# AN INTELLIGENT VEHICLE PRIORIZATION SYSTEM USING AI

**A PROJECT REPORT**

***Submitted by***


**DHIVYA A D**

**SATHVIKA A**

**SUJAINITHA G**

**THEJEAL SRI K**


***in partial fulfilment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

***in***

**ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**


**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

**SAMAYAPURAM – 621 112**


**MAY, 2025**

# AN INTELLIGENT VEHICLE PRIORIZATION SYSTEM USING AI

**A PROJECT REPORT**

***Submitted by***


**DHIVYA A D (811721243014)**

**SATHVIKA A (811721243049)**

**SUJAINITHA G (811721243055)**

**THEJEAL SRI K (811721243057)**


***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

***in***

**ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**


**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

**SAMAYAPURAM – 621 112**


**MAY, 2025**

# K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
## (AUTONOMOUS)
**SAMAYAPURAM – 621 112**

## BONAFIDE CERTIFICATE

Certified that this project report titled **"AN INTELLIGENT VEHICLE PRIORIZATION SYATEM USING AI"** is the bonafide work of **DHIVYA A D (811721243014), SATHVIKA A (811721243049), SUJAINITHA G(811721243055), THEJEAL SRI K (811721243057)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**SIGNATURE**

Dr. T.Avudaiappan M.E., Ph.D.,
**HEAD OF THE DEPARTMENT**

Department of Artificial Intelligence
K.Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

**SIGNATURE**

Mr. R. Roshan Joshua M.E.,
**SUPERVISOR**

ASSISTANT PROFESSOR
Department of Artificial Intelligence
K.Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

Submitted for the viva-voce examination held on ……

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

# DECLARATION

We jointly declare that the project report on **"AN INTELLIGENT VEHICLE PRIORIZATION SYSYTEM USING AI"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted on the partial fulfilment of the requirement of the award of Degree of **BACHELOR OF TECHNOLOGY**.

**Signature**

DHIVYA.A. D

SATHVIKA.A

SUJAINITHA.G

THEJEAL SRI.K

Place: Samayapuram

Date:

# ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

We are glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

We would like to thank **Dr. N. VASUDEVAN, M.E., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

We whole heartily thank to **Dr. T. AVUDAIAPPAN**, **M.E., Ph.D.,** Head of the department, **ARTIFICIAL INTELLIGENCE** for providing his encourage pursuing this project.

We express our deep and sincere gratitude to our project guide **Mr. R. ROSHAN JOSHUA**, **M.E.**, Department of **ARTIFICIAL INTELLIGENCE,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

We render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

# ABSTRACT

The intelligent vehicle prioritization system presents an AI-powered intelligent traffic management system designed to prioritize emergency vehicles at traffic signals, ensuring faster response times in urban areas. The system integrates deep learning techniques, including convolutional neural networks (CNNs) for analysing CCTV footage and Mel-frequency cepstral coefficients (MFCC) for processing emergency siren audio. By leveraging these technologies, the system detects emergency vehicles such as ambulance, fire trucks and so on in real-time and automatically adjusts traffic signals, allowing them to pass through without delays. Additionally, the system is designed for seamless integration with autonomous vehicles, enabling them to move out of the lane, further clearing the path for emergency vehicles. This approach minimizes human intervention and enhances overall traffic efficiency. IoT-based communication ensures real-time synchronization between traffic signals and emergency vehicle detection, optimizing urban mobility. By addressing the growing issue of traffic congestion in metropolitan areas, this model significantly reduces delays in emergency response, thereby decreasing mortality rates caused by late ambulance arrivals. The system is scalable and can be integrated into smart city infrastructures, offering a sustainable and efficient solution for modern urban traffic management.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | | |
|---|---|---|
| CMOS | - | Complementary Metal Oxide Semiconductor |
| GNN | - | Graph Neural Network |
| IOT | - | Internet of Things |
| MFCC | - | Mel-Frequency Cepstral Coefficients |
| R-CNN | - | Regional Convolutional Neural Network |
| ReLU | - | Rectified Linear Unit |
| RL | - | Reinforcement Learning |

# CHAPTER 1

# INTRODUCTION

The intelligent prioritization system focuses on developing an AI-driven traffic management system to prioritize emergency vehicles at traffic signals, reducing response time delays in urban areas. Using deep learning techniques like CNNs for video analysis and MFCC for siren detection, the system autonomously detects ambulances and adjusts traffic signals in real time. Integrated with IoT and autonomous vehicles, it ensures seamless lane clearance without human intervention. Designed for smart city infrastructure, this solution enhances emergency response efficiency, minimizing delays and improving urban mobility.

## 1.1 BACKGROUND

Rapid urbanization and increasing traffic congestion have significantly impacted emergency response times, often leading to critical delays for ambulances and other emergency vehicles. Traditional traffic management systems rely on manual intervention or pre-programmed signal cycles, which fail to adapt dynamically to real-time emergencies. With advancements in artificial intelligence and IoT, intelligent traffic control solutions have emerged to address this issue. By utilizing deep learning techniques like CNNs for video analysis and MFCC for siren detection, modern systems can accurately identify emergency vehicles and automatically adjust traffic signals. Integrating these technologies with autonomous vehicles further enhances lane clearance, reducing human dependency and optimizing traffic flow. The intelligent prioritization system aims to develop a scalable AI-powered system that prioritizes emergency vehicles, ensuring faster response times and minimizing casualties caused by delayed medical assistance. By leveraging smart city infrastructure, this solution contributes to the evolution of adaptive, real-time traffic management for improved urban mobility.

## 1.2 PROBLEM STATEMENT

The increasing traffic congestion in urban areas poses a major challenge to emergency response times, often leading to life-threatening delays for ambulances. Traditional traffic management systems lack real-time adaptability, relying on fixed signal cycles that do not prioritize emergency vehicles. Additionally, human intervention is often required to clear lanes, further delaying response times. Existing solutions do not leverage AI-driven automation to optimize traffic flow dynamically. To address this, an intelligent traffic control system utilizing deep learning and IoT is needed to detect ambulances in real time, adjust traffic signals, and ensure seamless passage, significantly improving emergency response efficiency.

## 1.3 AIM AND OBJECTIVE

### 1.3.1 Aim

The intelligent prioritization system aims to develop an AI-powered intelligent traffic management system that prioritizes emergency vehicles by detecting ambulances in real time and dynamically adjusting traffic signals. By integrating deep learning and IoT, the system ensures seamless passage, reducing delays and improving emergency response efficiency in urban areas.

### 1.3.2 Objective

The objective is to develop an AI-driven traffic management system that accurately detects emergency vehicles using deep learning techniques. The system will dynamically adjust traffic signals, integrate with IoT for real-time communication, and enhance autonomous vehicle coordination to ensure faster emergency response and improved urban traffic flow efficiency.

# CHAPTER 2

# LITERATURE SURVEY

## 2.1 SMART CITY TRAFFIC MANAGEMENT: ACOUSTIC-BASED VEHICLE DETECTION USING STACKING-BASED ENSEMBLE DEEP LEARNING APPROACH.

**Ahsan Shabbir, Ibrahim M. Almanjahie, Ammaranawazcheema, Inamullah.**

Stacking Ensemble Deep Learning (MLP, DNN, LSTM).

This study focuses on acoustic data analysis for real-time emergency vehicle detection in smart traffic management. A stacking ensemble deep learning model (MLP, DNN, LSTM) is used to classify emergency sirens from background traffic noises using microphone sensor data. Advanced feature extraction techniques (MFCC, spectral centroids, mel spectrogram, etc.) enhance accuracy. The proposed model achieves 99.12% accuracy, demonstrating its effectiveness in improving traffic flow and emergency response in smart cities.

**Merits**

Enhances emergency response efficiency, improves traffic management, and adapts to diverse urban conditions.

**Demerits**

Computationally intensive, complex to implement, and sensitive to environmental noise variations.

## 2.2 EVP-STC: EMERGENCY VEHICLE PRIORITY AND SELF-ORGANISING TRAFFIC CONTROL AT INTERSECTIONS USING INTERNET-OF-THINGS PLATFORM

**Ajmal Khan, Farman Ullah, Zeeshan Kaleem, You-Ze Cho, Shamsurrahman.**

Fuzzy Logic, Genetic Algorithm (GA), Reinforcement Learning (RL), Dijkstra's .

The study presents an Internet-of-Things (IoT)-based platform called Emergency Vehicle Priority and Self-Organizing Traffic Control (EVP-STC) to manage traffic congestion at intersections. The proposed EVP-STC system consists of three main components: an intersection controller that collects real-time traffic and emergency vehicle data, a road segment detection system using force resistive sensors to monitor vehicle density, and an emergency vehicle system that provides GPS coordinates to prioritize emergency vehicles at intersections. The system leverages ZigBee communication to transmit data and adjusts traffic light timings dynamically to minimize delays. Simulation results demonstrate the platform's effectiveness in reducing waiting times for emergency vehicles and optimizing traffic flow.

### Merits

Reduces traffic congestion, minimizes emergency vehicle delays, optimizes traffic signal timing, lowers fuel consumption, and enhances urban mobility.

### Demerits

High implementation cost, reliance on real-time data accuracy, potential sensor failures.

## 2.3 COORDINATE ATTENTION ENHANCED ADAPTIVE SPATIOTEMPORAL CONVOLUTIONAL NETWORKS FOR TRAFFIC FLOW FORECASTING.

**Siwei Wei, Sichen Shen, Donghua Liu, Yanan Song, Rong Gao, Chunzhi Wang.**

SpatioTemporal Graph Neural Networks (STGNN), Coordinate Attention Mechanism, Multi-Head Attention, Gated Fusion Mechanism.

Traffic flow prediction is a crucial aspect of intelligent transport systems, with Spatio Temporal Graph Neural Networks (STGNNs) being widely used. However, existing models struggle with spatial heterogeneity and the loss of spatial correlations in complex environments. To overcome these issues, this study introduces the Coordinate Attention Enhanced Adaptive Spatiotemporal Convolutional Network (CAAS). The CAAS model incorporates coordinate attention to model spatial heterogeneity and a spatial multi-head attention mechanism to capture intricate spatiotemporal dependencies. A gated fusion mechanism adaptively integrates temporal and spatial features for improved prediction accuracy. Experiments on PEMS04 and PEMS08 datasets demonstrate that CAAS outperforms existing methods.

**Merits**

Enhances spatial-temporal accuracy in traffic prediction.

**Demerits**

High computational costforreal-time use.

## 2.4 EMERGENCY VEHICLE DETECTION USING IOT IN SMART CITIES

**Mahesh Kumar Thota, Prathibhavani P.M, Shruthi Gujja, Deeksha Ravula.**

Computer Vision Algorithms, IoT-based Detection, Edge Computing-based Processing

This study explores the application of IoT and edge computing for real-time emergency vehicle detection in smart cities. It integrates sensor networks, computer vision algorithms, and edge computing to optimize response times. Key factors such as low latency, high accuracy, and resource efficiency are considered. The proposed system aims to enhance emergency response, improving urban safety and traffic management. Additionally, it reduces network congestion by processing data at the edge, minimizing reliance on cloud computing. The integration of AI-driven analytics can further refine detection accuracy and predictive capabilities. Future enhancements could incorporate V2X (Vehicle-to-Everything) communication, enabling seamless coordination between emergency vehicles, traffic signals, and city infrastructure for an intelligent and adaptive response system.

**Merits**

Enables real-time emergency vehicle detection with minimal latency.

**Demerits**

Requires high computational resources and infrastructure for deployment.

## 2.5 RFID BASED SMART TRAFFIC SYSTEM FOR EMERGENCY VEHICLES.

**Sathya D, C. Vinothini, S. Keerthi, Jagadeesan D, Nidhishree M S**

RFID-based Detection, Arduino-based Traffic Control.

This study addresses traffic congestion issues, especially at signalized junctions during emergencies, by developing an RFID-based traffic light control system. The system detects emergency vehicles via RFID transmission and utilizes an Arduino microcontroller to adjust traffic light sequences. Once the emergency vehicle passes, normal sequencing is restored. This approach minimizes delays, enhances road safety, and improves emergency response times. Additionally, it reduces fuel consumption and emissions by minimizing idle time at signals. The system can be expanded to include priority access for public transport and VIP convoys. Future enhancements could integrate real-world traffic conditions using IoT sensors and AI-based predictive analytics for a more advanced and adaptive traffic management system. Moreover, cloud-based data storage can enable real-time monitoring and optimization of traffic flow across multiple junctions.

**Merits**

Improves emergency response time by prioritizing emergency vehicles at traffic.

**Demerits**

Limited to vehicles equipped with RFID, requiring widespread adoption.

# CHAPTER 3

# SYSTEM ANALYSIS

## 3.1 EXISTING SYSTEM

These reviews the advancements in Emergency Vehicle Management (EVM) within Intelligent Transportation Systems (ITS), focusing on enhancing response times and safety. Emergency vehicles operate under time-sensitive conditions, requiring efficient route planning, real-time data retrieval, and advanced traffic management. The research highlights how smart city technologies, including IoT, AI, and big data analytics, can improve emergency response by optimizing travel routes, reducing delays, and ensuring seamless communication between vehicles and infrastructure.

The study explores various dimensions of EVM, such as accident detection, driver inattention monitoring, and traffic signal control, which are crucial for effective emergency response. It also discusses the role of Graph Neural Networks (GNNs), Convolutional Neural Networks (CNNs), and Vision Transformers in detecting driver distractions and improving road safety. Furthermore, the paper identifies key challenges in data privacy, processing limitations, and infrastructure constraints that hinder the implementation of ITS in EVM.

The authors emphasize the need for dynamic and real-time models to enhance emergency vehicle operations, ensuring faster and safer responses in urban environments. By leveraging VANETs, wireless sensor networks, and priority-based signaling, the study suggests potential improvements for reducing congestion and optimizing emergency services. Finally, the paper provides future research directions to further develop intelligent EVM solutions, integrating the latest advancements in AI-driven traffic control and smart city infrastructure.

## Algorithm Used

- **Graph Neural Networks (GNNs)**

Graph Neural Networks (GNNs) are deep learning models designed to handle graph-structured data, where relationships between entities are crucial. Unlike traditional machine learning models, GNNs can effectively process interconnected data, making them highly suitable for applications involving networks, social interactions, and road systems. In GNNs, each node in the graph learns representations by aggregating information from its neighbouring nodes. This message-passing mechanism allows the network to capture both local and global dependencies, improving predictive performance in complex environments.

GNNs play a key role in traffic flow prediction and route optimization. By modelling the road network as a graph, where intersections are nodes and roads are edges, the GNN can analyse real-time traffic data to determine optimal routes for emergency vehicles. This helps in identifying congestion patterns, predicting future traffic conditions, and dynamically rerouting vehicles to ensure the fastest response times. The ability of GNNs to adapt to real-time changes makes them a crucial component in enhancing emergency response systems.

- **Convolutional Neural Networks (CNNs)**

Convolutional Neural Networks (CNNs) are specialized deep learning models designed for image and video processing. They utilize convolutional layers to extract spatial features from images, making them highly effective for object detection, classification, and segmentation. CNNs consist of multiple layers, including convolutional, pooling, and fully connected layers, which help in detecting patterns such as edges, shapes, and textures.

This hierarchical learning structure enables CNNs to recognize objects with high accuracy, even under varying lighting and environmental conditions. CNNs are used for real-time image analysis from traffic cameras and drones.

They help in detecting obstacles, identifying accidents, and recognizing traffic signals. This information is then fed into the system to assist in dynamic rerouting and traffic management. By continuously analysing live traffic footage, CNNs enhance decision- making by providing accurate and up-to-date road conditions, ensuring that emergency vehicles can navigate through congested areas efficiently.

- **Reinforcement Learning (RL)**

Reinforcement Learning (RL) is a machine learning approach where an agent learns to make decisions by interacting with its environment. The agent receives rewards for desirable actions and penalties for undesirable ones, allowing it to optimize decision-making over time. RL is widely used in autonomous systems, robotics, and traffic management, where decision-making in dynamic and uncertain environments is crucial. Popular RL algorithms include Deep Q-Networks (DQN), Proximal Policy Optimization(PPO), and Advantage Actor-Critic (A2C).

RL is used to dynamically adjust traffic signals and optimize emergency vehicle routes. The system learns from past traffic patterns and continuously adapts to real-time conditions, reducing response times. For example, RL-based models can control smart traffic signals that prioritize emergency vehicles, clearing intersections before they arrive. Additionally, RL helps in making split-second decisions on rerouting strategies, ensuring the fastest and safest path is always chosen for emergency responders.

- **A* Algorithm**

The A* (A-star) algorithm is a graph traversal and pathfinding algorithm used to find the shortest route between two points. It combines Dijkstra's Algorithm (which finds the shortest path) and a heuristic function (which estimates the remaining cost to the goal).

The A* algorithm is used for real-time pathfinding for emergency vehicles. When an emergency call is received, the system calculates the shortest and least congested route using real-time traffic data. Unlike traditional GPS navigation, which only considers distance, A* considers factors such as road conditions, congestion levels, and traffic signals to ensure the fastest possible route. By integrating A* with machine learning models, the system can adapt dynamically, making it a highly effective tool for emergency response optimization.

### 3.1.1 Drawbacks

The drawbacks of the existing system include high computational requirements due to complex algorithms like GNNs and RL, making real-time processing challenging. Dependence on real-time data from traffic cameras, IoT sensors, and GPS may lead to inaccuracies if data transmission fails. The system's efficiency can be affected by unexpected road conditions, such as sudden accidents or construction work. Additionally, implementing such an advanced system requires significant infrastructure investment, which may not be feasible for all cities. Lastly, privacy concerns arise due to the constant monitoring of vehicles and road conditions.

## 3.2 PROPOSED SYSTEM

The proposed system detects ambulances using both audio and visual processing for enhanced accuracy. It captures real-time audio, extracts MFCC features, and compares them with a preloaded ambulance siren using correlation distance. Simultaneously, it processes CCTV footage using Inception V2 and R-CNN with transfer learning to detect ambulances in images. If either the siren or the ambulance is detected, an alert is triggered for emergency response. The intelligent prioritization system improves real-time emergency vehicle detection for smart traffic management and autonomous systems.

**Algorithm Used**

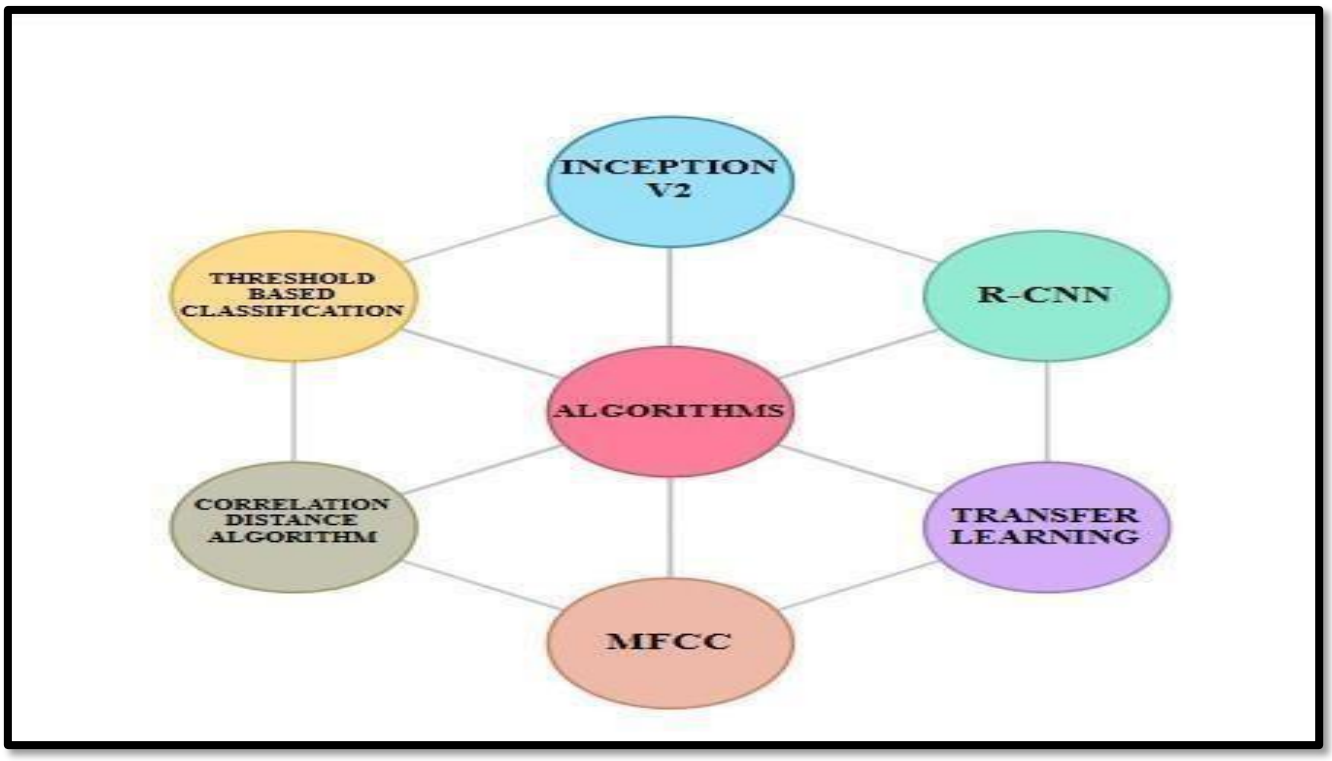


**Fig. 3.1 Algorithm Phase**

- **Inception V2**

Inception V2 is a deep learning model used for object detection in the proposed system, specifically for identifying ambulances in CCTV footage. It improves upon the original Inception architecture by reducing computational complexity while maintaining high accuracy. The model extracts multi-scale features using convolutional filters of different sizes, allowing it to recognize ambulances under various conditions such as different angles, lighting, and occlusions. Inception V2 is pre-trained on large datasets like COCO and ImageNet, which enables it to detect objects with minimal training on new datasets. By leveraging transfer learning, the model is fine-tuned on an ambulance-specific dataset to enhance its detection capabilities.

- **R-CNN (Region-Based Convolutional Neural Network)**

  R-CNN is used in this intelligent prioritization system for object detection by identifying regions of interest (ROIs) in images and classifying them into relevant categories, such as ambulances and non-emergency vehicles. The model works in three key steps: first, it generates multiple region proposals using a Selective Search algorithm; second, each region is processed through a CNN to extract feature representations; and third, a classifier (such as SVM or softmax) categorizes the regions. Since R-CNN is computationally expensive, its advanced versions like Fast R-CNN and Faster R-CNN improve efficiency by reducing redundant computations. Faster R-CNN is particularly useful in intelligent prioritization system as it incorporates a Region Proposal Network (RPN), which makes detection significantly faster. By combining R-CNN with Inception V2, the system achieves robust and efficient ambulance detection in CCTV footage.

- **Transfer Learning**

  Transfer learning is a crucial technique in intelligent prioritization system, allowing the system to leverage pre-trained deep learning models (such as Inception V2 and Faster R-CNN) instead of training them from scratch. Since training a deep neural network from the ground up requires vast amounts of labeled data and computational resources, transfer learning enables the model to reuse learned features from large datasets like ImageNet and COCO. This significantly reduces training time and improves detection accuracy. In the proposed system, transfer learning is applied by fine-tuning pre-trained models with a custom dataset containing ambulance images. This ensures that the model learns specific ambulance features, such as color, shape, and emergency symbols, while retaining general object recognition capabilities. By adjusting only the final layers of the pre-trained model, the system quickly adapts to the new task with minimal data. This approach enhances detection accuracy and makes real-time deployment feasible for smart surveillance applications.

- **MFCC (Mel-Frequency Cepstral Coefficients)**

MFCC is a widely used feature extraction technique for analyzing and identifying audio signals, particularly in speech and sound recognition tasks. MFCC is used to extract essential frequency features from real-time recorded audio and compare them with a preloaded ambulance siren. The process involves converting the audio signal into a spectrogram, applying the Mel scale filter banks, and computing the cepstral coefficients that represent the sound's characteristics. MFCC captures both temporal and spectral features, making it highly effective in distinguishing ambulance sirens from background noise. The extracted MFCC features are flattened into a feature vector, which is then used for similarity comparison with a reference siren. Since emergency sirens have distinct frequency patterns, MFCC ensures reliable detection even in noisy environments, enhancing the system's ability to identify approaching ambulances through sound analysis.

- **Correlation Distance Algorithm**

The Correlation Distance Algorithm, implemented using Scipy's spatial distance module, is used to measure the similarity between two MFCC feature vectors: one extracted from the reference ambulance siren and the other from real-time recorded audio. Correlation distance quantifies how similar the patterns of two signals are by analyzing the statistical correlation between their frequency components. A lower correlation distance indicates higher similarity, meaning the detected sound closely matches an ambulance siren. Correlation distance is computed after aligning the feature vector sizes to ensure a fair comparison. This approach helps in distinguishing sirens from other similar sounds, such as loud car horns or alarms, improving the system's accuracy. Since it operates on numerical feature representations rather than raw audio, it is computationally efficient and suitable for real-time emergency siren detection.

- **Threshold-Based Classification**

Threshold-based classification is used to decide whether an audio signal should be classified as an ambulance siren based on the computed correlation distance. In this method, a predefined similarity threshold (e.g., 0.8) is set, and if the computed correlation distance is below

this threshold, the system classifies the audio as an ambulance siren. This simple yet effective technique allows the system to make real-time decisions without the need for complex machine learning classifiers. The threshold can be fine-tuned based on empirical testing to balance false positives (misclassifying non-sirens as sirens) and false negatives (failing to detect actual sirens). By using threshold-based classification, the system ensures that only highly similar audio signals trigger an alert, making it robust against environmental noise and variations in siren sounds from different regions.

### 3.2.1 Advantages

The proposed system provides a multi-modal approach to ambulance detection by integrating both visual and audio recognition, ensuring high accuracy in real-world scenarios. By using Inception V2 and R-CNN, the system can efficiently detect ambulances in CCTV footage under various conditions, such as low lighting, occlusions, and different viewing angles. The use of transfer learning significantly reduces training time and computational costs while improving detection performance. Additionally, MFCC and correlation distance algorithms enable real-time siren detection, allowing the system to recognize ambulances even when they are not visible in the footage. The threshold-based classification ensures reliable decision-making by minimizing false alarms caused by background noise. The intelligent prioritization system enhances emergency response efficiency by notifying traffic management systems to clear pathways for ambulances, ultimately reducing delays in critical medical situations. Its scalability allows it to be integrated into smart cities, autonomous vehicles, and intelligent surveillance systems, making it a valuable tool for public safety.

By integrating Inception V2, R-CNN, transfer learning, MFCC, correlation distance, and threshold-based classification, the proposed system efficiently detects ambulances using both visual and audio data. This multi-modal approach enhances accuracy, ensuring real-time emergency vehicle detection for smart traffic management, autonomous vehicles, and surveillance systems.

# CHAPTER 4

# SYSTEM SPECIFICATIONS

## 4.1 HARDWARE SPECIFICATIONS

### 4.1.1 Hardware System Specification

- Computer (Minimum 4GB RAM, Dual-core processor, 500GB HDD/128GB SSD)
- Storage
- Power Supply (Converts AC to regulated DC for circuit operation)
- Transformer (Steps down voltage)
- Rectifier (Converts AC to DC)
- Smoothing & Regulator (Provides stable voltage)
- PIC Microcontroller
- Camera Module
- Microphone Module
- Stable internet connection
- Storage

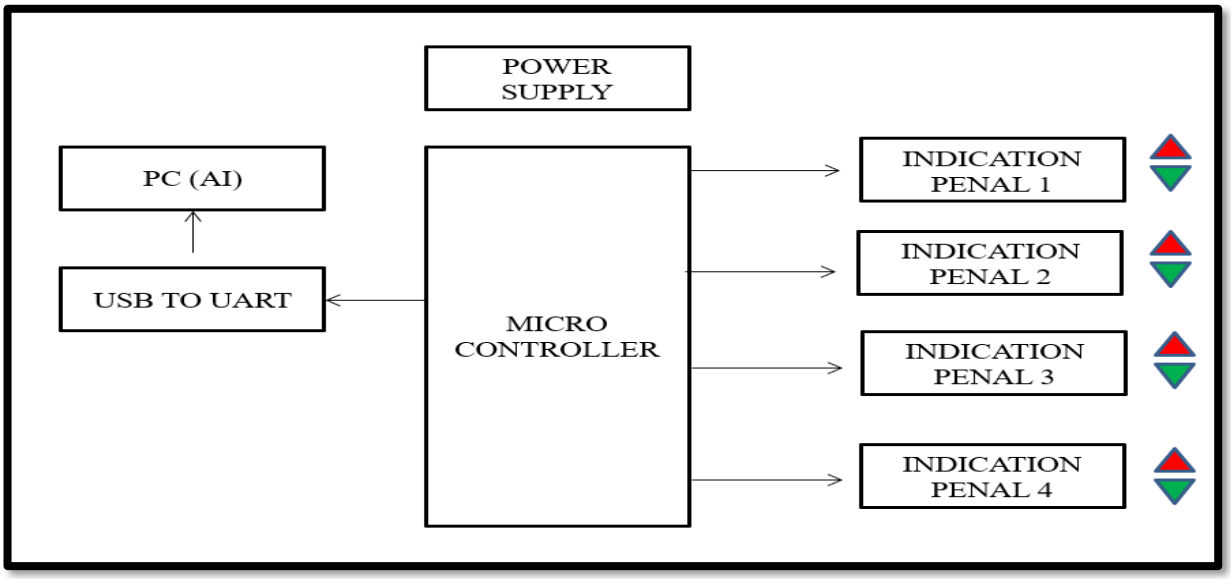


**Fig. 4.1 Hardware Block Diagram**

### 4.1.2 Hardware System Description

#### 4.1.2.1 Power Supply

Power supply is a reference to a source of electrical power. A device or system that supplies electrical or other types of energy to an output load or group of loads is called a power supply unit or PSU. The term is most commonly applied to electrical energy supplies, less often to mechanical ones, and rarely to others.

Power supplies for electronic devices can be broadly divided into linear and switching power supplies. The linear supply is a relatively simple design that becomes increasingly bulky and heavy for high current devices; voltage regulation in a linear supply can result in low efficiency. A switched-mode supply of the same rating as a linear supply will be smaller, is usually more efficient, but will be more complex.

**Linear Power supply**

An AC powered linear power supply usually uses a transformer to convert the voltage from the wall outlet (mains) to a different, usually a lower voltage. If it is used to produce DC, a rectifier is used. A capacitor is used to smooth the pulsating current from the rectifier. Some small periodic deviations from smooth direct current will remain, which is known as ripple. These pulsations occur at a frequency related to the AC power frequency (for example, a multiple of 50 or 60 Hz).

The voltage produced by an unregulated power supply will vary depending on the load and on variations in the AC supply voltage. For critical electronics applications a linear regulator will be used to stabilize and adjust the voltage. This regulator will also greatly reduce the ripple and noise in the output direct current. Linear regulators often provide current limiting, protecting the power supply and attached circuit from over current.

Adjustable linear power supplies are common laboratory and service shop test equipment, allowing the output voltage to be set over a wide range.

For example, a bench power supply used by circuit designers may be adjustable up to 30 volts and up to 5 amperes output. Some can be driven by an external signal, for example, for applications requiring a pulsed output.

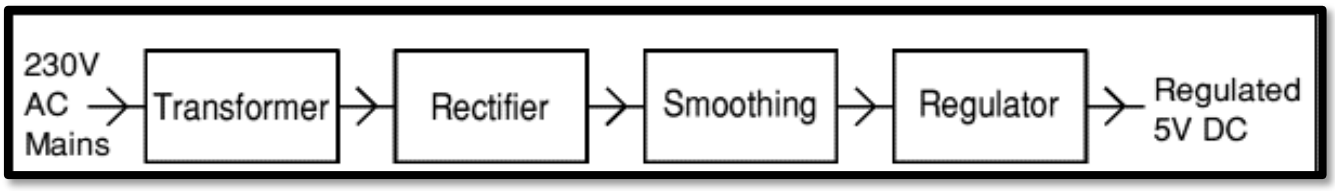


**Fig. 4.2 Regulated Power Supply System**

**Transformer**

Transformers convert AC electricity from one voltage to another with little loss of power. Transformers work only with AC and this is one of the reasons why mains electricity is AC.

The input coil is called the primary and the output coil is called the secondary. There is no electrical connection between the two coils; instead they are linked by an alternating magnetic field created in the soft-iron core of the transformer. The two lines in the middle of the circuit symbol represent the core.

Transformers waste very little power so the power out is (almost) equal to the power in. Note that as voltage is stepped down current is stepped up.The ratio of the number of turns on each coil, called the turn's ratio, determines the ratio of the voltages.

A step-down transformer has a large number of turns on its primary (input) coil which is connected to the high voltage mains supply, and a small number of turns on its secondary (output) coil to give a low output voltage.

$$\text{Turns ratio} = Vp/Vs = Nn/Ns \text{ and}$$
$$\text{Power out} = \text{Power in}$$
$$Vs*Is = Vp * Ip$$

Vp = primary (input) voltage
Np = number of turns on primary coil
Ip = primary (input) current

Vs = secondary (output) voltage
Ns = number of turns on secondary coil
Is = secondary (output) current

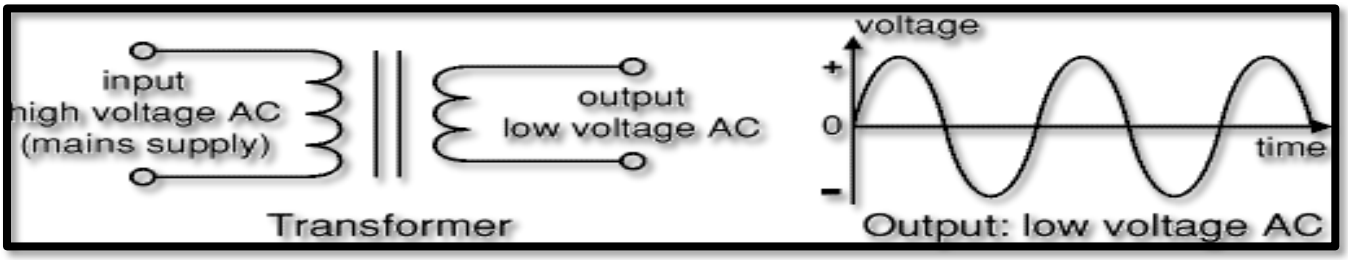


**Fig. 4.3 Working of Transformer**

The low voltage AC output is suitable for lamps, heaters and special AC motors. It is not suitable for electronic circuits unless they include a rectifier and a smoothing capacitor.

**Rectifier**

There are several ways of connecting diodes to make a rectifier to convert AC to DC. The bridge rectifier is the most important and it produces full-wave varying DC.

A full-wave rectifier can also be made from just two diodes if a centre-tap transformer is used, but this method is rarely used now that diodes are cheaper. A single diode can be used as a rectifier but it only uses the positive (+) parts of the AC wave to produce half-wave varying DC.

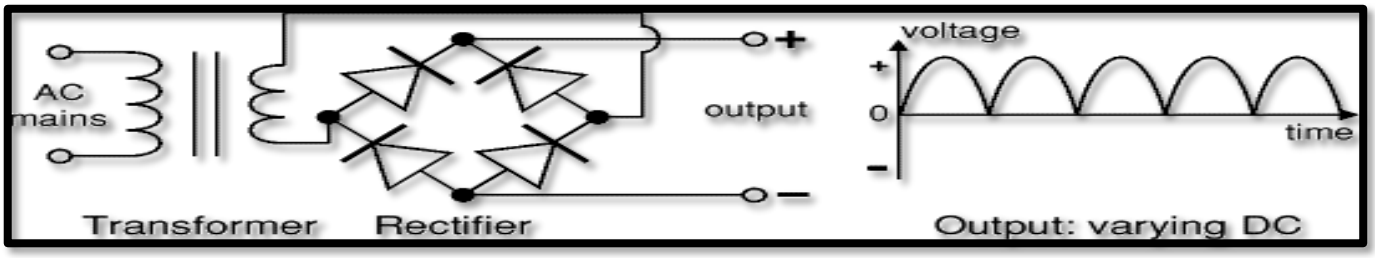


**Fig. 4.4 Working of Rectifier**

The varying DC output is suitable for lamps, heaters and standard motors. It is not suitable for electronic circuits unless they include a smoothing capacitor.

**Bridge rectifier**

A bridge rectifier can be made using four individual diodes, but it is also available in special packages containing the four diodes required. It is called a full-wave rectifier because it uses the entire AC wave (both positive and negative sections).

1.4 V is used up in the bridge rectifier because each diode uses 0.7V when conducting and there are always two diodes conducting, as shown in the diagram below. Bridge rectifiers are rated by the maximum current they can pass and the maximum reverse voltage they can withstand (this must be at least three times the supply RMS voltage so the rectifier can withstand the peak voltages). Please see the Diodes page for more details, including pictures of bridge rectifiers.

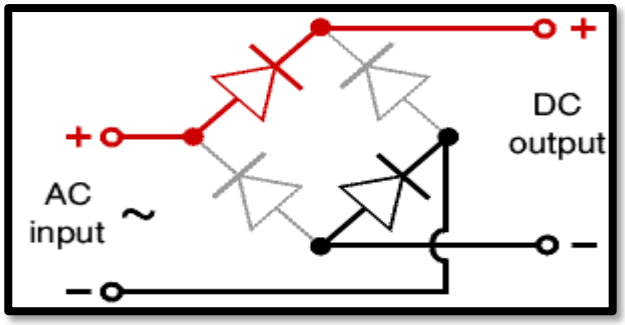


**Fig. 4.5 Bridge Rectifier**

Alternate pairs of diodes conduct, changing over the connections so the alternating directions of AC are converted to the one direction of DC.

Output: full-wave varying DC: (using the entire AC wave


**Fig.4.6 Full Wave Varying DC**

**Single diode rectifier**

A single diode can be used as a rectifier but this produces half-wave varying DC which has gaps when the AC is negative. It is hard to smooth this sufficiently well to supply electronic circuits unless they require a very small current so the smoothing capacitor does not significantly discharge during the gaps. Please see the Diodes page for some examples of

rectifier diodes

Output: half-wave varying DC (using only half the AC wave):

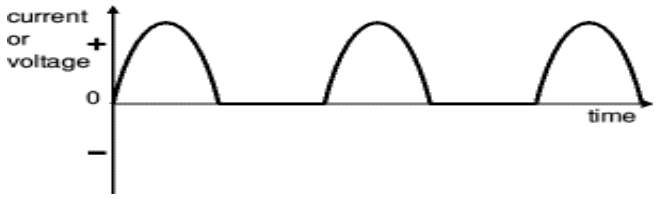


**Fig 4.7 Half Wave Varying DC**

**Smoothing**

Smoothing is performed by a large value electrolytic capacitor connected across the DC supply to act as a reservoir, supplying current to the output when the varying DC voltage from the rectifier is falling. The diagram shows the unsmoothed varying DC (dotted line) and the smoothed DC (solid line). The capacitor charges quickly near the peak of the varying DC, and then discharges as it supplies current to the output.

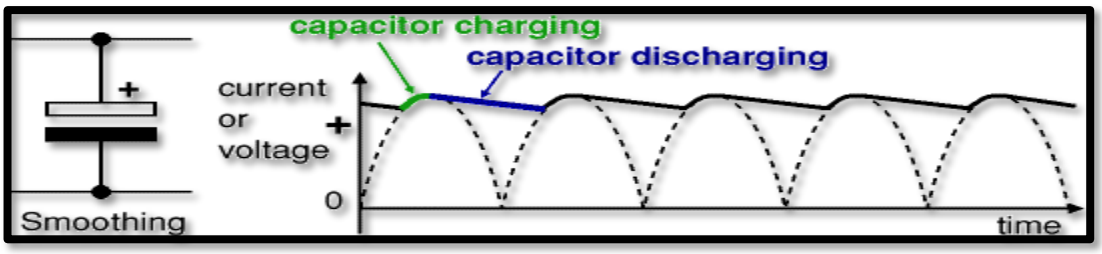


**Fig. 4.8 Capacitor**

Note that smoothing significantly increases the average DC voltage to almost the peak value (1.4 × RMS value). For example 6V RMS AC is rectified to full wave DC of about 4.6V RMS (1.4V is lost in the bridge rectifier), with smoothing this increases to almost the peak value giving 1.4 × 4.6 = 6.4V smooth DC.

Smoothing is not perfect due to the capacitor voltage falling a little as it discharges, giving a small ripple voltage.

For many circuits a ripple which is 10% of the supply voltage is satisfactory and the equation below gives the required value for the smoothing capacitor. A larger capacitor will give fewer ripples. The capacitor value must be doubled when smoothing half-wave DC.

Smoothing Capacitor for 10% ripple, C=5*10/vs.*f

C = smoothing capacitance in farads (F)

Io = output current from the supply in amps (A)

Vs = supply voltage in volts (V), this is the peak value of the unsmoothed DC

f = frequency of the AC supply in hertz (Hz), 50Hz in the UK.

**Regulator**

Voltage regulator ICs are available with fixed (typically 5, 12 and 15V) or variable output voltages. They are also rated by the maximum current they can pass. Negative voltage regulators are available, mainly for use in dual supplies. Most regulators include some automatic protection from excessive current ('overload protection') and overheating ('thermal protection').

The LM78XX series of three terminal regulators is available with several fixed output voltages making them useful in a wide range of applications.

One of these is local on card regulation, eliminating the distribution problems associated with single point regulation. The voltages available allow these regulators to be used in logic systems, instrumentation, HiFi, and other solid state electronic equipment. Although designed primarily as fixed voltage regulators these devices can be used with external components to obtain adjustable voltages and current.

Many of the fixed voltage regulator ICs has 3 leads and look like power transistors, such as the 7805 +5V 1A regulator shown on the right. They include a hole for attaching a heat sink if necessary.

i. Positive regulator

- input pin
- ground pin
- output pin

It regulates the positive voltage

ii. Negative regulator

- ground pin
- input pin

- output pin

It regulates the negative voltage

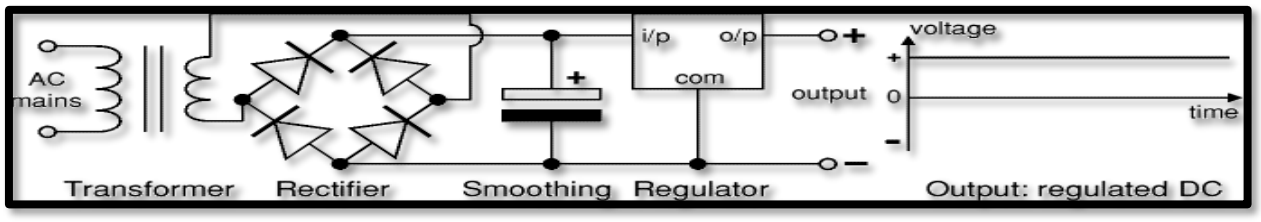


**Fig.4.9 Electronic Circuit Diagram**

The regulated DC output is very smooth with no ripple. It is suitable for all electronic circuits.

#### 4.1.2.2 PIC Microcontroller

PIC is a family of architecture microcontrollers made by Microchip Technology, derived from the PIC1640. Originally developed by General Instrument's Microelectronics Division. The name PIC initially referred to "Programmable Interface Controller".

PICs are popular with both industrial developers and hobbyists alike due to their low cost, wide availability, large user base, extensive collection of application notes, availability of low cost or free development tools, and serial programming (and re-programming with flash memory) capability.

The PIC microcontroller PIC16f877a is one of the most renowned microcontrollers in the industry. This controller is very convenient to use, the coding or programming of this controller is also easier. One of the main advantages is that it can be write-erase as many times as possible because it use FLASH memory technology. It has a total number of 40 pins and there are 33 pins for input and output. PIC16F877A is used in many pic microcontroller projects.

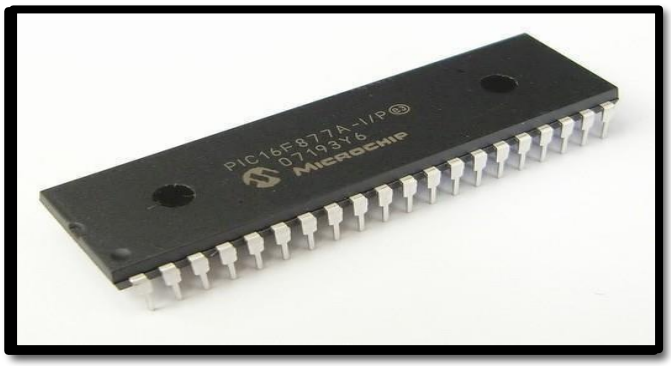


**Fig. 4.10 PIC Microcontroller**

Its flexible and can be used in areas where microcontrollers have never been used before as in coprocessor applications and timer functions etc. The microcontroller that has been used for the intelligent prioritization system is from PIC series. PIC microcontroller is the first RISC based microcontroller fabricated in CMOS (complementary metal oxide semiconductor) that uses separate bus for instruction and data allowing simultaneous access of program and data memory.

The main advantage of CMOS and RISC combination is low power consumption resulting in a very small chip size with a small pin count. The main advantage of CMOS is that has immunity to noise than other fabrication techniques.

**PIC 16877A**

Various microcontrollers offer different kinds of memories. EEPROM, EPROM, FLASH etc. are some of the memories of which FLASH is the most recently developed. Technology that is used in PIC 16877 is flash technology, so that data is retained even when the power is switched off.

Easy programming and erasing are other features of PIC 16F877. PIC16F877A microcontroller is used in the project.

**Table 4.1 Core Features**

| | |
|---|---|
| CPU | 8-bit PIC |
| Number of Pins | 40 |
| Operating Voltage (V) | 2 to 5.5 V |
| Number of I/O pins | 33 |
| ADC Module | 8ch, 10-bit |
| Timer Module | 8-bit(2), 16-bit(1) |
| Comparators | 2 |
| DAC Module | Nil |

| | |
|---|---|
| Communication | UART(1), SPI(1), I2C(1), MSSP(SPI/I2C) |
| External Oscillator | Up to 20Mhz |
| Internal Oscillator | Nil |
| Program Memory Type | Flash |
| Program Memory (KB) | 14KB |
| CPU Speed (MIPS) | 5 MIPS |

| | |
|---|---|
| RAM Bytes | 368 |
| Data EEPROM | 256 bytes |

**Peripheral Features**

- Timer0: 8bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during sleep via external clock/crystal
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- 10-bit multichannel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI (Master mode) and 12C (Master/ Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Brown – out detection circuitry for Brown out Reset (BOR)

**Pin Description**

PIC16F877A consists of 40 pins enclosed in 5 ports. Each port holds 8 pins which are bidirectional input/output pins. Pin diagram of PIC 16F877 is represented in Figure below.

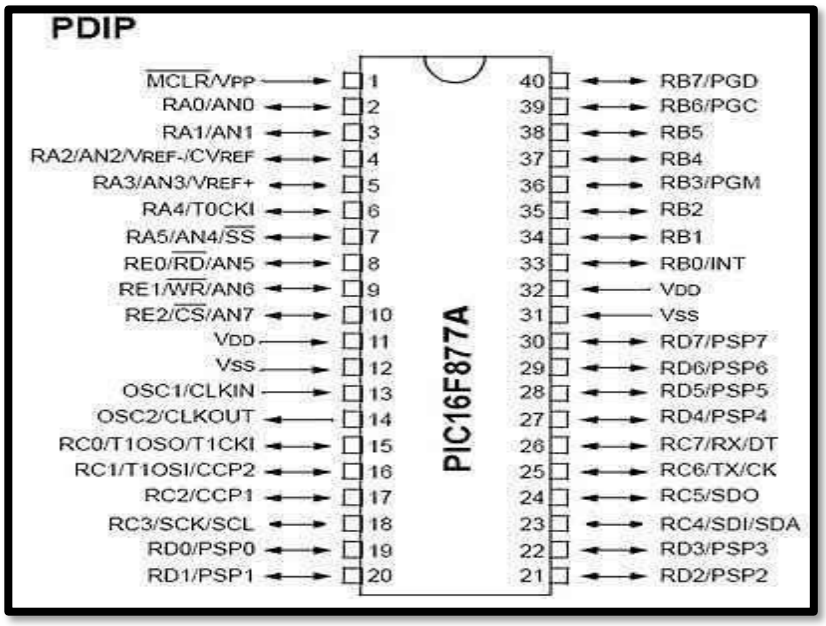


**Fig. 4.11 Pin Description Diagram**

### 4.1.2.3 LED Panel

A Traffic Indication LED Panel is a type of electronic display used to convey traffic-related information to drivers at intersections or along roads. It uses LED (Light Emitting Diode) technology to display different colors, typically red, green, and yellow (or amber), to control the flow of traffic.

Function of Traffic Indication LED Panels in the System:

1. Red Light (Stop) : When the system detects an emergency vehicle approaching a specific path, the corresponding indication panel turns green to allow the emergency vehicle to pass, while the other three indication panels turn red to stop other traffic and prevent congestion.

2. Green Light (Go): The indication panel for the path where the emergency vehicle is detected turns green to signal that this is the clear path for the emergency vehicle, allowing it to pass safely and without delay.
3. Yellow/Amber Light (Caution): Some systems may incorporate a yellow light to warn drivers about upcoming changes, like the approach of an emergency vehicle. This can act as a transitional state before switching from red to green or vice versa.

**Fig. 4.12 LED Panel**

Features of the Traffic Indication LED Panels

- Clear Visibility: LEDs are bright and visible from a distance, making them easy for drivers to see and follow, even in poor weather or at night.
- Quick Response: LED panels can switch between colors very quickly, providing real-time updates to drivers.
- Customization: These panels can be customized to display specific messages or warnings (such as "Emergency Vehicle Approaching").
- Energy Efficiency: LED lights are energy-efficient, long-lasting, and more durable than traditional light bulbs.

## 4.2 SOFTWARE SPECIFICATION

### 4.2.1 Software System Specification

- Programming Operating System : Windows 10/11
- Language Used : Python
- Backend : Python Script Window
- Frontend : Python Shell
- Software Description : Tensorflow, Keras, Numpy, Opencv, Pillow, Scikit-learn, object_detection.utils, Sounddevice, Librosa, Scipy.Spatial.

### 4.2.2 Software System Description

#### 4.2.2.1 MPLAB IDE Software

In MPLAB is a proprietary freeware integrated development environment for the development of embedded applications on PIC microcontrollers, and is developed by Microchip Technology.

MPLAB X is the latest edition of MPLAB and is developed on the NetBeans platform. MPLAB and MPLAB X support project management, code editing, debugging and programming of Microchip 8-bit, 16-bit and 32-bit PIC microcontrollers.

MPLAB is designed to work with MPLAB-certified devices such as the MPLAB ICD 3 and MPLAB REAL ICE, for programming and debugging PIC microcontrollers using a personal computer. PICK it programmers are also supported by MPLAB.

MPLAB 8.X is the last version of the legacy MPLAB IDE technology, custom built by Microchip Technology in Microsoft Visual C++. MPLAB supports project management, editing, debugging and programming of Microchip 8-bit, 16-bit and 32-bit Pic microcontrollers. MPLAB only works on Microsoft Windows. MPLAB is still available from Microchip's archives but is not recommended for new projects.

MPLAB supports the following compilers:

- MPLAB MPASM Assembler
- MPLAB ASM30 Assembler

- MPLAB C Compiler for PIC18
- MPLAB C Compiler for PIC24 and dsPIC DSCs
- MPLAB C Compiler for PIC32
- HI-TECH C

MPLAB X is the latest version of the MPLAB IDE built by Microchip Technology, and is based on the open-source NetBeans platform. MPLAB X supports editing, debugging and programming of Microchip 8-bit, 16-bit and 32-bit PIC microcontrollers.

MPLAB X is the first version of the IDE to include cross-platform support for Mac OS X and Linux operating systems, in addition to Microsoft Windows.

MPLAB X supports the following compilers:

- MPLAB XC8 — C compiler for 8-bit PIC devices
- MPLAB XC16 — C compiler for 16-bit PIC devices
- MPLAB XC32 — C/C++ compiler for 32-bit PIC devices
- HI-TECH C — C compiler for 8-bit PIC devices
- SDCC — open-source C compiler

HI-TECH C compiler for PIC10/12/16 MCUs (PRO)

This compiler has been discontinued and is no longer supported. This compiler has been replaced by the MPLAB® XC8 PRO (SW006021-2).

HI-TECH C Compiler for PIC10/12/16 MCUs - PRO fully implements the optimizations of Omniscient Code Generation™ - a whole-program compilation technology - to provide denser code and better performance on PIC MCUs. This ANSI C compiler integrates into Microchips MPLAB(R) IDE and is compatible with Microchip debuggers and emulators.

#### 4.2.2.2 Python Library

**Python Numpy**

NumPy plays a crucial role by enabling efficient numerical computations, which are essential for image and audio processing tasks. In the ambulance detection module, NumPy is

used for handling image arrays, converting pixel values into numerical matrices, and performing normalization operations.

By representing images as multi-dimensional arrays, NumPy facilitates fast manipulation of pixel data, making preprocessing steps like resizing, filtering, and augmentation more efficient. This is especially useful in deep learning models, where input data needs to be standardized for accurate predictions.

In the siren detection module, NumPy is used for processing and analyzing audio signals. The recorded ambulance siren is stored as a NumPy array, allowing mathematical operations like Fast Fourier Transform (FFT) to extract frequency components. Additionally, NumPy is used to compute Mel-Frequency Cepstral Coefficients (MFCCs), which help in distinguishing ambulance sirens from other background noises. The library's ability to perform matrix operations and similarity calculations enables efficient feature matching between the reference and detected siren sounds, ensuring accurate emergency vehicle identification.

Beyond image and audio processing, NumPy also supports data handling in model training and evaluation. It helps in creating structured datasets, managing large-scale numerical computations, and optimizing mathematical operations required for machine learning. Its integration with other libraries like TensorFlow and OpenCV ensures seamless execution of tasks across different modules, improving the overall efficiency and performance of the system.

**Open CV**

OpenCV OpenCV is a fundamental library primarily used for image and video processing tasks. In the ambulance detection module, OpenCV captures live frames from a camera, processes them, and applies techniques like resizing, grayscale conversion, and normalization to prepare images for deep learning models. It also helps in real-time object detection by drawing bounding boxes around detected ambulances, making visualization easier for users.

Additionally, OpenCV is used in the traffic signal automation module to detect ambulances approaching an intersection. By processing video feeds and analyzing object motion, it helps trigger the appropriate traffic light changes. Its seamless integration with TensorFlow and NumPy enhances real-time processing, ensuring quick and accurate emergency response.

How does computer recognize the image?

A lot of information is provided by human eyes based on what they observe. Machines are given the ability to observe everything, translate that vision into numbers, and store those numbers in memory. Here, the issue of how computers translate visuals into numbers arises. The pixel value is utilised to translate images into numbers, which is the answer.

The picture intensity at the particular location is represented by the numbers. In the above image, we have shown the pixel values for a grayscale image consist of only one value, the intensity of the black color at that location.

There are two common ways to identify the images:

**i. Grayscale**

Images that solely have the two hues black and white are known as grayscale images. Black is considered to have the lowest intensity whereas white has the highest, according to the contrast assessment of intensity.

**ii. RGB**

An RGB is a mixture of the colors red, green, and blue that results in a new color. Each pixel's value is extracted by the computer, which then organizes the information into an array for interpretation.

**Scikit-Learn**

Scikit-learn plays a crucial role, particularly in the machine learning aspects of ambulance detection and siren recognition. It is used for training and evaluating classification models, such as logistic regression or support vector machines (SVM), to distinguish ambulances from other vehicles based on extracted image or audio features. Its efficient implementation of feature scaling, splitting datasets, and cross-validation ensures the models generalize well to new data.

Additionally, in the siren detection module, Scikit-learn helps compare extracted audio features using similarity metrics and clustering algorithms. It provides essential tools for computing accuracy, precision, and recall, enabling fine-tuning of the detection model. With its user-friendly API and extensive functionality, Scikit-learn significantly enhances the machine.

**Pandas**

Pandas is extensively used for handling and processing structured data, ensuring efficient management of datasets related to ambulance detection and siren recognition. It facilitates the loading, cleaning, and manipulation of large image metadata and audio feature datasets, allowing seamless integration with machine learning models. Using Pandas DataFrames, the project organizes labeled data, such as image paths, bounding box coordinates, and extracted audio features, making it easier to preprocess and analyze.

Additionally, Pandas is instrumental in statistical analysis and performance evaluation. It helps in computing model accuracy, aggregating classification results, and visualizing trends over time. By enabling efficient data storage and retrieval, Pandas ensures a smooth workflow, making it an essential tool for managing the large volumes of information required for real-time ambulance detection and response systems.

**Tensorflow**

The TensorFlow plays a crucial role by providing a robust framework for building and deploying deep learning models for ambulance detection and siren recognition. It is used to train Convolutional Neural Networks (CNNs) for image-based detection and Recurrent Neural Networks (RNNs) or feature extraction techniques for audio classification. With TensorFlow's efficient computation graph, the project can leverage GPU acceleration to handle large datasets and complex neural network architectures, improving model performance and training speed.

Additionally, TensorFlow simplifies model deployment by supporting real-time inference on edge devices, allowing the ambulance detection system to function seamlessly in practical environments. The TensorFlow Model Zoo provides pre-trained object detection models like Faster R-CNN and SSD, which can be fine-tuned for ambulance recognition, significantly reducing training time while maintaining high accuracy. Its integration with TensorBoard also aids in monitoring model performance, tracking metrics, and optimizing hyperparameters for better generalization.

Following are the two TensorFlow — Convolutional Neural Networks

After understanding machine-learning concepts, we can now shift our focus to deep learning concepts. Deep learning is a division of machine learning and is considered as a crucial step taken by researchers in recent decades. The examples of deep learning implementation include applications like image recognition and speech recognition.

important types of deep neural networks:

- Convolutional Neural Networks
- Recurrent Neural Networks

In this chapter, we will focus on the CNN, Convolutional Neural Networks.

**Methodology**

Basics of convolutional neural networks The pooling layer, the rectified linear unit (ReLU) layer, the convolutional layer—commonly referred to as CONV2D—and a number of fully connected layers make up a deep convolutional neural network (CNN). To put it another way, a deep CNN is essentially made up of two subnetworks: a set of 2D convolutional layers and a traditional but deep neural network, as shown in Fig. 1. The foundational components of CNN are convolutional-2-dimensional (CONV2D) layers. In order to extract certain features or edges, a set of 2D filtering kernels (of size nn) are applied to an image (NN). The stride, kernel size, and number of kernels are important variables in the CONV2D layer operation.

The kernels can be design to extract a vertical, horizontal, diagonal edges and any other pattern from the input images. In fact, feature engineering and feature extraction from input images are carried out through the convolutional layers. The size of the input image is (NN3) if it is an RGB image, but the result of this convolution over a volume is 2D rather than a volume once more. To do this, a volume of size nn3 will be used to replicate each edge detector or kernel. The Rectified Linear Unit (ReLU), which converts all negative values produced by the convolution operation to zeroes, is typically placed below the convolutional layer.

• Pooling layers is one of the most common variations of this operation is max pooling. A max pooling layer follows a convolutional layer and performs two tasks: it determines the maximum value in the region and scales down the image. The quantity of downscaling is based on the pool size. Although average pooling is also an option, maximum pooling is more typical.

• Completely connected layers are comparable to typical neural network hidden layers, for which the number of outputs is the only factor to take into account during the training process.

The pre-trained model is essential for the usage of transfer learning. Several neural network architectures have been built by the deep learning community and are available in the Python and Keras libraries as well as the MATLAB deep learning toolbox.

For image classification and and object detection the most popular and state of the are the following

- AlexNet
- VGG-16
- VGG-19
- Inception V3
- XCeption
- ResNet-50

The differences among these networks are mainly the network architecture (the number and distribution and parameters of the convolution, pooling, and dropout, and fully connected layers ). And so, the overall number of trainable parameters, the computational complexity of the training process and attained performance.

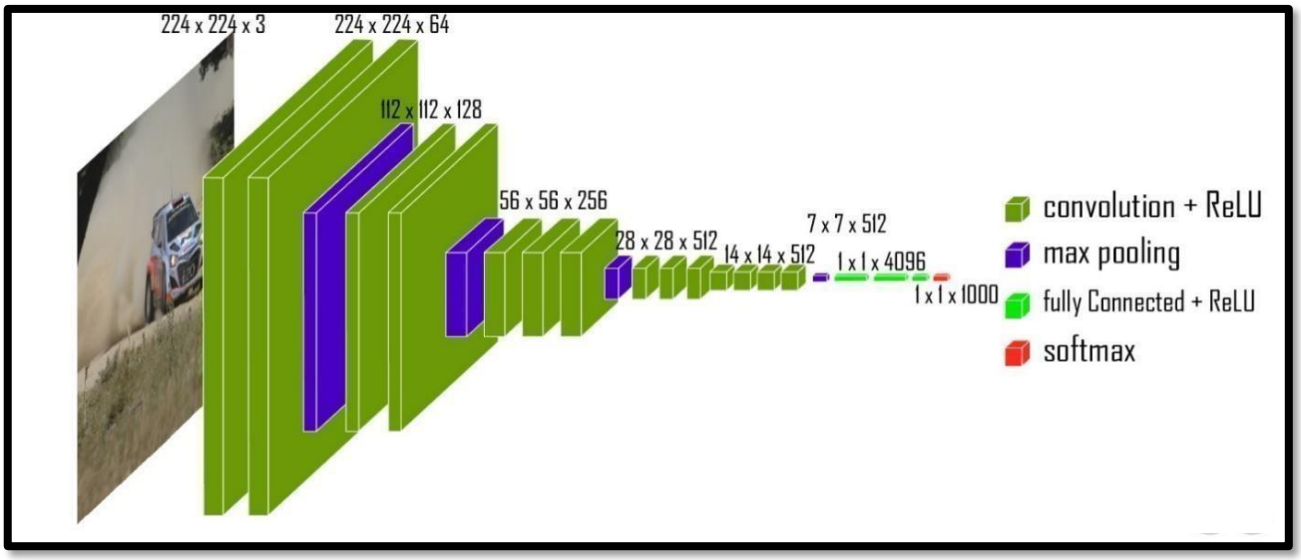


**Fig. 4.13 Layers of CNN**

**Python Pillow**

Pillow is essential in this intelligent prioritization system for handling image processing tasks, particularly for loading, modifying, and saving images used in ambulance detection. It provides an easy-to-use interface for working with different image formats, making it convenient for preprocessing operations such as resizing, cropping, and format conversion.

Before feeding images into deep learning models, Pillow ensures they are correctly formatted and standardized, improving the model's ability to learn meaningful patterns.

Moreover, Pillow assists in augmenting image datasets by applying transformations like rotation, flipping, and color adjustments, which help enhance the generalization of the model. Its integration with other libraries like NumPy and OpenCV allows seamless conversion between different image representations, making it a valuable tool in the preprocessing pipeline. This enhances the accuracy and robustness of the ambulance detection system by exposing the model to diverse real-world image variations.

Example:

Following example demonstrates the rotation of an image using python pillow:

```
from PIL import Image
#Open image using Image module
im = Image.open("images/cuba.jpg")
#Show actual Image im.show()
#Show rotated Image
im = im.rotate(45) im.show()
```

Output :

If you save the above program as Example.py and execute, it displays the original and rotated images using standard PNG display utility, as follows:

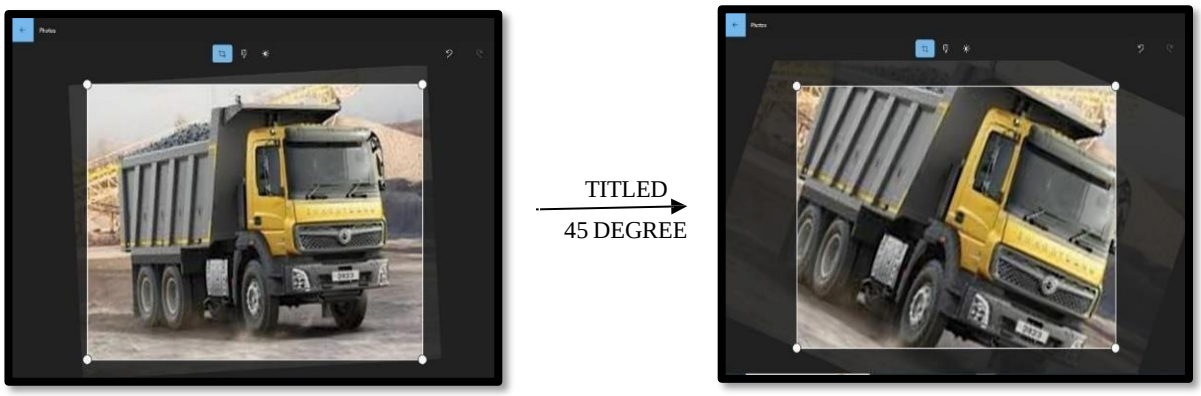


**Fig. 4.14 Data Augmentation by Tilting**

**Keras**

Keras plays a crucial role by providing a high-level, user-friendly interface for building and training deep learning models. It simplifies the implementation of convolutional neural networks (CNNs) for ambulance detection by offering pre-built layers, activation functions, and optimizers. With its easy-to-use API, Keras allows for quick prototyping and efficient model training, making it an ideal choice for deep learning applications. It seamlessly integrates with TensorFlow as a backend, ensuring scalability and performance while handling large datasets and complex neural network architectures.

Additionally, Keras supports various data preprocessing techniques, including image augmentation, normalization, and batch processing, which help improve model generalization. It also provides built-in callbacks for monitoring training progress, adjusting learning rates, and preventing overfitting. With its modular and flexible design, Keras enables smooth model experimentation, fine-tuning, and transfer learning, ensuring that the ambulance detection system achieves high accuracy and efficiency in real-world scenarios.

# CHAPTER 5
# SYSTEM DESIGN

## 5.1 SYSTEM ARCHITECTURE

A The system design consists of AI-powered traffic management integrating deep learning and IoT for real-time emergency vehicle detection. CNNs analyze CCTV footage, while MFCC processes siren audio to identify emergency vehicles. The system automatically adjusts traffic signals and communicates with autonomous vehicles to clear lanes. IoT-based synchronization ensures seamless coordination between signals and detection modules. This scalable solution enhances urban traffic efficiency and reduces emergency response delays.

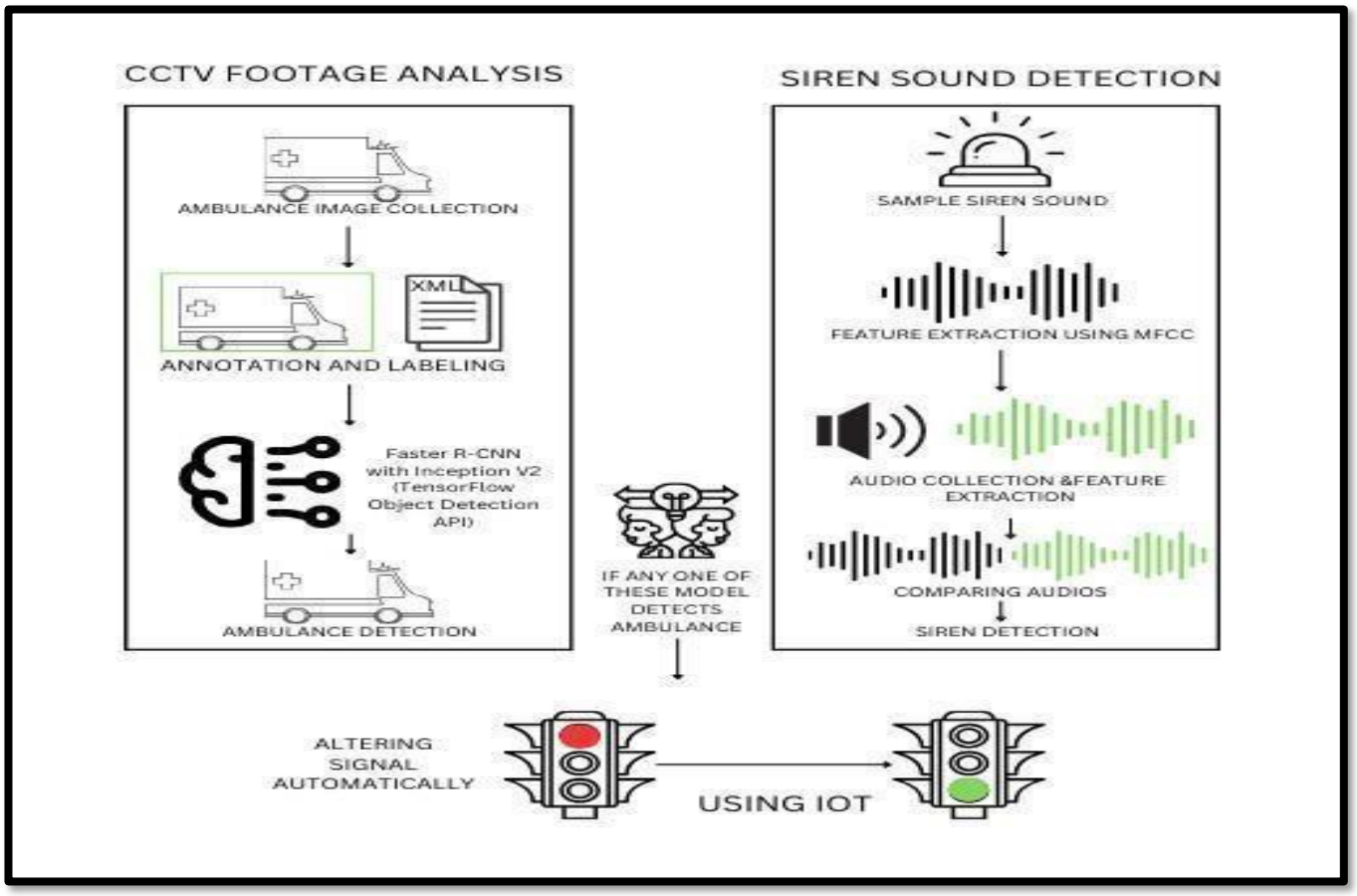


**Fig. 5.1 System Architecture**

## 5.2 DATA FLOW DIAGRAM

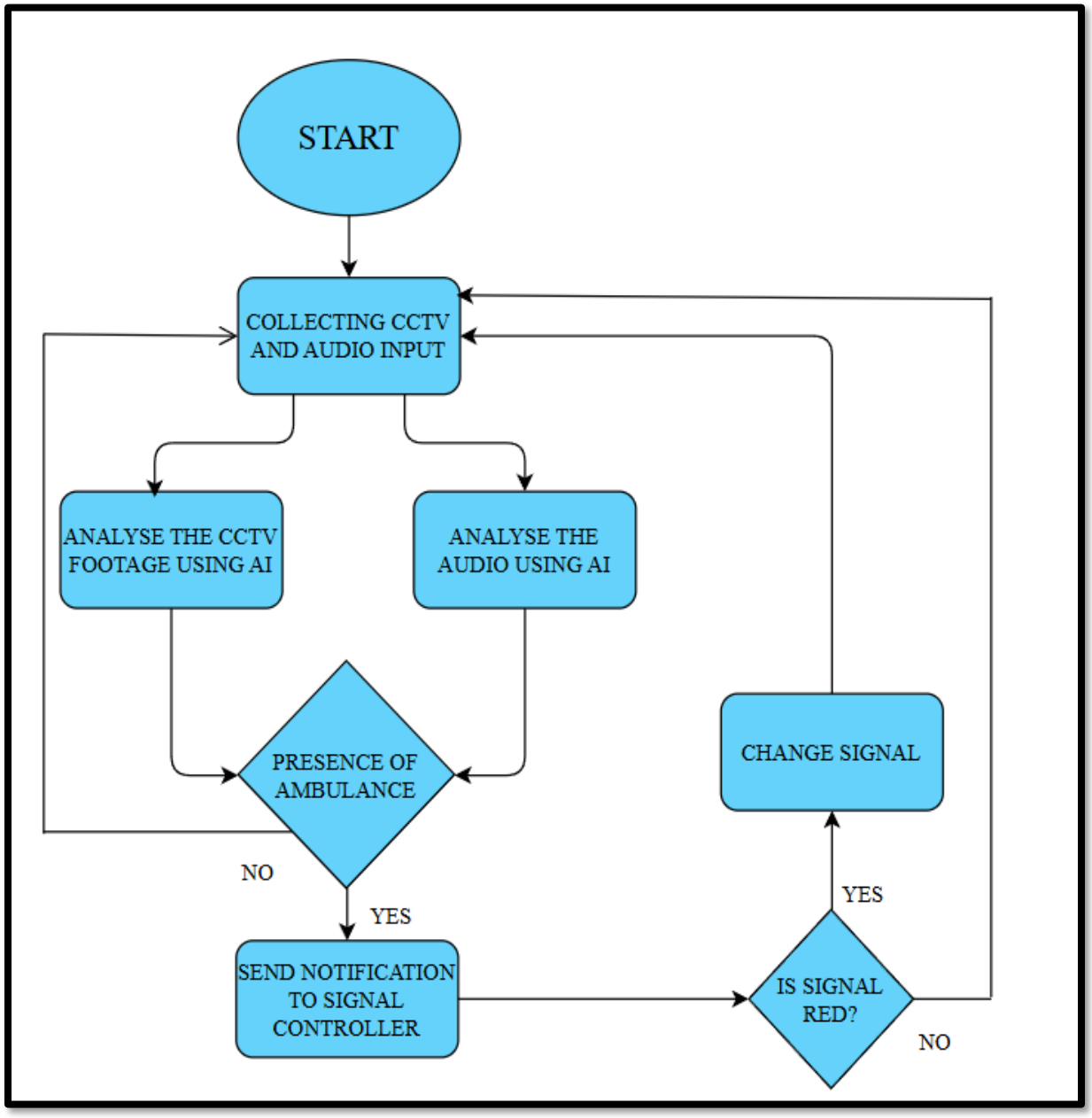


**Fig. 5.2 Data Flow Diagram**

## 5.3 USE CASE DIAGRAM

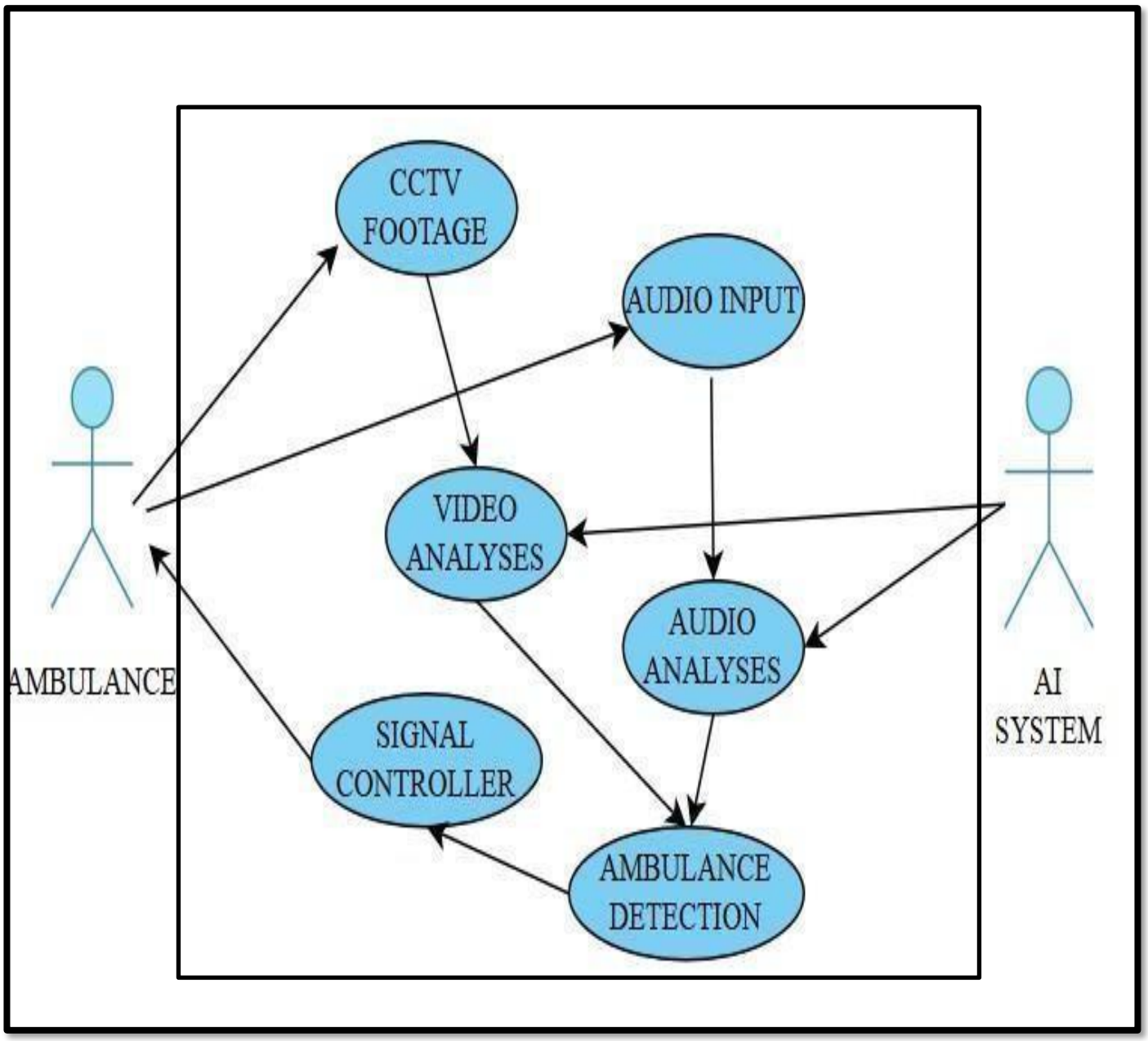


**Fig. 5.3 Use Case Diagram**

## 5.4 ACTIVITY DIAGRAM

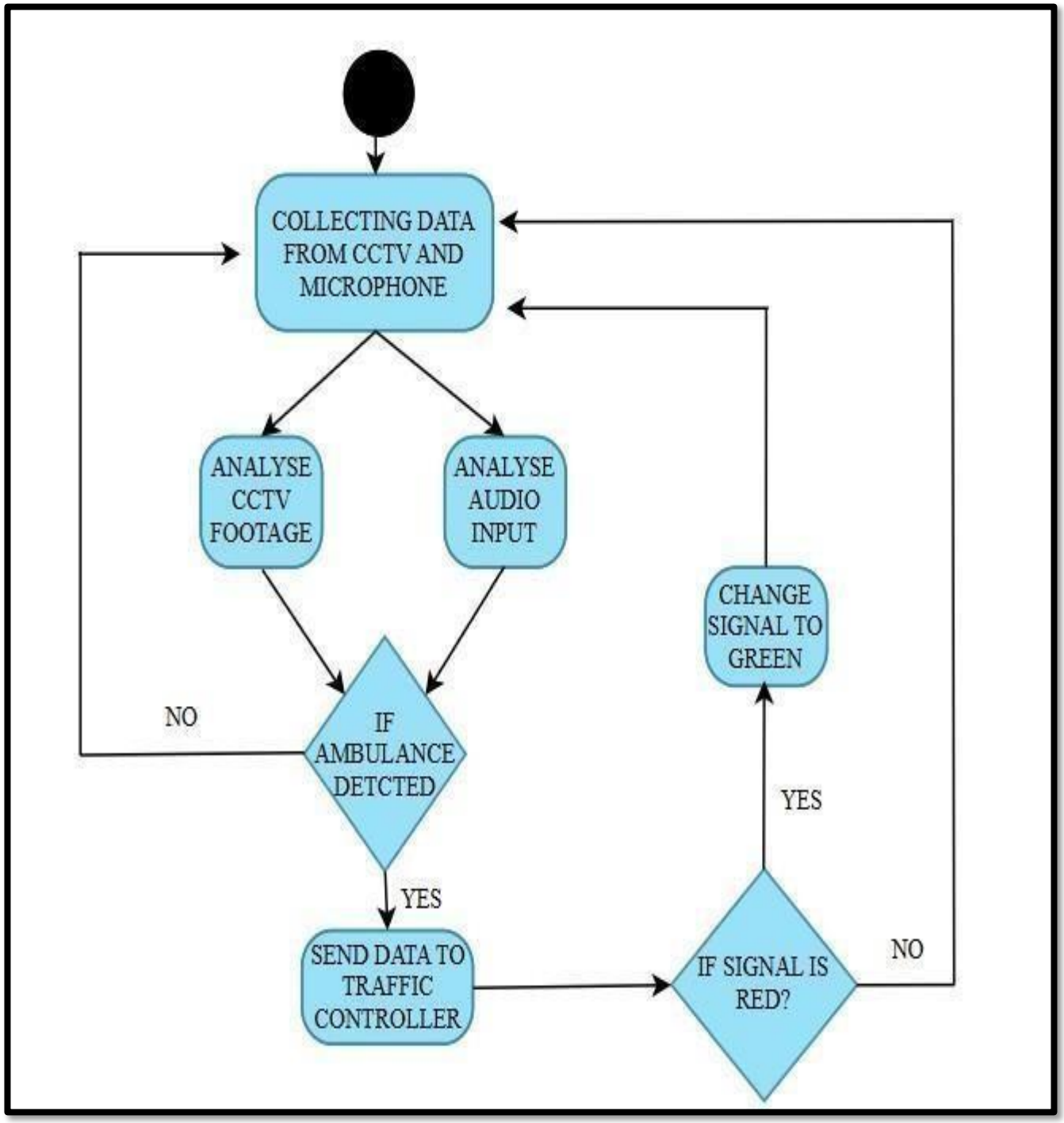


**Fig. 5.4 Activity Diagram**

## 5.5 SEQUENCE DIAGRAM

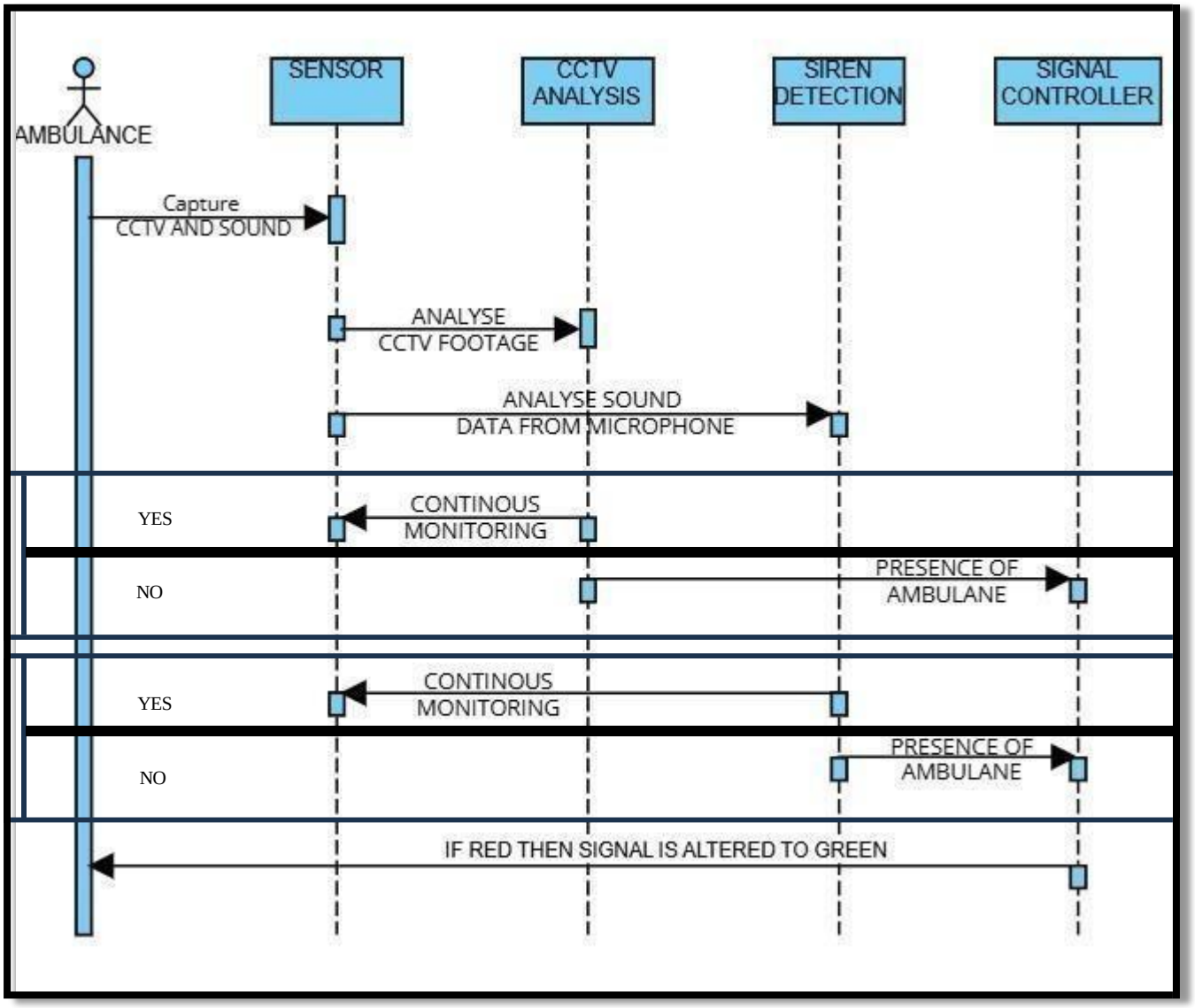


**Fig. 5.5 Sequence Diagram**

# CHAPTER 6

# MODULES DESCRIPTION

## 6.1 MODULES

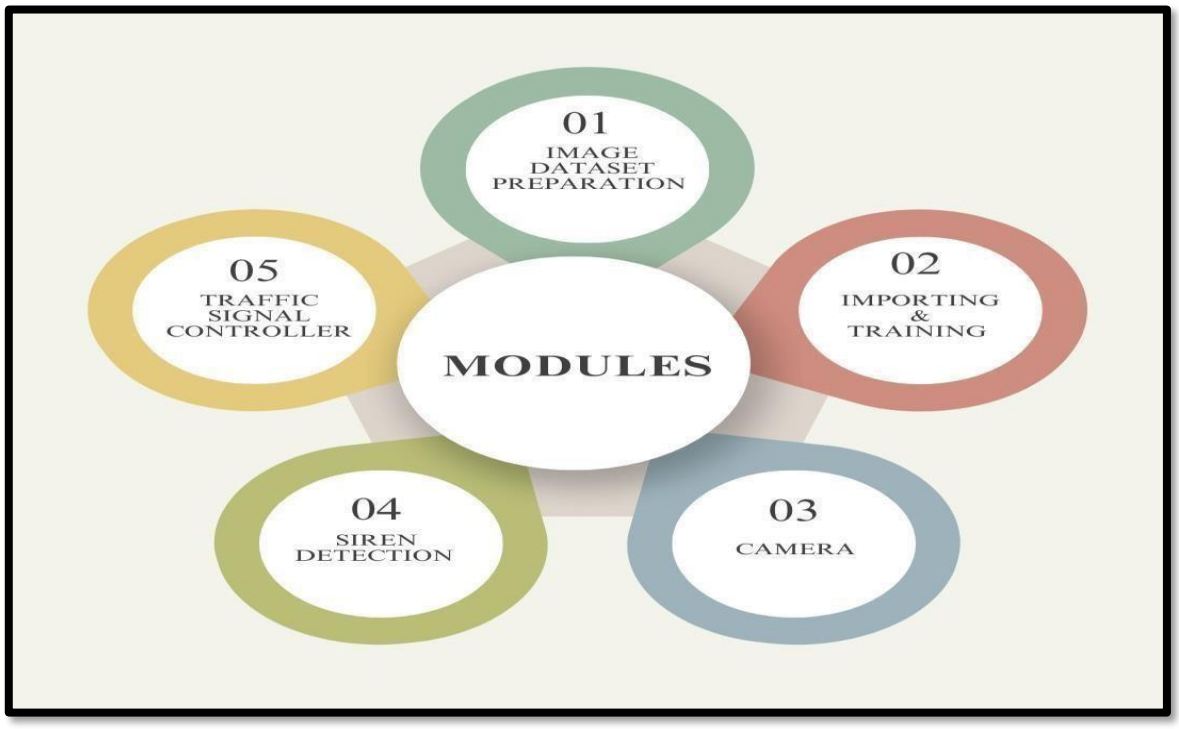


**Fig. 6.1 Modules**

## 6.2 IMAGE DATASET PREPARATORY MODULE

The Preparing a high-quality ambulance image dataset is a fundamental step in developing deep learning models for ambulance detection and classification. The accuracy and efficiency of the model depend heavily on the quality of the dataset. This module involves collecting diverse images, preprocessing them for consistency, annotating them for training purposes, and ensuring their quality through validation checks. Additionally, powerful libraries are utilized to automate and enhance each stage of the process.

**Image Dataset Collection**

To build a robust dataset, images of ambulances are collected from various sources. One of the primary methods involves capturing real-world images using cameras and drones, ensuring that different scenarios, such as moving ambulances in traffic, parked ambulances, and emergency scenarios, are included. Additionally, publicly available datasets and web scraping techniques are employed to gather more images from online repositories.

Web scraping allows the extraction of ambulance images from search engines and medical emergency-related websites, further expanding the dataset.

Crowdsourcing platforms also play a significant role in increasing dataset diversity. Platforms like Flickr, Unsplash, and crowd-labeled datasets contribute images taken from different angles, environments, and lighting conditions. The dataset is carefully curated to include various scenarios such as different weather conditions (rain, fog, clear skies) and times of the day (day, night, low-light conditions). This ensures that the deep learning model learns to generalize well across real-world variations rather than memorizing specific patterns.

**Image Preprocessing**

After collecting images, preprocessing is performed to standardize and enhance them for training. One of the key steps is resizing the images to a fixed dimension (such as 224x224 or 512x512 pixels) to maintain uniformity and reduce computational complexity. This ensures that all images have the same input size, allowing the model to process them efficiently.

Renaming is another crucial step, where a structured naming convention is applied, such as "ambulance_001.jpg," ensuring systematic dataset management and easy retrieval of images.

Normalization is then applied to scale pixel values to a specific range, typically between 0 and 1 or -1 and 1, to enhance computational efficiency and stabilize model training. Image augmentation is another essential preprocessing step that increases the dataset's diversity. Techniques such as rotation, flipping, brightness adjustment, cropping, and scaling are applied to create variations in images.

These transformations help prevent overfitting, allowing the model to recognize ambulances in different orientations, lighting conditions, and perspectives. By applying augmentation, the dataset effectively increases in size, improving the model's ability to generalize to unseen data.

**Image Labelling and Annotation**

For supervised deep learning models, proper labelling and annotation of images are crucial. The dataset is categorized into different classes, such as emergency ambulances, parked ambulances, and ambulances in traffic. Images are then annotated to help the model recognize and classify objects accurately. Annotation involves defining bounding boxes around ambulances, highlighting their exact position within an image. Tools like LabelImg, CVAT, and Roboflow are used to create these bounding boxes, ensuring precise object detection.

In cases where precise detection is required, segmentation masks are used instead of bounding boxes. This method helps identify the exact shape of the ambulance rather than just a rectangular box around it. Once the annotations are created, they are saved in widely accepted formats for compatibility with deep learning frameworks.

Formats such as COCO (JSON), Pascal VOC (XML), and YOLO (TXT) are used to store labelled data, ensuring smooth integration with models trained for classification, object detection, and segmentation tasks.

A critical step in this process is ensuring annotation accuracy. Any incorrectly labelled images can negatively impact the model’s learning, leading to poor predictions. As part of quality control, annotations are reviewed, corrected, and refined before proceeding to the training phase.

**Validation and Quality Check**

To maintain dataset integrity, a thorough validation check is performed. This step involves verifying annotation accuracy by checking whether bounding boxes correctly enclose ambulances and ensuring that segmentation masks align properly with object contours.

Data consistency is also examined, reviewing image resolution, clarity, and correctness to eliminate any mislabeled or distorted images. Duplicate removal is another crucial aspect, where identical images are identified and removed to prevent model bias toward repeated patterns. Augmented images are carefully reviewed to ensure that transformations do not distort the ambulance features. If augmentation techniques such as rotation or brightness adjustment significantly alter the image beyond recognition, such images are excluded from the final dataset. This validation process guarantees that only high-quality images with correct annotations contribute to the model training, ultimately improving its reliability and accuracy. Several Python libraries are employed to facilitate dataset preparation, preprocessing,and annotation. OpenCV is extensively used for image manipulation, including resizing, augmentation, and transformation operations. It provides efficient tools for processing large-scale image datasets with ease.

Pandas and NumPy are used for dataset organization, numerical computations, and managing metadata related to image annotations. These libraries help in structuring and analysing dataset attributes efficiently.

For deep learning applications, TensorFlow and Keras provide built-in utilities for data handling, preprocessing, and model integration. Albumentations is another powerful library specifically designed for image augmentation, offering advanced transformations such as random cropping, Gaussian noise addition, and contrast adjustments. Visualization plays a key role in dataset analysis, and libraries like Matplotlib and Seaborn are used to plot image distributions, visualize bounding boxes, and check dataset consistency. These libraries collectively streamline the dataset preparation workflow, ensuring efficiency and accuracy in model training.The image dataset preparatory module is a crucial step in training deep learning models for ambulance detection and classification. By systematically collecting diverse images, applying rigorous preprocessing techniques, ensuring accurate labeling, and validating data quality, the dataset is optimized for model learning.

## 6.3 TRAINING AND IMPORTING MODULE

Developing a deep learning model for real-time ambulance detection requires a structured approach, including importing essential libraries, preparing the dataset, selecting an appropriate model, training the model, and evaluating its performance. This module ensures the model is built efficiently, leveraging powerful image processing and deep learning techniques.

### Importing Essential Modules

To streamline the development process, essential Python libraries are imported at the beginning. This ensures code reusability, enhances scalability, and simplifies model implementation. The libraries used in this module cater to various tasks such as deep learning, image processing, numerical computation, and data visualization.

TensorFlow/Keras is the primary deep learning framework used for training and deploying the model. It provides high-level APIs for building Convolutional Neural Networks (CNNs) and optimizing model performance. OpenCV is utilized for image and video processing tasks, such as resizing, noise reduction, and real-time frame extraction from video streams. NumPy and Pandas play a crucial role in handling large datasets, performing numerical computations, and organizing metadata.

For data visualization, Matplotlib and Seaborn are used to generate plots, visualize training progress, and analyze dataset distributions. These libraries collectively facilitate a structured and efficient approach to developing the deep learning model.

### Data Preprocessing

Before training the model, ambulance images undergo preprocessing to standardize input features and improve learning efficiency. The dataset is collected from various sources, ensuring diverse lighting conditions, angles, and environments. Each image is resized to a fixed dimension (e.g., 224x224 pixels) to maintain consistency across training samples.

Normalization is applied by scaling pixel values between 0 and 1, which accelerates convergence and prevents numerical instability during training.

Augmentation techniques, such as flipping, rotation, brightness adjustments, and noise addition, are employed to artificially expand the dataset, improving the model's generalization ability.

After preprocessing, the dataset is split into three subsets:

- Training Set (70%) – Used to train the model and learn feature representations.
- Validation Set (20%) – Used to fine-tune hyperparameters and prevent overfitting.
- Testing Set (10%) – Used to evaluate final model performance on unseen data.

**Model Selection**

To ensure accurate and efficient emergency vehicle detection, a pre-trained deep learning model is selected and fine-tuned on the emergency vehicle dataset. The model must be capable of detecting ambulances, fire trucks, and police cars in real-world scenarios, including busy urban environments and varying lighting conditions.

Why Use Faster R-CNN with Inception V2?

Faster R-CNN with Inception V2 is chosen for this task due to its high accuracy and superior object detection performance. This model is well-suited for detecting small and medium-sized objects, making it ideal for recognizing emergency vehicles in complex backgrounds.

Key advantages include:

- High Accuracy: Faster R-CNN provides precise localization and classification of objects.

  Robust Small Object Detection: Its architecture ensures improved performance for detecting vehicles in crowded environments.
- Transfer Learning from COCO Dataset: Pre-trained on the Common Objects in Context (COCO) dataset, enabling the model to leverage prior knowledge and adapt quickly to

new datasets.

The model is downloaded from TensorFlow Model Zoo, which provides optimized models for object detection. By fine-tuning this pre-trained model, we can achieve high performance with relatively less training data.

**Model Configuration and Training**

Before training, the model's configuration file (pipeline.config) must be modified to align with the custom dataset.

Configurations Include:

- Number of Classes: The model is adapted for three emergency vehicle classes—ambulance, fire truck, and police car.
- Input Image Size Adjustment: Optimizing input dimensions ensures better feature extraction.
- Fine-Tuning Checkpoint: The model initializes from a COCO-trained Faster R-CNN checkpoint, accelerating convergence.
- Dataset Path Specification: Paths for training and testing datasets (train.record and test.record) are defined in the configuration.

Training Process

The model is trained using model_main_tf2.py, running for an extensive number of iterations (e.g., 200,000 steps) to ensure convergence.Real-time training progress is monitored via TensorBoard, which provides visual insights into model performance. Important metrics include:

- Loss: Expected to decrease steadily over training iterations.
- Mean Average Precision (mAP): Measures the accuracy of predictions across different confidence thresholds.
- Intersection over Union (IoU): Evaluates how well the predicted bounding boxes overlap with the ground truth.

**Model Configuration and Training**

Before training, the model's configuration file (pipeline.config) must be modified to align with the custom dataset.

Configurations Include:

- Number of Classes: The model is adapted for three emergency vehicle classes—ambulance, fire truck, and police car.
- Input Image Size Adjustment: Optimizing input dimensions ensures better feature extraction.
- Fine-Tuning Checkpoint: The model initializes from a COCO-trained Faster R-CNN checkpoint, accelerating convergence.
- Dataset Path Specification: Paths for training and testing datasets (train.record and test.record) are defined in the configuration.

Training Process

The model is trained using model_main_tf2.py, running for an extensive number of iterations (e.g., 200,000 steps) to ensure convergence.Real-time training progress is monitored via TensorBoard, which provides visual insights into model performance. Important metrics include:

- Loss: Expected to decrease steadily over training iterations.
- Mean Average Precision (mAP): Measures the accuracy of predictions across different confidence thresholds.
- Intersection over Union (IoU): Evaluates how well the predicted bounding boxes overlap with the ground truth.

**Model Evaluation**

Once training is complete, the model is tested on unseen images to assess its generalization capability. The evaluation phase involves multiple performance metrics to determine the model's accuracy and reliability.

Evaluation Metrics:

1. Precision & Recall: Determines how effectively the model identifies emergency vehicles while minimizing false positives.
2. Intersection over Union (IoU): Measures the overlap between predicted bounding boxes and actual ground truth labels.
3. Mean Average Precision (mAP): Averages the precision scores across different recall values, providing a comprehensive performance measure.

If the model exhibits suboptimal performance, hyperparameter tuning (e.g., adjusting the learning rate, batch size, or augmentation techniques) is applied to enhance accuracy and reduce misclassifications.

By leveraging Faster R-CNN with Inception V2 and transfer learning, an efficient and accurate ambulance detection system is developed. The model undergoes systematic training and evaluation, ensuring high performance in real-world conditions. If necessary, further refinements are made through hyperparameter tuning and additional dataset augmentation.

## 6.4 CCTV FOOTAGE ANALYSIS MODULE

This module is designed to detect and identify objects, specifically ambulances, in real-time using a TensorFlow Object Detection model. The system captures live video input, processes each frame, and classifies detected objects based on predefined labels. To ensure accurate and efficient detection, it employs advanced image processing techniques and deep learning models. The module is optimized for speed and responsiveness, allowing real-time decision-making.

### Model Loading and Initialization

The first step in the detection pipeline is loading the pre-trained object detection model. The frozen inference graph (frozen_inference_graph.pb), obtained from TensorFlow's Model Zoo, serves as the core detection engine. This graph contains a deep learning model trained on large datasets, enabling it to recognize and classify objects effectively.

To interpret the model's output, a label map utility is used, which maps numerical category indices to class labels. This ensures that the detected objects are correctly identified as ambulances or other relevant entities. Additionally, a TensorFlow session is initialized to handle the inference computations efficiently. By preloading the model into memory, the system can process incoming video frames with minimal delay, improving real-time performance.

**Image Capture and Preprocessing**

For real-time object detection, the system continuously captures video frames from a live camera feed using OpenCV. Each frame serves as input to the detection model, allowing it to analyse and classify objects dynamically. The captured frames can also be stored as images for further processing, logging, or debugging purposes.

Before feeding frames into the model, they undergo preprocessing to ensure compatibility. The dimensions of each frame are expanded to match the expected input shape of the model. Additional enhancements, such as resizing and normalizing pixel values, improve detection accuracy and efficiency. These preprocessing steps play a crucial role in optimizing the detection pipeline by reducing computational overhead and ensuring consistent input quality.

**Object Detection and Classification**

Once the frames are processed, the model extracts object information, including bounding boxes, class labels, and confidence scores. The bounding boxes define the location of detected objects, while the confidence scores indicate the likelihood of correct identification. The system compares detected objects against predefined class labels to determine their identities.

If an ambulance is detected with a confidence score above 70%, the system triggers an alert. This feature is crucial for emergency response scenarios, where quick and accurate

ambulance detection can facilitate faster decision-making.

The system's detection accuracy is further enhanced by applying techniques such as non-maximum suppression, which helps in filtering overlapping detections and ensuring reliable results.

**Visualization and Output**

To provide real-time feedback, the detected objects are visually highlighted by drawing bounding boxes around them. The detection results are displayed in a live OpenCV window, allowing users to monitor the system's performance interactively. The system also includes a mechanism to exit detection mode by pressing a specific key (e.g., 'q').

The real-time visualization helps users interpret detection results effortlessly. By integrating OpenCV's graphical capabilities, the system ensures that the output is both informative and user-friendly. These visual outputs, combined with accurate classification, make the module an effective tool for real-time ambulance detection.

**Libraries Used**

Several Python libraries are utilized to implement this module efficiently. NumPy is used for numerical operations, while OpenCV (cv2) handles image and video processing.

TensorFlow powers the deep learning model, enabling robust object detection. PIL (Python Imaging Library) assists with image handling, and object_detection.utils provides essential utilities for model deployment. These libraries work together to ensure seamless real-time object detection and classification.

## 6.5 SIREN SOUND DETECTION MODULE

This module is designed to identify and detect ambulance sirens in real-time using audio processing and machine learning techniques. By continuously monitoring environmental sounds, the system can recognize emergency sirens and trigger alerts accordingly.This capability is crucial for applications such as smart traffic management, where timely detection of ambulances can help optimize traffic signals and facilitate faster emergency response.

**Reference Siren Sound Processing**

To accurately detect ambulance sirens, the system first processes a pre-recorded reference siren sound (e.g., *ambulance.mp3*). This reference file serves as a baseline for comparison with real-time audio inputs. The key step in this process is extracting Mel-Frequency Cepstral Coefficients (MFCCs), which capture the unique frequency patterns of the siren.

MFCCs are widely used in speech and audio recognition as they represent the essential features of a sound signal while reducing noise and irrelevant variations. Once extracted, these MFCCs are converted into a flattened feature array, making it easier to compare against newly recorded audio samples. This pre-processing ensures that the system has a reliable reference for detecting sirens in real-world environments.

**Real-Time Audio Recording**

The system continuously listens for siren sounds using the sounddevice library, which provides real-time audio capture capabilities. To ensure efficient processing, audio is recorded in 5-second intervals at a sampling rate of 22,050 Hz.

This sample rate is sufficient for capturing the key frequency characteristics of an ambulance siren without unnecessary computational overhead. The recorded audio is then stored as a NumPy array, allowing for seamless integration with machine learning models and signal processing techniques. By segmenting the audio into fixed-duration samples, the system can analyse sound patterns dynamically, improving detection accuracy in noisy environments.

**Feature Matching and Similarity Check**

Once a new audio sample is recorded, the system extracts MFCCs from it, just as it did for the reference siren sound. These extracted features are then compared against the stored reference feature array to determine the similarity between the two sounds.

To ensure a fair comparison, the system applies feature normalization and scaling, making sure that the extracted features are uniform across different recording conditions.

The similarity between the reference and recorded siren is computed using correlation distance, a metric that measures how closely two feature sets match. A lower correlation distance indicates a higher likelihood that the recorded sound is an ambulance siren.

**Siren Detection and Alert Mechanism**

Based on the similarity check, the system calculates a Similarity Score. If this score meets or exceeds a predefined threshold (e.g., 0.8), the system confirms the presence of an ambulance siren and triggers an alert. If the similarity falls below the threshold, no siren is detected, preventing false alarms.

In real-world applications, this detection mechanism can be integrated into smart traffic control systems to automatically adjust traffic signals when an ambulance is approaching. By enabling seamless communication between emergency vehicles and road infrastructure, The intelligent prioritization system can help reduce emergency response times and improve urban mobility.

**Libraries Used**

Several Python libraries facilitate the development of this module. Sounddevice captures real-time audio, while Librosa handles feature extraction and MFCC computation. NumPy is used for numerical operations and data storage, ensuring efficient processing of audio signals. Finally, SciPy.Spatial provides advanced similarity computation techniques, enabling precise siren detection. Together, these libraries form a robust and scalable framework for real-time ambulance siren detection.

## 6.6 AUTOMATED TRAFFIC SIGNAL CONTROL MODULE

This module is designed to automate traffic signal control when an ambulance is detected, ensuring a smooth and prioritized passage for emergency vehicles. By integrating sensor-based detection with an PIC-controlled traffic management system, the module enhances urban mobility and reduces emergency response time. It allows real-time decision-making to modify traffic signals dynamically, giving priority to ambulances and minimizing delays in emergency situations.

**Power Supply and Management**

A stable power supply is essential for the continuous operation of the circuit components. The power system converts AC voltage to a regulated DC supply, ensuring a consistent power source for the PIC microcontroller and other connected devices. The transformer steps down the high-voltage AC from the mains supply to a lower, safer voltage level, while the rectifier converts AC to DC, making it suitable for electronic circuits. Additionally, a smoothing capacitor and a voltage regulator stabilize the DC output, eliminating fluctuations and ensuring uninterrupted functionality of the system.

**PIC Controller and Processing Unit**

The The PIC microcontroller serves as the central processing unit for the traffic control system. It continuously monitors input from sensors or cameras, processes the data, and makes adjustments. Equipped with both digital and analog I/O pins, the PIC microcontroller seamlessly interacts with external components such as infrared sensors, cameras, and traffic light controllers. Furthermore, it supports various communication protocols, including Pulse Width Modulation (PWM), Serial Communication (USART), and SPI, allowing effective coordination between different module.

**Traffic Signal Control and Automation**

Once the decision is made, the PIC triggers the traffic signal adjustments. The green light is activated for the ambulance's lane, ensuring a clear path, while red lights are simultaneously applied to other lanes to halt general traffic. This automated control mechanism helps in reducing congestion and delays, making way for emergency vehicles to pass through efficiently. After the ambulance has cleared the intersection, the system resets itself to its normal functioning mode, restoring traffic signals to their standard operation.

**System Integration and Real-World Applications**

The intelligent prioritization system can be integrated with smart city infrastructure, where traffic signals communicate with a central monitoring unit for optimized traffic flow management. With advancements in IoT, this setup can be connected to cloud-based traffic management systems, enabling real-time monitoring and remote control.

# CHAPTER 7
# CONCLUSION AND FUTURE ENHANCEMENT

## 7.1 CONCLUSION

The proposed Emergency Vehicle Detection System provides an innovative solution to reduce delays faced by ambulances in congested urban areas. By integrating computer vision, deep learning, and audio-based detection, the system accurately identifies emergency vehicles and ensures immediate traffic light adjustments to prioritize their movement. The use of wireless communication and IoT-based automation enables seamless data transmission between detection units and traffic control systems, ensuring real-time decision-making.

The intelligent prioritization system enhances emergency response efficiency, road safety, and overall traffic management, contributing to the development of smart city infrastructure. By minimizing delays for emergency vehicles, the project aims to save lives and optimize urban mobility. Future enhancements could include improving detection accuracy in complex environments, integrating (Vehicle-to-Everything) communication, and expanding the system to detect . Overall, The intelligent prioritization system represents a significant advancement in intelligent traffic control, making cities safer and more efficient.

## 7.2 APPLICATIONS

Smart traffic management systems play a crucial role in dynamically controlling traffic flow by prioritizing emergency vehicles, thereby reducing delays at intersections. By integrating with emergency response systems, these technologies enhance the efficiency of ambulances, ensuring a clear path during critical situations. The implementation of IoT-based automation in smart cities further optimizes traffic flow, utilizing AI-driven decision-making for seamless coordination. In the healthcare sector, such advancements significantly reduce patient transport time, allowing ambulances to reach medical facilities faster.

Additionally, law enforcement and disaster management teams benefit from these systems, as they assist in navigating through congested roads during emergencies. Automated traffic control centers further enhance real-time traffic monitoring and management.

## 7.3 FUTURE ENHANCEMENT

Future enhancements for this intelligent prioritization system focus on integrating advanced technologies to further improve ambulance detection and traffic management. One key improvement is the incorporation of AI-powered predictive analytics, which can anticipate ambulance arrival times based on real-time traffic data and optimize signal control accordingly. Machine learning models can be trained on historical data to enhance detection accuracy, reducing false positives and improving system reliability.

Another significant enhancement is the deployment of a cloud-based monitoring system that allows centralized traffic control authorities to oversee multiple intersections in real time. By leveraging IoT connectivity, data from cameras, sensors, and siren detectors can be transmitted to cloud servers, enabling remote monitoring and decision-making. This would facilitate seamless coordination between emergency response units and traffic control centers, ensuring a faster response time.

Lastly, expanding the system to cover larger urban areas with AI-driven traffic flow analysis can help optimize overall city traffic management. By incorporating crowd and congestion analysis, traffic signals can be dynamically adjusted based on the movement patterns of all vehicles, further improving emergency response times while maintaining smooth traffic flow for the general public. These advancements will make the system more robust, scalable, and adaptable to future smart city infrastructures.

## APPENDIX A

## SOURCE CODE

### CCTV Footage Analysis

```
from import numpy as np
import os
import sys
import tensorflow as tf
from distutils.version import StrictVersion
from collections import defaultdict
from PIL import Image
from object_detection.utils import ops as utils_ops

if StrictVersion(tf.__version__) < StrictVersion('1.9.0'):
  raise ImportError('Please upgrade your TensorFlow installation to v1.9.* or later!')

from utils import label_map_util
from utils import visualization_utils as vis_util

MODEL_NAME = 'inference_graph'
PATH_TO_FROZEN_GRAPH = MODEL_NAME + '/frozen_inference_graph.pb'
PATH_TO_LABELS = 'training/labelmap.pbtxt'

detection_graph = tf.Graph()
with detection_graph.as_default():
  od_graph_def = tf.GraphDef()
  with tf.gfile.GFile(PATH_TO_FROZEN_GRAPH, 'rb') as fid:
    serialized_graph = fid.read()
    od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')

category_index = label_map_util.create_category_index_from_labelmap(PATH_TO_LABELS,
use_display_name=True)

def run_inference_for_single_image(image, graph):
    if 'detection_masks' in tensor_dict:
        # The following processing is only for single image
        detection_boxes = tf.squeeze(tensor_dict['detection_boxes'], [0])
        detection_masks = tf.squeeze(tensor_dict['detection_masks'], [0])
        real_num_detection = tf.cast(tensor_dict['num_detections'][0], tf.int32)
        detection_boxes = tf.slice(detection_boxes, [0, 0], [real_num_detection, -1])
```

```
            detection_masks, detection_boxes, image.shape[0], image.shape[1])
        detection_masks_reframed = tf.cast(
            tf.greater(detection_masks_reframed, 0.5), tf.uint8)
        tensor_dict['detection_masks'] = tf.expand_dims(
            detection_masks_reframed, 0)
    image_tensor = tf.get_default_graph().get_tensor_by_name('image_tensor:0')

    # Run inference
    output_dict = sess.run(tensor_dict,
                           feed_dict={image_tensor: np.expand_dims(image, 0)})

    # all outputs are float32 numpy arrays, so convert types as appropriate
    output_dict['num_detections'] = int(output_dict['num_detections'][0])
    output_dict['detection_classes'] = output_dict[
        'detection_classes'][0].astype(np.uint8)

    output_dict['detection_boxes'] = output_dict['detection_boxes'][0]
    output_dict['detection_scores'] = output_dict['detection_scores'][0]
    global a2
    if 'detection_masks' in output_dict:
        output_dict['detection_masks'] = output_dict['detection_masks'][0]
    if output_dict['detection_classes'][0] == 1 and output_dict['detection_scores'][0] > 0.70:
      print('AMBULANCE')
    return output_dict
a1=0
a2=0
import cv2
cap = cv2.VideoCapture(0)
try:
    with detection_graph.as_default():
        with tf.Session() as sess:
                # Get handles to input and output tensors
                ops = tf.get_default_graph().get_operations()
                all_tensor_names = {output.name for op in ops for output in op.outputs}
                tensor_dict = {}
                for key in [
                  'num_detections', 'detection_boxes', 'detection_scores',
                  'detection_classes', 'detection_masks'
                ]:
                    tensor_name = key + ':0'
                    if tensor_name in all_tensor_names:
                        tensor_dict[key] = tf.get_default_graph().get_tensor_by_name(
                      tensor_name)
```

```
            while True:

                (__, image_np) = cap.read()
                # Expand dimensions since the model expects images to have shape: [1, None, None, 3]
                image_np_expanded = np.expand_dims(image_np, axis=0)
                cv2.imwrite('capture.jpg',image_np)
                # Actual detection.
                output_dict = run_inference_for_single_image(image_np, detection_graph)
                # Visualization of the results of a detection.
                vis_util.visualize_boxes_and_labels_on_image_array(
                    image_np,
                    output_dict['detection_boxes'],
                    output_dict['detection_classes'],
                    output_dict['detection_scores'],
                    category_index,
                    instance_masks=output_dict.get('detection_masks'),
                    use_normalized_coordinates=True,
                    line_thickness=8)
                cv2.imshow('object_detection', cv2.resize(image_np,(800,600)))
                if cv2.waitKey(1)& 0xFF == ord('q'):
                    cap.release()
                    cv2.destroyAllWindows()
                    break
except Exception as e:
    print(e)
    #cap.release()
```

**Siren Detection**

```
import sounddevice as sd
import numpy as np
import librosa
import scipy.spatial

def load_ambulance_sound(file_path="ambulance.mp3"):
    ref_audio, ref_sr = librosa.load(file_path, sr=22050)
    ref_mfcc = librosa.feature.mfcc(y=ref_audio, sr=ref_sr)
    return ref_mfcc.flatten(), ref_sr

def record_audio(duration=5, sr=22050):
    print("🎙 Listening for siren...")
    audio = sd.rec(int(duration * sr), samplerate=sr, channels=1, dtype=np.float32)
    sd.wait()
    print("✅ Audio recorded successfully")
```

```
    return np.squeeze(audio), sr

def match_array_size(ref, test):
    min_length = min(len(ref), len(test))
    return ref[:min_length], test[:min_length]

def compare_audio(ref_mfcc, ref_sr, test_audio, test_sr):
    test_mfcc = librosa.feature.mfcc(y=test_audio, sr=test_sr).flatten()
    ref_mfcc, test_mfcc = match_array_size(ref_mfcc, test_mfcc)
    distance = scipy.spatial.distance.correlation(ref_mfcc, test_mfcc)
    print(f"🔍 Similarity Score: {distance}")
    return distance

def main():
    ref_mfcc, ref_sr = load_ambulance_sound()
    test_audio, test_sr = record_audio()
    similarity = compare_audio(ref_mfcc, ref_sr, test_audio, test_sr)

    if similarity < 0.8: # Adjusted threshold
        print("🚑🚨 Ambulance Siren Detected! 🚨🚑")
    else:
        print("❌ No siren detected. Try again.")

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

**PIC Microcontroller**

```
#include<pic.h>
__CONFIG(0X3F72);
unsigned char ser1='<',T=0,K=0;
#define R1 RB0
#define O1 RB1
#define G1 RB2
#define R2 RB3
#define O2 RB4
#define G2 RB5
#define R3 RB6
#define O3 RB7
#define G3 RC0
#define R4 RC1
#define O4 RC2
#define G4 RC3
#define L1 RC4
```

```
#define L2 RC5
void delay(unsigned int a)
{
while(a--);
}
void delay3()
{
T0CS=0;
T0SE=0;
PSA=0;
PS0=1;
PS1=1;
PS2=1;
TMR0=60;
while(T0IF==0);
T0IF=0;
TMR0=0;
}
void delay1()
{
unsigned int i=0;
for(i=0;i<=200;i++)
{
delay3();
}
}
void delay2()
{
unsigned int i=0;
for(i=0;i<=100;i++)
{
delay3();
}
}
// GPS INTIALIZE FUNCTION
void gps_init()
{
TXSTA=0X24;
RCSTA=0X90;
SPBRG=25;
GIE=1;
PEIE=1;
RCIE=1;
}
```

```
void interrupt rcx()
{
if(RCIF==1)
{
RCIF=0;
ser1=RCREG;

if(ser1=='1')
{
R1=0;O1=1;G1=0;
R2=0;O2=1;G2=0;
R3=0;O3=1;G3=0;
R4=0;O4=1;G4=0;
delay2();
R1=0;O1=0;G1=1;
R2=1;O2=0;G2=0;
R3=1;O3=0;G3=0;
R4=1;O4=0;G4=0;
delay1();
}
if(ser1=='2')
{
R1=0;O1=1;G1=0;
R2=0;O2=1;G2=0;
R3=0;O3=1;G3=0;
R4=0;O4=1;G4=0;
delay2();
R1=1;O1=0;G1=0;
R2=0;O2=0;G2=1;
R3=1;O3=0;G3=0;
R4=1;O4=0;G4=0;
delay1();
}
if(ser1=='3')
{
R1=0;O1=1;G1=0;
R2=0;O2=1;G2=0;
R3=0;O3=1;G3=0;
R4=0;O4=1;G4=0;
delay2();
R1=1;O1=0;G1=0;
R2=1;O2=0;G2=0;
R3=0;O3=0;G3=1;
```

```
R4=1;O4=0;G4=0;
delay1();
}
if(ser1=='4')
{
R1=0;O1=1;G1=0;
R2=0;O2=1;G2=0;
R3=0;O3=1;G3=0;
R4=0;O4=1;G4=0;
delay2();
R1=1;O1=0;G1=0;
R2=1;O2=0;G2=0;
R3=1;O3=0;G3=0;
R4=0;O4=0;G4=1;
delay1();
}
}
}
void main()
{
TRISB=0X00;
PORTB=0X00;

TRISC=0x80;
PORTC=0x00;
gps_init();
delay(5000);
while(1)
{
R1=0;O1=0;G1=1;
R2=1;O2=0;G2=0;
R3=1;O3=0;G3=0;
R4=1;O4=0;G4=0;
delay1();
R1=0;O1=1;G1=1;
R2=1;O2=0;G2=0;
R3=1;O3=0;G3=0;
R4=1;O4=0;G4=0;
delay2();
R1=1;O1=0;G1=0;
R2=0;O2=0;G2=1;
R3=1;O3=0;G3=0;
R4=1;O4=0;G4=0;
delay1();
```

```
R1=1;O1=0;G1=0;
R2=0;O2=1;G2=1;
R3=1;O3=0;G3=0;
R4=1;O4=0;G4=0;
delay2();
R1=1;O1=0;G1=0;
R2=1;O2=0;G2=0;
R3=0;O3=0;G3=1;
R4=1;O4=0;G4=0;
delay1();
R1=1;O1=0;G1=0;
R2=1;O2=0;G2=0;
R3=0;O3=1;G3=1;
R4=1;O4=0;G4=0;
delay2();
R1=1;O1=0;G1=0;
R2=1;O2=0;G2=0;
R3=1;O3=0;G3=0;
R4=0;O4=0;G4=1;
delay1();
R1=1;O1=0;G1=0;
R2=1;O2=0;G2=0;
R3=1;O3=0;G3=0;
R4=0;O4=1;G4=1;
delay2();

}
}
```

## APPENDIX B

## SCREENSHOTS

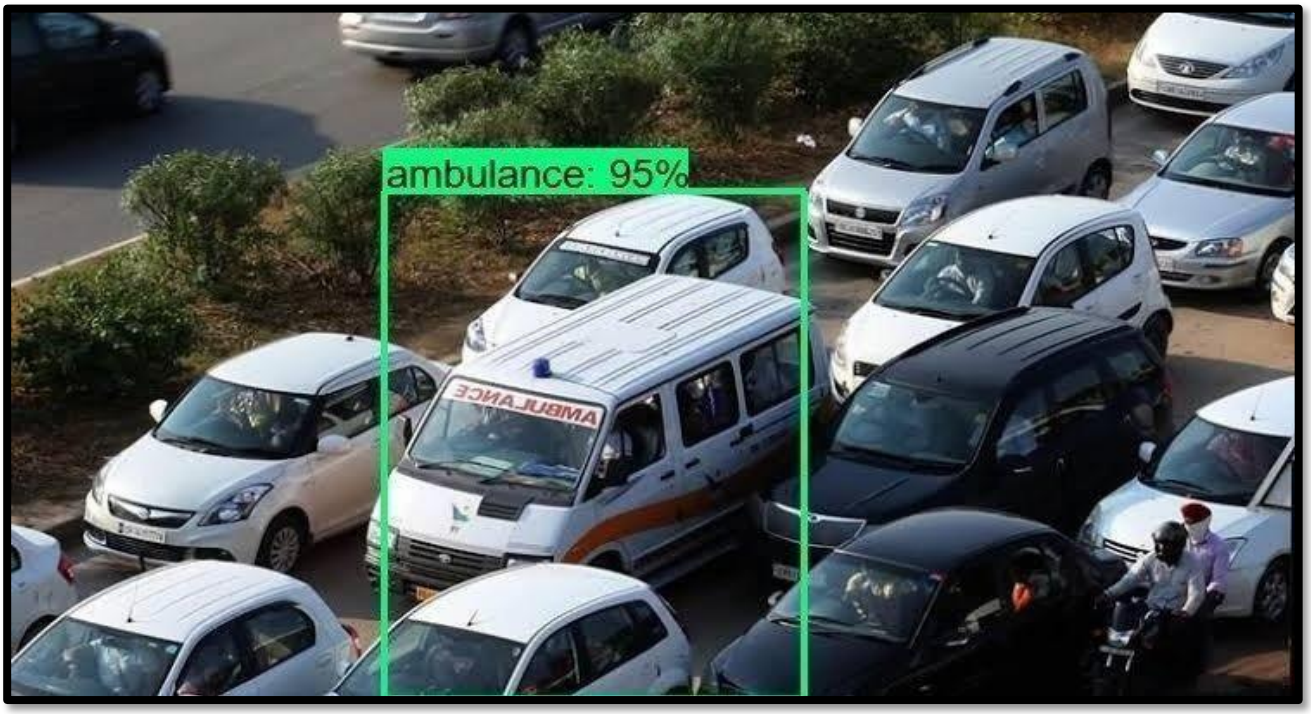


**Fig. B.1 Ambulance Detection from CCTV**

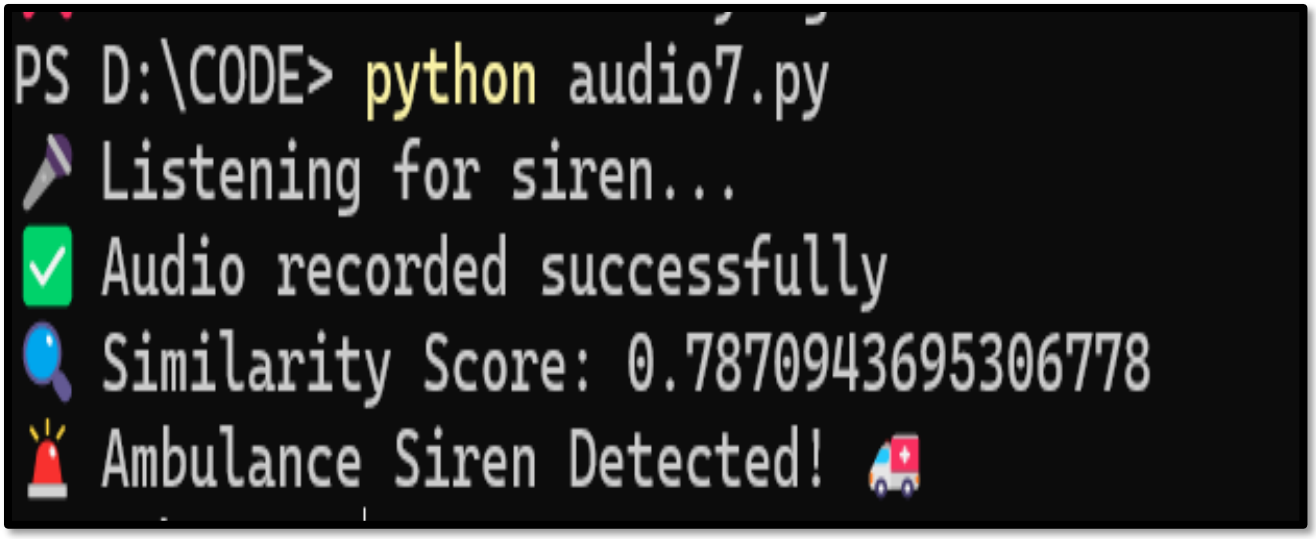


**Fig. B.2 Siren Detection**

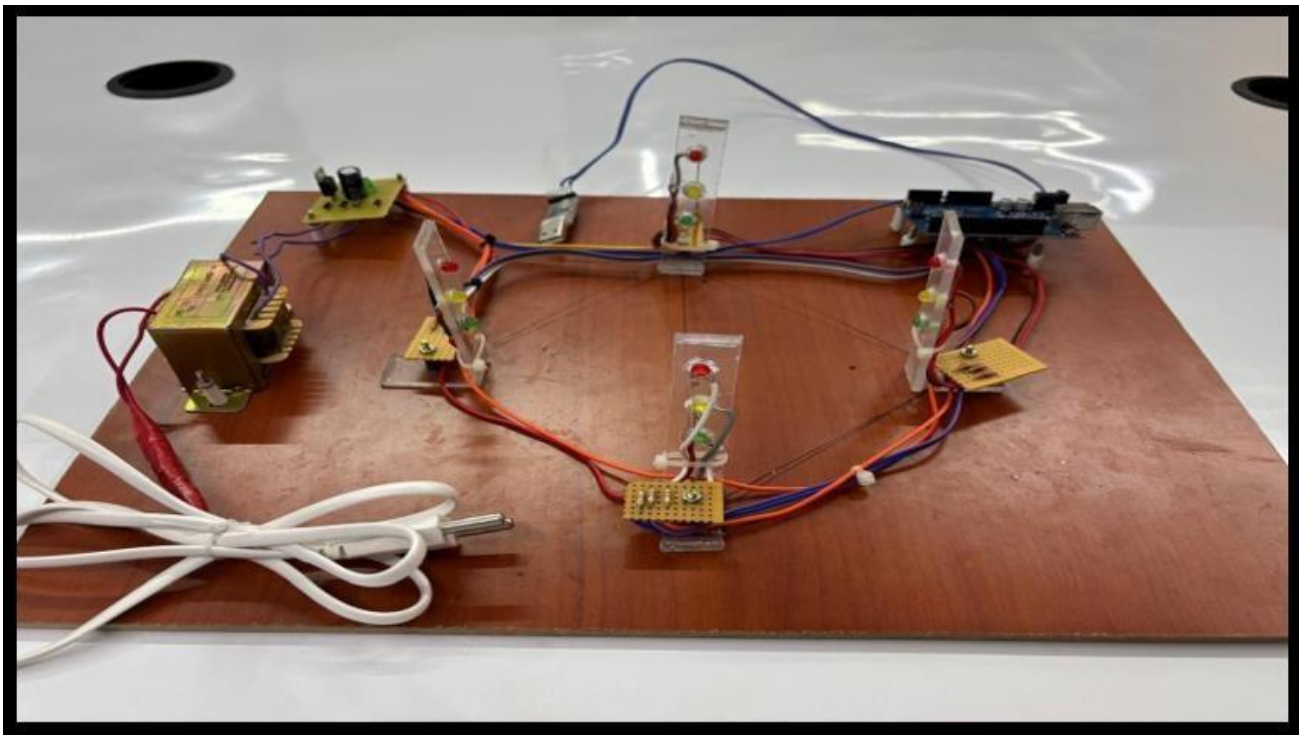

**Fig. B.3 Intersection Signal Hardware**

## REFERENCES


1. Liu, Z., Li, Y., Wu, Y.b. “Multiple UAV formations delivery task planning based on a distributed adaptive algorithm”. J Frankl Inst, 360, 3047−3076, 2023.

2. He, Y., Liu, Y., Yang, L., Qu, X. “Deep adaptive control: Deep reinforcement learning-based adaptive vehicle trajectory control algorithms for different risk levels”. IEEE Trans Intell Veh, 9, 1654–1666, 2023.

3. Guo, N., Zhang, X., Zou, Y. “Real-time predictive control of path following to stabilize autonomous electric vehicles under extreme drive conditions”. Automot Innov, 5, 453−470, 2022.

4. Jaswanth, M., Narayana, N. K. L., Rahul, S., Supriya, M. “Autonomous Car Controller using Behaviour Planning based on Finite State Machine”. In: 2022 6th International Conference on Trends inElectronics and Informatics(ICOEI), 296–302, 2022.

5. Li, H., Liu, W., Yang, C., Wang, W., Qie, T., Xiang, C. “An optimization-based path planning approach for autonomous vehicles using the DynEFWA-artificial potential field”. IEEE Trans Intell Veh, 7, 263−272, 2022.

6. Liang, Y., Li, Y., Yu, Y., Zhang, Z., Zheng, L., Ren, Y. “Path-following control of autonomous vehicles considering coupling effects and multi-source system uncertainties”. Automot Innov, 4, 284−300, 2021.

7. Liu, Y., Lyu, C., Zhang, Y., Liu, Z., Yu, W., Qu, X. “DeepTSP: Deep traffic state prediction model based on large-scale empirical data”. Commun Transport Res, 1, 100012, 2021.

# A.V.C. COLLEGE OF ENGINEERING

ESTD 1996

| Affiliated to Anna University, Chennai | Approved by AICTE, New Delhi |
| Accredited by NAAC with 'A' Grade (3rd Cycle) & NBA (CSE, EEE, ECE and MECH) |
| An ISO 9001:2015 Certified Institution |
Mannampandal, Mayiladuthurai District, Tamilnadu, India, Pincode: 609305.

QUALITY SYSTEM CERTIFICATION DNV ISO 9001

Certificate of Participation

## DEPARTMENT OF INFORMATION TECHNOLOGY & COMPUTER SCIENCE AND ENGINEERING

*Jointly Organizes*

*International Conference on*

## INNOVATIVE RESEARCH IN ENGINEERING AND TECHNOLOGY

(ICIRET-2025)

*This is to certify that Mr./Ms./Dr.* A.D. DHIVYA, Final Year, AIDS,
K. Ramakrishnan College of Technology

*has presented the research paper entitled* AN INTELLIGENT VEHICLE PRIORIZATION SYSTEM USING AI

*in the International Conference on Innovative Research in Engineering and Technology (ICIRET-2025) with* **ISBN No. 978-81-986065-2-5** *organized by the Department of Information Technology & Computer Science and Engineering, A.V.C. College of Engineering, Mayiladuthurai, India on* **28th March, 2025.**

CO-ORDINATOR | CONVENER | PRINCIPAL | DIRECTOR

# A.V.C. COLLEGE OF ENGINEERING

ESTD 1996

| Affiliated to Anna University, Chennai | Approved by AICTE, New Delhi |
| Accredited by NAAC with 'A' Grade (3rd Cycle) & NBA (CSE, EEE, ECE and MECH) |
| An ISO 9001:2015 Certified Institution |
Mannampandal, Mayiladuthurai District, Tamilnadu, India, Pincode: 609305.

QUALITY SYSTEM CERTIFICATION DNV ISO 9001

Certificate of Participation

## DEPARTMENT OF INFORMATION TECHNOLOGY & COMPUTER SCIENCE AND ENGINEERING

*Jointly Organizes*

*International Conference on*

## INNOVATIVE RESEARCH IN ENGINEERING AND TECHNOLOGY

(ICIRET-2025)

*This is to certify that Mr./Ms./Dr.* A. SATHVIKA, Final Year, AIDS,
K. Ramakrishnan College of Technology

*has presented the research paper entitled* AN INTELLIGENT VEHICLE PRIORIZATION SYSTEM USING AI

*in the International Conference on Innovative Research in Engineering and Technology (ICIRET-2025) with* **ISBN No. 978-81-986065-2-5** *organized by the Department of Information Technology & Computer Science and Engineering, A.V.C. College of Engineering, Mayiladuthurai, India on* **28th March, 2025.**

CO-ORDINATOR | CONVENER | PRINCIPAL | DIRECTOR

# A.V.C. COLLEGE OF ENGINEERING

ESTD 1996

| Affiliated to Anna University, Chennai | Approved by AICTE, New Delhi |
| Accredited by NAAC with 'A' Grade (3rd Cycle) & NBA (CSE, EEE, ECE and MECH) |
| An ISO 9001:2015 Certified Institution |
Mannampandal, Mayiladuthurai District, Tamilnadu, India, Pincode: 609305.

QUALITY SYSTEM CERTIFICATION DNV ISO 9001

Certificate of Participation

## DEPARTMENT OF INFORMATION TECHNOLOGY & COMPUTER SCIENCE AND ENGINEERING

*Jointly Organizes*

*International Conference on*

## INNOVATIVE RESEARCH IN ENGINEERING AND TECHNOLOGY

(ICIRET-2025)

*This is to certify that Mr./Ms./Dr.* G. SUJAINITHA, Final Year, AIDS,
K. Ramakrishnan College of Technology

*has presented the research paper entitled* AN INTELLIGENT VEHICLE PRIORIZATION SYSTEM USING AI

*in the International Conference on Innovative Research in Engineering and Technology (ICIRET-2025) with* **ISBN No. 978-81-986065-2-5** *organized by the Department of Information Technology & Computer Science and Engineering, A.V.C. College of Engineering, Mayiladuthurai, India on* **28th March, 2025.**

CO-ORDINATOR | CONVENER | PRINCIPAL | DIRECTOR

# A.V.C. COLLEGE OF ENGINEERING

ESTD 1996

| Affiliated to Anna University, Chennai | Approved by AICTE, New Delhi |
| Accredited by NAAC with 'A' Grade (3rd Cycle) & NBA (CSE, EEE, ECE and MECH) |
| An ISO 9001:2015 Certified Institution |
Mannampandal, Mayiladuthurai District, Tamilnadu, India, Pincode: 609305.

QUALITY SYSTEM CERTIFICATION DNV ISO 9001

Certificate of Participation

## DEPARTMENT OF INFORMATION TECHNOLOGY & COMPUTER SCIENCE AND ENGINEERING

*Jointly Organizes*

*International Conference on*

## INNOVATIVE RESEARCH IN ENGINEERING AND TECHNOLOGY

(ICIRET-2025)

*This is to certify that Mr./Ms./Dr.* K. THEJEAL SRI, Final Year, AIDS,
K. Ramakrishnan College of Technology

*has presented the research paper entitled* AN INTELLIGENT VEHICLE PRIORIZATION SYSTEM USING AI

*in the International Conference on Innovative Research in Engineering and Technology (ICIRET-2025) with* **ISBN No. 978-81-986065-2-5** *organized by the Department of Information Technology & Computer Science and Engineering, A.V.C. College of Engineering, Mayiladuthurai, India on* **28th March, 2025.**

CO-ORDINATOR | CONVENER | PRINCIPAL | DIRECTOR