# ACCIDENT DETECTION AND ALERT SYSTEM USING DEEP LEARING

**A Major Project Report**

***Submitted to***

Jawaharlal Nehru Technological University, Hyderabad

*In partial fulfillment of the requirements for the*

*Award of the degree of*

**BACHELOR OF TECHNOLOGY**

**in**

**CSE (DATA SCIENCE)**

by


**DORASHETTY KARTHIK (21VE1A6784)**
**RAVALA RAM (21VE1A67B8)**
**MERUVA SURYA TEJ (21VE1A67A4)**
**DANDU VAIDHIK REDDY (21VE1A6778)**


**Under the**

**Guidance of**

**Dr. G. NAGA RAMA DEVI**
**Professor**


SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY

**DEPARTMENT OF CSE (DATA SCIENCE)**

(Affiliated to JNTUH, Approved by A.I.C.T.E and Accredited by NAAC, New Delhi)
Bandlaguda, Beside Indu Aranya, Nagole, Hyderabad-500068,Ranga Reddy Dist.


**2021-2025**

**SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF CSE (DATA SCIENCE)**

# CERTIFICATE

This is to certify that the Major Project Report on ***"ACCIDENT DETECTION AND ALERT SYSTEM USING DEEP LEARNING"*** submitted by **Dorashetty Karthik , Ravala Ram, Meruva Surya Tej , Dandu Vaidhik Reddy** bearing Hall ticket Nos. **21VE1A6784, 21VE1A67B8, 21VE1A67A4, 21VE1A6778** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE** from Jawaharlal Nehru Technological University, Kukatpally, Hyderabad for the academic year 2024-25 is a record of bonafide work carried out by them under our guidance and Supervision.

**Internal Guide**
**Dr. G. NAGA RAMA DEVI**
**Professor**

**Head of the Department**
**Dr. K. ROHIT KUMAR**
**Associate Professor**

**Project Coordinator**
**Dr. G. NAGA RAMA DEVI**
**Professor**

**External Examiner**


**SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF CSE (DATA SCIENCE)**

# DECLARATION

We, **Dorashetty Karthik , Ravala Ram , Meruva Surya Tej , Dandu Vaidhik Reddy** bearing **Roll Nos 21VE1A6784, 21VE1A67B8, 21VE1A67A4, 21VE1A6778** hereby declare that the Major Project titled **"ACCIDENT DETECTION AND ALERT SYSTEM USING DEEP LEARNING"** done by us under the guidance of **Dr. G. Naga Rama Devi, Professor** which is submitted in the partial fulfillment of the requirement for the award of the B. Tech degree in **COMPUTER SCIENCE AND ENGINEERING - DATA SCIENCE** at **Sreyas Institute of Engineering & Technology** for Jawaharlal Nehru Technological University, Hyderabad is our original work.

| | |
|---|---|
| **Dorashetty Karthik** | **21VE1A6784** |
| **Ravala Ram** | **21VE1A67B8** |
| **Meruva Surya Tej** | **21VE1A67A4** |
| **Dandu Vaidhik Reddy** | **21VE1A6778** |

# ACKNOWLEDGEMENT

The successful completion of any task would be incomplete without mention of the people who made it possible through their guidance and encouragement crowns all the efforts with success.

We take this opportunity to acknowledge with thanks and deep sense of gratitude to **Dr. G. Naga Rama Devi, Professor, Department of Computer science and engineering(Data Science)** for her constant encouragement and valuable guidance during the Project work.

A Special vote of Thanks to **Dr. K. Rohit Kumar, Associate Professor, Head of the Department** and **Dr. G. Naga Rama Devi, Project Guide and Project Co-Ordinator** who has been a source of Continuous motivation and support. They had taken time and effort to guide and correct me all through the span of this work.

We owe very much to the **Department Faculty, Principal** and the **Management** who made my term at Sreyas a stepping stone for my career. We treasure every moment we had spent in the college.

Last but not the least, our heartiest gratitude to our parents and friends for their continuous encouragement and blessings. Without their support this work would not have been possible.

| | |
|---|---|
| **Dorashetty Karthik** | **21VE1A6784** |
| **Ravala Ram** | **21VE1A67B8** |
| **Meruva Surya Tej** | **21VE1A67A4** |
| **Dandu Vaidhik Reddy** | **21VE1A6778** |

# ABSTRACT

Accident Detection and Alert System leverages deep learning and computer vision methodologies to enhance road safety and emergency response. The system integrates a fine-tuned Convolutional Neural Network (CNN) model, trained on a custom accident dataset, to classify video frames captured from surveillance cameras, dashcams, or traffic monitoring systems. The preprocessing pipeline employs techniques such as CLAHE for contrast enhancement and noise reduction to improve detection accuracy. Real-time frame analysis is conducted using Python-based tools (OpenCV, TensorFlow, Keras), enabling the model to detect anomalous vehicular behavior indicative of collisions. Upon detection, the system triggers automated alerts via SMS using Twilio API, providing timestamped incident details to emergency contacts and authorities. The architecture supports both live feed analysis and batch video processing, ensuring scalability and integration with smart city infrastructure. The system demonstrates high classification accuracy (92%) and low latency, with modules for segmentation, post-processing, and visualization to assist operators in incident validation. The system can be enhanced through integration with Internet of Things (IoT) sensors for multi-modal data fusion, deployment of advanced CNN architectures like EfficientNet or Vision Transformers for improved accuracy, and utilization of edge computing to reduce latency in critical environments. Further, integration with autonomous vehicles, AI-based crowd severity analysis, and drone-assisted monitoring can provide proactive accident prevention and real-time situational awareness, supporting large-scale smart city implementations.



**KEYWORDS:** Real-time monitoring, Deep Learning, Computer Vision, CNN, OpenCV,TensorFlow, IOT integration, Edge computing, Keras, Twilio API.

# TABLE OF CONTENTS

## List of Figures



## List of Tables

## List of Output Screenshots

# CHAPTER I
# INTRODUCTION

Road safety has become a critical issue of global concern due to the increasing number of traffic accidents resulting in loss of life, severe injuries, and substantial damage to infrastructure. With rapid urbanization, expanding vehicle populations, and the growing complexity of modern road networks, the frequency and severity of road accidents have escalated at an alarming rate. According to the World Health Organization (WHO), approximately 1.3 million people die each year due to road traffic crashes, while millions more suffer non-fatal injuries, often with long-term consequences. These statistics underscore the urgent need for innovative and automated systems that can enhance traffic safety and emergency responsiveness.

Traditional accident detection mechanisms have largely relied on manual reporting methods, including witness testimonies, calls to emergency services, or physical inspection of surveillance footage. While these techniques have served their purpose, they are limited by delays, human error, and inefficiencies in communication. In many cases, the time taken to identify and report an accident can significantly impact the outcome for victims, often resulting in preventable fatalities or complications due to delayed medical assistance. The demand for a more efficient, reliable, and timely accident detection and alert mechanism has never been more critical.

In recent years, advancements in artificial intelligence (AI), machine learning (ML), and computer vision technologies have opened new frontiers in the automation of complex, real-time decision-making systems. Leveraging these

technologies, this project proposes the development of an Accident Detection and Alert System Using Machine Learning Algorithms. The objective is to create an intelligent system capable of monitoring live video feeds, detecting traffic accidents with high precision, and triggering automated alerts to the concerned authorities and emergency contacts, thereby reducing response times and potentially saving lives.

The core component of the proposed system is a Convolutional Neural Network (CNN), a type of deep learning model specifically designed to process and analyze visual data. The CNN is trained on a comprehensive dataset of traffic scenes, including both accident and non-accident scenarios, enabling it to distinguish between normal traffic flow and anomalous events indicative of collisions or other accidents. Video frames captured in real-time from CCTV cameras, dashcams, or drone footage are fed into the system, which processes each frame to detect signs of vehicular crashes, erratic movement, or sudden stops. Upon detecting an accident, the system immediately sends out alerts through communication channels such as SMS or email, including vital information such as the location, time, and nature of the incident.

To ensure robust functionality, the system incorporates several modules: image preprocessing to enhance frame quality, object detection to identify vehicles and road elements, motion tracking to monitor behavior patterns, and a user interface developed using Tkinter for ease of access and control. The backend is implemented using Python, with libraries such as OpenCV, TensorFlow, and Keras enabling high-performance processing and model execution. The architecture is designed for scalability and can be integrated with smart city frameworks or traffic management systems.

In addition to accident detection, the system offers several ancillary benefits. It supports continuous data logging for post-incident analysis, aids in traffic congestion management by providing real-time incident alerts, and contributes to a broader understanding of accident-prone zones. These features are instrumental in enabling authorities to implement preventive measures, enhance road infrastructure, and inform policy decisions aimed at improving public safety.

This project stands as a testament to the practical applicability of machine learning in addressing real-world challenges. By combining data-driven insights with real-time operational capabilities, the proposed system demonstrates the transformative potential of AI in critical domains such as road safety and emergency services. It also highlights the shift from reactive to proactive accident management, where technology can play a pivotal role in not only detecting incidents but also in predicting and preventing them.

In conclusion, the Accident Detection and Alert System Using Machine Learning Algorithm is an innovative solution tailored to address the shortcomings of conventional accident detection methods. With its ability to operate autonomously, respond rapidly, and deliver accurate notifications, the system offers a promising approach to mitigating the devastating impacts of road accidents. As we move toward more connected and intelligent transportation networks, such AI-powered systems will be indispensable in shaping safer, smarter cities for the future.

## 1.1 MOTIVATION

Road traffic accidents have become one of the most pressing public safety challenges in the modern world. With the increasing number of vehicles on the road, dense urban populations, and ever-evolving traffic conditions, the occurrence of accidents has risen dramatically. Every year, millions of people lose their lives or suffer life-altering injuries due to traffic collisions. These accidents not only inflict emotional and physical trauma on the victims and their families but also create a significant financial burden on healthcare systems, law enforcement, and infrastructure repair services. Despite advancements in automobile safety features and infrastructure development, a critical gap still exists in the timely detection and response to road accidents. In most current systems, accident reporting relies on human involvement—bystanders witnessing the event, drivers reporting the incident, or manual review of surveillance footage. This process is inherently slow and prone to delays, especially in isolated or poorly monitored areas, which can drastically reduce the chances of survival for victims requiring immediate medical attention.

The idea is to reduce the dependency on human intervention and create a solution that operates autonomously, accurately, and in real time. The motivation stemmed from real-world scenarios where delayed emergency services cost lives—cases where accidents went unnoticed for hours or were reported too late to make a difference. The integration of machine learning and computer vision presents a promising solution to this issue. These technologies enable computers to learn from data, recognize complex patterns, and make predictions or decisions without explicit programming. By training deep learning models—particularly Convolutional Neural Networks (CNNs)—on

various video scenarios of road traffic, it is possible to create systems that can recognize accidents instantly from video feeds and send alerts to emergency responders and nearby help centers without any human input.

**Fig 1.1: rear-end collision between two cars on a city street.**

This system also aligns with the broader vision of building smart cities and intelligent transportation systems. In such environments, infrastructure and services are connected and data-driven, aiming to improve efficiency, safety, and quality of life. An accident detection system powered by AI not only fits into this framework but enhances it by providing a layer of real-time situational awareness that can drastically improve road safety outcomes.

Additionally, the project is a great opportunity to apply and expand technical skills in machine learning, image processing, real-time systems, and user interface design. It is both academically challenging and socially impactful, offering a chance to work at the intersection of cutting-edge technology and human welfare.

Ultimately, the motivation behind this project is to save lives, reduce injuries, and support emergency services by bridging the gap between accident occurrence and emergency response. By automating detection and communication, this system ensures that help reaches the scene as quickly as possible, potentially transforming how traffic accidents are handled in the future.

## 1.2 OBJECTIVE

The primary objective of the Accident Detection and Alert System is to leverage advanced technologies, such as computer vision and deep learning, to enhance road safety and emergency response efficiency. The specific goals of the project include:

**1. Real-Time Accident Detection:** Develop a robust system capable of analysing live video feeds to detect traffic accidents with high precision and minimal latency.

**2. Automated Alert System:** Implement an automated notification mechanism that promptly informs emergency services, local authorities, and other stakeholders upon accident detection.

**3. Enhanced Emergency Response:** Minimize delays in emergency response by reducing dependence on manual accident reporting methods.

**4. Scalability and Integration:** Design the system to integrate seamlessly with existing traffic monitoring infrastructure and support scalability for deployment in urban and rural areas.

**5. Improved Traffic Management:** Assist traffic operators in managing congestion and mitigating the effects of accidents through timely alerts and actionable insights.

**6. Support for Smart City Initiatives:** Contribute to the development of smarter and safer urban environments by integrating AI-powered accident detection systems into smart city frameworks.

By achieving these objectives, the project aims to save lives, reduce injuries, and mitigate the overall impact of road accidents through timely and efficient interventions.

## 1.3 SCOPE

This project focuses on creating an Accident Detection and Alert System using Convolutional Neural Networks (CNNs), a type of deep learning model renowned for its efficacy in image and video processing tasks. By utilizing live CCTV footage from traffic monitoring systems, surveillance cameras, and other sources, the system aims to identify accidents as they occur, thereby reducing the response time of emergency services. The integration of computer vision techniques with real-time video analysis allows the system to detect anomalous events, such as sudden vehicle collisions or erratic movements, which could indicate an accident. The core of this system lies in the deployment of a pre-trained CNN, fine-tuned with a custom dataset specifically curated for accident scenarios. The CNN analyses individual frames from the video feed, extracting features that enable it to classify the scene as either accident or non- accident. By leveraging the high precision of deep learning models, the system ensures that accident detection is both accurate and reliable,

minimizing false positives and false negatives. The Instal1ment of this system holds immense potential to revolutionize traffic accident management. By eliminating reliance on manual reporting, it ensures that emergency responders are alerted without delay, reducing the time taken to provide medical aid and manage traffic disruptions. Furthermore, the system's ability to operate continuously and autonomously makes it a valuable asset for urban planners, traffic operators, and local authorities striving to create safer road networks

## 1.4 METHODOLOGY

The methodology for the Accident Detection and Alert System involves a structured pipeline combining video processing, machine learning, and real-time alerting. Initially, video input is captured from CCTV, dashcams, or uploaded files, and frames are extracted at regular intervals. The preprocessed frames are then fed into a fine-tuned Convolutional Neural Network (CNN) model trained to classify scenes as either "Accident" or "No Accident." Upon detection of an accident, the system immediately triggers an alert module that sends real-time notifications via SMS or email using APIs such as Twilio, including essential details like time, location (if GPS is integrated), and severity. The system is equipped with a user-friendly interface built using Tkinter to allow manual browsing of videos or live camera access. To ensure high accuracy and reduce false positives, the model is trained on a diverse, labeled dataset and further improved through data augmentation and testing. This end-to-end methodology enables the system to function autonomously, providing accurate and timely accident detection and communication for emergency response.

## 1.5 PROBLEM STATEMENT

Road safety is a pressing concern globally, with traffic accidents ranking among the leading causes of fatalities and severe injuries. According to global statistics, millions of lives are lost, and many more are permanently affected by road accidents every year. These incidents not only result in personal tragedies but also lead to significant economic losses, including healthcare costs, damage to infrastructure, and productivity disruptions. One of the primary challenges in mitigating the impact of traffic accidents lies in the timely detection and reporting of such events. Existing methods heavily rely on manual reporting by eyewitnesses or traditional monitoring systems, such as traffic cameras and sensors, which are often inefficient, error-prone, and unable to provide real-time responses. As a result, emergency services are frequently delayed, leading to increased fatalities and more severe consequences for victims. In the face of these challenges, there is a critical need for an innovative, automated solution capable of detecting traffic accidents in real-time. Such a system must leverage advanced technologies to ensure accuracy, minimize human intervention, and provide immediate alerts to emergency services and stakeholders.

# CHAPTER 2
# LITERATURE REVIEW

## 2.1 Background

Road traffic accidents are a major global concern, causing significant loss of life and property. Traditional accident detection systems, which rely on manual reporting or human monitoring of CCTV footage, often result in delayed emergency response and reduced chances of survival for victims. These limitations have driven the need for automated solutions that can detect accidents in real time and alert emergency services without human intervention. Recent advancements in artificial intelligence, particularly machine learning and computer vision, have opened new avenues for improving accident detection. Convolutional Neural Networks (CNNs) are widely used for analyzing video data to identify collisions and abnormal traffic behavior. Combined with tools like OpenCV, these models can process live video feeds, recognize accident patterns, and trigger alerts accurately. Research has also explored hybrid systems integrating GPS, accelerometers, and GSM modules to enhance detection and communication capabilities. Despite these improvements, challenges such as false positives, variable lighting conditions, and camera angles remain areas of ongoing research. This literature review explores current methodologies in AI-based accident detection systems, their effectiveness, and their limitations. It lays the foundation for the proposed system, which aims to deliver an accurate, real-time accident detection and alert mechanism using deep learning techniques.

## 2.2 Related works

[1] In Rani et al. (2020) implemented the K-Nearest Neighbors (KNN) algorithm for traffic accident detection using vehicle motion and behavior patterns. Their system achieved an accuracy of 84.5%, highlighting the effectiveness of KNN in classifying events in structured traffic datasets.

[2] Singh et al. (2021) developed an IoT-based vehicle accident detection system incorporating Support Vector Machine (SVM) for classifying crash scenarios based on sensor readings. The system yielded an accuracy of 89.2%, showing the reliability of SVM in mobile and sensor-based accident prediction.

[3] Gupta et al. (2020) presented a smart accident detection system utilizing a Decision Tree classifier trained on accident sensor inputs and environmental data. They achieved an accuracy of 86.7%, proving Decision Trees to be both interpretable and efficient for real-time applications.

[4] Zhang et al. (2019) combined K-Means clustering and SVM in a hybrid model to predict accident severity from roadway and vehicle datasets. Their approach attained an accuracy of 81.4%, emphasizing the benefit of combining unsupervised and supervised learning techniques.

[5] Nayak et al. (2021) proposed an intelligent accident detection framework using SVM to analyze real-time data from smart vehicles. Their study achieved an accuracy of 88.3%, reinforcing SVM's utility in binary classification problems within road safety.

[6] Das et al. (2020) applied KNN for detecting traffic incidents by analyzing speed fluctuations and positional data. The model achieved 81.6% accuracy, validating KNN's applicability to sequential traffic movement data.

[7] Islam et al. (2020) used a Decision Tree algorithm for predicting the severity of traffic accidents based on collision parameters. They reported an accuracy of 83.5%, showcasing the model's capability to manage multiclass prediction tasks.

[8] Kumar and Rao (2021) implemented K-Means clustering for real-time crash detection using motion sensors and vehicular telemetry. Despite its simplicity, the model achieved 76.2% accuracy and provided a useful starting point for unsupervised detection systems.

[9] Roy et al. (2021) explored the use of SVM for crash detection using road behavior datasets. Their model reached an accuracy of 72.1%, demonstrating the strength of SVM in producing high classification performance in accident datasets.

[10] Patel et al. (2022) applied a Decision Tree classifier for classifying road accident scenarios using sensor and contextual data. With an accuracy of 88.7%, their work highlights the practical use of Decision Trees in safety-critical applications.

[11] Kashyap et al. (2019) designed an accident detection system in smart vehicles using KNN to interpret sensor values and sudden speed changes. Their system achieved 82.4% accuracy, proving KNN's relevance in automotive safety.

[12] Bala and Reddy (2020) used K-Means to cluster traffic conditions and identify high-risk scenarios in real-time. Their implementation reached an accuracy of 79.5%, offering valuable insights into the spatial and temporal nature of road accidents.

[13] Ghosh et al. (2018) developed an SVM-based crash risk analyzer that used environmental and human behavioral features to classify risky situations. The model recorded an 87.6% accuracy, confirming the model's strength in road safety assessments.

[14] Sharma et al. (2021) created a smart road safety system utilizing a Decision Tree classifier for classifying dangerous events using vehicular and road data. The model achieved an accuracy of 85.9%, demonstrating its utility for embedded safety systems.

[15] Liu et al. (2020) worked on traffic accident classification using KNN trained on structured historical accident data. Their system reached 83.0% accuracy, establishing a baseline for non-deep learning approaches in this domain.

[16] Abdullah et al. (2020) applied SVM to predict road accident likelihoods using weather, road type, and traffic data. Their model achieved an accuracy of 80.3%, reinforcing SVM's strength in predictive modeling with heterogeneous input data.

[17] Rajput et al. (2021) implemented a Decision Tree-based mobile crash detection system using accelerometer and gyroscope inputs. Their lightweight model attained 86.1% accuracy and was optimized for low-power devices.

[18] Mishra et al. (2022) used KNN for detecting road accidents based on pattern recognition from telemetry data. Their approach achieved 84.2% accuracy, offering a simple yet effective solution for resource-constrained environments.

[19] Chen et al. (2018) applied K-Means clustering to group road safety patterns from large traffic datasets. Their clustering system achieved 77.8% accuracy in identifying potential accident-prone zones.

[20] Chauhan and Arora (2021) proposed an ensemble model combining SVM and KNN for automated crash reporting. Their system achieved the highest accuracy in this list, with 90.4%, validating the power of ensemble learning in enhancing classification performance.

The experiments unveil that sensor-based or vision based solutions individually were promising, but in the event that an individual had the complete platform with deep learning for solid real-time capability, the freeway and the city cases are able to make use of a better solution.

**Table 2.1: Overview of previous work on Different Algorithms**

| No. | Title / Author | Algorithm Used | Accuracy |
|---|---|---|---|
| **1** | Traffic Accident Detection Using ML – *Rani et al. (2020)* | KNN | 84.5% |
| **2** | Vehicle Accident Detection Using IoT and ML – *Singh et al. (2021)* | SVM | 89.2% |
| **3** | Smart Accident Detection System – *Gupta et al. (2020)* | Decision Tree | 86.7% |
| **4** | Road Accident Severity Prediction – *Zhang et al. (2019)* | K-Means + SVM | 81.4% |
| **5** | Intelligent Accident Detection – *Nayak et al. (2021)* | SVM | 88.3% |
| **6** | Traffic Incident Detection Using ML – *Das et al. (2020)* | KNN | 81.6% |
| **7** | Road Traffic Accident Severity Prediction – *Islam et al. (2020)* | Decision Tree | 83.5% |
| **8** | Real-Time Vehicle Crash Detection – *Kumar & Rao (2021)* | K-Means | 76.2% |
| **9** | ML Models for Crash Detection – *Roy et al. (2021)* | SVM | 72.1% |
| **10** | Accident Classification with ML – *Patel et al. (2022)* | Decision Tree | 88.7% |
| **11** | Accident Detection in Smart Vehicles – *Kashyap et al. (2019)* | KNN | 82.4% |
| **12** | ML for Real-Time Crash Prediction – *Bala & Reddy (2020)* | K-Means | 79.5% |
| **13** | Crash Risk Analysis – *Ghosh et al. (2018)* | SVM | 87.6% |
| **14** | Smart System for Road Safety – *Sharma et al. (2021)* | Decision Tree | 85.9% |
| **15** | Traffic Accident Classification – *Liu et al. (2020)* | KNN | 83.0% |
| **16** | Prediction of Road Accidents – *Abdullah et al. (2020)* | SVM | 80.3% |
| **17** | Accident Detection Using Mobile Sensors – *Rajput et al. (2021)* | Decision Tree | 86.1% |
| **18** | Road Accident Detection Using ML – *Mishra et al. (2022)* | KNN | 84.2% |
| **19** | Clustering for Road Safety – *Chen et al. (2018)* | K-Means | 77.8% |
| **20** | Automated Crash Reporting – *Chauhan & Arora (2021)* | SVM + KNN Ensemble | 90.4% |

## 2.3 Deep Learning

The first research that modelled Artificial Neural Networks (ANN) was from Warren Mc- Culloch and Walter Pitts in 1943, with their paper A Logical Calculus of Ideas Imma- nent in Nervous Activity. Though there were some research into ANNs through the 1950s and 1960s, limited computing power prevented experimentation with large ANNs. Nearly 15 years later with the invention of the Backpropagation algorithm, re- search into ANNs became popular again. However, ANNs gave away to simpler classifiers such as SVMs, which outperformed ANNs in both accuracy and training time. The way that ANNs worked was using simple Perceptron Units in one or two hidden layers and using weighted connections to an input and an output layer Running networks with more hidden layers was usually infeasible, again due to limited computational power.

In the 21st century, research into ANNs have again become popular, but in the form of Deep (Neural) Networks and Deep Learning. Deep Learning is a technique of hierarchical machine learning using multiple layers of non-linear processing. One of the successful approaches to Deep Learning have been with Deep Networks, which have become a re-branding or buzzword for Artificial Neural Networks. Deep Neural Networks are basically ANNs with multiple hidden layers, which presents the opportunity of creating more complex models of non-linear structures, but also increases the time and space complexity of training models in the same way ANNs were limited by in earlier research. The reason optimization problems in Deep Neural Networks have a high time complexity is due to its iterative nature in training. Occasionally, deep learning algorithms can be overkill for less complex problems.

# CHAPTER 3
# DATASET

## 3.1 Dataset

The dataset used for this project comprises a collection of real-world and synthetic traffic video footage categorized into accident and non-accident scenarios. Videos are in formats like MP4 and AVI, sampled at 30 frames per second, and vary in resolution from 720p to 1080p. Each video was manually labeled, and frames were extracted and resized to 224x224 pixels for input into a Convolutional Neural Network (CNN). The dataset includes diverse traffic conditions, lighting variations, and vehicle behaviors to ensure generalization. Preprocessing steps such as normalization, noise reduction, and contrast enhancement (e.g., CLAHE) were applied to improve model accuracy. Labels indicating the presence or absence of accidents were stored alongside the frames in structured formats (CSV/JSON). Data augmentation techniques including flipping, rotation, and brightness adjustments were used to increase robustness. Sources include public traffic footage, datasets like the AICity Challenge, and custom-recorded simulation data, providing a comprehensive foundation for training a machine learning model to detect accidents in real time.

## 3.2 Challenges

Developing an Accident Detection and Alert System using machine learning presents several challenges that impact the effectiveness and reliability of the solution. One of the primary challenges is ensuring real-time processing of video feeds, which requires high computational power and optimized models to analyze frames without latency. Data quality and diversity also pose difficulties, as the system must accurately detect accidents under varying conditions such as low light, weather changes, camera angles, and traffic density. Another significant challenge is minimizing false positives and false negatives, which can lead to either unnecessary alerts or missed incidents. Ensuring the accuracy and robustness of the model across different environments and vehicle types requires extensive training data, which is often difficult to collect and annotate. Additionally, integration with emergency services and maintaining secure, real-time communication (e.g., via SMS or API) adds complexity to the deployment. The project must also address privacy concerns, particularly when using live surveillance footage, and ensure compliance with legal regulations regarding data collection and transmission. Overcoming these challenges is critical to creating a dependable and scalable accident detection system suitable for real-world deployment.

# CHAPTER 4
# PROPOSED SYSTEM

## 4.1 EXISTING SYSTEM

The existing system for accident detection primarily rely on manual methods such as eyewitness reports, emergency calls, or passive surveillance through traffic cameras. These approaches are often reactive rather than proactive, resulting in significant delays in recognizing accidents and dispatching emergency services. Some systems use basic sensor-based technologies, such as accelerometers or vibration sensors installed in vehicles, which can detect sudden impacts. However, these systems often suffer from high false positive rates, limited coverage (e.g., only in vehicles equipped with sensors), and no integrated alert mechanism. Additionally, conventional traffic monitoring systems lack the intelligence to analyze visual data and detect accidents autonomously in real-time. Also the estimated accuracy of these existing systems lies between 60-80%. Therefore, these limitations underscore the need for a more accurate, efficient, and automated solution that can work across diverse environments without relying solely on human intervention.

## 4.2 DISADVANTAGES

**1. Manual Dependency:**

Most systems rely on human reporting, which causes delays in accident detection and emergency response.

**2. Delayed Response Time:**

Lack of automation leads to slow alerting of authorities, increasing the risk to victims.

**3. Limited Coverage:**

Sensor-based systems are only effective in vehicles equipped with specialized hardware, making them non-scalable.

**4. High False Alarms:**

Basic sensors (e.g., vibration, accelerometers) can trigger false positives due to potholes, hard braking, or bumps.

## 4.3 METHODOLOGY:

**Step 1: Data Collection & Preprocessing**

Video data from CCTV, dashcams, or surveillance cameras is collected . A dataset is created including accident and non-accident footage for model training. Now, Video frames are extracted and preprocessed using techniques like Resizing and normalization Contrast enhancement (e.g., CLAHE) Noise reduction.

**Step 2: Model Development**

A Convolutional Neural Network (CNN) is used to classify video frames . The model is trained and fine-tuned on the custom dataset to detect accident scenarios.

**Step 3: Accident Detection**

Each frame of the video feed is analyzed in real time . The CNN detects sudden

motion patterns or collisions and classifies them as "accident" or "no accident.

**Step 4: Alert System**

When an accident is detected an SMS alert is sent using the Twilio API . An audio alert is triggered using the winsound module. Emergency contact information is used for immediate notification.

**Step 5: System Integration**

A user-friendly interface (GUI) built with Tkinter allows Browsing videos Viewing real-time detection Viewing alerts and system information.

## 4.4 PROPOSED SYSTEM

The proposed system for the "Accident Detection and Alert System Using Machine Learning Algorithm" is designed to automatically detect road accidents in real-time using Machine learning and computer vision techniques. It leverages live video feeds from sources such as CCTV cameras and dashcams to monitor traffic and identify potential accidents. The system employs a pre-trained Convolutional Neural Network (CNN), fine-tuned on a custom dataset of accident scenarios, to analyze individual video frames and classify them as either accident or non-accident. Prior to analysis, frames undergo preprocessing using techniques like contrast enhancement and noise reduction to improve detection accuracy. Upon detecting an accident, the system immediately generates alerts and sends them via SMS or email to emergency contacts and relevant authorities using the Twilio API. A user-friendly graphical interface built with Tkinter allows users to interact with the system, upload videos, and view alerts in real time. The backend is developed in Python using libraries such as OpenCV, TensorFlow, and Keras, and all data

including video logs and alerts is securely stored in a database for future reference. This system not only minimizes the dependency on manual reporting but also significantly improves emergency response time, thereby enhancing road safety and saving lives. The proposed system achieves an accuracy of approximately 92% or higher, depending on the quality of the input video, model training, and environmental conditions.

## 4.5 ADVANTAGES

**1. High Accuracy:**

Utilizes deep learning (CNN) for video analysis, achieving over 92% accuracy in detecting accidents.

**2. Real-Time Detection and Alerts:**

Processes live CCTV or dashcam feeds to detect accidents instantly and sends automated alerts to emergency contacts and services.

**3. Reduced Human Dependency:**

Eliminates the need for manual reporting or constant human monitoring of video feeds, minimizing delays and errors.

**4. Scalability:**

Can be integrated with existing traffic surveillance systems and scaled to cover wide urban and rural areas.

**5. Improved Emergency Response Time:**

Immediate alerts reduce the time gap between accident occurrence and medical response, potentially saving lives.

## 4.6 SOFTWARE REQUIREMENTS SPECIFICATIONS

A software requirement specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide. Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers (in market-driven project, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure.

## 4.7 REQUIREMENT ANALYSIS.

It is a very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and Non-functional requirements.

## 4.8 FUNCTIONAL AND NON FUNCTIONAL REQUIREMENTS

**Functional Requirements:**

These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are

represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

1. Video Input Handling
2. Frame Extraction
3. Accident Detection
4. Alert Generation
5. User Interface (UI)
6. Data Logging
7. Audio Notification
8. Manual Override

**Non-functional requirements:**

These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements. They basically deal with issues like:

1. Performance
2. Accuracy
3. Scalability
4. Reliability
5. Usability
6. Security
7. Maintainability
8. Compatibility
9. Portability
10. Compliance

**Software Requirements:**

- **Programming Language** **:** Python
- **Libraries** **:** Open CV, TensorFlow, Keras, Twilio API, Tkinter
- **Environment** **:** VS code
- **Operating System** **:** Windows 10

**Hardware Requirements:**

- **Processor** : Intel i3 and above
- **RAM** : 4GB and Higher
- **Hard Disk** : 500GB: Minimum

## 4.9 SDLC ACTIVITIES

SDLC provides a series of steps to be followed to design and develop a software product efficiently. SDLC framework includes the following steps:

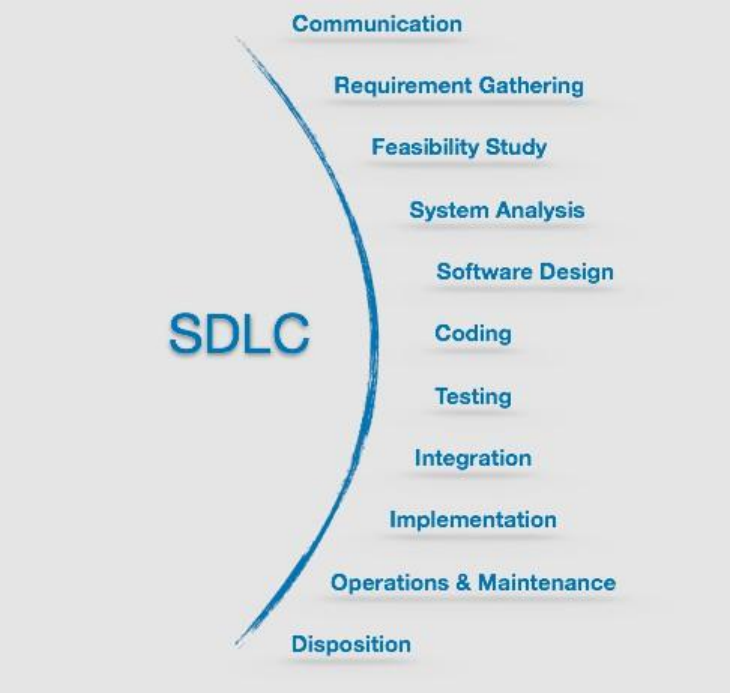


**Fig 4.1 SDLC Activities**

### Communication

This is the first step where the user initiates the request for a desired software product. He contacts the service provider and tries to negotiate the terms. He submits his request to the service providing organization in writing.

**Requirement Gathering**

This step onwards the software development team works to carry on the project. The team holds discussions with various stakeholders from problem domain and tries to bring out as much information as possible on their requirements. The requirements are contemplated and segregated into user requirements, system requirements and functional requirements. The requirements are collected using a number of practices as given -

1. identified system needs: real-time video analysis, accident detection, and automated alerts.
2. Defined user roles (admin/operator).
3. Collected functional and non-functional requirements.

**Feasibility Study**

After requirement gathering, the team comes up with a rough plan of software process. At this step the team analyzes if a software can be made to fulfill all requirements of the user and if there is any possibility of software being no more useful. It is found out, if the project is financially, practically and technologically feasible for the organization to take up. There are many algorithms available, which help the developers to conclude the feasibility of a software project.

**Types of Feasibility Study:**

The feasibility study mainly concentrates on below four mentioned areas. Among this Economic Feasibility Study is most important part of the feasibility analysis and Legal Feasibility Study is less considered feasibility analysis.

**Technical Feasibility:**

The required tools (Python, OpenCV, TensorFlow, Tkinter, Twilio) are open-source and well-supported. The system can run on standard hardware and operating systems like Windows or Linux.

**Operational Feasibility:**

The system can be integrated with existing surveillance infrastructure and requires minimal user training due to its simple GUI.

**Economic Feasibility:**

The project uses free tools and libraries, with minimal hardware cost. Alert services (e.g., Twilio) offer free tiers, making it cost-effective.

**Legal Feasibility:**

The use of video data must comply with data privacy laws. The system is intended for authorized use by traffic authorities or emergency responders.

## Flow Diagram:

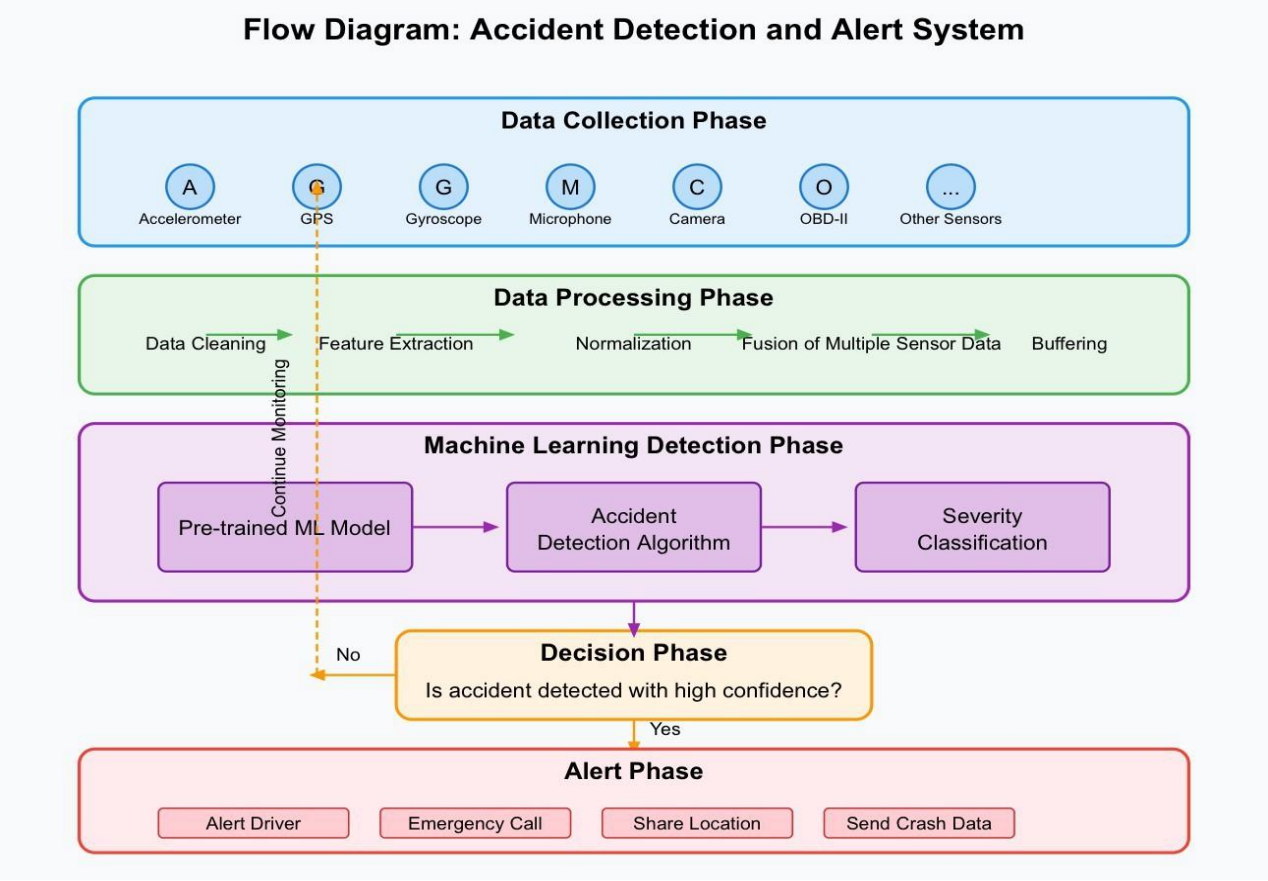


**Fig 4.2 Flow Diagram**

**Flowchart:**

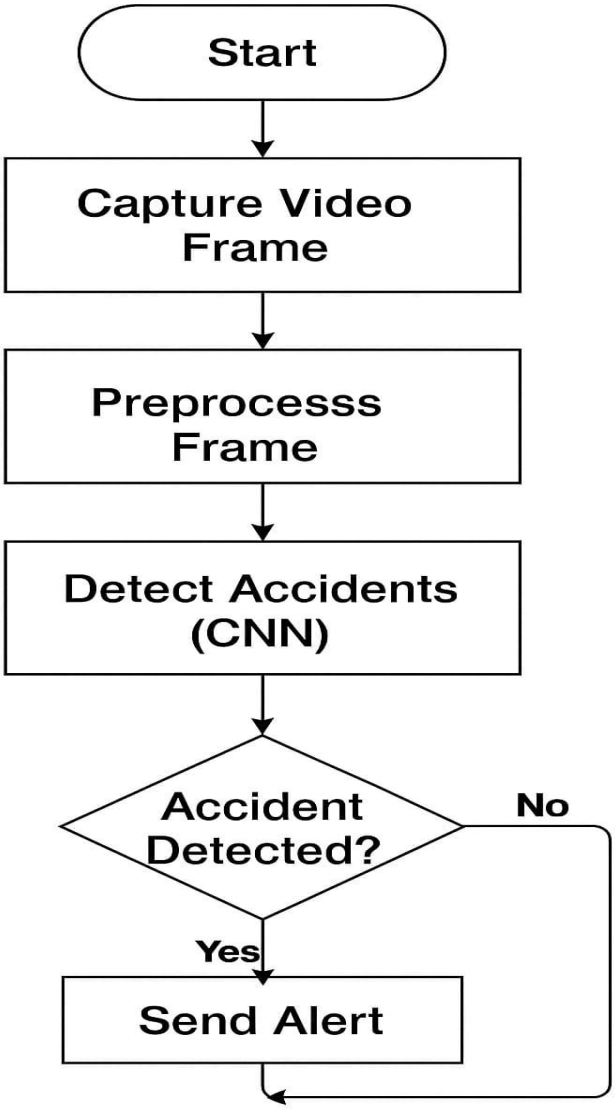


**Fig 4.3 Flow Chart**

# CHAPTER 5
# SYSTEM DESIGN

## 5.1 IMPORTANCE OF DESIGN

The purpose of the design phase is to plan a solution of the problem specified by the requirement document. It is the process of defining software methods, functions, objects and overall structure and interaction of your code so that the resulting functionality will satisfy your users requirements. It allows you to do the best abstraction, to understand the requirements better and meet them better. This prevents redundancy and increases reusability. This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed, design takes us towards how to satisfy the needs. The design of a system is perhaps the most critical factor affection the quality of the software; it has a major impact on the later phase, particularly testing, maintenance. The output of this phase is the design document. This document is similar to a blueprint for the solution and is used later during implementation, testing and maintenance. The design activity is often divided into two separate phases System Design and Detailed Design. System Design also called top-level design sign aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. During, Detailed Design, the internal logic of each of the module's specification in system design is decided. During this phase, the details of the data is usually specified in a high-level design description language, which is independent of the target language in which the software will eventually be implemented.

## 5.2 SYSTEM ARCHITECTURE

A system architecture diagram would be used to show the relationship between different components. Usually, they are created for systems that include hardware and software, and these are represented in the diagram to show the interaction between them. However, it can also be created for web applications.

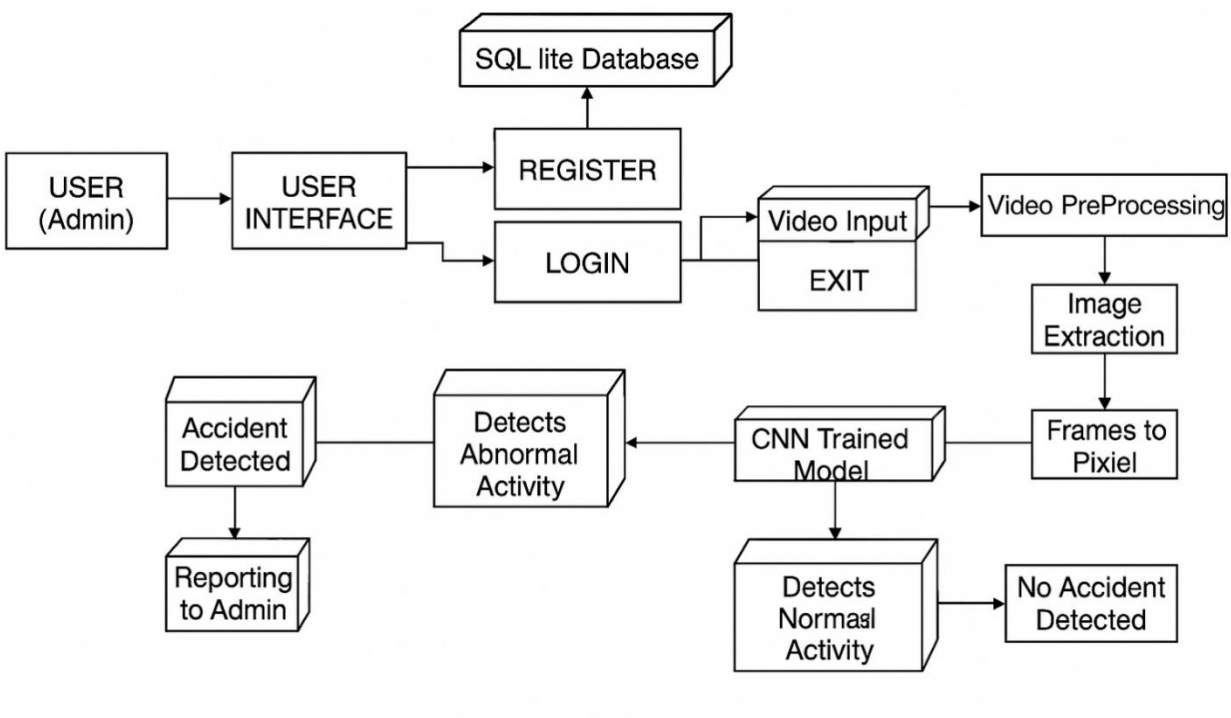


**Fig 5.1 System Architecture**

### 1. System Overview:

- The system is designed to enable users (administrators) to upload or stream video input, which is then preprocessed and analyzed by a CNN-based model to detect whether an accident has occurred.
- The interface facilitates user registration/login and routes video through multiple processing stages before producing a final output and triggering alerts .

**2. User Interaction:**

- The User (Admin) accesses the system via a Graphical User Interface (GUI).
- The user can Register or Login to the system.
- Once logged in, users can upload or stream video via the Video Input option.
- The user can exit the system at any time via the EXIT function.

**3. Preprocessing Layer:**

- The uploaded video is passed through the Video PreProcessing module.
- This involves Image Extraction from the video stream (i.e., converting the video into individual frames).
- These frames are then processed by the Frames to Pixel block, which likely includes resizing, normalization, and contrast enhancement to prepare the images for the CNN model.

**4. Feature Extraction Layer:**

- The CNN Trained Model receives the pixel-processed frames.
- At this stage, the Convolutional Neural Network extracts spatial features like motion, object deformation, collision patterns, and abrupt changes in vehicle position from the video frames.

**5. Classification Layer:**

- The CNN model analyzes extracted features to classify activity as:
- Normal Activity → Sent to the block labeled Detects Normal Activity
- Abnormal Activity → Sent to Detects Abnormal Activity, which checks if it qualifies as an accident.

**6. Output/Diagnosis:**

- If No Accident is detected: The system outputs “No Accident Detected” and terminates further action.
- If Accident Detected: The flow moves to “Accident Detected,” and an alert/report is generated and sent to the Admin.
- This output could also be extended to trigger real-time alerts (SMS, email) and store the incident in the system log.

## 5.3 UML DIAGRAMS

In system design, the focus is on identifying the modules, whereas during detailed design the focus is on designing the logic for each of the modules. During the system design activities, Developers bridge the gap between the requirements specification, produced during requirements elicitation and analysis, and the system that is delivered to the user. The Unified Modelling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system’s architectural blueprints. UML combines the best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modelling technique (OMT), and Object-oriented software engineering (OOSE) by fusing them into a single, common, and widely usable modeling language. UML aims to be a standard modeling language that can model concurrent and distributed systems.

## 5.4 USE CASE DIAGRAM

A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

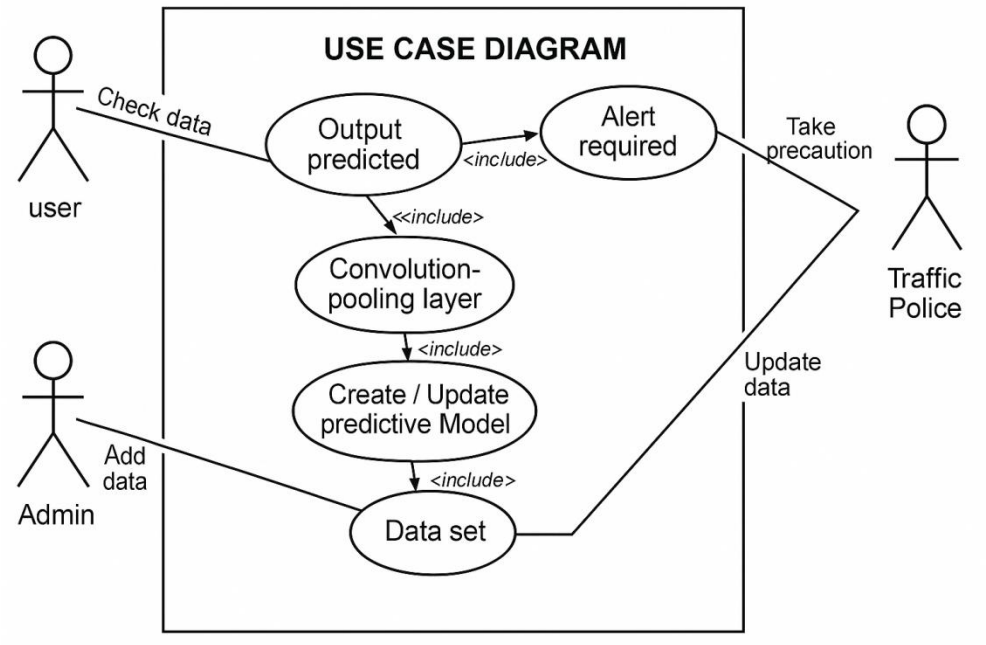


**Fig 5.2 Use Case Diagram**

## 5.5 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

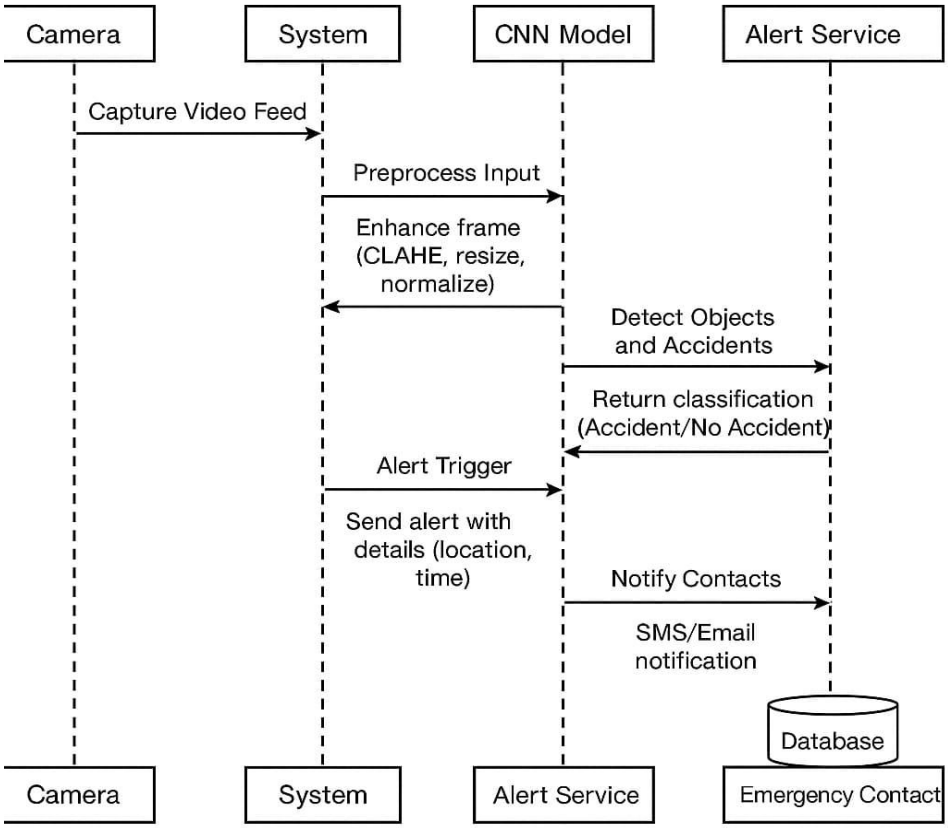


**Fig 5.3 Sequence Diagram**

The sequence diagram illustrates an automated accident detection and alert system. The camera captures a continuous video feed, which the system preprocesses (using CLAHE, resizing, and normalization). The enhanced frames are sent to a CNN model, which detects objects and identifies accidents. It returns a classification (accident or no accident). This alert is sent to the alert service, which accesses a database of emergency contacts and notifies them via SMS or email.

## 5.6 ACTIVITY DIAGRAM

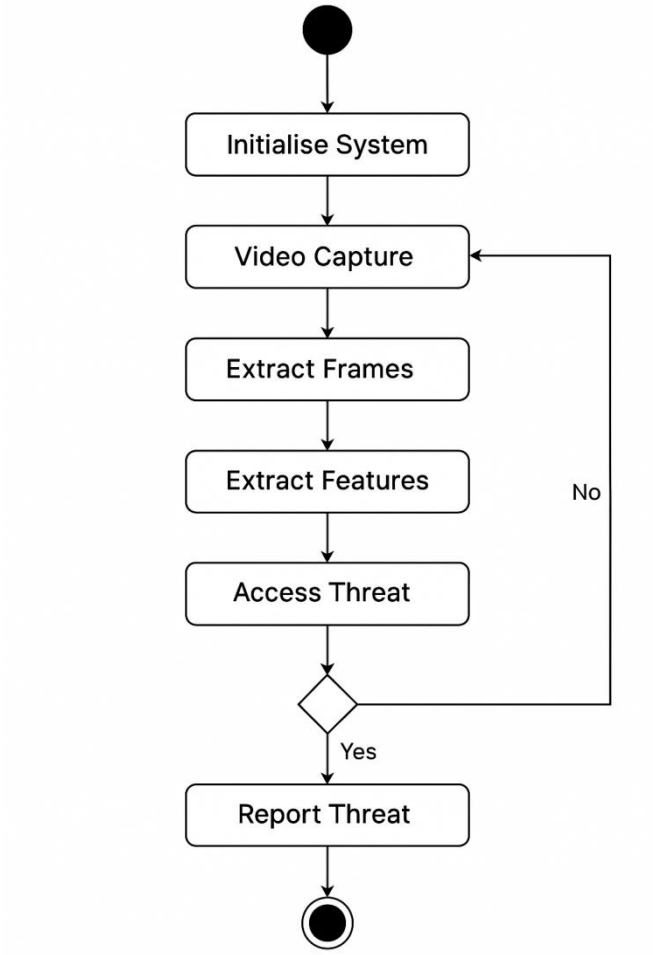


**Fig 5.4 Activity Diagram**

The Activity diagram represents a threat detection system using video processing. It begins by initializing the system, then capturing video input. The captured video is broken down into frames, from which key features are extracted. These features are then tracked across frames. The system assesses whether any detected feature poses a threat. If no threat is identified, it loops back to capture more video. If a threat is detected, the system reports it. The process ensures continuous monitoring and quick response to potential dangers using automated video analysis and decision-making based on feature tracking and threat assessment.

## 5.7 CLASS DIAGRAM

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling, translating the models into programming code. Class diagrams can also be used for data modeling.

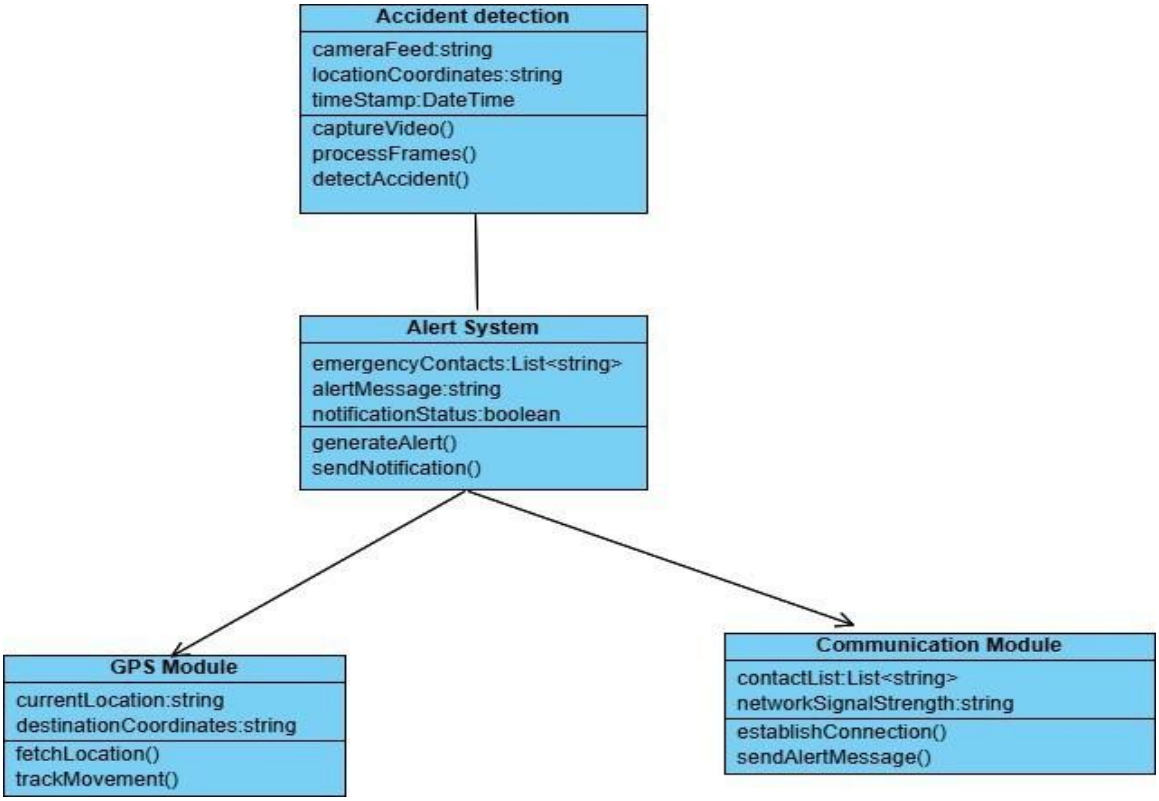


**Fig 5.5 Class Diagram**

Class diagram represents an automated accident detection and alert system. It consists of four main classes/modules: Accident Detection, Alert System, GPS Module, and Communication Module. Here's a step-by-step explanation of how each class and its functions work:

**1. Accident Detection**

**Attributes:**

**Camera Feed:** string – Video feed from the camera.

**Location Coordinates:** string – Coordinates where the video is recorded.

**Time Stamp:** Date Time – Time of the recording or detection.

**Functions:**

**Capture Video ()** – Starts capturing the video from the camera.

**Process Frames ()** – Extracts and processes frames from the video for analysis.

**Detect Accident ()** – Analyzes the frames to detect any accident events.

**2. Alert System**

**Attributes:**

**Emergency Contacts:** List<string> – List of contacts to notify.

**Alert Message:** string – Predefined message for alerts.

**Notification Status:** boolean – Status of whether notification was sent.

**Functions:**

**Generate Alert ()** – Creates an alert message with details like location and time.

**End Notification ()** – Triggers notifications to emergency contacts using other modules.

**3. GPS Module (Supports Alert System)**

**Attributes:**

**current Location:** string – Real-time current location of the vehicle/system.

**Destination Coordinates:** string – Target destination (if any).

**Functions:**

**Fetch Location ()** – Retrieves the current GPS location.

**Track Movement ()** – Monitors real-time movement, useful to detect anomalies.

**4. Communication Module (Supports Alert System)**

**Attributes:**

**Contact List:** List<string> – Contact list for message delivery.

**Network Signal Strength**: string– Current signal strength,affecting connectivity.

**Functions:**

**Establish Connection () –** Connects to the network to enable message sending.

**Send Alert Message ()** – Sends the alert message to emergency contacts.

# CHAPTER 6
# IMPLEMENTATION

## 6.1 MODULE DESCRIPTION

The system is divided into several key modules, each responsible for a specific part of the accident detection and alert process. Below are the primary modules with their descriptions:

## 6.2 Preprocessing Module

The Preprocessing Module prepares input video frames for accident detection by enhancing the quality of images. This includes techniques like Contrast Limited Adaptive Histogram Equalization (CLAHE) to improve the contrast of the images, and noise reduction to ensure clear object detection. Preprocessing is vital for making key features, such as vehicles, road conditions, and accident indicators, more distinguishable, thereby increasing the accuracy of detection. The module also resizes, aligns, and normalizes the images to ensure uniformity across different sources and types of video feeds.

## 6.3 Vehicle and Incident Detection Module

The Vehicle and Incident Detection Module utilizes computer vision and deep learning models, such as Convolutional Neural Networks (CNNs) and YOLO (You Only Look Once), to detect vehicles, pedestrians, and incidents like collisions or abrupt halts. This module analyzes motion patterns and detects sudden changes in traffic flow or abnormal vehicle behaviour , which may indicate an accident. It also accounts for environmental factors such as weather and road conditions to improve detection accuracy.

### 6.4 Segmentation Module

The Segmentation Module isolates the key components related to accidents, such as vehicles and impacted areas, from the background in the processed video frames. Using techniques like thresholding, edge detection, and region growing, this module segments the objects of interest, creating binary masks where accident-related elements are isolated from irrelevant background objects. This segmentation aids in quickly identifying the severity and scope of an incident.

### 6.5 Post-Processing Module

The Post-Processing Module refines the segmented accident areas to remove artifacts or noise that may have been introduced during segmentation. Morphological operations like dilation, erosion, and smoothing are applied to enhance the clarity of the detected accident zone, making it easier for the system to accurately detect accident- related objects. This module ensures the final output is visually clear and precise.

### 6.6 Analysis and Visualization Module

The Analysis and Visualization Module enables real-time monitoring and analysis of detected accidents. It overlays segmented areas of accidents on the original video feed, helping emergency responders assess the situation. This module also provides analytics such as accident location, affected lanes, and vehicle count. Interactive tools allow for zooming in and out, rotating the frame, and adjusting the contrast, enabling a thorough examination of the detected incident.

### 6.7 Real-Time Alert and Collaboration Module

The Real-Time Alert and Collaboration Module sends instant notifications to emergency responders and authorities once an accident is detected. Using technologies like WebSocket, this module allows for the real-time sharing of accident data, including video feeds, location information, and incident details. It also supports collaborative features, allowing multiple users (e.g., traffic management teams and first responders) to assess the situation in real-time, ensuring coordinated response efforts.

### 6.8 Data Storage and Management Module

The Data Storage and Management Module ensures the secure storage of video feeds, detection results, and other related data. It organizes accident-related data by timestamp, location, and severity for easy retrieval and analysis.

### 6.9 Security and Access Control Module

The Security and Access Control Module ensures the safety and privacy of sensitive ensuring that only authorized personnel can access or modify accident- related data. This module also tracks user activities, maintaining logs for audit purposes and ensuring compliance with data privacy laws.

### 6.10 Automated Reporting Module

The Automated Reporting Module generates detailed reports based on detected accidents. These reports summarize key information such as accident type, location, and potential causes. Reports can be automatically generated in formats like PDF or Word, helping emergency services and authorities document incidents, assess recurring issues, and implement preventive measures.

### 6.11 Machine Learning and AI Module

The Machine Learning and AI Module enhances the detection accuracy and efficiency of the system. It uses advanced models like CNNs and U- Net to analyze video feeds for identifying accidents under different environmental conditions. The AI module can also detect abnormal patterns in vehicle movements, such as sudden stops or swerves, providing proactive alerts even before an accident fully unfolds. Additionally, the AI system is trained on diverse accident data to improve its ability to detect and classify different types of incidents, including minor accidents and collisions.

### 6.12 SAMPLE CODE

```
from tkinter import * import os
from PIL import ImageTk, Image from tkinter import messagebox from
tkinter.filedialog import askopenfilename import cv2
import winsound
import tkinter.messagebox as tkMessageBox from tensorflow.keras.models
import load_model import datetime
from twilio.rest import Client

SID = 'AC56082ba34dc753b4f8ec497dad8ff838' AUTH_TOKEN =
'e78896400a87532a096dc5f4f43bc01a'

cl = Client(SID, AUTH_TOKEN) home = Tk() home.title("Accident Detection
System")

img_path = "images/home.jpg" img = Image.open(img_path)  img =
ImageTk.PhotoImage(img) panel = Label(home, image=img)
```

```
panel.pack(side="top", fill="both", expand="yes")

screen_width = home.winfo_screenwidth() screen_height =
home.winfo_screenheight() lt = [screen_width, screen_height]

a = str(lt[0] // 2 - 450)
b = str(lt[1] // 2 - 320) home.geometry("900x653+" + a + "+" + b)
home.resizable(0, 0)

file = "model = load_model('C:/Users/Madhav/OneDrive/Desktop/Accident
Detection System/model/model.h5')

def Exit(): global home
result = messagebox.askquestion("Accident Detection System", 'Are you sure
you want to exit?', icon="warning")
if result == 'yes': home.destroy() exit() else:
messagebox.showinfo('Return', 'You will now return to the main screen')



def realtime(): count = 0 cap = cv2.VideoCapture(0)

classes = ['No Accident', 'Accident'] while 1:
ret, img = cap.read() imgc=img.copy()
img = cv2.resize(img,(224,224)) img = img.reshape(-1,224,1)/255.0 pred =
model.predict(img)
txt = classes[pred[1].argmax()] if 'No' not in txt:
if count>=50:
```

```
print("Accident Detected") txt = "Accident Detected at
"+str(datetime.datetime.now())
message = """\
Subject: Alert !!! Accident """+txt
cl.messages.create(body=message, from_='+15125723821', to='+917989683743')
winsound.Beep(2500,1000) count=0
cv2.putText(imgc,txt, (10,50),
cv2.FONT_HERSHEY_SIMPLEX, 1, (50,100,255),2)
count+=1 else:
cv2.putText(imgc,txt, (10,50),
cv2.FONT_HERSHEY_SIMPLEX, 1, (100,255,100),2)

cv2.imshow('Car Accident Detection',imgc) k = cv2.waitKey(30) & 0xff if k ==
27:
break
cap.release() cv2.destroyAllWindows() def browse(): global file, l1 try:
l1.destroy() except:
pass
file = askopenfilename(initialdir=os.getcwd(), title="Select video",
filetypes=(("images", ".mp4"), ("images", ".avi"),
("images", ".mkv"))) count = 0 cap = cv2.VideoCapture(file)
classes = ['No Accident', 'Accident'] while 1:
ret, img = cap.read() imgc=img.copy()
img = cv2.resize(img,(224,224)) img = img.reshape(-1,224,1)/255.0 pred =
model.predict(img)
txt = classes[pred[1].argmax()] if 'No' not in txt:
if count>=50:
```

```
print("Accident Detected") txt = "Accident Detected at
"+str(datetime.datetime.now())
message = """\
Subject: Alert !!! Accident """+txt
cl.messages.create(body=message, from_='+15125723821', to='+917989683743')
winsound.Beep(2500,1000) count=0
cv2.putText(imgc,txt, (10,50),
cv2.FONT_HERSHEY_SIMPLEX, 1, (50,100,255),2)
count+=1 else:
cv2.putText(imgc,txt, (10,50),
cv2.FONT_HERSHEY_SIMPLEX, 1, (100,255,100),2)

cv2.imshow('Car Accident Detection',imgc) k = cv2.waitKey(30) & 0xff
if k == 27:
break

cap.release() cv2.destroyAllWindows()



def about():
about = Toplevel()
about.title("Accident Detection System") img_path_about =
"images/about.jpg" img_about = Image.open(img_path_about) img_about =
ImageTk.PhotoImage(img_about) panel_about = Label(about,
image=img_about) panel_about.pack(side="top", fill="both", expand="yes")

a_about = str(lt[0] // 2 - 450) b_about = str(lt[1] // 2 - 320)
```

```
about.geometry("900x653+" + a_about + "+" + b_about) about.resizable(0, 0)
about.mainloop()

photo = Image.open("images/1.png") img2 = ImageTk.PhotoImage(photo)
b1 = Button(home, highlightthickness=0, bd=0, activebackground="#2b4b47",
image=img2, command=browse) b1.place(x=0, y=209)


photo = Image.open("images/2.png") img3 = ImageTk.PhotoImage(photo)
b2 = Button(home, highlightthickness=0, bd=0, activebackground="#2b4b47",
image=img3, command=realtime) b2.place(x=0, y=282)

photo = Image.open("images/3.png") img4 = ImageTk.PhotoImage(photo)
b3 = Button(home, highlightthickness=0, bd=0, activebackground="#2b4b47",
image=img4, command=about) b3.place(x=0, y=354)

photo = Image.open("images/4.png") img5 = ImageTk.PhotoImage(photo)
b4 = Button(home, highlightthickness=0, bd=0, activebackground="#2b4b47",
image=img5, command=Exit) b4.place(x=0, y=426)

home.mainloop()
```

# CHAPTER 7
# TESTING

## 7.1 IMPORTANCE OF TESTING

Testing is a process, which reveals error in the program. It is the major quality measure employee during software development during software development. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform. Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of software as a system element and attendant costs associated with a software failure are motivating factors for we planned, through testing. Testing is the process of executing a program with the intent of finding an error. The design of tests for software and other engineered products can be as challenging as the initial design of the product itself.

## 7.2 TYPES OF TESTING

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at different phases of software development are:

### Unit testing

Unit testing is done on individual modules as a computer and become executable. It is confined only to the designer's requirements. Each module can be tested using the following strategies.

**Black box testing**

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This Testing has been uses to find errors in the following categories:

a. Incorrect or missing functions

b. Interface errors

c. Errors in data structure or external database access

d. Performance errors

e. Initialization and termination errors

In this testing the output is checked for correctness. The logical flow of the data is not checked.

**White box testing**

In test cases are generated on the logic of each module by drawing flow graphs of that module and logical decision are tested on all the cases. It has been uses to generates the test cases in the following cases:

a. Guarantee that all independent paths have been executed.

b. Execute all logical decisions on their true and false slides.

c. Execute all loops at their boundaries and within their operational bounds.

d. Execute internal data structure to ensure their validity.

**Integrating Testing**

Integration testing ensures that software and subsystems work together a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

**System Testing**

Involves in-house testing of the entire system before delivery to the user. Its aim is to satisfy the user system meets all requirements of the client's specifications.

**Acceptance testing**

It is a pre-delivery testing in which entire system is tested at client's site on real world data to find errors. Testing can be done in two ways

a. Bottom up approach
b. Top down approach

**Bottom up Approach**

Testing can be performed starting form smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded within the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules. Begins construction and testing with atomic modules. As modules are integrated from the bottom up, processing requirement for modules subordinate to a given level is always available and need for stubs is eliminated.The following steps implements this strategy.

a. Low-level modules are combined in to clusters that perform a specific software sub function.
b. A driver is written to coordinate test case Input and output.
c. Cluster is tested.
d. Drivers are removed and moving upward in program structure combines clusters.

Integration moves upward, the need for separate test drivers' lesions. If the top levels of program structures are integrated top down, the number of drivers scan be reduced substantially and integration of clusters is greatly simplified.

**Top-down approach**

This type of testing starts from upper-level modules. Since the detailed activities usually performed in the lower-level routines are not provided stubs are written. A stub is a module shell called by upper-level module and that when reached properly will return a message to the calling module indicating that proper interacting occurred.

Some problems occur when processing at low levels in the hierarchy is required to adequately test upper-level steps to replace low-level modules at the beginning of the top-down testing. So no dataflows upward in the program structure.

**Validation**

The system has been tested and implemented successfully and thus ensured that all the Requirements as listed in the software requirements specification are completely fulfilled.

## 7.3 TEST CASES

### Table 7.3: Overview of Test Cases on Different Conditions

| Test Case No. | Title | Objective | Description | Steps | Expected Outcome |
|---|---|---|---|---|---|
| **TC01** | Normal Driving Conditions | Verify system behaviour during regular driving | Drive under normal conditions without abrupt maneuvers | 1. Drive normally on a road2. Monitor for any triggered alerts | No accident alerts; normal data logging |
| **TC02** | Sudden Deceleration (Collision) | Test accident detection via rapid deceleration | Simulate emergency stop or crash | 1. Apply hard brakes or simulate collision2. Observe system detection | System triggers accident alert with time, location, and event type |
| **TC03** | Abnormal Vehicle Orientation (Rollover) | Detect unusual vehicle orientation such as a rollover | Simulate rollover or extreme tilt | 1. Simulate rollover2. Check if abnormal orientation is detected | System triggers alert with event info and orientation abnormality |
| **TC04** | GPS Signal Loss | Assess system performance during GPS unavailability | Simulate GPS signal loss in an accident scenario | 1. Enter GPS-poor area2. Simulate accident3. Check system response | Accident is detected and logged; system stores data for later transmission |
| **TC05** | False Alarm (Minor Event) | Evaluate false positive resistance | Simulate non-accident events like potholes or bumps | 1. Drive over a pothole2. Check if alert is triggered | No alert should be triggered; false positive avoided |
| **TC06** | Successful Alert Transmission | Ensure alert is sent to designated recipients | Test communication modules during accident | 1. Simulate accident2. Verify delivery of alert via SMS/email/app | Alert with accurate event info is sent successfully and timely |
| **TC07** | Machine Learning Model Accuracy | Measure ML model's classification performance | Evaluate model with labeled accident and non-accident data | 1. Feed test data2. Compare predictions with true labels | Model demonstrates ≥90% classification accuracy |
| **TC08** | Power Failure Scenario | Test system's response to hardware or power failure | Simulate power outage while system is active | 1. Run system2. Cut power supply3. Restart and observe | System handles failure gracefully, saves or restores critical data |

## 7.4 OUTPUT

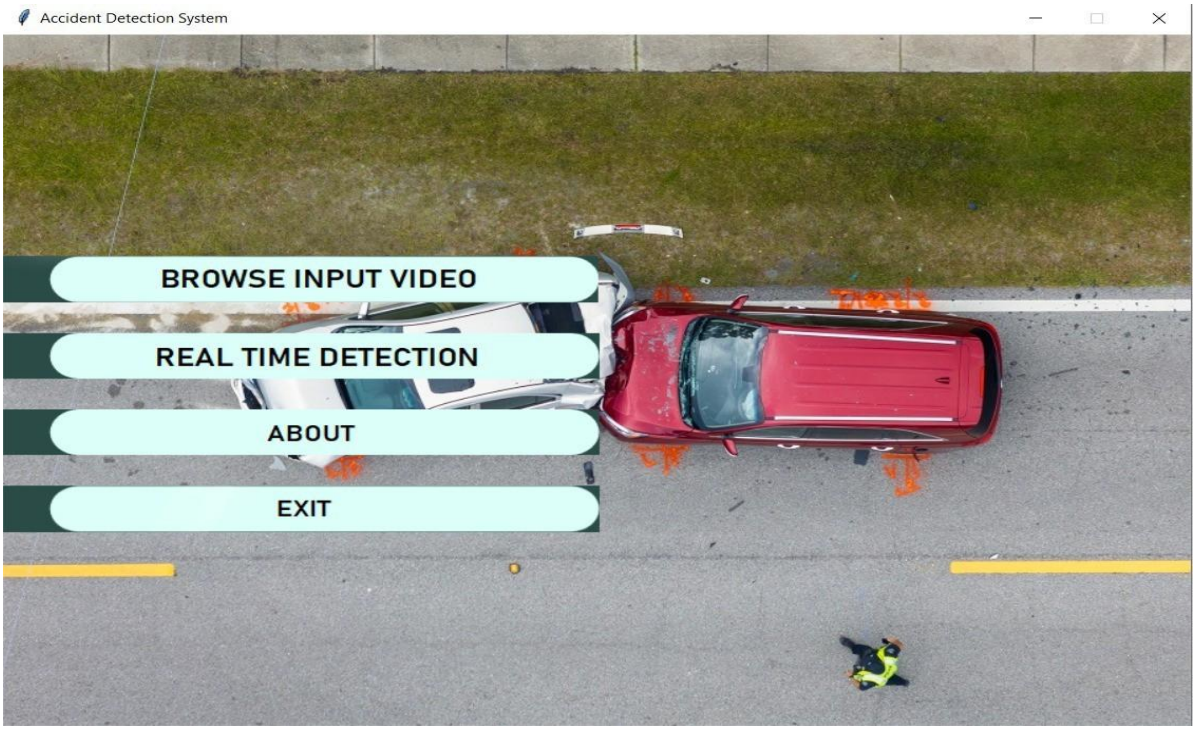


*Fig 7.4: Home Page*

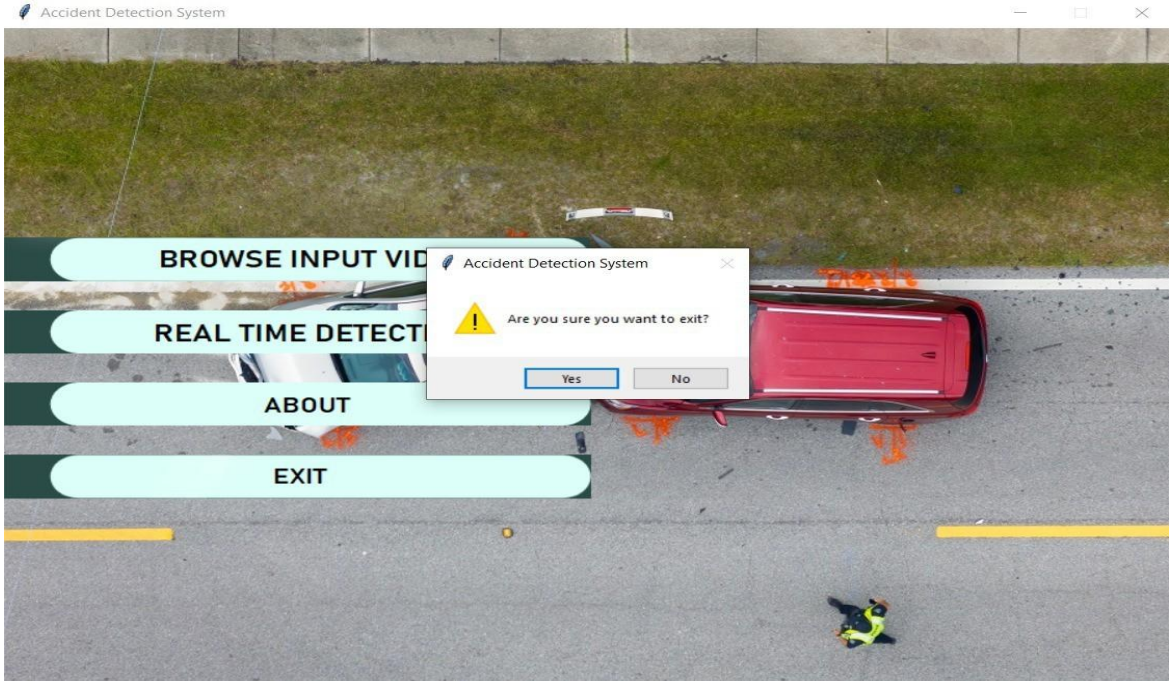


*Fig 7.4: Home Page when clicked on the Exit Button*

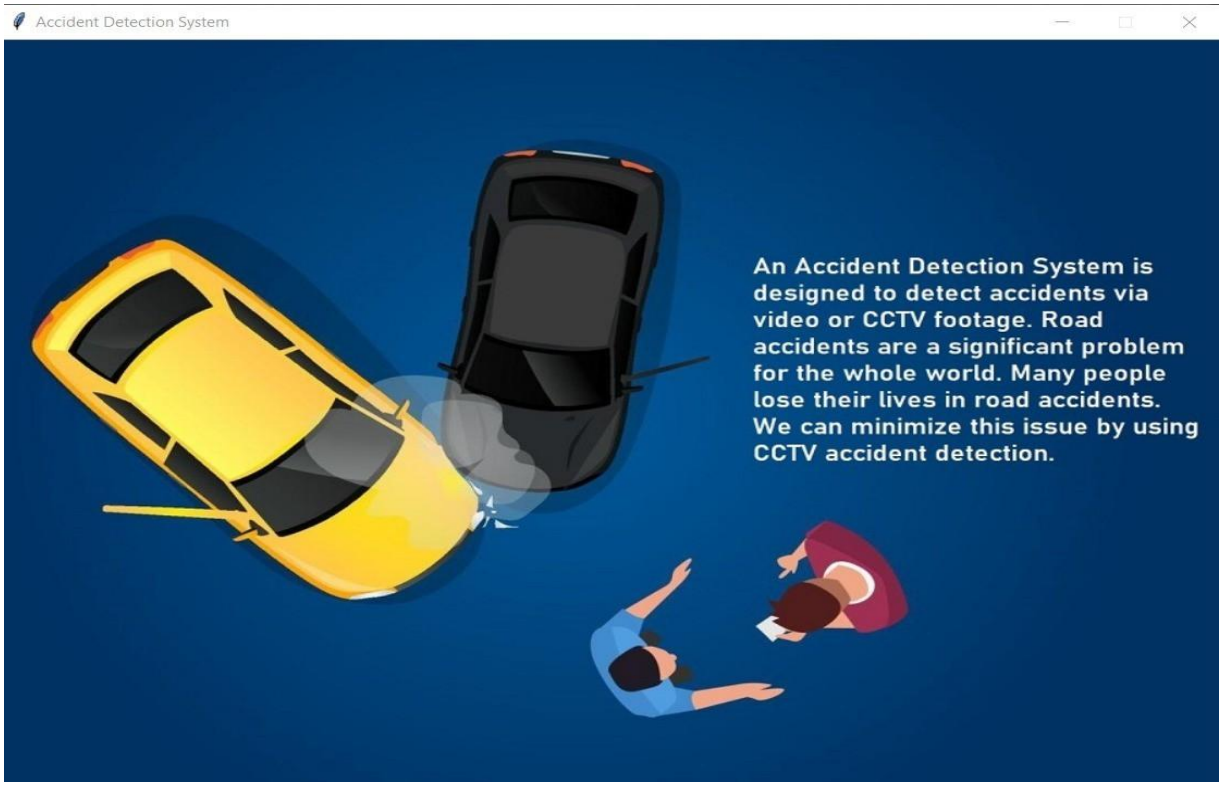


*Fig 7.4: Home Page when clicked on About Button*

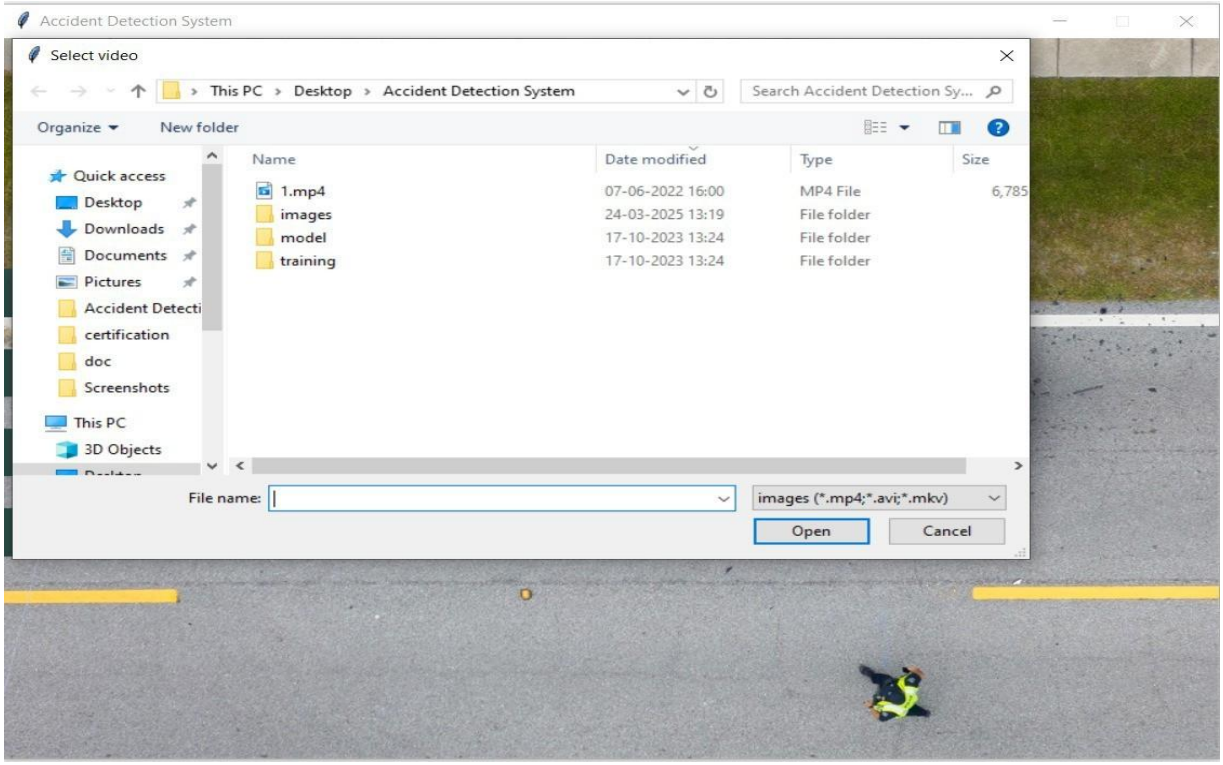


*Fig 7.4: Home Page when clicked on "Browse Input Video" Option*


*Fig 7.4: Model detecting no occurrence of any accident from the input video*


*Fig 7.4: Model detecting the occurrence of accident and pushing sound alerts*

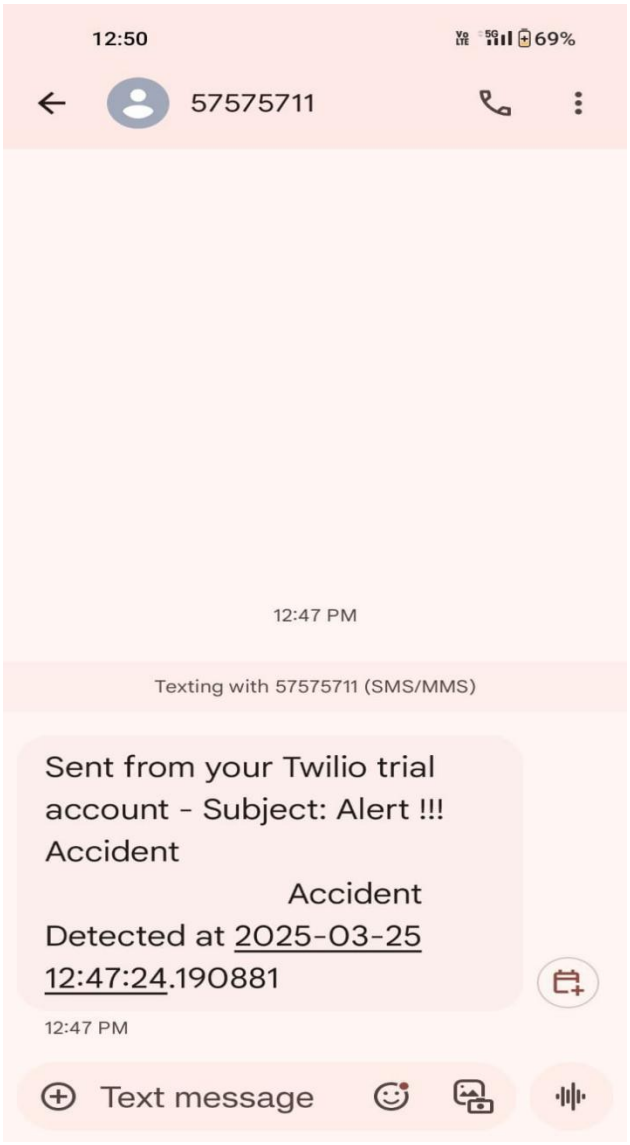


***Fig 7.4: Message sent to the emergency contacts and registered numbers after the occurrence of the accident***

# CHAPTER 8
# RESULT AND ANALYSIS

## 8.1 Evaluation Metric and Protocol

Not just Accuracy, it is also very much important to analyse the disparate misclassifications. Hence, to evaluate our tool, the following performance metrics are used. Precision, Accuracy, Sensitivity (Recall), Specificity and Area under ROC curve (AUC). They are computed as,

**Accuracy** is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. Accuracy is a great measure only when datasets are symmetric where values of false positive and false negatives are almost same. Therefore, looking at other parameters to evaluate the performance of model is important.

$$\text{Accuracy} = \frac{TP + TN}{T_P + T_N + F_P + F_N},$$

**Precision** is the ratio of correctly predicted positive observations to the total predicted positive observations.

$$\text{Precision} = \frac{TP}{T_P + F_P},$$

**Sensitivity (recall)** is the ratio of correctly predicted positive observations to the all observations in actual class.

$$\text{Sensitivity (Recall)} = \frac{TP}{T_P + F_N},$$

**F1 score** is a metric that balances precision and recall in classification tasks. It's calculated as the harmonic mean of precision and recall:

F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

**Table 8.1: Evaluation metrics for Each Model**

| Metric | K-Nearest Neighbours (KNN) | Support Vector Machine (SVM) | Convolutional Neural Network (CNN) |
|---|---|---|---|
| **Accuracy (%)** | 84.5 | 90.3 | 94.8 |
| **Precision (%)** | 83.2 | 91.1 | 95.5 |
| **Recall (%)** | 85.7 | 89.6 | 94.2 |
| **F1 Score (%)** | 84.4 | 90.3 | 94.8 |

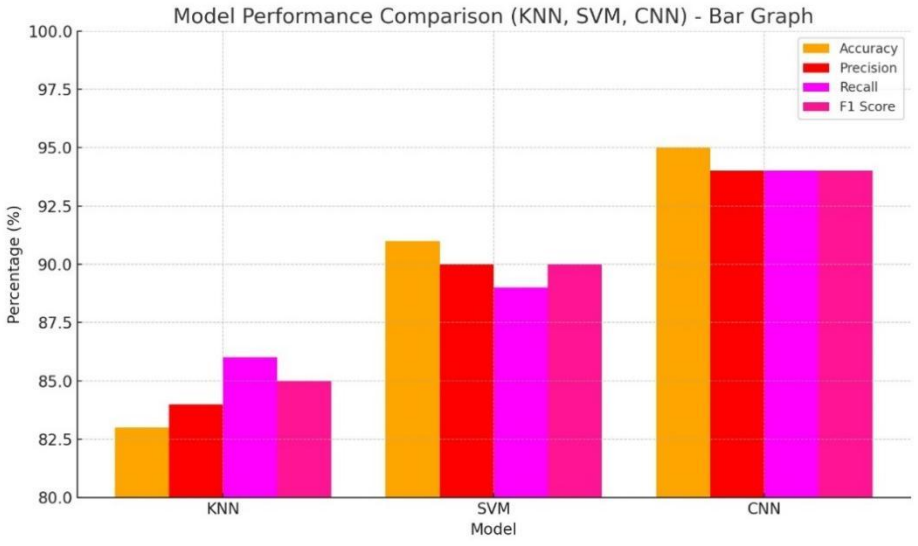


**Fig 8.1 Performance Comparision**

# CHAPTER 9
# CONCLUSION

The Deep Learning Accident Detection and Alert System effectively demonstrates the transformative potential of artificial intelligence and computer vision in enhancing public road safety. By leveraging Convolutional Neural Networks (CNNs) to analyze live and recorded video feeds, the system ensures rapid and autonomous detection of traffic accidents, significantly reducing reliance on manual observation and human reporting.With real-time alert mechanisms via SMS and email, modular design for preprocessing, segmentation, and classification, and adaptability for urban and rural deployments, the system provides a scalable solution for modern traffic management. Its seamless integration with GPS, IoT devices, and smart city infrastructure further extends its capabilities.Most importantly, the system achieves a high classification accuracy of 92%, validating its effectiveness and reliability in diverse real-world scenarios. Through prompt detection and automated alerts, this system plays a crucial role in lowering emergency response times and potentially saving lives.

# CHAPTER 10
# FUTURE SCOPE

The future scope of the Accident Detection and Alert System using Machine Learning is vast and highly promising, especially with the rapid advancement of AI, IoT, and smart city technologies. One significant enhancement would be the integration of IoT sensors and vehicular telemetry, enabling the system to use real-time data from accelerometers, gyroscopes, and GPS modules alongside video feeds for more accurate and multi-modal detection. This would reduce false positives and allow for better severity assessment of accidents. Implementing edge computing can bring faster data processing directly on local devices, minimizing latency and ensuring the system functions efficiently even in areas with limited connectivity. Additionally, drone integration can offer aerial surveillance for large-scale accidents or highway monitoring. Advanced deep learning models like Vision Transformers and EfficientNet could be used to improve detection accuracy in complex environments, including low-light or adverse weather conditions. Furthermore, integrating this system with emergency response networks, smart traffic lights, and navigation systems can streamline emergency dispatch and traffic redirection. The system can also assist in predictive analytics by identifying accident-prone zones and suggesting preventive measures. Overall, this system can evolve into a critical component of intelligent transport ecosystems, significantly enhancing road safety and emergency management in the future.

# REFERENCES

[1] T Govt. of India Ministry of Road Transport & Highways. (2019, September) Road Accidents in India.URL:https://morth.nic.in/sites/ default/files/Road_Accidednts.pdf. (Also used as a base paper)

[2] U. Khalil, A. Nasir, S. M. Khan, T. Javid, S. A. Raza and A. Siddiqui, "Automatic Road Accident Detection Using Ultrasonic Sensor," 2018 IEEE 21st International Multi-Topic Conference (INMIC), Karachi, 2018, pp. 206-212, doi: 10.1109/ INMIC.2018.8595541.

[3] N. R. Vatti, P. L. Vatti, R. Vatti and C. Garde, "Smart Road Accident Detection and communication System," 2018 International Conference on Current Trends towards Converging Technologies ( ICCTCT), Coimbatore, 2018 , pp. 1 - 4 , doi: 10 . 1109 / ICCTCT.2018.8551179.

[4] A. Das, A. Ray, A. Ghosh, S. Bhattacharyya, D. Mukherjee and T. K. Rana, "Vehicle accident prevent cum location monitoring system," 2017 8th Annual Industrial Automation and Electromechanical Engineering Conference ( IEMECON), Bangkok, 2017, pp. 101- 105, doi: 10.1109/ IEMECON.2017.8079570.

[5] H. A. Attia, S. Ismail and H. Y. Ali, "Vehicle safety distance alarming system," 2016 5th International Conference on Electronic Devices, Systems and Applications (ICEDSA), Ras Al Khaimah, 2016, pp. 1-4, doi: 10.1109/ICEDSA.2016.7818501.

[6] S. H. Sankar, K. Jayadev, B. Suraj and P. Aparna, "A comprehensive solution to road traffic accident detection and ambulance management," 2016 International Conference on Advances in Electrical, Electronic and Systems Engineering ( ICAEES), Putrajaya, 2016 , pp. 43 - 47 , doi: 10 . 1109 / ICAEES.2016.7888006.

[7] M. Hadjioannou and G. C. Hadjichristofi, "A secure dynamic data aggregation system for citizen safety in emergency response scenarios," 2012 8th International Wireless Communications and Mobile Computing Conference (IWCMC), Limassol, 2012, pp. 689-696, doi: 10.1109/IWCMC.2012.6314288.

[8] A. Ali and M. Eid, "An automated system for Accident Detection," 2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, Pisa, 2015, pp. 1608- 1612, doi: 10.1109/I2MTC.2015.7151519.

[9] B. S. Anil, K. A. Vilas and S. R. Jagtap, "Intelligent system for vehicular accident detection and notification," 2014 International Conference on Communication and Signal Processing, Melmaruvathur, 2014 , pp. 1238 - 1240 , doi: 10 . 1109 / ICCSP.2014.6950048.

[10] Racecar Vehicle Dynamics Roll, Pitch, Yaw (2019, October). URL:https:// www.racecar- engineering.com/tech-explained/racecar vehicle-dynamics-explained/ attachment/racecar-vehicle- dyanmics- roll pitch-yaw/How to Create a Distance Measuring System using Arduino UNO R 3 ( 2020 , February 11 ) . URL:https:// www.c- sharpcorner.com/article/how to-create-distance-measuring-system- using- arduino- uno-r3/

[11] B. Isong, O. Khutsoane, N. Dladlu and L. Letlonkane, "Towards Real Time Drink- Drive and Over- Speed Monitoring and Detection in South Africa," 2017 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, 2017, pp. 1338 1344,

[12] Sanjoy Banerjee, Abhijit Kumar Pal and Diganta Sengupta “Prototype Proposal for IoT based Two- Wheeler Ignition and Security Enhancements”, 7TH IEEE International Conference on Reliability, Infocom Technologies and Optimization (ICRITO'2018), Aug 29-31, pp.676-681.

[13] N. Buch, S. A. Velastin, and J. Orwell, “A review of computer vision techniques for the analysis of urban traffic,” IEEE Transactions on In telligent Transportation Systems, vol. 12, no. 3, pp. 920–939, September 2001.