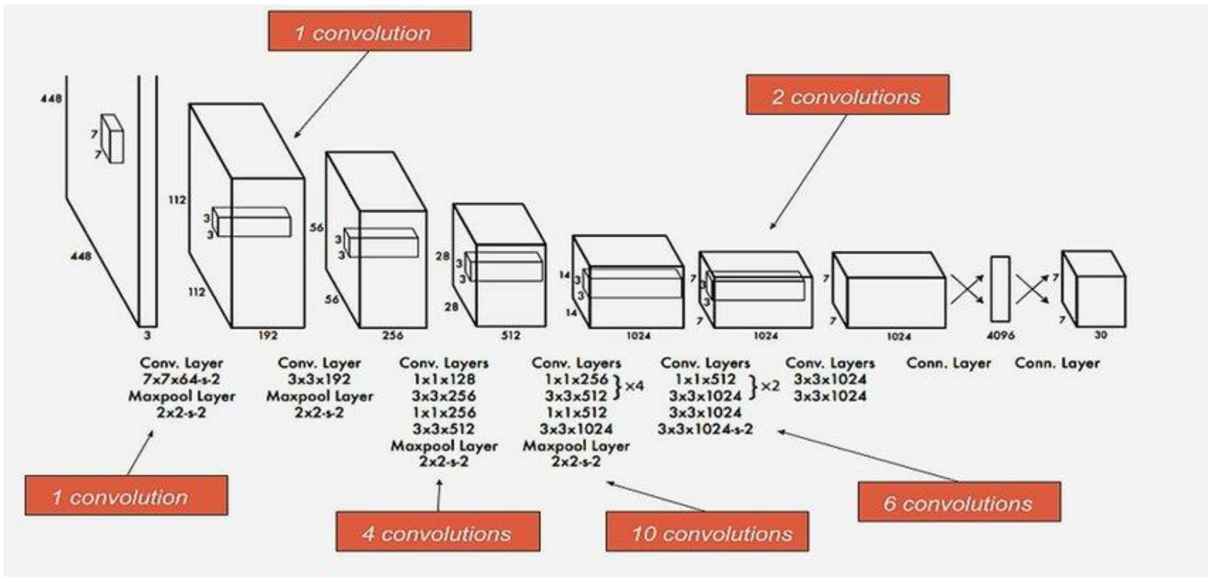

1 convolution
2 convolutions
448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer
1 convolution
4 convolutions
10 convolutions
6 convolutions

Figure 5.4.1: Working of Convolutional layers

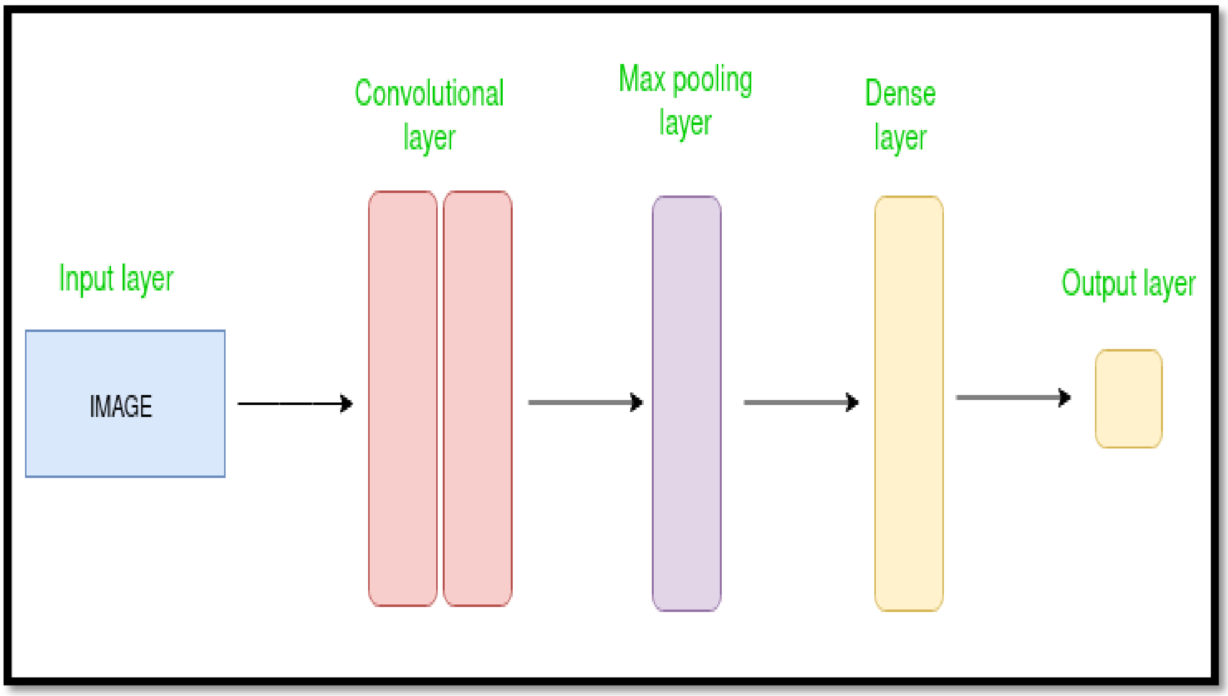


Figure 5.4.2: CNN layers

A convolutional neural network is a feed-forward network with the ability of extracting topological properties from the input image. It extracts features from the raw image and then a classifier classifies extracted features. CNNs are invariance to distortions and simple geometric transformations like translation, scaling, rotation and squeezing. Convolutional Neural Networks combine three architectural ideas to ensure some degree of shift, scale, and distortion invariance: local receptive fields, shared weights, and spatial or temporal sub-sampling. This feature which allows reducing the number of trainable parameters is called weight sharing technique and is applied in all CNN layer. With local receptive fields, elementary visual features including edges can be extracted by neurons. To extract the same visual feature, neurons at different locations can share the same connection structure with the same weights. The output of such a set of neurons is a feature map. This operation is the same as a convolution of the input image with a small size kernel. Multiple feature maps can be applied to extract multiple visual features across the image. Subsampling is used to reduce the resolution of the feature map, and hence reduce the sensitivity of the output to shifts and distortions. In our proposed CNN structure, multiple features can be extracted from each original eye data, and each feature has $n3$ dimensions.

5.5 MODULE: REAL-TIME INTERFACE

The real-time interface module provides continuous monitoring and control over traffic light optimization based on ambulance detection. This module consists of computer vision-based vehicle recognition and audio signal processing. Cameras at traffic signals capture vehicle movement, while microphones detect ambulance sirens. The YOLO-based model processes images in real-time, identifying emergency vehicles. Simultaneously, the CNN model classifies detected siren sounds to confirm an approaching ambulance. Once detection is verified, the system automatically adjusts traffic signals to green in the ambulance’s direction and red in other directions, ensuring an unobstructed route for emergency vehicles. Additionally, the real-time interface allows system operators to monitor live data, override controls if needed, and update detection models periodically for continuous improvement.

5.6 MODULE: CLASSIFY THE VEHICLES

The vehicle classification module ensures that the trained model is tested and evaluated for real-time deployment. The YOLO model classifies vehicles into categories such as ambulances, fire trucks, police cars, and general traffic. Data augmentation techniques (random rotations, brightness adjustments) are applied to improve model robustness. The system uses transfer learning to fine-tune pre-trained CNN models like ResNet or MobileNet, adapting them for specific vehicle classification tasks. Evaluation metrics such as accuracy, precision, recall, and F1-score are computed to validate model performance. Misclassified cases are analyzed to refine detection accuracy further. Finally, the classified vehicle data is integrated with traffic light optimization, allowing emergency vehicles to receive priority passage through congested intersections.

CHAPTER 6

SYSTEM DESIGN

6.1 UML DIAGRAMS

6.1.1 USE CASE DIAGRAM

A use case diagram is a type of UML diagram that represents the interactions between an actor (a user or system) and a system under various scenarios. The diagram provides a visual representation of the system's functionalities and the interactions between the actors and the system.

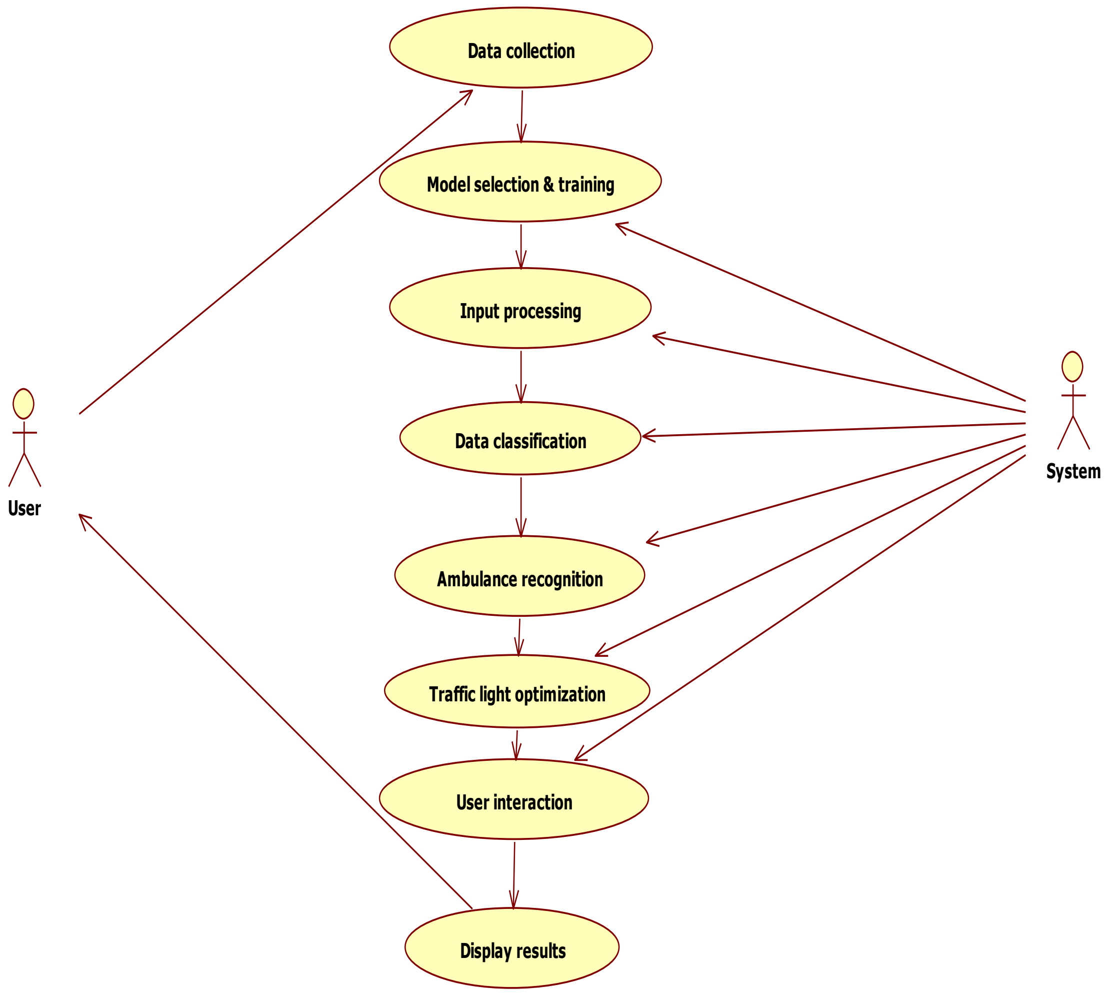


Figure 6.1.1: UseCase Diagram

6.1.2 SEQUENCE DIAGRAM

A sequence diagram is a visual representation of the interactions between different objects or components in a system or process over time. In the case of a currency recognition project for visually impaired people, a sequence diagram could illustrate the interactions between the user, the smartphone app, and the banknote recognition software.

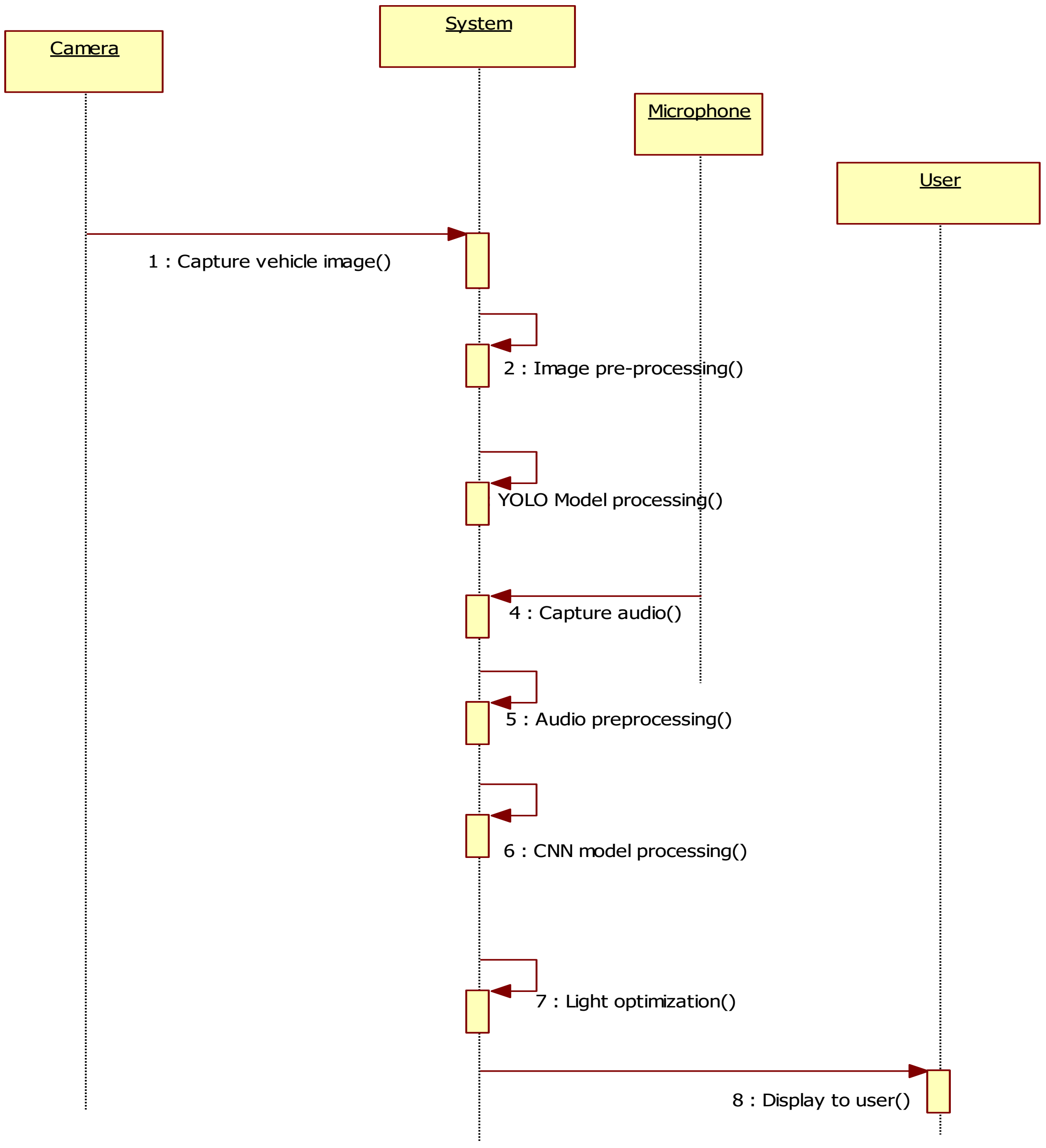


Figure 6.1.2: Sequence Diagram

6.1.3 ACTIVTY DIAGRAM

The activity diagram shows a unique kind of state diagram in which the majority of states are action states and the majority of transitions are brought about by the fulfillment of actions in the source states. One may refer to the action as a system operation. As a result, the control flow is transferred across operations. This flow may occur concurrently, forked, or sequentially. Activity diagrams use a variety of features to address various forms of flow control.

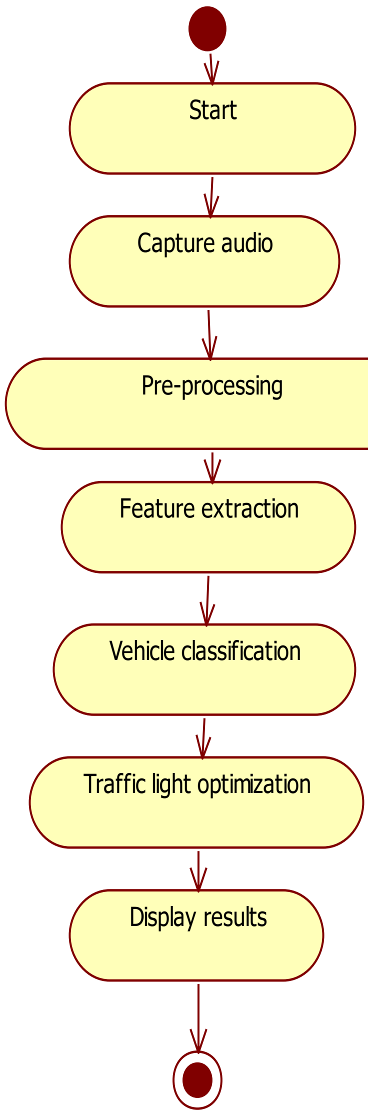


Figure 6.1.3: Activity Diagram

6.1.4 CLASS DIAGRAM

A class diagram is a type of UML (Unified Modeling Language) diagram that provides a visual representation of the classes, attributes, and methods of an object-oriented program or system. It shows the relationships and dependencies between the different classes in the system and how they interact with one another.

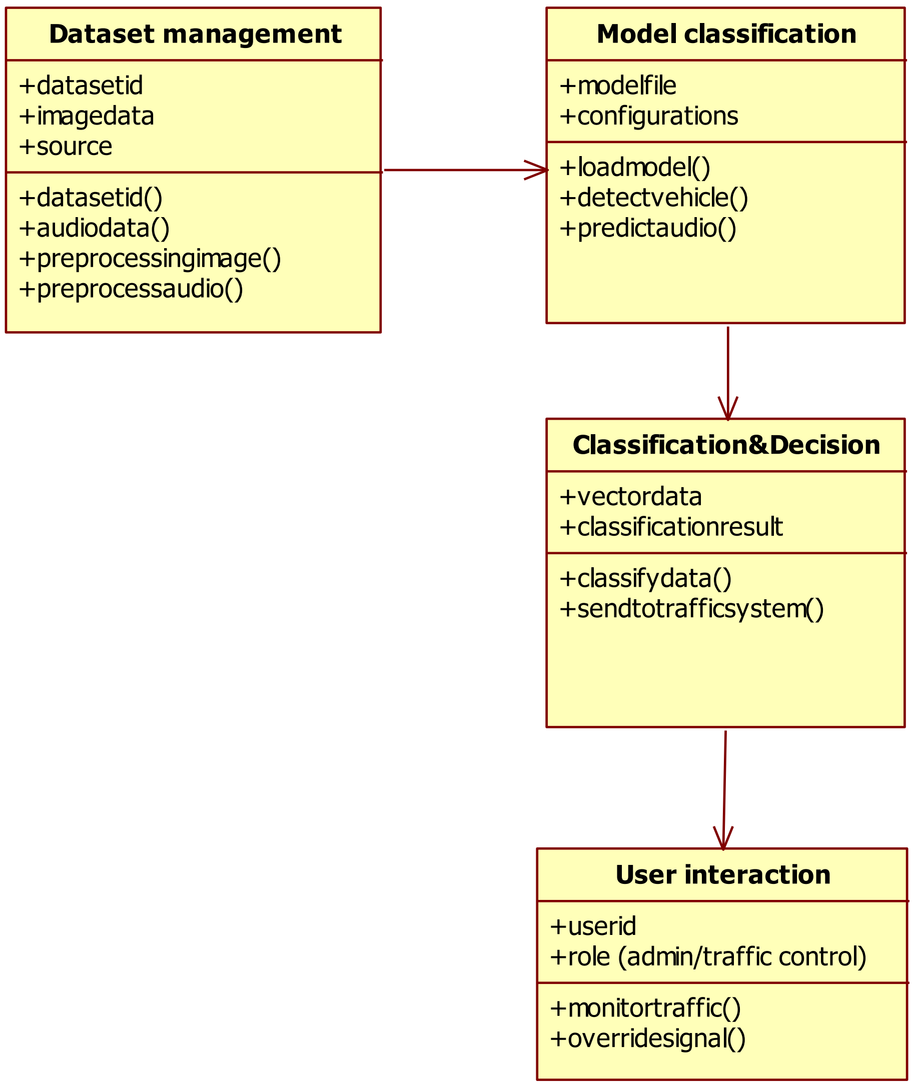


Figure 6.1.4: Class Diagram

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

Efficient emergency response in urban environments is crucial to saving lives, and traffic congestion poses a significant challenge to ambulances reaching their destinations on time. This project addresses this critical issue by developing an intelligent traffic light control system that prioritizes emergency vehicles using a combination of computer vision, audio processing, and deep learning algorithms. The system integrates YOLO-based vehicle detection and CNN-based sound classification to accurately identify ambulances in real-time. Once detected, the system dynamically adjusts traffic signal timings to create a clear path for the emergency vehicle, reducing unnecessary delays and improving overall traffic management. By leveraging advanced deep learning techniques and real-time communication, the system enhances the speed and efficiency of emergency response while ensuring compliance with traffic regulations.

The proposed framework is tested across varied real-world scenarios to ensure accuracy, reliability, and effectiveness. The system significantly outperforms traditional machine learning approaches in recognizing emergency vehicles, offering higher precision and faster processing speeds. Its ability to adapt to different traffic conditions makes it a scalable and robust solution for modern urban traffic infrastructure. With continued improvements and integration with smart city initiatives, this intelligent traffic control system has the potential to revolutionize emergency response management, reducing fatalities and enhancing public safety. The implementation of such AI-driven solutions paves the way for a more efficient, responsive, and intelligent traffic ecosystem that prioritizes human lives over congestion.

7.2 FUTURE ENHANCEMENTS

- **Integration with IoT and Smart Traffic Systems**: Future improvements can involve integrating IoT-based smart sensors at traffic intersections to enhance real-time detection and decision-making. By connecting the system to a centralized cloud-based traffic control platform, authorities can receive live updates and optimize traffic flow dynamically. This would improve the coordination of emergency vehicle movement across multiple intersections, ensuring smooth and uninterrupted passage.
- **Implementation of 5G and Edge Computing for Faster Processing**: Leveraging 5G networks and edge computing can significantly reduce latency in data processing and decision-making. Instead of relying on centralized servers, processing vehicle images and siren audio at the edge (i.e., directly at the traffic signals) can enhance response time. This will allow for near-instantaneous traffic light adjustments, minimizing delays and improving emergency response efficiency.
- **Multi-Modal Emergency Vehicle Recognition**: The system can be extended to support multiple emergency vehicles, such as fire trucks and police cars, with unique prioritization rules. By training the model on a broader dataset and refining classification algorithms, the system can differentiate between various emergency vehicles and adjust traffic signals based on their urgency levels.
- **Integration with GPS and Navigation Systems**: By connecting with real-time GPS navigation systems used by emergency responders, the system can predict the arrival of an ambulance and preemptively clear traffic. This predictive approach can further reduce waiting times by synchronizing multiple traffic signals along the ambulance’s route, ensuring seamless passage.

APPENDIX A

SOURCE CODE

```
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
from keras.models import model_from_json
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
batch_size = 32
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# All images will be rescaled by 1./255
train_datagen = ImageDataGenerator(rescale=1/255)
# Flow training images in batches of 128 using train_datagen generator
train_generator = train_datagen.flow_from_directory(
'DataSet/train/',  # This is the source directory for training images
target_size=(200, 200),  # All images will be resized to 200 x 200
batch_size=batch_size,
# Specify the classes explicitly
 classes = ['Ambulance','Bicycle','Bus','Car','Motorcycle','Tank','Taxi','Truck'],
# Since we use categorical_crossentropy loss, we need categorical labels
class_mode='categorical')
import tensorflow as tf
model = tf.keras.models.Sequential([
# Note the input shape is the desired size of the image 200x 200 with 3 bytes color
# The first convolution
tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(200, 200, 3)),
tf.keras.layers.MaxPooling2D(2, 2),
# The second convolution
tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
tf.keras.layers.MaxPooling2D(2,2),
# The third convolution
tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
tf.keras.layers.MaxPooling2D(2,2),
# The fourth convolution
tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
tf.keras.layers.MaxPooling2D(2,2),
# The fifth convolution
tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
tf.keras.layers.MaxPooling2D(2,2),
# Flatten the results to feed into a dense layer
tf.keras.layers.Flatten(),
# 128 neuron in the fully-connected layer
tf.keras.layers.Dense(128, activation='relu'),
# 5 output neurons for 5 classes with the softmax activation
tf.keras.layers.Dense(8, activation='softmax')])
model.summary()
from tensorflow.keras.optimizers import RMSprop
early = tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=5)
model.compile(loss='categorical_crossentropy',
optimizer=RMSprop(lr=0.001),
metrics=['accuracy'])
total_sample=train_generator.n
n_epochs = 10
history = model.fit_generator(
train_generator,
steps_per_epoch=int(total_sample/batch_size),
epochs=n_epochs,
verbose=1)
model.save('model.h5')
acc = history.history['accuracy']
loss = history.history['loss']
epochs = range(1, len(acc) + 1)
# Train and validation accuracy
```

```
plt.plot(epochs, acc, 'b', label='Training accurarcy')

plt.title('Training  accurarcy')

plt.legend()

plt.figure()

# Train and validation loss

plt.plot(epochs, loss, 'b', label='Training loss')

plt.title('Training  loss')

plt.legend()

plt.show()

import sys

import pathlib

working_dir_path = pathlib.Path().absolute()

if sys.platform.startswith('win32'):

    TRAINING_FILES_PATH = str(working_dir_path) + '\\features\\'

    SAVE_DIR_PATH = str(working_dir_path) + '\\joblib_features\\'

    MODEL_DIR_PATH = str(working_dir_path) + '\\model\\'

    TESS_ORIGINAL_FOLDER_PATH = str(working_dir_path) +
'\\sound_set_data\\'

    EXAMPLES_PATH = str(working_dir_path) + '\\examples\\'

else:

    TRAINING_FILES_PATH = str(working_dir_path) + '/features/'

    SAVE_DIR_PATH = str(working_dir_path) + '/joblib_features/'

    MODEL_DIR_PATH = str(working_dir_path) + '/model/'
```

```
   TESS_ORIGINAL_FOLDER_PATH = str(working_dir_path) + '/sound_set_data/'
   EXAMPLES_PATH = str(working_dir_path) + '/examples/'
"""
This files creates the X and y features in joblib to be used by the predictive models.
"""


import os
import time
import joblib
import librosa
import numpy as np
from config import SAVE_DIR_PATH
from config import TRAINING_FILES_PATH
class CreateFeatures:
@staticmethod
def features_creator(path, save_dir) -> str:
"""
This function creates the dataset and saves both data and labels in two files, X.joblib
and y.joblib in the joblib_features folder. With this method, you can persist your
features and train quickly new machine learning models instead of reloading the
featuresevery time with this pipeline.
"""
```

```
lst = []

start_time = time.time()

for subdir, dirs, files in os.walk(path):
for file in files:
print(file)
try:
# Load librosa array, obtain mfcss, store the file and the mcss information in a new
array
X, sample_rate = librosa.load(os.path.join(subdir, file),
res_type='kaiser_fast')
mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate,
n_mfcc=40).T, axis=0)
# The instruction below converts the labels (from 1 to 8) to a series from 0 to 7
# This is because our predictor needs to start from 0 otherwise it will try to predict
also 0.
file = str(file[0:3])
print(file)
if file == 'amb':
file = 0
else:
file = 1
print(file)
```

```
arr = mfccs, file

print(arr)

lst.append(arr)

# If the file is not valid, skip it

except ValueError as err:

print(err)

continue

print("--- Data loaded. Loading time: %s seconds ---" % (time.time() - start_time))

# Creating X and y: zip makes a list of all the first elements, and a list of all the
second elements.

X, y = zip(*lst)

# Array conversion

X, y = np.asarray(X), np.asarray(y)

# Array shape check

print(X.shape, y.shape)

# Preparing features dump

X_name, y_name = 'X.joblib', 'y.joblib'

joblib.dump(X, os.path.join(save_dir, X_name))

joblib.dump(y, os.path.join(save_dir, y_name))

return "Completed"

if __name__ == '__main__':

print('Routine started')
```

```
FEATURES = CreateFeatures.features_creator(path=TRAINING_FILES_PATH,
save_dir=SAVE_DIR_PATH)
print('Routine completed.')


"""
This file can be used to try a live prediction.
"""
import keras
import librosa
import numpy as np


from config import EXAMPLES_PATH
from config import MODEL_DIR_PATH
class LivePredictions:
"""
Main class of the application.
"""
def __init__(self, file):
"""
Init method is used to initialize the main parameters.
"""
self.file = file
self.path = MODEL_DIR_PATH + 'sound_Model.h5'
self.loaded_model = keras.models.load_model(self.path)
def make_predictions(self):

"""
Method to process the files and create your features.
"""
data, sampling_rate = librosa.load(self.file)
mfccs = np.mean(librosa.feature.mfcc(y=data, sr=sampling_rate, n_mfcc=40).T,
axis=0)
x = np.expand_dims(mfccs, axis=2)
x = np.expand_dims(x, axis=0)
predictions = self.loaded_model.predict_classes(x)
print( "Prediction is", " ", self.convert_class_to_emotion(predictions))
@staticmethod
def convert_class_to_emotion(pred):
"""
Method to convert the predictions (int) into human readable strings.
"""
label_conversion = {'0': 'ambulance',
'1': 'traffic'
}
for key, value in label_conversion.items():
if int(key) == pred:
```

```
label = value
return label
if __name__ == '__main__':
live_prediction = LivePredictions(file=EXAMPLES_PATH + 'sound_1.wav')
live_prediction.loaded_model.summary()
live_prediction.make_predictions()
live_prediction = LivePredictions(file=EXAMPLES_PATH + 'sound_401.wav')
live_prediction.make_predictions()
"""
Neural network train file.
"""
import os
import joblib
import numpy as np
import matplotlib.pyplot as plt
from keras.layers import Dense
from keras.layers import Conv1D
from keras.layers import Flatten
from keras.layers import Dropout
from keras.layers import Activation
from keras.models import Sequential
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

from sklearn.model_selection import train_test_split

from config import SAVE_DIR_PATH

from config import MODEL_DIR_PATH

class TrainModel:

@staticmethod

def train_neural_network(X, y) -> None:

"""

This function trains the neural network.

"""

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=42)

x_traincnn = np.expand_dims(X_train, axis=2)

x_testcnn = np.expand_dims(X_test, axis=2)

print(x_traincnn.shape, x_testcnn.shape)

model = Sequential()

model.add(Conv1D(64, 5, padding='same',

input_shape=(40, 1)))

model.add(Activation('relu'))

model.add(Dropout(0.2))

model.add(Flatten())

model.add(Dense(2))

model.add(Activation('softmax'))
```

```
print(model.summary)

model.compile(loss='sparse_categorical_crossentropy',

optimizer='rmsprop',

metrics=['accuracy'])

cnn_history = model.fit(x_traincnn, y_train,

batch_size=16, epochs=50,

validation_data=(x_testcnn, y_test))

# Loss plotting

plt.plot(cnn_history.history['loss'])

plt.plot(cnn_history.history['val_loss'])

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

plt.savefig('loss.png')

plt.close()

# Accuracy plotting

plt.plot(cnn_history.history['accuracy'])

plt.plot(cnn_history.history['val_accuracy'])

plt.title('model accuracy')

plt.ylabel('acc')

plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.savefig('accuracy.png')
predictions = model.predict_classes(x_testcnn)
new_y_test = y_test.astype(int)
matrix = confusion_matrix(new_y_test, predictions)
print(classification_report(new_y_test, predictions))
print(matrix)
model_name = 'sound_Model.h5'
# Save model and weights
if not os.path.isdir(MODEL_DIR_PATH):
os.makedirs(MODEL_DIR_PATH)
model_path = os.path.join(MODEL_DIR_PATH, model_name)
model.save(model_path)
print('Saved trained model at %s ' % model_path)
if __name__ == '__main__':
print('Training started')
print(SAVE_DIR_PATH + 'X.joblib')
X = joblib.load(SAVE_DIR_PATH + 'X.joblib')
y = joblib.load(SAVE_DIR_PATH + 'y.joblib')
NEURAL_NET = TrainModel.train_neural_network(X=X, y=y)
from flask import Flask, render_template, flash, request, session, send_file
from flask import render_template, redirect, url_for, request
```

```
import warnings
import datetime
app = Flask(__name__)
app.config['DEBUG']
app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'
@app.route("/")
def homepage():
return render_template('index.html')
@app.route("/Test")
def Test():
return render_template('Test.html')
@app.route("/ImageTest")
def ImageTest():
return render_template('ImageTest.html')
@app.route("/imgetest", methods=['GET', 'POST'])
def imgetest():
if request.method == 'POST':
import tensorflow as tf
import numpy as np
from keras.preprocessing import image
file = request.files['fileupload']
file.save('static/upload/Test.jpg')
```

```
fname = 'static/upload/Test.jpg'

import warnings

warnings.filterwarnings('ignore')

classifierLoad = tf.keras.models.load_model('model.h5')

test_image = image.load_img('static/upload/Test.jpg', target_size=(200, 200))

test_image = np.expand_dims(test_image, axis=0)

result = classifierLoad.predict(test_image)

print(result[0][0])

res = ''

result1 = ''

if result[0][0] == 1:

res = 'Ambulance'

result1 = 'Emergency'

return render_template('Test.html', result=res, gry=fname)

elifresult[0][1] == 1:

res = 'Bicycle'

result1 = 'Normal'

elifresult[0][2] == 1:

res = 'Bus'

result1 = 'Normal'

elifresult[0][3] == 1:

res = 'Car'
```

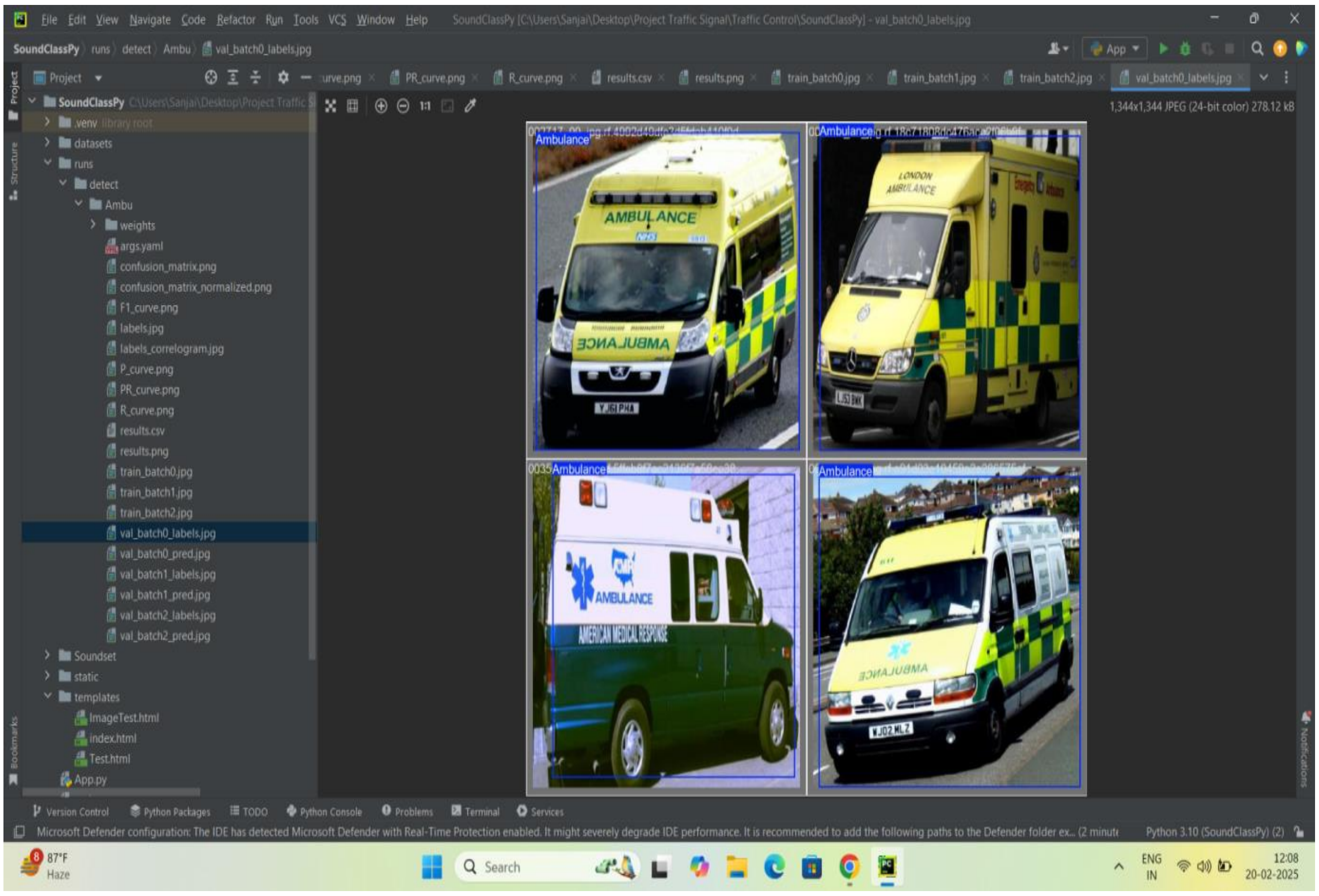
```
result1 = 'Normal'
elifresult[0][4] == 1:
res = 'Motorcycle'
result1 = 'Normal'
elifresult[0][5] == 1:
res = 'Tank'
result1 = 'Normal'
elifresult[0][6] == 1:
res = 'Taxi'
result1 = 'Normal'
elifresult[0][7] == 1:
res = 'Truck'
result1 = 'Normal'
return render_template('ImageTest.html', result=res, result1=result1, gry=fname)
@app.route("/testsound", methods=['GET', 'POST'])
def testsound():
if request.method == 'POST':
file = request.files['fileupload']
file.save('static/Out/' + file.filename)
live_prediction = LivePredictions(file='static/Out/' + file.filename)
live_prediction.loaded_model.summary()
live_prediction.make_predictions()
```

```
res = session["out"]
gry = ''
if res == "ambulance":
res = 'Ambulance'
result1 = 'Emergency'
gry = 'static/emoji/green.jpg'
elif res == "traffic":
res = 'Traffic'
result1 = 'Normal'
gry = 'static/emoji/red.jpg'
return render_template('Test.html', result=res, result1=result1, gry=gry)
import keras
import librosa
import numpy as np
from config import EXAMPLES_PATH
from config import MODEL_DIR_PATH
class LivePredictions:
"""
Main class of the application.
"""
def __init__(self, file):
"""
```

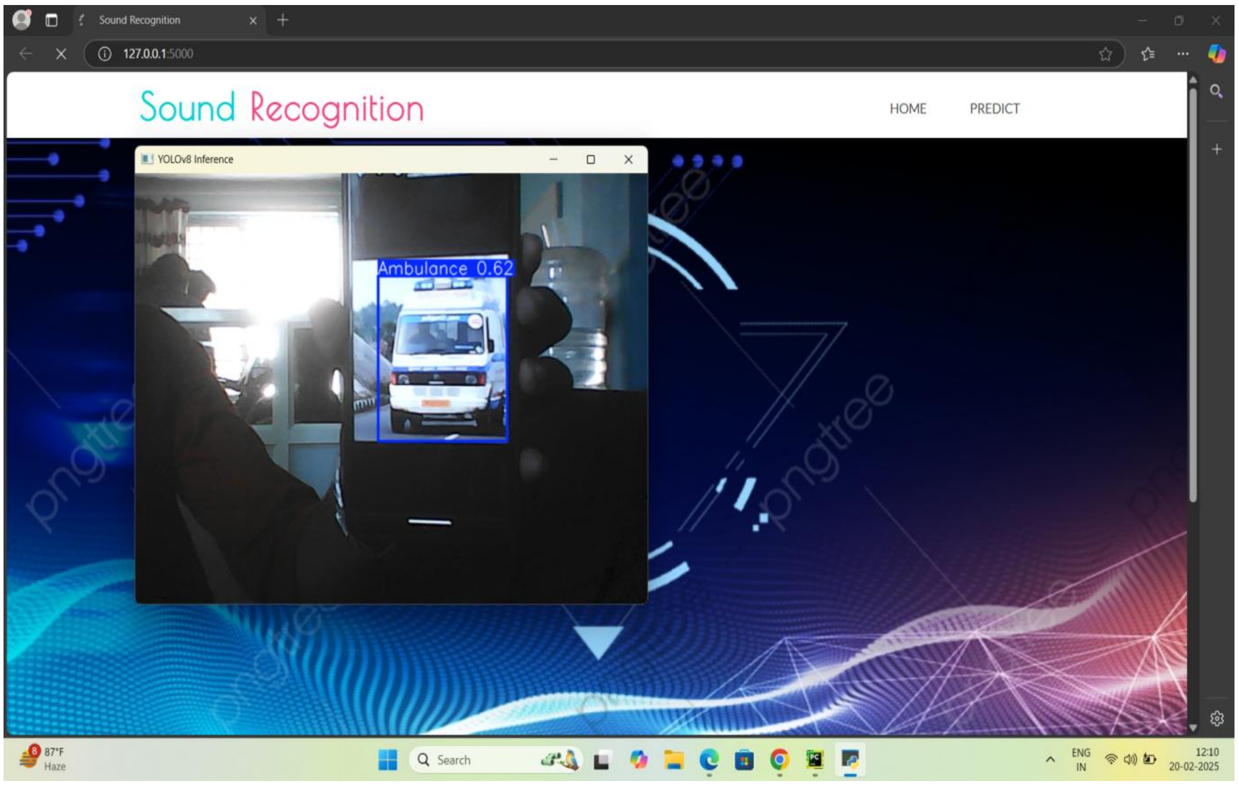
```
Init method is used to initialize the main parameters.
"""
self.file = file
self.path = MODEL_DIR_PATH + 'sound_Model.h5'
self.loaded_model = keras.models.load_model(self.path)
def make_predictions(self):
"""
Method to process the files and create your features.
"""
data, sampling_rate = librosa.load(self.file)
mfccs = np.mean(librosa.feature.mfcc(y=data, sr=sampling_rate, n_mfcc=40).T,
axis=0)
x = np.expand_dims(mfccs, axis=2)
x = np.expand_dims(x, axis=0)
predictions = self.loaded_model.predict_classes(x)
print("Prediction is", " ", self.convert_class_to_emotion(predictions))
session["out"] = self.convert_class_to_emotion(predictions)
@staticmethod
def convert_class_to_emotion(pred):
"""
Method to convert the predictions (int) into human readable strings.
"""
label_conversion = {'0': 'ambulance',
'1': 'traffic',

}

for key, value in label_conversion.items():

if int(key) == pred:

label = value

return label

def sendmsg(targetno, message):

import requests

requests.post(

"http://smsserver9.creativepoint.in/api.php?username=fantasy&password=596692&t
o=" + targetno + "&from=FSSMSS&message=Dear user  yourmsg is " + message + "
Sent By FSMSG
FSSMSS&PEID=1501563800000030506&templateid=1507162882948811640")

if __name__ == '__main__':

app.run(debug=True, use_reloader=True)
```

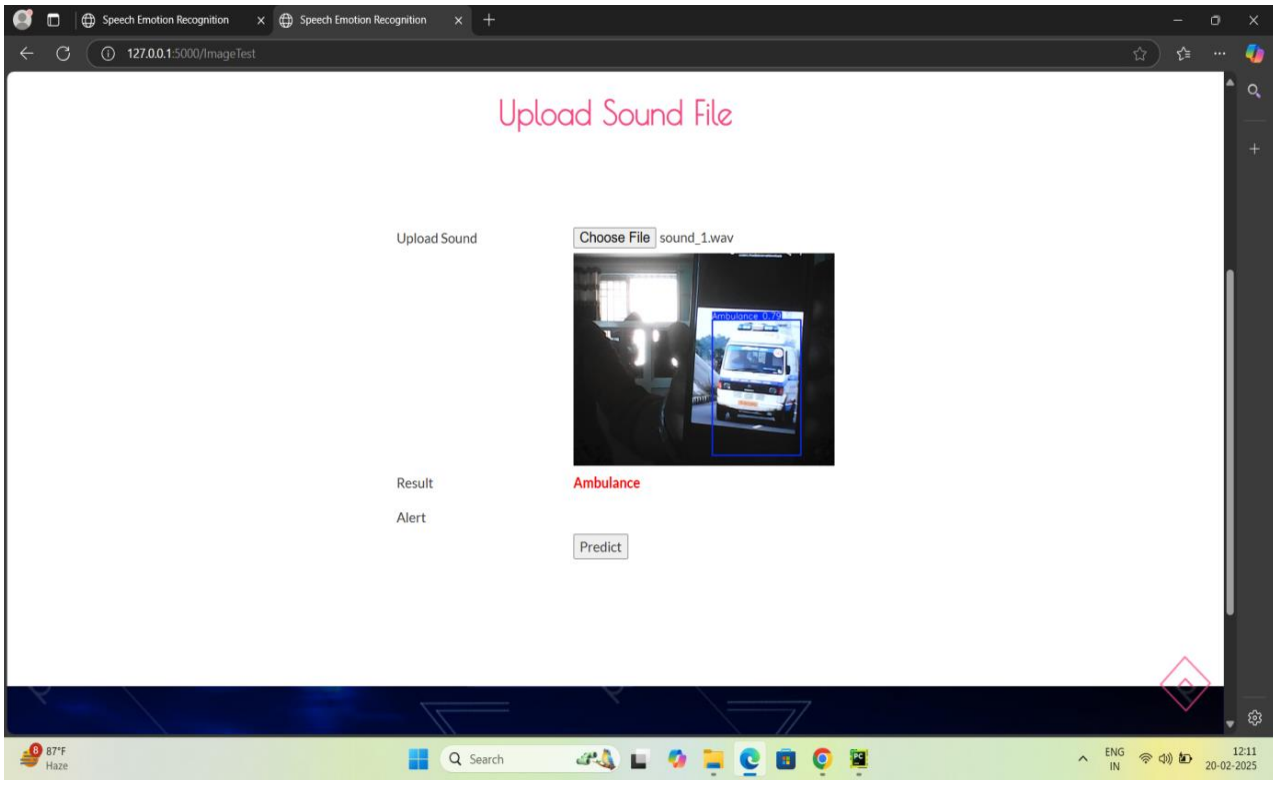
APPENDIX B

SCREENSHOTS



Microsoft Defender configuration: The IDE has detected Microsoft Defender with Real-Time Protection enabled. It might severely degrade IDE performance. It is recommended to add the following paths to the Defender folder ex... (2 minute



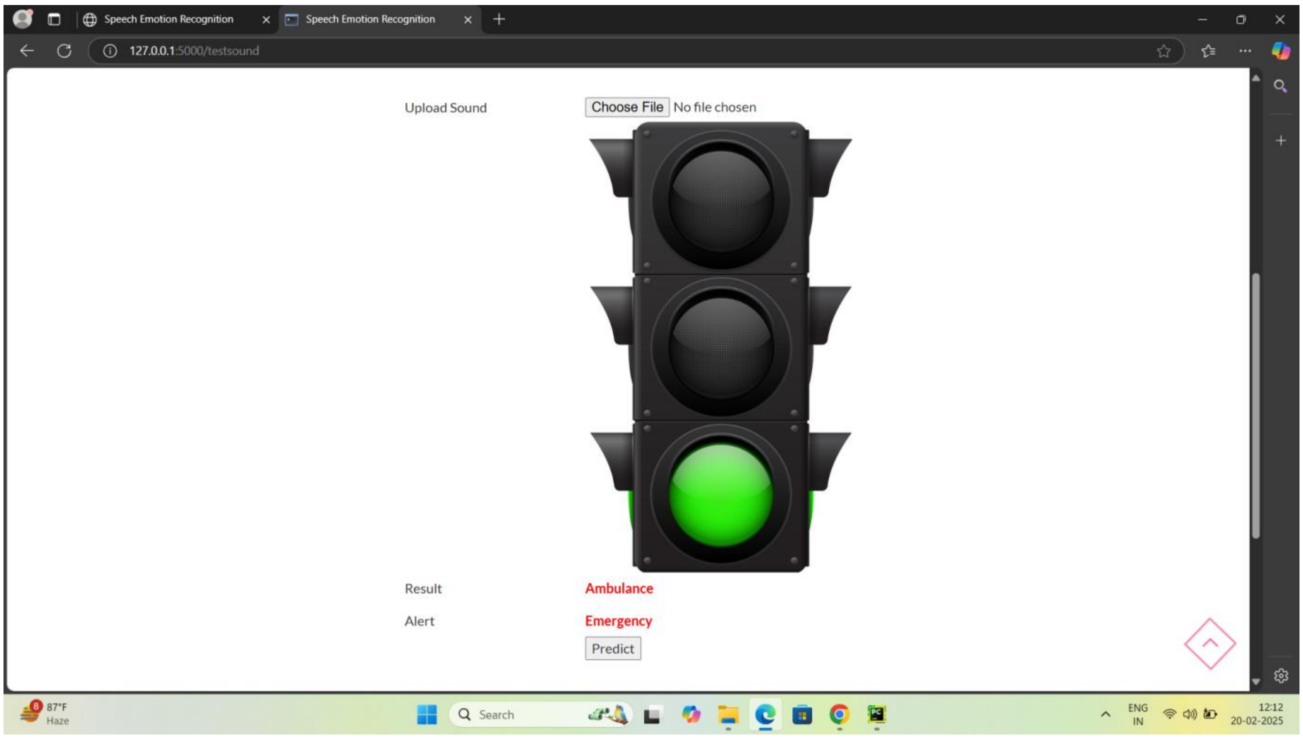


Upload Sound File

Upload Sound sound_1.wav

Result Ambulance

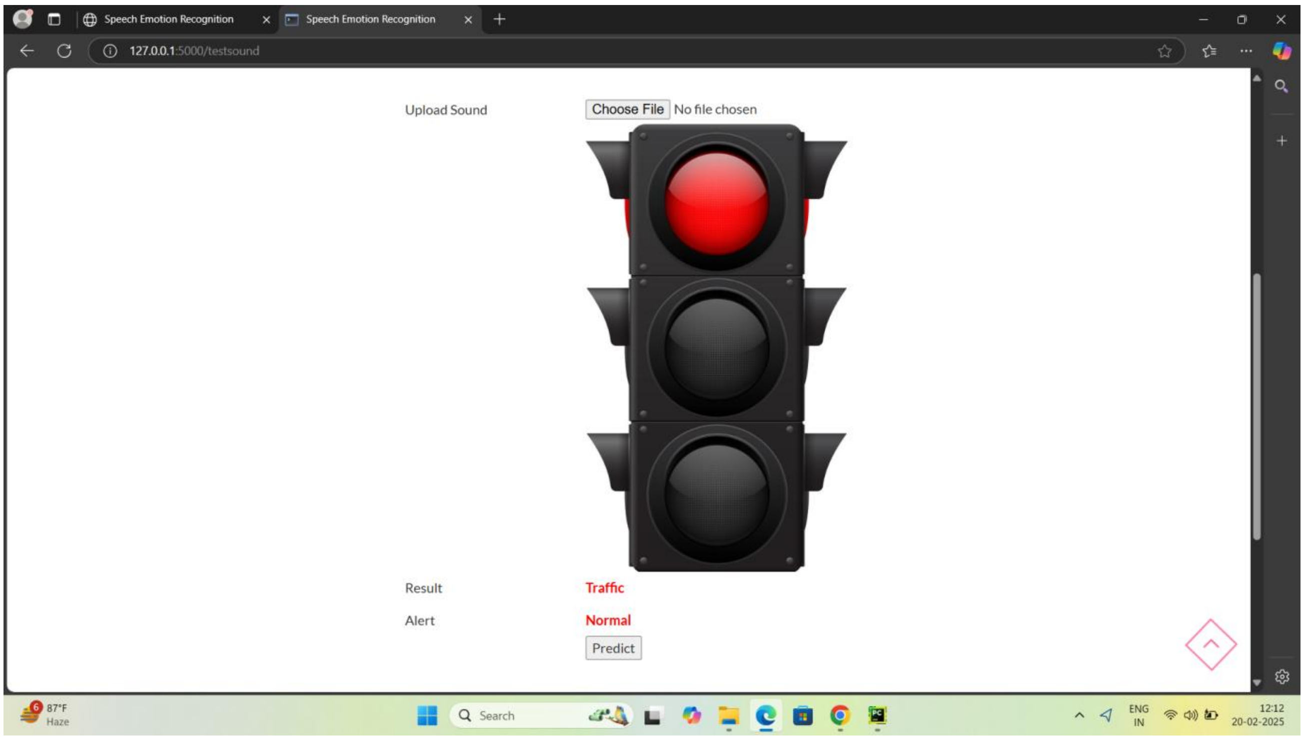
Alert



Upload Sound Choose File No file chosen

Result Ambulance

Alert Emergency



Upload Sound Choose File No file chosen

Result Traffic

Alert Normal

REFERENCES

[1] Chowdhury, Abdullahi, et al. "IoT-based emergency vehicle services in intelligent transportation system." Sensors 23.11 (2023): 5324.

[2] Arikumar, Kochupillai Selvaraj, et al. "V2x-based highly reliable warning system for emergency vehicles." Applied Sciences 13.3 (2023): 1950.

[3] Mittal, Usha, and Priyanka Chawla. "Acoustic based emergency vehicle detection using ensemble of deep learning models." Procedia Computer Science 218 (2023): 227-234.

[4] Mei, Hao, et al. "Libsignal: an open library for traffic signal control." Machine Learning 113.8 (2024): 5235-5271.

[5] Naeem, Awad Bin, et al. "Intelligent road management system for autonomous, non-autonomous, and VIP vehicles." World Electric Vehicle Journal 14.9 (2023): 238.

[6] Jurczenia, Karol, and Jacek Rak. "A survey of vehicular network systems for road traffic management." *IEEE Access* 10 (2022): 42365-42385.

[7] Gholamhosseinian, Ashkan, and Jochen Seitz. "A comprehensive survey on cooperative intersection." *IEEE Access* 10 (2022): 7937-7972.

[8] Creß, Christian, Zhenshan Bing, and Alois C. Knoll. "Intelligent transportation systems using roadside infrastructure: A literature survey." IEEE Transactions 25.7 (2023): 6309-6327.

[9] Aoki, Shunsuke, and Ragunathan Rajkumar. "Safe intersection management with cooperative perception for mixed traffic of human-driven and autonomous vehicles." IEEE Open Journal of Vehicular Technology 3 (2022): 251-265.

[10] Butt, Faran Awais, et al. "On the integration of enabling wireless technologies and sensor fusion for next-generation connected and autonomous vehicles." IEEE Access 10 (2022): 14643-14668.