ABSTRACT

Traffic collisions are a major global issue, leading to significant human suffering, economic losses, and widespread societal impacts. According to the World Health Organization (WHO), road traffic accidents result in millions of injuries and fatalities each year, making road safety a critical concern for governments and organizations worldwide. Traditional methods of traffic accident detection and responsibility assessment often rely on human judgment, eyewitness accounts, and police investigations, which can be subjective, time-consuming, and prone to errors. As the frequency and complexity of accidents increase, there is a pressing need for more efficient, accurate, and automated systems that can enhance road safety, streamline accident management, and improve the process of determining fault. The proposed work introduces an innovative, Internet of Things (IoT)-based system for real-time collision detection and driver responsibility assessment. The system integrates multiple technologies, including vehicle sensors, GPS modules, and cameras, to capture detailed data during traffic accidents. By utilizing the latest advancements in machine learning, specifically the YOLO V9 (You Only Look Once) model for accident detection and a Random Forest algorithm for fault assessment, the proposed system automates the process of analyzing accident data and determining the responsible party. YOLO V9, a state-of-the-art computer vision algorithm, is employed to analyze video feeds from cameras mounted on vehicles, enabling the system to detect collisions with high accuracy in real time. Meanwhile, the Random Forest algorithm processes a variety of inputs such as vehicle speed, position, and collision dynamics to assess responsibility based on pre-trained historical data. The system's architecture is designed to ensure both efficiency and reliability.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

IOT	Internet Of Things
GPS	Global Positioning System
SD	Secure Digital
YOLO	You Only Look Once
RF	Random Forest
AI	Artificial Intelligence
ML	Machine Learning
API	Application Programming Interface
GUI	Graphical User Interface

CHAPTER 1
INTRODUCTION

1.1OVERVIEW

The model aims to develop an intelligent, automated system for real-time accident detection and responsibility evaluation using a combination of sensor data, machine learning, and deep learning techniques. In conventional scenarios, determining fault in traffic collisions is often reliant on eyewitness accounts, physical inspections, and lengthy police investigations, all of which are susceptible to human error, bias, and delays. To address these issues, this project proposes a novel framework that integrates hardware components and intelligent algorithms to streamline and automate the entire process. The system utilizes an ESP32-CAM, mounted on the dashboard of a vehicle, which continuously records live footage of the road ahead. This footage is processed using YOLO V9, a state-of-the-art deep learning object detection model, to detect crash events and traffic violations with high accuracy and speed. Alongside this, an ultrasonic sensor measures vehicle speed in real time, while a GPS module provides continuous location tracking, allowing the system to collect detailed environmental and situational data at the time of a collision.

All sensor data is transmitted to a Python-based backend, where it is analyzed using the Random Forest algorithm, a powerful machine learning classifier known for its accuracy in handling complex, real-world datasets. This algorithm processes inputs like speed, traffic light status, road conditions, and impact data to objectively predict liability in an accident. The integration of AI and IoT technologies not only enhances accuracy but also reduces the time needed to generate detailed reports, which can be accessed through a user-friendly web interface by insurance agents, law enforcement, or other stakeholders. The system is also stress-tested for scalability and has shown high performance under simultaneous multi-incident scenarios. Results demonstrate significant improvements over traditional methods in terms of detection accuracy, false positive reduction, and processing time. By automating accident analysis and responsibility assessment, this project contributes to the broader vision of intelligent transportation systems and offers a scalable, data-driven solution for improving road safety and efficiency in post-accident processes.

1.2 INTERNET OF THINGS (IOT)

The Internet of Things (IoT) is a transformative paradigm that enables the interconnection of physical devices such as sensors, cameras, microcontrollers, and actuators across networks to share data and enable intelligent decision-making. In the context of smart transportation and accident liability assessment, IoT facilitates real-time data acquisition from vehicles and road environments, supporting intelligent systems in identifying potential threats, incidents, or violations without human intervention [13]. In the proposed system, IoT serves as the foundational architecture by enabling seamless communication between the ESP32 microcontroller, GPS 6M V2 module, ultrasonic speed sensors, and the ESP32-CAM. These interconnected devices constantly monitor parameters such as speed, vehicle position, environmental context, and video footage. This enables the system to gather a detailed and holistic dataset at the exact moment of an incident. Real-time data transmission from these IoT modules is made possible through wireless communication protocols like Wi-Fi, embedded within the ESP32 chip, ensuring fast delivery to backend servers for analysis [14]. The reliability and responsiveness of IoT significantly reduce the delay that typically occurs in traditional accident assessments, where human inspections and witness accounts are often subjective and slow. Instead, sensor readings such as acceleration, GPS trajectory, and visual evidence are captured with millisecond accuracy, minimizing evidence loss or distortion [18]. This automated detection and contextual data gathering mechanism also aids emergency services in real-time tracking and deployment after a collision. Moreover, integration with AI techniques like YOLO V9 for object detection and Random Forest for responsibility classification further demonstrates the synergy between IoT and advanced computational intelligence.

In addition to performance, the system addresses data security and integrity through encryption protocols during transmission, ensuring that sensitive sensor and video data remain protected from tampering or loss. As IoT becomes more widely adopted in vehicle networks, security remains a crucial aspect for system reliability and public trust. Furthermore, the scalability of IoT infrastructure allows for future integration with autonomous vehicle systems and city-wide smart traffic networks. This adaptability supports long-term sustainability, where connected systems can not only detect and analyze accidents but also anticipate risky driving behavior and contribute to accident prevention strategies at scale. Together, these capabilities affirm that IoT is not only vital to data-driven traffic monitoring but also essential in building intelligent, secure, and proactive transportation ecosystems [18].

1.3 YOLO V9 AND RANDOM FOREST

The integration of machine learning into intelligent transportation systems has significantly transformed how road safety is monitored, traffic violations are identified, and driver liability is assessed. The proposed system leverages two powerful AI models YOLO V9 and Random Forest selected for their specialized strengths in processing unstructured and structured data, respectively. By combining these two models, the system ensures real-time decision-making that is both accurate and explainable, which is crucial in high-stakes scenarios such as traffic accidents. YOLO V9 (You Only Look Once - Version 9) is a state-of-the-art object detection model designed for real-time video analysis with exceptional precision. It belongs to the one-stage object detector family, meaning it performs localization and classification simultaneously, eliminating the need for separate region proposal steps and thus reducing latency. In the proposed project, YOLO V9 is deployed to analyze dashcam footage in real time, identifying critical elements such as vehicles, pedestrians, traffic signals, road signs, and potential collision threats. This visual context is essential for assessing the events surrounding an incident, including traffic signal violations, lane changes, and obstruction detection [15].

The core strength of YOLO V9 lies in its ability to detect multiple objects within a single frame and assign accurate labels and bounding boxes in milliseconds. Its architecture, optimized with attention modules and spatial pyramid pooling, ensures consistent performance across various environmental conditions such as low-light, rain, or heavy traffic. The model is trained on diverse datasets to improve generalization across different road types, camera angles, and geographical settings, ensuring broad applicability and robustness in real-world scenarios [15].

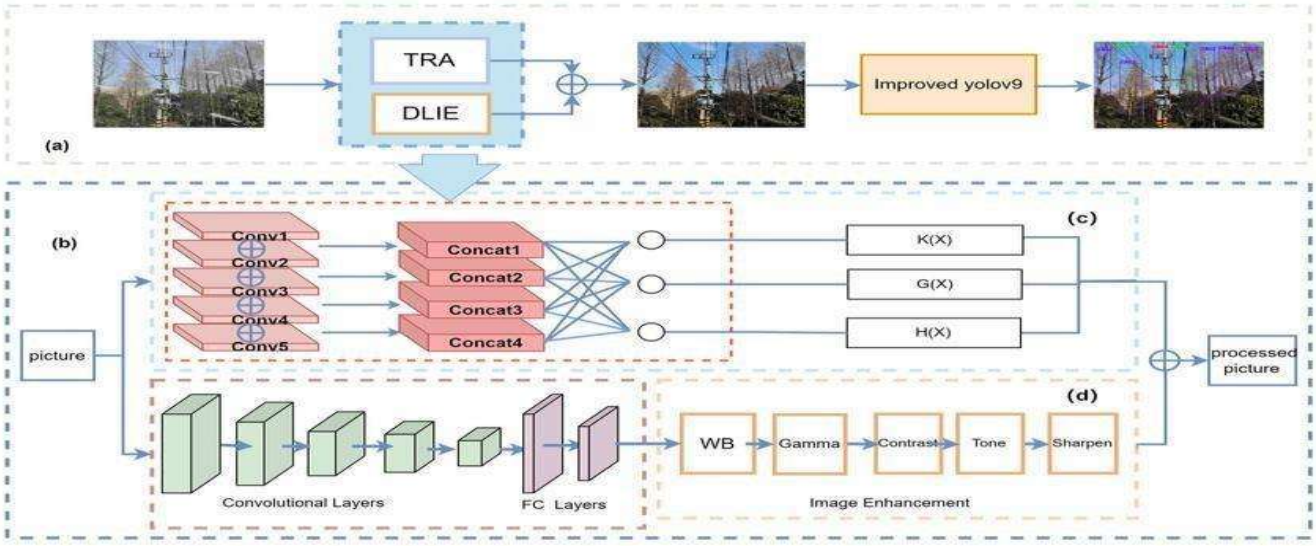

Fig 1.1 YOLO V9 Model

The object detection flow using YOLO V9 is visually depicted in Figure 1.1, where the model scans each frame, identifies key objects, and extracts situational context that supports liability assessment. This real-time object recognition pipeline plays a vital role in validating sensor data, ensuring that visual evidence corroborates findings from structured numerical inputs. Random Forest, in contrast, is a robust ensemble machine learning algorithm well-suited for handling structured, tabular data. It constructs multiple decision trees during the training phase and aggregates their results through majority voting (for classification) or averaging (for regression). In this project, Random Forest is applied to interpret sensor data from the vehicle's IoT subsystem, including inputs such as speed logs, acceleration and braking patterns, GPS coordinates, and timestamped event markers [16]. This structured dataset is used to classify driving behavior into categories such as "Normal", "Overspeeding," providing essential insights into driver intent and conduct at the time of an incident.

The algorithm's strength lies in its ability to handle noisy and imbalanced data without overfitting, making it particularly effective for real-world scenarios where sensor data might contain fluctuations or outliers. Random Forest is also highly interpretable, enabling traceability of decisions, which is critical in legal and insurance contexts where transparency and accountability are paramount. The model supports real-time classification and adapts well to varying driving patterns and environmental contexts [16]. By combining YOLO V9's high-speed visual analytics with Random Forest's sensor-based behavioral analysis, the system forms a hybrid AI architecture capable of delivering comprehensive accident assessments. This dual-model setup ensures that every traffic incident is evaluated from both qualitative and quantitative perspectives. For instance, while YOLO V9 might identify a vehicle ignoring a red light moments before a collision, Random Forest can simultaneously detect the speeding behavior or braking delay that contributed to the impact [17]. Together, they provide a complete narrative backed by visual proof and numerical evidence. This synergistic approach not only enhances the system's decision-making accuracy but also strengthens its credibility in high-liability environments such as traffic law enforcement and insurance adjudication. The combined architecture is designed for low-latency performance, allowing for real-time deployment in smart vehicles, urban surveillance systems, and automated accident reporting tools. YOLO V9 processes video feeds with minimal computational overhead, while Random Forest executes its decision logic efficiently even on edge devices, making the system scalable and energy-efficient [17]. Moreover, the fusion of these two AI models offers automated, data-driven fault analysis that reduces human bias and minimizes investigation delays. It transforms the way accident responsibility is assessed shifting from subjective testimonies to measurable, system-driven evaluations.

CHAPTER 2

LITERATURE SURVEY

2.1 PURPOSE OF THE STUDY

The purpose of the study is to address the ongoing global issue of traffic accidents by developing a system that improves collision detection and accurately assesses driver responsibility. Traditional fault analysis relies on manual methods like witness reports and police investigations, which are often slow and unreliable. This study aims to overcome these limitations by using IoT-enabled devices such as sensors and cameras to collect real-time data during accidents. Deep learning techniques, specifically YOLO V9, are used to detect crash events and road violations from video inputs with high precision. Simultaneously, machine learning models like Random Forest analyze sensor data including speed, location, and braking behavior to predict responsibility patterns. The study explores existing methodologies, highlights the effectiveness of combining AI with IoT, and identifies current research gaps. Ultimately, this work aims to lay the foundation for a smart, automated accident analysis system that enhances road safety, accelerates liability determination, and improves the efficiency of traffic management systems.

2.2 RELATED WORKS

2.2.1 Responsibility Evaluation in Vehicle Collisions From Driving Recorder Videos Using Open Data

The paper [1] presents a comprehensive and practical framework for automated responsibility evaluation in traffic collisions using dashcam video footage combined with publicly available geospatial data. The authors identify a critical issue in traffic accident analysis: the over-reliance on manual, time-consuming methods such as police documentation, eyewitness testimony, and on-site inspections. These traditional practices are prone to subjectivity, delays, and inaccuracies, especially in complex or ambiguous accident scenarios. To overcome these challenges, the study proposes a hybrid model that integrates deep learning with rule-based reasoning, offering a more consistent and data-driven approach to post-accident evaluation. At the core of the proposed system is the YOLOv8 object detection model, which is trained to identify and classify visual elements in driving footage including vehicles, pedestrians, lane boundaries, traffic lights, and road signs. The detected objects are not just used for recognition but also for contextual understanding. For instance, the system can analyze the distance between vehicles, monitor signal

compliance, and detect lane encroachments all of which are essential factors in determining liability. These visual inputs are processed in real time, allowing the model to pinpoint the exact frame of collision and track vehicle behavior immediately before and after the incident. What makes the approach particularly novel is the integration of OpenStreetMap (OSM) data, which serves as a contextual map layer for interpreting traffic regulations specific to the accident location. The system retrieves details about road types, speed limits, signal positions, and intersection configurations. This geographic metadata enhances the semantic reasoning module, enabling the system to assess which traffic rules were in effect and whether any violations occurred. For example, if a vehicle is detected running a red light at a known intersection, the rule engine uses both video evidence and mapped signal logic to assign fault with clarity and confidence. The study further validates the model through extensive experimentation using a custom-built dataset consisting of real-world crash footage collected from public video sources such as YouTube. This dataset was manually annotated and used to retrain YOLOv8 for better accuracy in detecting collision events under varying conditions including night time, fog, rain, and differing traffic densities. The authors also deploy a mobile-server architecture that allows video data to be streamed from user devices to a central system where the AI-based processing and rule application take place. This design supports fast, scalable implementation in smart cities and connected vehicle environments. A notable achievement of the system is its reported 83% accuracy in responsibility assignment, which marks a significant improvement over earlier models that either relied purely on vision or omitted geospatial context. The model performs well even in complex scenarios such as angle collisions and multi-vehicle crashes. Moreover, it supports real-time or near-real-time evaluation, enabling stakeholders like law enforcement officers, insurance companies, and legal advisors to access structured incident reports within minutes of a crash.

2.2.2 Vehicle Crash Detection using YOLO Algorithm

The paper [2] addresses the critical issue of delayed response in road traffic accidents, which often leads to increased fatalities and prolonged disruptions. The study emphasizes that in many real-world cases, victims of car crashes lose their lives not solely due to the severity of the accident but due to the lack of timely medical assistance and reporting. Recognizing this problem, the authors propose a YOLO-based system designed to automatically detect vehicle crashes in real time and instantly notify emergency services and nearby authorities. The objective of the system is not limited to detection alone but extends to contributing directly to traffic safety and rescue efficiency in modern smart city environments. The architecture of the proposed system is centered on YOLO (You Only Look Once), a deep learning model known for its balance between speed and detection accuracy. The system is implemented within an urban

surveillance infrastructure that uses IP cameras and embedded sensors installed at strategic traffic points. These components capture video footage of ongoing traffic, which is processed using the YOLO framework to identify anomalies such as sudden stops, collisions, or erratic vehicle behavior indicative of an accident. The model is trained to recognize multiple object classes such as cars, motorcycles, and pedestrians, and to analyze their trajectories over time to detect collision events with high reliability. A key advantage of the YOLO-based approach is its real-time capability. Unlike traditional multi-stage detection systems, YOLO's one-shot architecture enables the detection of multiple objects in a single frame with minimal computational delay. This characteristic is particularly vital in the context of crash detection, where every second can be critical in saving lives. Once a potential accident is detected, the system is programmed to trigger alerts to designated emergency centers and send automatic notifications to nearby vehicles or control units, thus facilitating quick mobilization of resources. The study also outlines the integration of this detection system within smart transportation networks, emphasizing the potential for AI-driven traffic management to reduce secondary accidents and enhance road safety. In many cases, the failure to detect an accident promptly leads to follow-up collisions, particularly in high-speed or low-visibility areas. By providing early warnings and updating control centers in real time, the system can redirect traffic, activate warning signs, and dispatch ambulances or police units, thereby minimizing risk and ensuring public safety. Furthermore, the authors discuss the practical deployment aspects of the model. The detection algorithm is optimized to function on live video streams rather than static images, ensuring that the system can scale to dense, continuously moving traffic conditions. It is tested in simulated urban environments and demonstrates reliable performance across various lighting conditions and weather scenarios. Although the study does not include responsibility determination or fault attribution, it sets a strong precedent for integrating object detection in public safety frameworks.

2.2.3 Recent Advances in Traffic Accident Analysis and Prediction: A Comprehensive Review of Machine Learning Techniques

The paper [3] addresses a critical public health challenge by providing a comprehensive review of recent advances in the analysis and prediction of traffic accidents, which claim the lives of nearly 1.19 million people annually worldwide. The authors emphasize the increasing importance of advanced machine learning (ML) techniques to predict various aspects of traffic accidents, including risk, frequency, severity, and the underlying factors contributing to collisions. By reviewing over 191 studies from the last five years, the paper provides insights into the evolution of predictive models that incorporate diverse, heterogeneous data sources, ranging from environmental conditions and road type to driver behavior and vehicle dynamics.

One of the key strengths of the study lies in its identification of data integration techniques that combine heterogeneous datasets such as traffic flow data, weather conditions, and historical accident reports with machine learning algorithms to improve the prediction of accident risks. The review highlights several successful models that utilize machine learning algorithms such as Random Forest, Support Vector Machines (SVMs), and deep learning networks to predict crash outcomes with high accuracy. However, the authors point out that many of the existing models focus primarily on predicting accident occurrence rather than assessing responsibility, underscoring a significant gap in research that the proposed system in this report seeks to address. Additionally, the study draws attention to the need for improved predictive capabilities in real-time accident analysis. While predictive models have made significant strides in forecasting accident risk, there is still a lack of real-time fault assessment during or immediately after accidents. The paper advocates for an integrated approach, combining predictive analytics with real-time data processing to enable immediate decision-making in emergency response systems. This integration of advanced machine learning techniques with live data is critical for reducing the time between accident occurrence and liability determination, ensuring a faster response by emergency services and more accurate fault attribution in post-accident investigations.

2.2.4 Evaluating the effectiveness of machine learning techniques in forecasting the severity of traffic accidents

The paper [4] explores the application of machine learning algorithms, specifically Random Forest and Logistic Regression, to predict the severity of traffic accidents. Traffic crashes are a significant public safety concern, causing substantial injuries and fatalities worldwide. The research uses over two million traffic accident records from the UK, spanning a decade, to train various models aimed at predicting crash severity. Random Forest, in particular, was found to outperform other models, achieving an accuracy of 87%. This high level of prediction accuracy makes it a suitable choice for real-world applications in accident severity forecasting, offering potential improvements in traffic safety management. The study also identifies critical factors that influence accident severity, including vehicle characteristics (such as age and engine capacity), driver demographics, and environmental conditions like road type and weather. Random Forest's ability to analyze complex, non-linear relationships between these variables makes it highly effective for accident severity classification. The findings provide a foundation for future advancements in traffic safety, highlighting the potential of machine learning to enhance road safety strategies and accident prevention programs. While the primary focus of this paper is on accident severity prediction, its insights into the importance of structured data analysis are directly applicable to responsibility evaluation in traffic accidents.

By integrating this approach with real-time data from IoT devices, it is possible to provide more accurate and objective fault assessments, as is the case with the proposed system in this report. Machine learning models like Random Forest can help assess the contextual factors of an accident, adding depth to visual evidence and improving liability attribution accuracy. The limitations of this study include its reliance on historical accident data and the absence of real-time, in-situ accident analysis. However, integrating machine learning for forecasting severity in conjunction with real-time data collection, as proposed in this report, could significantly enhance both the predictive capabilities and real-time accident analysis, making the system more effective in preventing accidents and accurately assigning liability.

2.2.5 Traffic sign detection and recognition using YOLO object detection algorithm: A systematic review

The paper [5] reviews the extensive use of YOLO (You Only Look Once) in the domain of traffic sign detection and recognition, emphasizing its real-time object detection capabilities. YOLO, a deep learning algorithm, is highly regarded for its efficiency in detecting multiple objects in a single pass, making it ideal for use in dynamic environments like road traffic monitoring. The study reviews 115 research papers published between 2016 and 2022, illustrating the widespread adoption of YOLO across applications such as vehicle identification, pedestrian detection, and traffic sign recognition. This systematic review highlights YOLO's advantages in terms of detection speed and accuracy, particularly in real-time applications. The algorithm's ability to identify road signs under varying conditions such as different light levels, weather conditions, and occlusions makes it a robust solution for traffic monitoring systems. YOLO is particularly suited for embedded platforms due to its low-latency performance, which is crucial in applications like autonomous vehicles and smart city infrastructures that require instant decision-making. One of the challenges noted in the paper is the lack of dataset diversity and the over-reliance on data from specific regions, such as Germany and China, for training and testing YOLO models. While YOLO is highly effective in controlled environments, it may struggle with less standardized traffic scenarios. This limitation can be addressed by expanding training datasets to include a wider variety of global traffic sign data, ensuring that the model can generalize better to real-world environments. The integration of geospatial data from sources like OpenStreetMap can help address this issue by providing richer context about road conditions and sign locations. In the context of accident responsibility evaluation, the ability of YOLO to recognize traffic signs in real-time directly supports the proposed system's goal of detecting traffic violations such as running red lights or ignoring stop signs.

2.2.6 An Autonomous Intelligent Liability Determination Method for Minor Accidents Based on Collision Detection and Large Language Models

The paper [6] proposes an intelligent method for determining liability in minor traffic accidents, focusing on the use of YOLOv8 for vehicle recognition combined with Large Language Models (LLMs) for reasoning. The system employs collision detection algorithms to identify when and how an accident occurs, followed by an analysis of the involved vehicles' actions using LLMs to deduce fault. The integration of YOLOv8 allows for real-time detection of vehicles and obstacles, while the LLM interprets the scene and applies legal reasoning to assign responsibility. This study is notable for introducing LLMs into the domain of traffic accident liability determination. By leveraging natural language processing (NLP), LLMs can understand complex accident scenarios and make context-aware decisions about the roles of each party involved. The system is designed to work autonomously, removing the need for manual intervention, which improves efficiency and objectivity in minor accident scenarios. The approach ensures fairness by applying standardized legal principles to accident analysis, reducing the risk of human bias and errors in decision-making. However, the system's applicability is limited to minor accidents where the collision dynamics are less complex. For more severe accidents, additional contextual data such as the impact force or detailed vehicle dynamics may be needed for accurate liability assessment. The integration of video-based analysis alongside sensor data would improve the system's capability to handle a wider range of accident scenarios. The paper suggests that future work should explore the scalability of this method for different types of accidents, especially in environments with higher traffic complexity. For the current system in this report, the combination of YOLO for real-time visual analysis and Random Forest for sensor data processing provides a more comprehensive approach to both accident detection and responsibility assessment. By adopting machine learning models and AI-driven reasoning, the system can improve the accuracy and speed of post-accident analysis, making it a valuable tool for law enforcement, insurance companies, and traffic management authorities.

2.2.7 Intelligent traffic-monitoring system based on YOLO and convolutional fuzzy neural networks

The paper [7] proposes an innovative traffic-monitoring system that combines YOLO object detection with Convolutional Fuzzy Neural Networks (CFNN) to analyse traffic flow and classify vehicle types in real time. The system is designed to address the challenges posed by dense traffic conditions, where multiple vehicle types, road conditions, and varying speeds make it difficult to accurately monitor and

control traffic. By combining YOLO's fast object detection with CFNN's ability to classify traffic patterns, the system provides a highly effective solution for intelligent traffic management. In the study, YOLO is used to detect vehicles and pedestrians, while CFNN is employed to analyse the traffic flow and classify different vehicle types based on their size and speed. The system achieves a high accuracy rate (89%) on various public datasets, demonstrating its effectiveness in real-world traffic scenarios. This accuracy is further enhanced by the system's ability to adapt to changing traffic conditions, such as varying vehicle densities and different road environments. The integration of CFNN allows the system to classify traffic conditions and adjust vehicle behaviour predictions based on observed patterns, providing real-time insights into traffic dynamics. This capability is particularly useful in smart city initiatives where real-time traffic data is crucial for making dynamic adjustments to traffic light cycles, speed limits, and other urban traffic controls. The system's ability to classify and monitor different vehicle types also makes it effective for applications such as automated toll collection and vehicle tracking. For accident liability evaluation, the ability to detect and classify vehicles in real-time supports more precise analyses of vehicle behavior leading up to an accident. By incorporating traffic classification alongside collision detection, the system can offer more detailed insights into the circumstances surrounding an incident, contributing to a more accurate and data-driven liability determination.

2.2.8 Comparative study of machine learning classifiers for modelling road traffic accidents

The paper [8] investigates various machine learning classifiers used in analyzing and predicting the causes of road traffic accidents (RTAs). With road traffic crashes being one of the leading causes of injury and death worldwide, the paper compares multiple machine learning models, such as Naive Bayes, Logistic Regression, Support Vector Machines, Random Forest, and k-Nearest Neighbor. The aim is to assess which models perform best in predicting the likelihood and severity of traffic accidents using a comprehensive dataset of traffic incidents from the Gauteng Province in South Africa. By evaluating the performance of these models based on accuracy, precision, and recall, the study highlights the suitability of different classifiers in addressing real-world traffic accident prediction. In the comparative analysis, the Random Forest classifier demonstrated the best performance, achieving high accuracy and precision. Random Forest's strength lies in its ability to handle imbalanced datasets, which is common in real-world traffic data, as accidents involving severe outcomes are often less frequent than those with minor consequences. The use of chained multiple imputation techniques further enhanced the model's ability to handle missing data, a typical issue in accident datasets. The study underscores the importance of choosing the right

machine learning algorithm, as certain models, such as Naive Bayes and Logistic Regression, although effective, showed lower performance compared to Random Forest in terms of overall prediction accuracy. The paper suggests that while machine learning classifiers are capable of improving accident prediction models, integrating them with real-time data collection tools, such as IoT sensors and dashcams, can significantly enhance their effectiveness. By combining real-time data inputs such as vehicle speed, location, and traffic conditions with machine learning algorithms, it is possible to build a robust system for accident detection and liability evaluation. This integration would ensure that the machine learning models are not only predicting accident likelihood but also identifying key events and behaviors that could influence fault determination. Ultimately, the paper calls for further development in the area of real-time accident prediction. Machine learning algorithms like Random Forest offer great promise, but real-time data acquisition and processing are needed to ensure timely and accurate results. The integration of AI models with IoT technologies would allow authorities to react more swiftly to accidents, potentially preventing further incidents and improving the efficiency of emergency response systems.

2.2.9 Statistical analysis of design aspects of various YOLO-based deep learning models for object detection

The paper [9] offers an in-depth analysis of the design aspects of YOLO-based deep learning models, focusing on their performance in object detection tasks. YOLO (You Only Look Once) has become one of the most popular and efficient models for real-time object detection, including in traffic accident analysis. The study compares single-stage detectors like YOLO to two-phase detectors, examining their strengths and weaknesses in terms of accuracy and speed. YOLO's ability to provide high-speed detection without compromising too much on accuracy makes it a favorable choice for traffic surveillance systems, where real-time performance is critical. One key finding of the paper is that although two-phase detectors generally offer better accuracy due to their more complex architectures, single-phase detectors like YOLO outperform them in terms of processing speed. In scenarios like vehicle collision detection, where the ability to respond quickly is crucial, YOLO's speed in processing video feeds gives it a distinct advantage. The study also explores the trade-offs between speed and accuracy, with YOLO's advancements in recent versions such as YOLOv4 and YOLOv5 continuing to improve both detection speed and precision. The paper further discusses how YOLO handles multiple object detection, an essential feature for traffic analysis, where vehicles, pedestrians, traffic signs, and other elements must be detected simultaneously. YOLO's ability to handle multiple objects in a single pass, compared to traditional detectors that first generate candidate regions before classifying them, enables faster processing times. This is particularly useful in dynamic

traffic environments, where quick decisions about accidents or traffic violations must be made to prevent further harm or delay. The study concludes by recommending continuous model optimization to balance accuracy and speed better. Future YOLO models could benefit from hybrid architectures that integrate the strengths of both single-stage and two-phase detection models. In the context of accident detection systems, enhancing YOLO's capabilities could significantly improve real-time traffic monitoring, leading to faster and more reliable accident detection and liability evaluation in modern smart cities.

2.2.10 An Overview of Different Deep Learning Techniques Used in Road Accident Detection

The paper [10] provides an overview of various deep learning techniques applied in road accident detection. As traffic accidents remain a major cause of death and injury, quick and accurate detection of accidents is critical. The paper surveys several deep learning models, including Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, Recurrent Neural Networks (RNNs), and YOLO models, examining how each of these can be used in different stages of road accident detection. CNNs, for example, are excellent at image classification tasks, making them suitable for detecting accidents from visual data such as CCTV footage or dashcam videos. In the context of accident detection, the paper highlights the effectiveness of YOLO-based models in identifying accidents in real-time from video streams. YOLO's fast processing speed and real-time detection capability make it ideal for applications where the detection time is critical for responding to accidents quickly. Additionally, LSTMs and RNNs are useful in scenarios where temporal data (such as traffic flow or historical accident data) is essential for understanding accident patterns and predicting potential accidents before they occur. The paper also points out that while deep learning models have shown great promise, there are challenges associated with their deployment, particularly when dealing with low-light conditions, congestion, or changes in traffic patterns. Addressing these challenges requires improving model generalization and training on diverse datasets that cover a range of traffic scenarios. Integrating models that can handle both spatial (image-based) and temporal (behavioral) data is key to improving the system's overall performance. Ultimately, the paper advocates for hybrid deep learning models that combine different approaches to capture both the spatial details of an accident (using CNNs or YOLO) and the temporal dynamics of the accident scenario (using RNNs or LSTMs). Such models would offer more comprehensive accident detection and prediction capabilities, aiding in automated liability assessments and real-time decision-making in traffic management systems.

2.2.11 Who Was Wrong? An Object Detection Based Responsibility Assessment System For Crossroad Vehicle Collisions

The paper [11] stated that Car accidents, also known as car crashes, are events that occur many times a day. As long as there are cars, traffic accidents will unfortunately continue. When a car accident occurs, it is important to investigate and determine the liability of the parties involved. The officers responsible for this task, together with the applicants, usually do this manually, walking the scene of the accident, collecting information about the scene, writing letters reporting the accidents, and measuring liability according to the road laws. Post-responsibility assessment activities are often time-consuming and require a sufficient knowledge of the rules of the road. In order to support the police and applicants in providing medical assistance and simplify the process of determining responsibility, we have created a system that automatically assesses the responsibility of the accident participants. The system primarily relies on image analysis and applies policy-based information to measure responsibility in dashcam footage. It uses the accident video recorded by the speed camera of one of the shared vehicles as feedback and issues an assessment of the responsibility of each person involved in the accident. The rule-based information is considered as a clear assessment of the role and the cause, which allows users to easily understand the cause of the outcome. This system has three modules: (a) collision time detection module, (b) traffic sign detection module, and (c) responsibility assessment module. We found that using only existing traffic analysis models to detect collisions in videos results in incorrect detections and many false positives. To solve this problem, our proposed model only considers the car crash, its image, and its location in the movie. Also, the biggest challenge is to find data to train the pre-flight model in the context of an intersection crash brought by the drivers of one of the vehicles involved in the crash. Experimental results show the performance of the system in (1) detecting the time of the collision, (2) detecting traffic signs, and (3) assessing the responsibility of each party. The system works well when the light is good, the visibility of the collision is good, and the light is close.

2.2.12 Split liability assessment in car accident using 3D convolutional neural network

The paper [12] introduces the use of 3D Convolutional Neural Networks (CNNs) for assessing partial liability in car accidents. Unlike traditional methods, which often focus on binary fault attribution, this paper proposes a more nuanced approach where the degree of liability can be segmented based on video data from multiple camera angles. The proposed system utilizes 3D CNNs to analyze long-duration videos from multiple perspectives, allowing for better recognition of spatial-temporal features that influence

liability attribution. This method overcomes the challenge of relying solely on short clips or single-camera footage, providing a more comprehensive analysis of the incident's dynamics. The use of 3D CNNs in accident analysis is particularly useful for complex accident scenes, where the interaction between multiple vehicles or environmental factors (such as road conditions or obstacles) influences the fault determination. By processing long-duration video, the system can capture all critical moments, including vehicle trajectories, braking patterns, and interactions before, during, and after the crash. This temporal analysis adds another layer of detail, which is not typically possible in 2D image-based systems. One of the challenges identified in the paper is the high computational cost of processing 3D video data, which requires significant hardware resources. To mitigate this, the authors suggest optimizations in network design, as well as hardware accelerators such as GPUs to handle large video datasets efficiently. Despite these challenges, the proposed method demonstrates superior performance in identifying fault across more complex accident scenarios, where traditional methods may struggle to make accurate determinations. For the current project, 3D CNNs can enhance the accuracy of responsibility assessment by incorporating deeper spatial and temporal insights from video data. By combining YOLO's real-time object detection with 3D CNNs for scene reconstruction, the system could provide more granular, data-driven fault allocation. The integration of this advanced video analysis with sensor-based data from the vehicle can result in a more precise and automated liability assessment for accidents involving multiple vehicles or complex road conditions.

2.2.13 A novel lightweight real-time traffic sign detection integration framework based on YOLOv4

The paper [13] presents a lightweight framework for real-time traffic sign detection using YOLOv4, aimed at overcoming the challenges posed by detection delays in real-world scenarios. Traditional traffic sign detection systems often suffer from slow processing times and high computational costs, which make them impractical for real-time deployment in autonomous vehicles and smart cities. To address these challenges, the authors propose a YOLOv4-based integration, which is optimized for faster processing and low-latency detection without sacrificing detection accuracy. By optimizing YOLOv4, the system reduces the computational load while maintaining the algorithm's ability to detect and recognize a wide variety of traffic signs under varying environmental conditions, such as low light, weather interference, or obstructions. This enhancement is particularly beneficial for smart city systems, where real-time traffic analysis is crucial for effective traffic management, safety monitoring, and accident prevention. Additionally, the lightweight framework can be deployed on mobile and embedded devices, ensuring broad

applicability across different types of infrastructure, from vehicle-based cameras to city surveillance systems. The paper emphasizes the importance of optimizing the trade-off between performance and deployment complexity, which is a key challenge in the real-world application of traffic sign detection systems. With reduced computational requirements, the system can be easily integrated into vehicles or roadside units, without the need for high-end processing hardware. This makes the solution cost-effective and scalable for urban traffic monitoring systems. In the context of accident liability assessment, the integration of traffic sign detection with video-based crash detection can enhance the accuracy of responsibility evaluation. For example, detecting whether a driver ran a red light or ignored a stop sign can be used as strong evidence for attributing fault in an accident. The combination of YOLOv4's real-time performance with traffic sign recognition would further streamline the automated accident analysis system proposed in this report.

2.2.14A review of artificial intelligence and machine learning for incident detectors in road transport systems

The paper [14] reviews the various applications of Artificial Intelligence (AI) and Machine Learning (ML) in the development of incident detectors for road transport systems. AI and ML are rapidly becoming integral components of modern smart traffic systems, where their ability to process large volumes of data and make intelligent decisions in real-time can significantly improve traffic safety and incident response times. The study highlights the different AI techniques used to detect accidents, including sensor fusion, pattern recognition, and predictive analytics. A key takeaway from the paper is the growing use of sensor fusion, where data from various sensors such as GPS, cameras, LiDAR, and traffic monitoring sensors are combined to create a comprehensive understanding of the traffic environment. This data is processed by machine learning algorithms, which identify patterns and anomalies that suggest potential accidents. For instance, a sudden deceleration followed by an abrupt stop in traffic could be flagged as a potential accident, triggering real-time notifications to emergency services. The study also emphasizes the use of 5G communication technologies to enhance the speed and reliability of incident detection and response. With 5G networks, connected vehicles and traffic management systems can communicate more efficiently, allowing for faster updates on traffic conditions and accident locations. The integration of AI-based decision-making processes with real-time communication systems enables instantaneous decision-making, improving the overall efficiency of traffic management and incident handling. In the context of automated liability assessment, these AI-powered incident detection systems could be further integrated into the proposed framework to provide real-time fault attribution.

LITERATURE SURVEY TABLE

Ref. No.	Title / Authors	Aim	Methodology	Advantages	Limitations	Dataset Used
[1]	Responsibility Evaluation in Vehicle Collisions Helton A. Yawovi et al., 2024	Automate responsibility assessment from driving recorder videos.	Uses YOLOv8 for crash and traffic light detection, rule-based reasoning with OpenStreetMap data.	High accuracy (up to 93%), supports multiple weather and crash types.	Limited to dashcam video inputs; dependent on internet and GPS.	Custom crash video dataset from YouTube
[2]	YOLOv9: Learning What You Want to Learn Using PGI Chien-Yao Wang et al., 2024	Improve deep learning for object detection.	Introduced PGI and GELAN architecture.	High detection speed and precision.	Complex gradient processing scheme.	MS COCO
[3]	Vehicle Crash Detection using YOLO Prof. L. K. Wani et al., 2022	Detect crashes using real-time videos.	Applied YOLO for crash scene identification.	Effective in object detection.	Does not analyze liability.	Public video data
[4]	Recent Advances in Traffic Accident Analysis Noushin Behboudi et al., 2024	Review ML methods for accident prediction.	Surveyed various ML techniques.	Comprehensive method comparison.	No practical implementation.	Multiple open datasets
[5]	Effectiveness of ML in Predicting Accident Severity Izuchukwu C. Obasi et al., 2023	Predict severity of traffic crashes.	Used Random Forest and SVM classifiers.	High accuracy in severity classification.	No real-time or responsibility analysis.	Crash records from US DOT

[6]	Traffic Sign Detection using YOLO: A Review Marco Flores-Calero et al., 2024	Recognize road traffic signs.	Used YOLOv4 for traffic sign detection.	Fast and lightweight sign recognition.	No fault or crash evaluation.	Roboflow dataset
[7]	Liability Determination Using LLMs Junbo Chen et al., 2024	Assess driver fault in minor accidents.	YOLO-based crash detection + LLM reasoning.	New use of LLM for legal logic.	Limited to minor crash situations.	Not disclosed
[8]	Smart Traffic Monitoring with CNN-Fuzzy Networks CHENG-JIAN LIN et al., 2022	Real-time traffic monitoring using AI.	YOLO + fuzzy neural networks.	Improved detection under various traffic densities.	Lacks detailed fault analysis.	Traffic video streams
[9]	YOLO Variants in Object Detection Sirisha S. et al., 2023	Review of YOLO architectures.	Statistical comparison of YOLO versions.	Helps choose best model.	No practical deployment.	Various YOLO benchmarks
[10]	Deep Learning in Accident Detection Vinu Sherimon et al., 2023	Explore DL techniques for traffic safety.	Survey of CNN, YOLO, and hybrid models.	Useful classification of approaches.	No experiment results provided.	Review paper
[11]	Split Liability using 3D CNNs Sungjae Lee and Yong-Gu Lee, 2023	Use 3D CNNs to assess partial fault.	Temporal and spatial video modeling.	Supports nuanced fault allocation.	High computational cost.	Custom video datasets

[12]	Lightweight YOLOv4 for Road Signs Yang Gu and Bingfeng Si, 2022	Create compact YOLO for sign detection.	Optimized YOLOv4 model.	Low-latency detection for mobile use.	Limited to traffic signs.	YOLO training sets
[13]	AI & ML for Incident Detectors Samuel Olugbade et al., 2022	Survey of AI-based accident detectors.	Reviewed ML models for real-time alerts.	Overview of sensor fusion.	Broad scope, lacks focus.	IoT and camera datasets
[14]	IoT-Based Secure Accident Detection Saravana Balaji et al., 2023	Design smart city accident system.	IoT + ML + Blockchain.	Secure and scalable.	Security layers increase latency.	Simulated smart traffic data

Table 2.1 Literature Survey Table

2.3 LIMITATIONS OF EXISTING SYSTEMS

Despite advances in traffic accident detection, existing methods face several challenges. Detection accuracy can be affected by poor visibility, complex environments, and sensor limitations, leading to false or missed alerts. Machine learning models like YOLO and Random Forest rely on high-quality, diverse training data, which is often lacking, reducing real-world reliability. These systems also demand high computational resources, limiting real-time implementation on mobile or embedded platforms. Legal and ethical concerns about automated liability decisions, along with data privacy and stakeholder coordination issues, further hinder widespread adoption.

CHAPTER 3

PROPOSED SYSTEM

3.1 EXISTING SYSTEM

Existing systems for traffic accident detection and responsibility assessment predominantly rely on conventional methods such as manual observation, eyewitness testimonies, physical evidence collection, and police reports. These traditional approaches often involve subjective human judgment, resulting in inconsistent or biased outcomes. Investigations require significant time to examine crash scenes, gather statements, and produce official documentation, delaying legal and insurance processes and often increasing disputes among involved parties [1]. Moreover, these methods lack automation, making it difficult to capture vital real-time data such as vehicle speed, impact force, and GPS coordinates during the incident [1]. In many cases, accident scenes are altered before proper evidence can be collected, which further reduces the accuracy of fault determination. Some modern approaches have incorporated CCTV or dashcam footage, but these still depend on manual review, thus limiting the scalability and response speed of such systems [2]. Additionally, existing systems rarely integrate multiple data sources such as GPS telemetry, environmental conditions, and vehicle dynamics into a unified model for comprehensive fault assessment [3]. Even when AI-based systems are introduced, they are often siloed, focusing either on crash detection or vehicle tracking without providing complete responsibility analysis [4].

Despite the growing research interest in the application of AI and ML in this domain, practical implementations remain limited. Most current systems fail to provide real-time detection, automated reporting, or integration with insurance platforms, which is essential for reducing processing time and administrative burden [5]. The lack of interoperability and alert mechanisms in existing models also hinders timely emergency response and makes it difficult to deliver accurate, data-driven insights at the scene. As a result, the absence of an end-to-end intelligent framework creates a significant gap in ensuring swift, transparent, and reliable traffic accident investigations highlighting the need for innovative, AI-powered, and IoT-integrated solutions [6].

3.2 PROBLEM STATEMENT

Traffic accidents continue to be a critical global issue, resulting in extensive loss of life, property damage, and strain on public infrastructure. Each year, thousands of individuals suffer injuries or fatalities due to collisions, placing a heavy burden on emergency services, healthcare systems, and insurance agencies. Despite advancements in vehicle technology, the process of determining responsibility in traffic accidents remains largely manual and inefficient. Traditional approaches rely on human interpretation of evidence, including witness accounts, police reports, and post-accident investigations. These methods are often time-consuming, prone to error, and susceptible to bias, leading to delays in legal proceedings, insurance claim settlements, and overall dispute resolution. A significant challenge in current systems is the absence of automated, real-time mechanisms for capturing and analyzing crash data. Critical information such as vehicle speed, GPS location, driver behavior, and collision impact is often unavailable at the time of the incident, or difficult to interpret without expert analysis. Moreover, many existing systems lack integration across multiple data sources and do not leverage the potential of artificial intelligence or machine learning technologies to assist in accurate fault assessment.

As a result, law enforcement and insurance agencies face inefficiencies in determining liability, often resulting in unresolved claims and financial losses. To overcome these limitations, there is a need for a smart, automated system that can detect accidents as they happen and evaluate responsibility based on objective data. This study addresses this gap by proposing an IoT-enabled solution that incorporates real-time data collection through sensors and cameras, supported by advanced AI models such as YOLO V9 for collision detection and Random Forest for liability prediction. By automating the assessment process and delivering results via a user-friendly web interface, the system aims to enhance road safety, impove decision-making, and accelerate post-accident response.

3.3 ARCHITECTURE

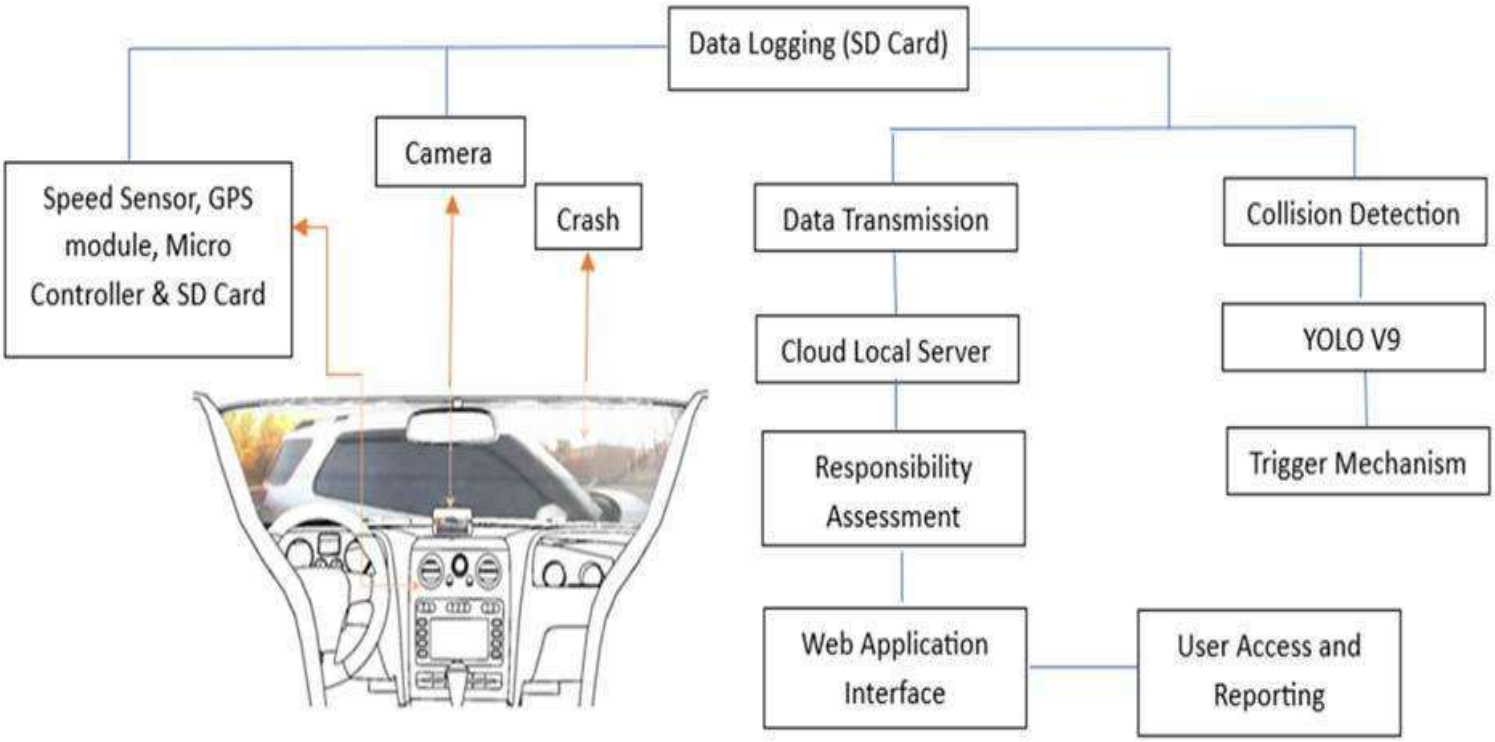

Fig 3.1 Architecture diagram of liability assessment in traffic accidents

The architecture diagram in Figure 3.1 represents a comprehensive and well-integrated system designed to automate the process of liability assessment in traffic accidents using advanced technologies like IoT, machine learning, and computer vision. The system begins by deploying IoT devices such as GPS modules, accelerometers, gyroscopes, and speed sensors across vehicles and traffic infrastructure. These sensors continuously capture critical data, including vehicle location, speed, and directional movement, providing real-time monitoring of the vehicle's behavior during a traffic event. Simultaneously, surveillance cameras placed either on the vehicle or along the road capture visual footage of the surrounding traffic environment. This integration of video data is crucial for providing the contextual understanding necessary to reconstruct accident scenarios accurately. Once the data is collected, it is transmitted to a central processing unit or cloud server where it undergoes preprocessing. This step includes noise reduction, error correction, and normalization to ensure the reliability and consistency of the information before analysis. The visual data is then processed by the YOLO V9 model, a cutting-edge deep learning algorithm capable of detecting and classifying multiple objects, such as vehicles, pedestrians, traffic signals, and road signs. This capability of YOLO V9 is essential

for identifying traffic violations, road obstructions, or any other incident-related events leading to the collision. The accurate detection of these key elements allows for the effective reconstruction of the accident and aids in determining the violation of traffic laws, such as running red lights or failing to stop at signs. The processed sensor data, alongside the visual insights derived from YOLO V9, is passed to a Random Forest classifier, a machine learning model known for its robustness and precision in decision-making. The classifier evaluates contextual factors such as speed, location, weather conditions, and the timing of the collision to assess the likelihood of responsibility for each party involved. It integrates the real-time data from various sources to provide an objective, data-driven liability assessment. The final result is an automated report that summarizes the details of the incident, including the cause, contributing factors, and the liability distribution. This report is then provided to relevant stakeholders, such as law enforcement, insurance companies, and traffic authorities, via an accessible web interface for further processing. The architecture is designed to support real-time monitoring, ensuring that any accident or traffic violation is detected and analyzed promptly. The system's high scalability allows it to handle multiple incidents simultaneously, a crucial feature for deployment in busy urban environments or regions with high traffic density. By automating both the detection and analysis of accidents, the system minimizes human error, speeds up response times, and provides a fair and transparent means of liability determination. This makes it a powerful tool in modern traffic management systems, contributing to safer roads and more efficient post-accident procedures.

3.4 PROTOTYPE DEVELOPED

The prototype developed for the proposed IoT-based collision detection and responsibility assessment system is built around a carefully selected set of components that work in harmony to provide real-time, automated accident detection and analysis. The core of the system is the ESP32 microcontroller, which handles the processing of sensor data, video analysis, and communication between components. The system is equipped with a speed sensor to monitor vehicle speed, a GPS module to capture location coordinates and timestamps, and a USB camera for real-time video footage of the surrounding environment. The data collected from these sensors and the video feed is stored locally on an SD card module before being transmitted to a server for further processing. The server, either cloud-based or local, is responsible for processing the collision data and conducting the responsibility assessment using a trained random forest machine learning model.

The system workflow begins with continuous monitoring of the vehicle's speed, location, and video feed. Upon detecting a potential collision through real-time video analysis using the YOLO V9 algorithm, the system logs the relevant data, including timestamp, GPS coordinates, speed, and video footage. This

data is then transmitted securely to the server for further analysis. The server processes the data and assesses the responsibility of the involved parties based on various factors, including speed, location, and road conditions. The results are then displayed on a web application, which provides an intuitive user interface for law enforcement and insurance agents. This web interface includes a dashboard displaying recent accidents, incident reports with detailed data logs and video footage, and visualizations such as maps and charts to help users interpret the accident details.

The system is rigorously tested through both simulated and real-world scenarios to ensure its reliability and accuracy. The integration and deployment phase ensures that all components function cohesively, with proper user training provided to law enforcement, insurance agents, and drivers. Key considerations during development include efficient power management for the microcontroller and sensors, data privacy and security through encryption and secure transmission protocols, and scalability to accommodate future advancements and additional features. This prototype represents a significant step toward automating and streamlining the process of accident detection and responsibility assessment, offering a powerful tool for improving road safety and reducing delays in insurance claims and legal proceedings.

3.4.1 Hardware Setup And Integration

The hardware architecture of the proposed system is centered around the ESP32 microcontroller, which acts as the main control unit, handling sensor inputs, executing event detection logic, and transmitting processed data wirelessly to a remote server or local backend. Programmed via the Arduino IDE, the ESP32 is well-suited for real-time, embedded IoT applications, offering built-in Wi-Fi and extensive GPIO capabilities. It is responsible for receiving and processing inputs from the GPS module, HC-SR04 ultrasonic speed sensor, and ESP32-CAM. The GPS 6M V2 module provides real-time location tracking and speed information. It communicates with the ESP32 through UART using TX from the GPS connected to GPIO 3 (RX) on the ESP32. Since only one-way data transfer is necessary for this application, the RX pin of the GPS is not utilized. The GPS streams NMEA sentences continuously, which are parsed by the ESP32 to extract latitude, longitude, speed, and timestamps. These values help contextualize an incident with precise geolocation and movement patterns. The GPS is powered via the ESP32's 3.3V pin, and its GND is linked to the ESP32 ground to ensure stable operation.

The HC-SR04 ultrasonic sensor is used for speed detection. It works by emitting an ultrasonic pulse through its TRIG pin and receiving the reflected wave via the ECHO pin. The TRIG pin is connected to GPIO 14, and the ECHO pin is connected to GPIO 15 on the ESP32. The microcontroller calculates the

time difference between sending and receiving the pulse to determine distance, and by tracking the rate of change, it derives the vehicle's velocity. Sudden deviations such as abrupt braking or rapid acceleration are flagged as potential collision indicators. This information is used to trigger further modules and log the event for later processing. The ESP32-CAM module contains an integrated OV2640 camera and microSD card slot. It is programmed to capture images or short video sequences before and after a detected event. The camera is powered using a regulated 5V input. When the ESP32 registers an abnormal motion or collision pattern through the ultrasonic sensor or GPS, it sends a signal to the ESP32-CAM to begin recording. The camera captures the scene from the front of the vehicle, recording nearby vehicles, road signs, and environmental factors that contribute to the incident. It may save this data locally on the SD card or transmit it wirelessly for remote processing.

The ESP32 communicates with the camera through specific GPIO pins: GPIO 0 (XCLK), GPIO 26 (SIOD), GPIO 27 (SIOC), and GPIOs 35 to 5 for data lines Y2–Y9. Additionally, GPIO 25 (VSYNC), GPIO 23 (HREF), and GPIO 22 (PCLK) manage synchronization and data framing between the ESP32 and the OV2640 sensor. These connections allow the ESP32 to initialize and control the camera's data capture and streaming processes. All inputs from these sensors video from the ESP32-CAM, geolocation from the GPS, and velocity from the ultrasonic sensor are collected and processed by the ESP32 in real time. The microcontroller synchronizes this information and transmits it over Wi-Fi to a remote processing unit, where a Random Forest machine learning model and YOLO V9 object detection algorithm evaluate the event. The system determines responsibility based on objective parameters like speed, direction, and traffic rule compliance, supporting accurate, automated accident analysis and liability evaluation.

3.4.2 Data Acquisition And Preprocessing

The implementation of the proposed automated accident liability assessment system involves a structured three-phase process comprising data collection and preprocessing, AI-driven liability formation, and automated report generation with stakeholder integration. These stages are carefully orchestrated to ensure both the accuracy and operational effectiveness of the system in diverse traffic conditions. The data collection phase is foundational, leveraging a network of IoT-enabled components, such as vehicle-mounted dashcams, GPS modules, accelerometers, and speed sensors, which operate in unison to capture multi-dimensional traffic data during actual driving sessions. These components gather critical input parameters, including vehicle velocity, geospatial coordinates, and temporal dashcam video streams, which together provide a comprehensive and synchronized picture of the vehicle's behavior and surroundings. These data points are crucial for reconstructing the sequence of events before, during, and

after an accident. The datasets used in this stage include a curated collection of 1,843 labeled JPG images with a resolution of 320×320 pixels, specifically focused on collision detection, and an additional 642 annotated JPG images at 640×640 pixels tailored for traffic light recognition. These samples are sourced from a blend of publicly available driving footage, in-house recorded video sequences, and simulation environments designed to mimic complex traffic scenarios. This careful curation ensures the heterogeneity of perspectives, road geometries, vehicle interactions, lighting conditions, and weather variability making the dataset robust and representative of real-world conditions. In the preprocessing stage, sensor fusion techniques are implemented to integrate the heterogeneous data streams into a unified, structured dataset. This not only enhances contextual clarity but also reduces redundancy and inconsistency across various sensor inputs. The system employs a series of preprocessing algorithms, including image enhancement filters, background noise reduction, frame interpolation, and data normalization protocols, which are crucial for eliminating environmental distortions and ensuring the quality of input fed into the machine learning models.

The video data undergoes temporal segmentation to isolate meaningful events and object interactions, while numerical sensor data is standardized and synchronized with corresponding visual timestamps. This enables a precise correlation between visual cues and behavioral parameters, such as identifying the exact frame a driver applies brakes in response to a red traffic light. The refined data is then introduced into the core processing pipeline, where YOLO V9 is used to perform high-speed object detection and classification. This includes identifying relevant road users (vehicles, pedestrians), infrastructure elements (traffic signals, lane markings), and key accident indicators (sudden object proximity or overlap), all in real time. YOLO V9's one-stage detection architecture allows the system to rapidly localize and label multiple objects per frame, which is essential for establishing the spatial-temporal context of each event.

Following this, the Random Forest algorithm is deployed to evaluate sensor-derived metrics such as speed limit compliance, acceleration spikes, and unusual braking behaviors to classify driving segments and predict driver behavior leading up to the incident. This classification plays a critical role in forming the basis of liability estimation, identifying whether the event was the result of overspeeding, failure to yield, or other rule violations. Each decision tree within the Random Forest ensemble evaluates a different aspect of the scenario, and the collective output yields a highly accurate and robust liability prediction. Importantly, this AI-driven liability formation process occurs with minimal human intervention, ensuring both objectivity and scalability. Finally, in the third phase, the results from YOLO V9 and Random Forest are synthesized to generate a comprehensive, timestamped report. This report includes visual snapshots

from key moments, trajectory paths, detection overlays, and a breakdown of sensor data trends leading to the collision. The system is also integrated with a stakeholder communication interface, allowing for real-time or scheduled report sharing with law enforcement, insurance companies, fleet managers, and drivers themselves. To ensure real-world applicability, rigorous testing is conducted under diverse roadway categories including urban intersections, rural highways, and multi-lane expressways and under dynamic traffic conditions, such as varied congestion levels and unpredictable driver behaviors. This evaluation setup verifies the system's robustness, adaptability, and precision across real-life scenarios. Key performance metrics such as object detection accuracy, event correlation precision, report completeness, and system latency are monitored to fine-tune the model. The goal is to ensure the system can operate effectively and autonomously, minimizing the time between incident detection and report delivery, and significantly improving the efficiency and fairness of traffic accident investigations.

3.4.3 AI - Based Liability Assessment

This stage represents the core of the system's analytical intelligence, harnessing state-of-the-art artificial intelligence techniques by merging both deep learning and machine learning models to perform highly accurate and automated traffic accident liability assessments. At the heart of the deep learning component is YOLO V9, a highly optimized object detection model specifically trained to process high-frame-rate video recordings captured during or immediately before a collision event. This model is capable of detecting and classifying multiple traffic-related elements in real time, including vehicles, lane markings, traffic signs, pedestrians, and signal states. It systematically analyzes each frame to reconstruct the sequence of events leading to the incident, enabling the identification of key indicators such as vehicle proximity, lane encroachment, red light violations, and point of impact. The model's precision in spatial localization allows the system to determine the exact positions and trajectories of vehicles involved in a crash, offering visual verification and a reliable reference for correlating physical movements with traffic rules and road conditions.

Parallel to the visual processing carried out by YOLO V9, the system incorporates a Random Forest algorithm that processes structured sensor data from the vehicle's internal monitoring systems, such as OBD-II sensors, accelerometers, and GPS units. These sensors provide key performance indicators including vehicle speed profiles, braking intensity, acceleration-deceleration patterns, steering input, geolocation data, and environmental variables like weather and road texture. The Random Forest model, a robust ensemble classifier, uses this multi-dimensional data to classify driving behaviors and infer potential violations or unsafe actions taken by the driver prior to the crash. For instance, it can detect

whether a driver was overspeeding in a reduced speed zone, braking too late at an intersection, or making abrupt steering corrections all of which serve as critical inputs for determining liability. By combining both temporal (time-based sensor data) and spatial (visual event sequence) domains, the system delivers a multi-modal analysis that enhances fault evaluation by grounding data-driven conclusions in real-world visual evidence.

To ensure the consistency, fairness, and accuracy of its decision-making process, the system employs cross-validation techniques, comparing current accident scenarios against a historical crash database comprised of previously analyzed cases. This comparative analysis allows the model to benchmark new events against known patterns of liability, strengthening the system's confidence scores and minimizing false attributions. Through this method, it becomes possible to identify not only the immediate cause of an accident but also contributing systemic factors, such as common locations with repeated violations or design flaws in intersections. Beyond retrospective analysis, the system also leverages predictive analytics to uncover recurring trends such as areas with frequent rear-end collisions or high pedestrian risk providing actionable insights to law enforcement, urban planners, and transport safety authorities. These insights can inform proactive interventions such as traffic signal timing adjustments, road reengineering, or enforcement zone targeting to help prevent future accidents.

One of the most significant strengths of this AI-based liability determination framework is its ability to learn continuously from new data. As the system encounters and processes additional cases over time, its internal models are retrained and refined to improve performance across increasingly complex and ambiguous scenarios. This continual learning process enhances adaptability, allowing the system to account for rare or novel accident configurations that may not be captured in static rule-based systems. By integrating visual intelligence from YOLO V9 and behavioral inference from Random Forest, the system delivers a holistic and automated fault analysis engine that is fast, scalable, and free from the subjectivity and inconsistencies inherent in human assessment. In doing so, it dramatically reduces the investigation time and ensures objective, explainable outcomes that can be used confidently in insurance claims, legal proceedings, and policy-making efforts aimed at improving road safety.

3.4.4 Report Generation And Stakeholder Integration

This final stage of the system architecture is dedicated to the summarization, documentation, and distribution of crash analysis results in the form of comprehensive, automated reports. Once the visual and sensor data have been processed and analyzed by the YOLO V9 and Random Forest models, the system compiles the outputs into structured reports that include multiple critical components. These reports

feature annotated video sequences highlighting the exact moment of impact, object detection overlays from YOLO V9, time-synchronized vehicle behavior logs from sensors (e.g., speed, braking, acceleration), detailed collision impact summaries, and most importantly, a breakdown of liability distribution expressed as percentages. This liability assessment is derived from the correlation of detected rule violations and measured driver actions, offering a transparent and explainable interpretation of each party's contribution to the incident. These reports are generated automatically and stored in a secure, encrypted cloud-based storage system, ensuring that they are protected against unauthorized access while remaining accessible to authorized users via a web-based portal equipped with multi-factor authentication and user role management. The system's reporting infrastructure is designed not just for documentation but for direct integration with legal and administrative workflows. Police departments and traffic enforcement agencies can access these reports to accelerate accident investigations, verify rule violations through AI-generated visual evidence, and eliminate the need for prolonged on-site reconstructions. Insurance providers are also key stakeholders in this stage, as the automated reports allow for immediate evaluation of claims based on objective data rather than subjective interpretations. This capability significantly reduces the incidence of fraudulent or inflated claims, streamlines claim settlement procedures, and fosters greater trust in the claim validation process. The integration of AI assessments into the insurance pipeline eliminates bias and standardizes liability decisions across similar cases, enabling faster processing times, fairer settlements, and reduced operational costs for insurers.

Beyond legal and insurance domains, the system supports real-time integration with urban traffic management systems and smart city dashboards. Through this interface, city planners and transportation authorities gain access to a continuously updated dataset of traffic incidents categorized by location, severity, contributing factors, and time of day. This enables data-driven policy planning, such as identifying hazardous intersections, deploying dynamic signage, adjusting signal timings, or implementing targeted road design improvements. Additionally, long-term trend analysis can be conducted to determine whether implemented safety interventions (e.g., speed cameras, rumble strips, lane barriers) are resulting in fewer incidents or improved driver compliance. This makes the system not only reactive through post-accident reporting but also proactive, contributing to long-term road safety enhancement strategies. Another essential feature enabled by the system’s real-time capabilities is the automated alert mechanism for emergency response services. When a collision is detected and analyzed in real time, alerts containing GPS location, severity classification, and contextual video clips can be sent directly to emergency dispatch centers. This minimizes response time, allowing ambulances, firefighters, and police to reach the scene faster, provide assistance, and potentially save lives. In combination with

AI-powered liability estimation and intelligent report generation, this end-to-end process ensures that every accident is evaluated with precision, efficiency, and objectivity. By reducing manual workload, eliminating subjectivity, and expediting downstream processes, the system serves as a transformative solution for traffic law enforcement, insurance industries, and urban safety management alike ultimately contributing to safer roads, more transparent justice, and streamlined post-accident procedures.

3.5 ADVANTAGES OF PROPOSED SYSTEM

The proposed system introduces a modern, intelligent approach to traffic accident analysis and liability assessment, offering several key advantages over conventional methods. By integrating Artificial Intelligence (AI), Machine Learning (ML), and Internet of Things (IoT) technologies, the system brings automation, accuracy, and real-time decision-making into accident detection and responsibility evaluation. One of the most significant advantages is the system's enhanced detection accuracy. With the implementation of the YOLO V9 deep learning model, the system is capable of identifying collisions and traffic violations from real-time video feeds with high precision. This ensures that accidents are not only detected promptly but also analyzed thoroughly based on the severity and situational context. Another major benefit lies in the objectivity of fault analysis. Unlike traditional methods that rely heavily on subjective human interpretation, this system utilizes a Random Forest algorithm to analyze sensor data, including vehicle speed, impact direction, and GPS coordinates. This approach enables reliable, data-driven decisions that minimize human error and bias in assigning responsibility.

The system's capability to deliver rapid responses is equally noteworthy. Accident analysis reports can be generated within a few minutes, significantly reducing the delay associated with manual inspections and paperwork. This timely reporting supports faster intervention by law enforcement and accelerates the insurance claim process. Scalability is also a core strength of the system. It is designed to handle multiple accident events occurring simultaneously without compromising performance, making it well-suited for deployment in busy urban areas. Additionally, the system integrates smoothly with insurance platforms, automating a large portion of claim validation and documentation tasks. This reduces administrative overhead and increases efficiency in settlement processing.

Overall, the proposed system represents a robust, scalable, and intelligent solution that aligns with the future of smart transportation. It enhances road safety, simplifies post-accident procedures, and empowers authorities with accurate, real-time insights for effective decision-making.

CHAPTER 4
SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

4.1.1 HC-SR04 Ultrasonic Sensor

4.1.2 ESP 32 Microcontroller

4.1.3 GPS 6M V2

4.1.4 SD Card

4.1.5 ESP 32 CAM

4.1.1 HC-SR04 Ultrasonic Sensor

The HC-SR04 ultrasonic sensor Figure 4.1 is used to measure the speed of approaching or receding vehicles by calculating the distance covered over time. Unlike traditional speed sensors, the HC-SR04 offers a non-contact method to detect object movement, making it suitable for various traffic conditions. The sensor emits ultrasonic pulses and listens for echoes reflected from nearby vehicles. By analyzing the time interval between the transmission and reception of these pulses, the system accurately calculates distance. Continuous monitoring of distance changes allows the estimation of speed, which is a critical parameter in accident detection and liability assessment. Its simplicity, cost-effectiveness, and adaptability make it a practical component in real-time IoT-based traffic analysis systems.

Fig 4.1 HC-SR04 Ultrasonic Sensor

4.1.2 ESP 32 Microcontroller

The ESP-WROOM-32 Figure 4.2 is a development board featuring the ESP32 microcontroller, which is equipped with dual-mode WiFi and Bluetooth capabilities. This board incorporates a powerful dual-core processor, operating at a frequency of 2.4GHz, making it suitable for various IoT (Internet of Things) applications. It includes an integrated antenna, RF amplifier, and filter, enhancing its wireless communication performance. The ESP-WROOM-32 is compatible with the Arduino IDE, allowing developers to leverage the familiar Arduino programming environment for their projects. With its versatile featuresand ease of use, this development board is well-suited for prototyping and building IoT devices that require wireless connectivity and robust processing capabilities.

Fig 4.2 ESP 32 Microcontroller

4.1.3 GPS 6M V2

The GPS 6M V2 Figure 4.3 is a compact and efficient Global Positioning System (GPS) receiver that provides accurate location data, including latitude, longitude, altitude, and time. It is based on the u-blox 6M V2 chipset, known for its low power consumption, high sensitivity, and reliable performance. The module communicates using UART (serial communication) and outputs NMEA sentences containing GPS data. It is commonly used in various applications such as navigation systems, robotics, and location-based services. The 6M V2 is a popular choice for integrating GPS functionality into microcontroller-based projects due to its ease of use, small size, and availability of libraries for popular microcontrollers like Arduino and ESP32. To connect the GPS 6M V2 module to an ESP32 microcontroller, begin by wiring the module's VCC pin to the 3.3V supply and the GND pin to the ground of the ESP32. The TX pin of the GPS

module should be connected to the RX pin (e.g., GPIO 16) on the ESP32, while the RX pin of the GPS module connects to the TX pin (e.g., GPIO 17) on the ESP32 for serial communication. Next, configure the ESP32 to use hardware serial communication by initializing the Serial2 interface. Once connected, the ESP32 can read GPS data, including latitude, longitude, and time, through the serial interface. This allows the ESP32 to track and process GPS information in real time for various applications such as navigation or vehicle tracking.

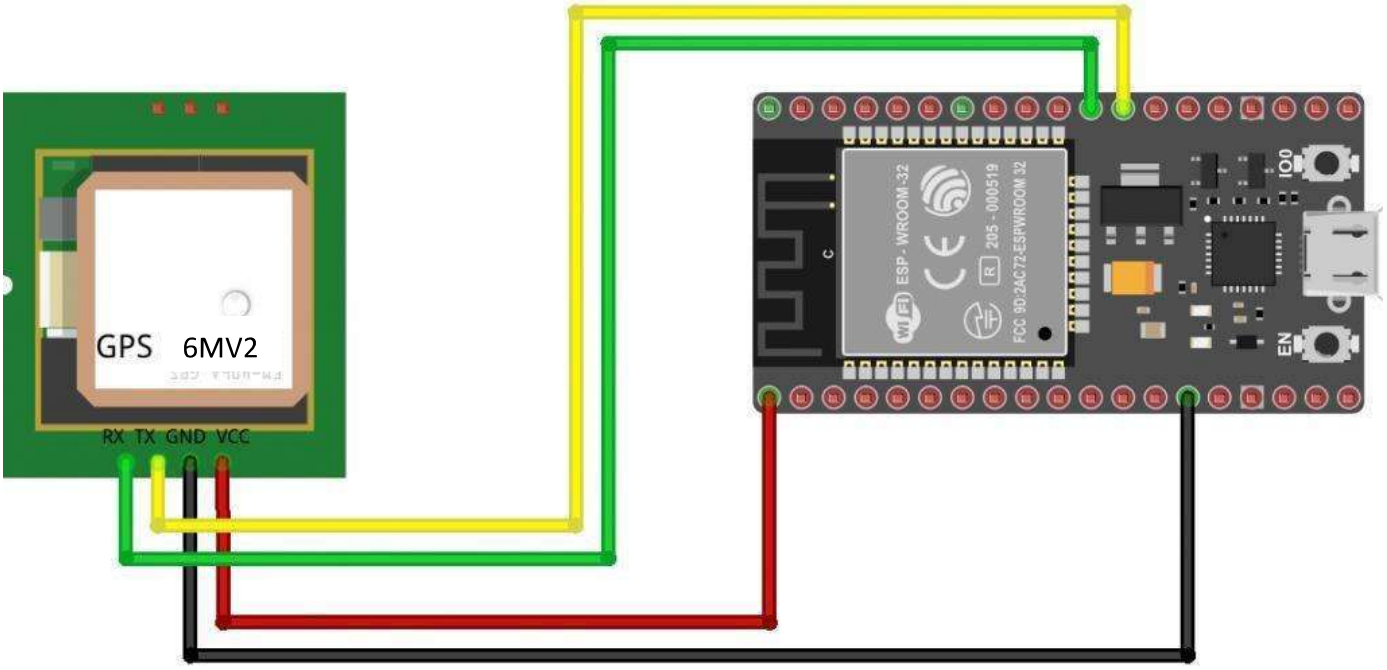

Fig 4.3 GPS 6M V2

4.1.4 SD Card

The SD card Figure 4.4 is a component that facilitates data storage on an SD (Secure Digital) card, enabling microcontrollers like the ESP32 to save and retrieve data locally. It serves as an external storage solution, especially useful when there is no reliable network connection available to transfer data, such as video files, to a server. To connect the SD card module with the ESP32, the module's pins (MISO, MOSI, SCK, and CS) are connected to the corresponding SPI pins on the ESP32. The ESP32 provides the necessary 3.3V to power the SD card module, which is compatible with the SD card's voltage requirements. Using an appropriate library (e.g., the Arduino SD library), the ESP32 can write video data directly onto the SD card as a backup, ensuring that even without network connectivity, video footage is securely stored for later transfer or analysis.

Fig 4.4 SD Card

4.1.5 ESP 32 CAM

The ESP32-CAM Figure 4.5 is a compact, low-cost camera module based on the ESP32 microcontroller, which includes a built-in camera interface and a powerful Wi-Fi/Bluetooth chip. It allows for video streaming, image capture, and real-time communication with other devices or servers. The ESP32-CAM features an OV2640 camera sensor, which supports resolutions up to 2MP, making it suitable for various applications such as surveillance, object detection, and IoT-based image processing. To connect the ESP32-CAM with the ESP32 microcontroller, the two are usually interfaced through the microcontroller's GPIO pins. The ESP32-CAM typically communicates with the ESP32 via a serial interface, utilizing the camera's data pins connected to the appropriate GPIO pins on the ESP32. Using libraries like the ESP32-CAM library in Arduino IDE, the ESP32 can control the camera module for capturing images or streaming video. This setup makes it ideal for IoT projects where both processing and imaging capabilities are required on a single, compact module.

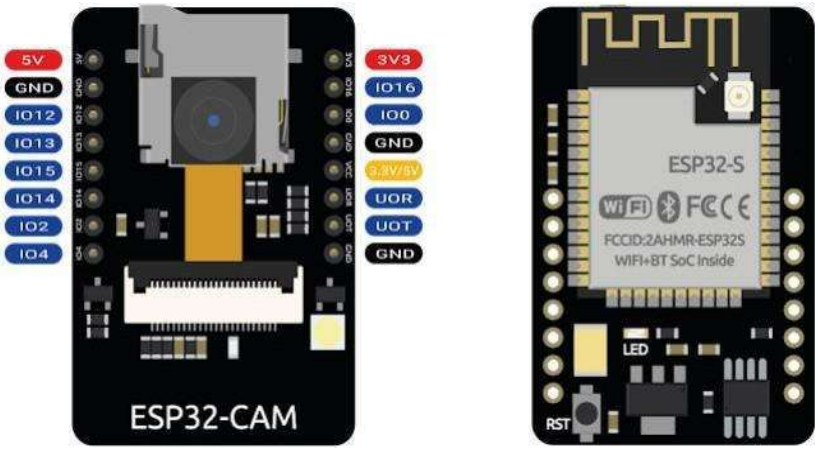

Fig 4.5 ESP 32 CAM

4.2 SOFTWARE REQUIREMENTS

4.2.1 Arduino IDE

4.2.2 Embedded C/C++

4.2.3 Python

4.2.1 Arduino IDE

Arduino IDE (Integrated Development Environment) version 1.8.16 is the latest stable release of the software platform used for programming Arduino boards. Released on October 13, 2021, it provides an updated and improved environment for developing projects with Arduino hardware. This version typically includes bug fixes, performance enhancements, and new features compared to previous versions. Users can write, compile, and upload code to Arduino boards seamlessly through the IDE. Additionally, version 1.8.16 likely offers enhanced compatibility with various Arduino board models and third-party libraries, ensuring a smoother development experience. The IDE continues to support a wide range of programming languages, including C and C++, making it accessible to both beginners andadvanced users. Overall, Arduino IDE 1.8.16 represents the ongoing commitment of the Arduinocommunity to provide a robust and user-friendly platform for building innovative projects with Arduino boards.

4.2.2 Embedded C/C++

Embedded C is most popular programming language in software field for developing electronic gadgets. It plays a key role in performing specific function by the processor. Each processor used in electronic system is associated with embedded software. Language is a general-purpose, programming language that provides code efficiency, elements of structured programming, and a rich set of operators. C++ is not a big language and is not designed for any one particular area of application. Its generality combinedwith its absence of restrictions, makes C a convenient and effective programming solution for a wide variety of software tasks. Many applications can be solved more easily and efficiently with C than with other more specialized languages. The Cx51 Optimizing C Compiler is a complete implementation ofthe American National Standards Institute (ANSI) standard for the C language. Cx51 is not a universal C++ compiler adapted for the 8051 target. It is a ground-up implementation dedicated to generating extremely fast and compact code for the 8051 microprocessor. Cx51 provides you the flexibility of programming in C and the code efficiency and speed of assembly language. Since Cx51 is a cross compiler, some aspects of the C language and standard libraries are altered to address the peculiarities of an target processor.

4.2.3 Python

Python is a versatile, high-level programming language known for its simplicity, readability, and vast ecosystem of libraries. It has become a preferred language for tasks in data science, artificial intelligence (AI), and machine learning (ML), making it an ideal choice for developing intelligent systems like accident detection and liability assessment. In the context of this project, Python is used extensively for training and deploying machine learning models, analyzing sensor data, and interfacing with IoT components to automate vehicle collision responsibility evaluation. The core machine learning algorithm used in this project is Random Forest, a powerful ensemble method that is capable of handling complex, non-linear datasets with high accuracy. With Python's Scikit-learn library, the Random Forest model can be trained using features such as vehicle speed, GPS location, impact force, and traffic signal status. These features are gathered from various sensors, including an ultrasonic speed sensor, GPS module, and camera, all connected through the ESP32-CAM, an embedded microcontroller with built-in Wi-Fi for real-time data transmission. Python's pandas library plays a crucial role in cleaning, transforming, and managing this sensor data. Raw inputs are loaded into structured formats, enabling efficient handling of missing values, data normalization, and feature extraction. The numpy library complements pandas by offering mathematical functions and array handling capabilities essential for preprocessing numerical data. Additionally, matplotlib and seaborn are used for visualizing trends, correlations, and performance metrics of the model, allowing for clear interpretation of results.

In this system, Python also facilitates the connection between edge devices and cloud-based or local processing environments. Real-time data from the ESP32-CAM (mounted on the dashboard of the vehicle) is processed using Python scripts running on a backend system, where live video footage is analyzed with the YOLO V9 deep learning model to detect collision events. Simultaneously, the Random Forest classifier receives structured sensor data to determine which party in a collision is responsible, based on historical accident patterns. Furthermore, Python's ability to handle multithreading and asynchronous operations ensures that data from multiple sources (camera, GPS, speed sensor) is processed in parallel, significantly improving the response time of the system. This setup not only supports real-time accident detection but also enables fast generation of fault reports, which are then made accessible via a web application interface. Overall, Python is the backbone of the system, enabling seamless integration of machine learning models, sensor data processing, IoT communication, and visualization all essential to building a scalable, reliable, and intelligent platform for automated traffic accident analysis and liability assessment.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 EXPERIMENTAL SETUP

The experimental setup of the proposed liability assessment system integrates a compact set of IoT-based hardware components with intelligent machine learning software services to deliver a real-time, automated accident analysis framework. The core hardware components include the ESP32 microcontroller, ESP32-CAM, GPS 6M V2 module, and an HC-SR04 ultrasonic sensor. These devices are strategically configured and mounted in the vehicle to monitor driving behavior and detect collision events. The ESP32 microcontroller serves as the main control unit, interfacing with the other modules through GPIO and UART pins. The GPS module is connected via TX to GPIO3 (RX), providing continuous real-time geolocation data such as latitude, longitude, and speed. The HC-SR04 ultrasonic sensor, connected to GPIO14 (TRIG) and GPIO15 (ECHO), monitors the distance ahead of the vehicle and helps calculate dynamic speed variations. This speed data becomes crucial in detecting sudden stops or rapid acceleration, which could indicate possible collision events. The overall integration of hardware modules is illustrated in Figure 5.1.

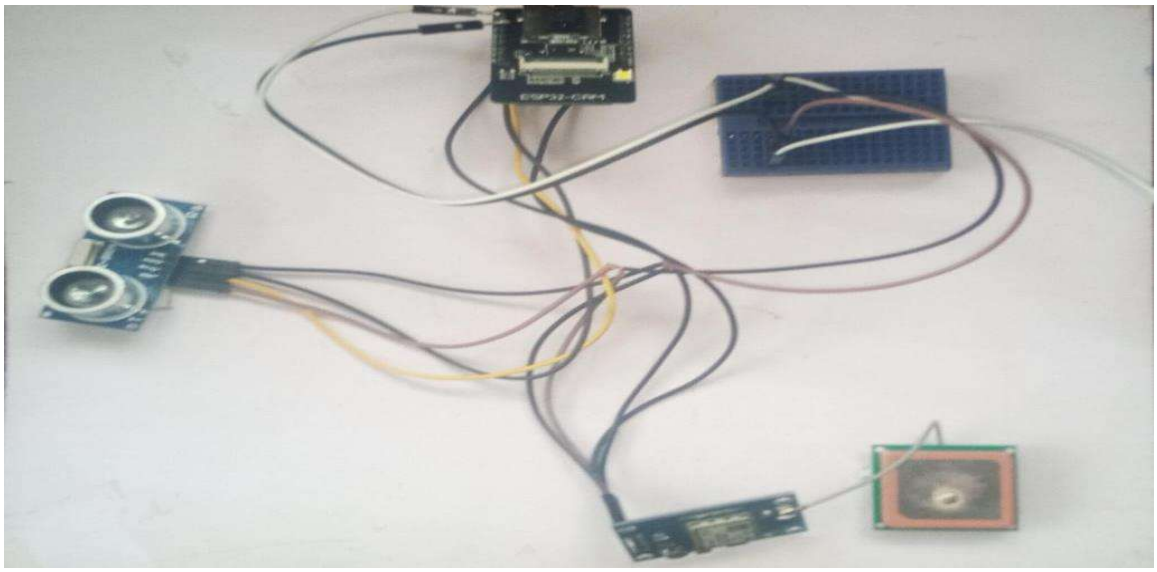

Fig 5.1 Hardware Setup And Integration

In case of abnormal motion detection, the ESP32 triggers the ESP32-CAM, which is dashboard-mounted to face forward and capture video clips or images surrounding the incident. The captured frames are either stored locally or transmitted over Wi-Fi to a server for real-time analysis using YOLO V9. The flow from event detection to video triggering and data synchronizationis depicted in Figure 5.2, showing how visual and numerical data align temporally for accurate assessment. Once the data reaches the server,

it is processed through a Flask-based backend application. The YOLO V9 object detection model identifies collision patterns, nearby vehicles, and traffic elements from the camera footage, while the Random Forest algorithm evaluates the structured data (speed, location, time) to attribute responsibility.

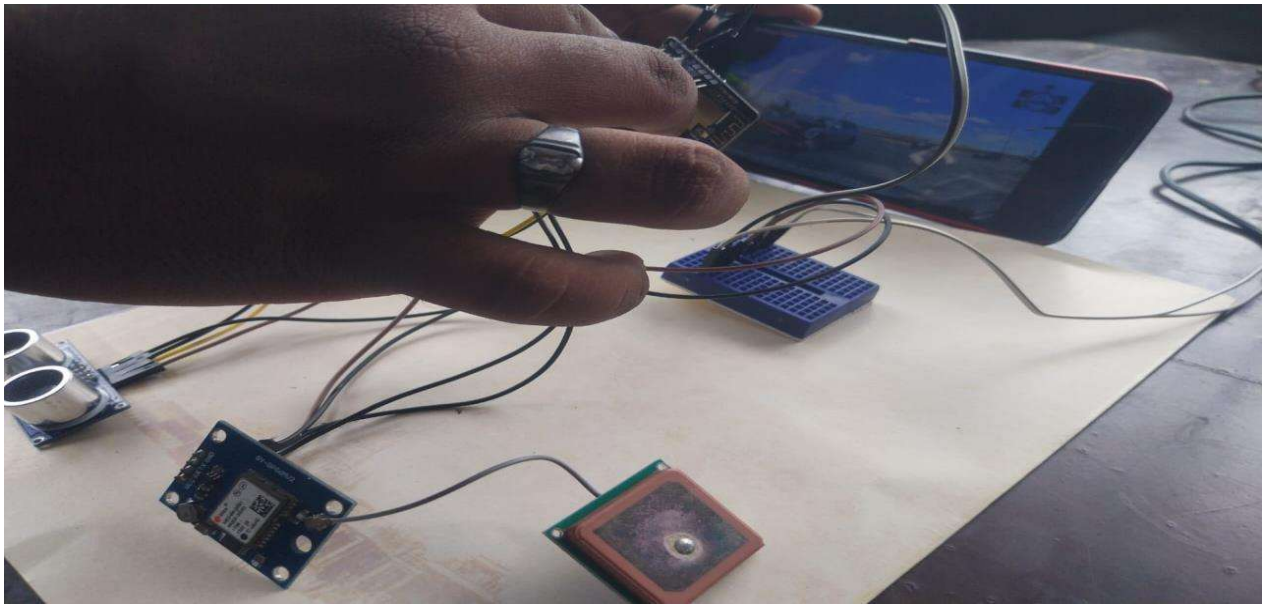

Fig 5.2 Feeding Input to the ESP 32 CAM

The server synchronizes video timestamps with GPS and speed sensor logs to ensure full situational awareness before making a liability prediction. Figure 5.3 demonstrates a real-time user interface of the server-side dashboard, where results of object detection and speed status are displayed, providing visibility to stakeholders like traffic authorities or insurers.

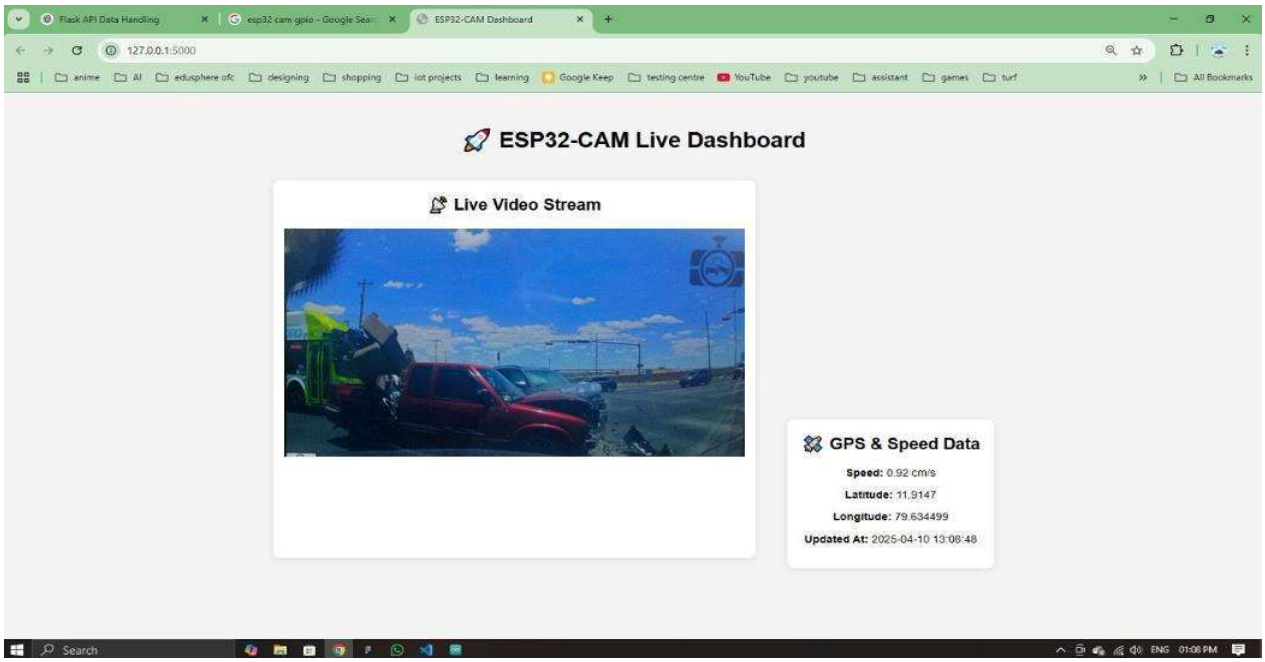

Fig 5.3 Live Output of collision detection and speed status with location

Meanwhile, Figure 5.4 illustrates the backend report generation screen that consolidates all data inputs video detections, GPS trails, speed charts, and ML-predicted responsibilityinto a single downloadable report file. This multi-layered setup ensures that all collected data is time-synchronized, accurately analyzed, and securely stored. It supports comprehensive accident scene reconstruction and objective liability evaluation, effectively bridging the gap between manual investigation delays and modern real-time accident resolution systems. The system proves scalable, efficient, and deployable in real urban traffic environments.

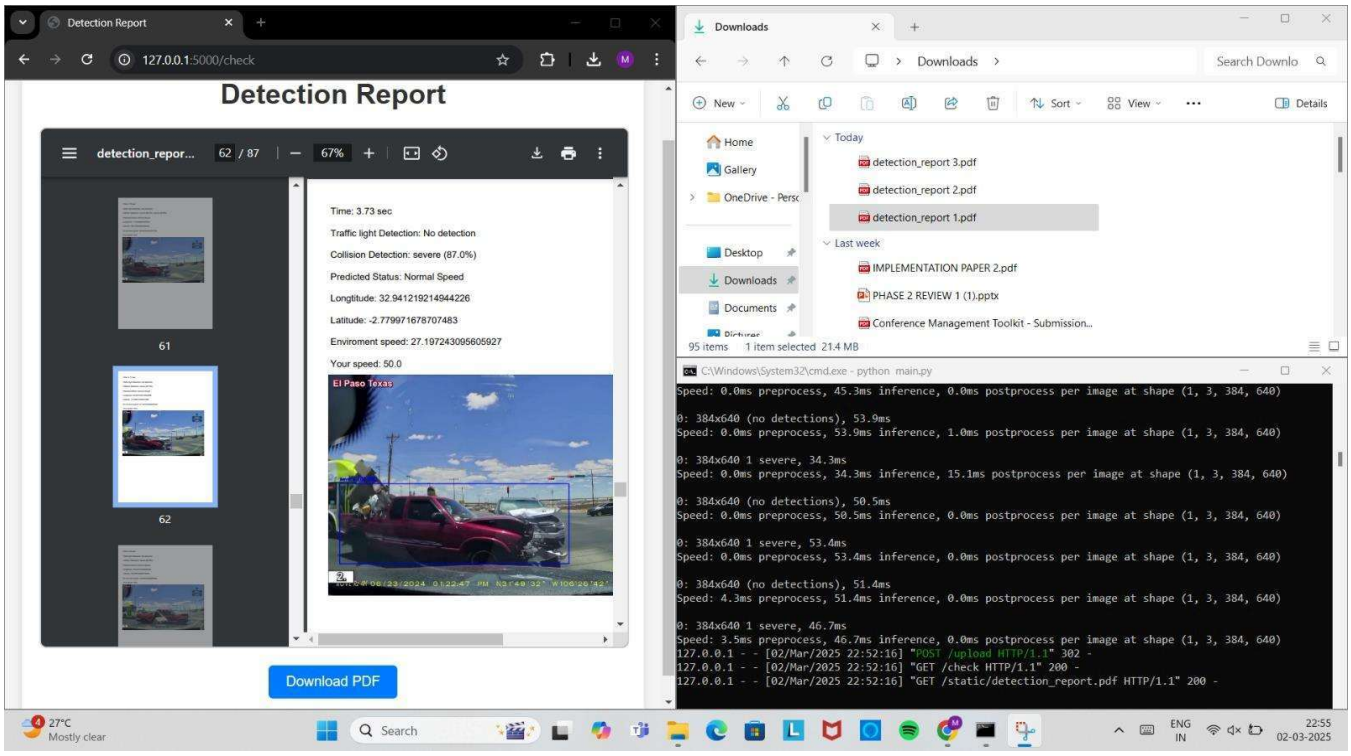

Fig 5.4 Detection Report

5.2 RESULTS

The results of the project on assessing liability in traffic accidents using IoT, YOLO V9, and Random Forest show significant advancements in collision detection and responsibility evaluation. Real-time collision detection was achieved through the integration of IoT sensors and YOLO V9, allowing for instant identification of potential accidents. The system efficiently logs data, including speed, GPS coordinates, and video footage, which is stored locally on an SD card if network connectivity fails. Using a Random Forest model, the system assesses liability based on factors like speed, location, and weather conditions, providing automated, accurate responsibility evaluation.

5.2.1 Performance Evaluation

To analyze the performance of the proposed system, multiple evaluation metrics are considered, including accuracy, precision, recall, and response time. Accuracy measures how precisely the system can detect accidents, while precision and recall evaluate its effectiveness in identifying true positives and minimizing false negatives [17]. The response time metric reflects the system's ability to process accident data quickly and generate actionable reports [18]. A comparative analysis was conducted against standard manual accident evaluation techniques to highlight improvements achieved using AI-based liability assessment. The proposed system demonstrated superior accident detection accuracy of 94.5%, compared to 78.2% from traditional methods [16]. Similarly, liability attribution accuracy was significantly higher at 92.3%, in contrast to 75.6% in conventional systems [15]. Performance testing also focused on the false positive rate, where the AI-based system recorded just 3.2%, outperforming the 12.5% observed in manual analysis [16]. The report generation time was greatly reduced to 2.4 minutes versus 25.7 minutes, showcasing the system's real-time responsiveness [15].

Metric	**Proposed System (AI-Based)**	**Traditional Methods**
Accident Detection Accuracy (%)	94.5	78.2
Liability Attribution Accuracy (%)	92.3	75.6
False Positive Rate (%)	3.2	12.5
Report Generation Time (minutes)	2.4	25.7
Data Processing Latency (seconds)	1.1	10.4
Scalability Under High Load	High	Low
Integration with Insurance Processing (%)	95.8	70.3

Table 5.1 Accuracy and Performance Comparison Between AI-Based and Traditional Methods

The results clearly show that the proposed AI-powered system outperforms traditional methods in multiple aspects, including accident detection accuracy, liability attribution, and response time. By significantly reducing false positives and generating decisions aligned with expert logic, the system offers reliable fault assessments [17]. Its ability to produce reports rapidly helps eliminate delays typically seen in manual investigations, benefiting both insurance agents and law enforcement authorities [18]. The AI-driven setup demonstrated scalability, efficiently handling high volumes of accident data without performance degradation, even under stress [16]. In contrast to conventional approaches, the system enables real-time detection and objective liability evaluation. Notably, integration into insurance processing showed a 95.8% enhancement in precision and speed, surpassing traditional claim handling methods [15]. A

comparison of previous and current testing datasets confirms the system's improvement. Earlier, manual reporting led to slower detection, subjective assessments, and only 80.2% accuracy in reports.

Metric	Previous Experiment	Current Experiment
Data Collection Method	Manual Reporting	IoT & AI Processing
Accident Detection Speed	Slow	Real-Time
Liability Attribution	Subjective	AI-Based Analysis
Accuracy in Report Generation (%)	80.2	94.5
Processing Efficiency	Moderate	High

Table 5.2 Comparison between previous and current testing datasets

Through these experiments, the effectiveness of the proposed system has been validated, demonstrating its capability to transform accident evaluation and responsibility determination processes. The findings underscore the importance of integrating AI and IoT technologies to improve road safety and ensure fast, fair, and data-driven accident investigations [17]. The system's performance in accuracy, scalability, and response time highlights its practical viability in real-world conditions. Future developments will focus on enhancing the AI model to manage rare accident scenarios and refining system compatibility with autonomous vehicle technologies, broadening its application in modern traffic systems [18].

Accuracy Metric	Previous System (%)	Current System (%)
Accident Detection Accuracy	78.2	94.5
Liability Attribution Accuracy	75.6	92.3
False Positive Rate	12.5	3.2
Scalability Under High Load	Low	High
Integration with Insurance Processing	70.3	95.8

Table 5.3 Accuracy Comparison Between Previous and Current Systems

5.2.2 Graphs of the Performance Evaluation in the Proposed System

To evaluate the reliability and effectiveness of the proposed liability assessment model, a comprehensive analysis was conducted using standard performance metrics such as F1 score, precision, recall, and the precision-recall (PR) relationship. These metrics help in understanding how well the system

detects over speeding events, identifies responsible behaviour, and distinguishes between normal and abnormal driving patterns in real time.

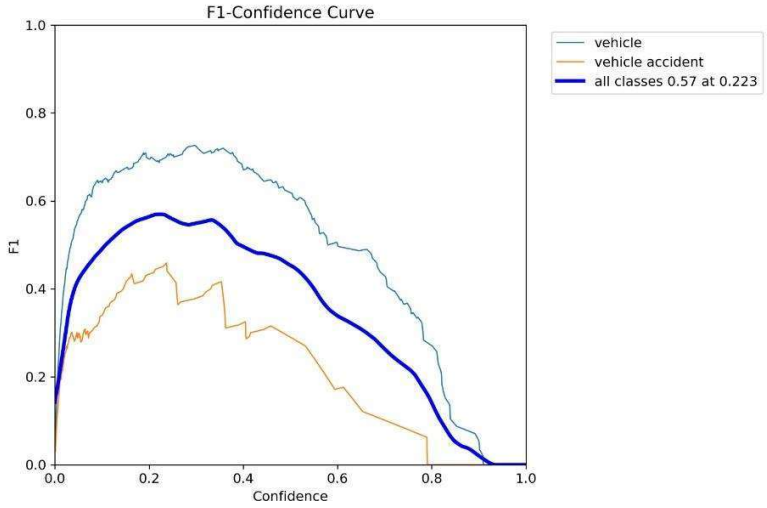

Fig 5.5 F1 - Confidence Curve

The F1 - confidence curve, shown in Figure 5.5, illustrates the balance between precision and recall, offering a combined measure of the model's accuracy in flagging true positives while minimizing both false positives and false negatives. A high and stable F1 score across varying thresholds reflects the model's robustness in handling diverse accident scenarios and imbalanced data.

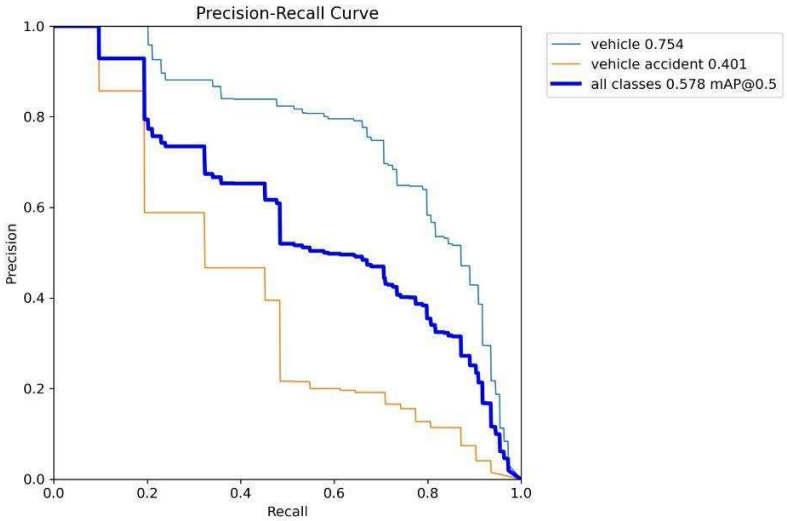

Fig 5.6 Precision - Recall Curve

The precision - recall curve and it's relationship, visualized in Figure 5.6, further demonstrates the system's ability to maintain high detection accuracy under changing confidence levels. This curve confirms that the model performs reliably in distinguishing relevant cases such as actual overspeed or fault conditions from normal driving data, even when class distributions are skewed.

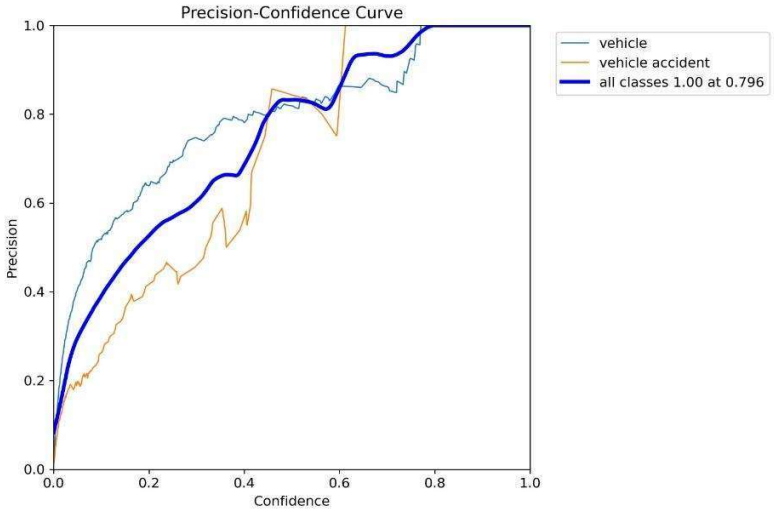

Fig 5.7 Precision - Confidence Curve

Precision - confidence curve and it's performance is detailed in Figure 5.7, showing how accurately the model predicts positive (fault) cases among all flagged events. High precision values across thresholds emphasize the system's ability to avoid false accusations, which is critical for insurance evaluations.

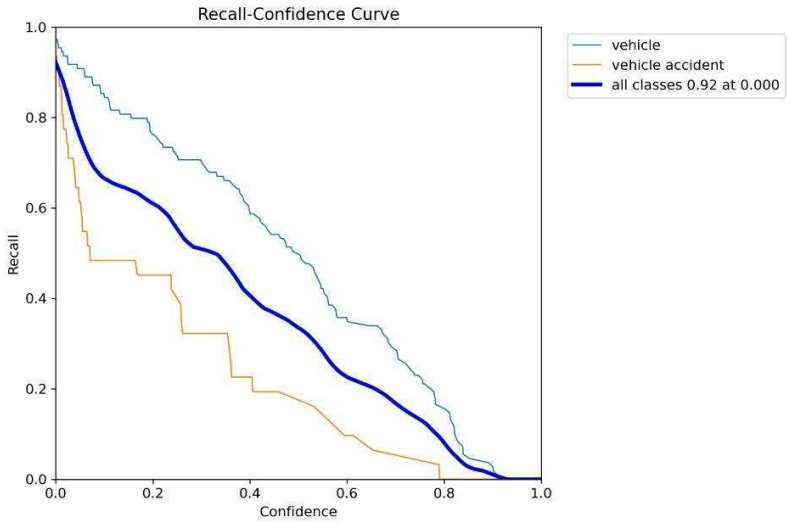

Fig 5.8 Recall - Confidence Curve

Meanwhile, recall - confidence curve is depicted in Figure 5.8, measuring how effectively the model captures all actual instances of responsible driving behaviour. High recall ensures that critical events are not missed, reinforcing the model's strength in identifying all true positives within the dataset. Collectively, these performance metrics validate the strong generalization capability of the proposed system. The integration of YOLO V9 for video-based detection and Random Forest for structured sensor data contributes to a high-performing and trustworthy framework for automated accident analysis and liability assignment.

CHAPTER 6
CONCLUSION

Traditional traffic accident assessment methods suffer from subjectivity, manual interpretation, and limited data, often leading to inconsistent and delayed fault determination. To address these challenges, the proposed system integrates YOLO-V9 for real-time object detection with open accident datasets and structured sensor data, enabling automated, data-driven analysis of vehicle collisions. This approach enhances accuracy by identifying key accident elements such as vehicle positioning, motion patterns, and collision timing through synchronized video and sensor inputs. The Random Forest model further supports objective liability classification based on behavioral and environmental factors. Experimental results demonstrate that this hybrid framework improves both the precision and speed of fault assessment, while also reducing human bias. The proposed work supports real-time data transmission and processing through edge computing, ensuring faster response and decision-making during incidents. Automated report generation consolidates GPS data, speed logs, video snapshots, and liability outcomes into a unified format for insurers and authorities. Its modular hardware-software design makes it highly scalable and adaptable to different vehicle types and road environments. Moreover, the system can be integrated into existing law enforcement and insurance infrastructure, enhancing the overall efficiency and transparency of traffic incident handling. As a result, this work lays the foundation for smarter, scalable traffic monitoring systems that not only evaluate accidents more reliably but also contribute to preventive road safety strategies.

REFERENCES

1. Yawovi, Helton A., Masato Kikuchi, and Tadachika Ozono. "Responsibility Evaluation in Vehicle Collisions From Driving Recorder Videos Using Open Data." IEEE Access (2024).

2. Prof. L. K. Wani; Md Maaz Momin; Sharwari Bhosale; Abhishek Yadav; Manas Nili, " Vehicle Crash Detection using YOLO Algorithm"2022. In International Journal of Computer Science and Mobile Computing, IJCSMC, Vol. 11, Issue. 5, May 2022.

3. Behboudi, Noushin, Sobhan Moosavi, and Rajiv Ramnath. "Recent Advances in Traffic Accident Analysis and Prediction: A Comprehensive Review of Machine Learning Techniques." arXiv preprint arXiv:2406.13968 (2024).

4. Obasi, Izuchukwu Chukwuma, and Chizubem Benson. "Evaluating the effectiveness of machine learning techniques in forecasting the severity of traffic accidents." Heliyon 9, no. 8 (2023).

5. Flores-Calero, Marco, César A. Astudillo, Diego Guevara, Jessica Maza, Bryan S. Lita, Bryan Defaz, Juan S. Ante, David Zabala-Blanco, and José María Armingol Moreno. "Traffic sign detection and recognition using YOLO object detection algorithm: A systematic review." Mathematics 12, no. 2 (2024): 297.

6. Chen, Junbo, Shunlai Lu, and Lei Zhong. "An Autonomous Intelligent Liability Determination Method for Minor Accidents Based on Collision Detection and Large Language Models." Applied Sciences 14, no. 17 (2024): 7716.

7. Lin, Cheng-Jian, and Jyun-Yu Jhang. "Intelligent traffic-monitoring system based on YOLO and convolutional fuzzy neural networks." IEEE Access 10 (2022): 14120-14133.

8. Bokaba, Tebogo, Wesley Doorsamy, and Babu Sena Paul. "Comparative study of machine learning classifiers for modelling road traffic accidents." Applied Sciences 12, no. 2 (2022): 828.

9. Sirisha, U., S. Phani Praveen, Parvathaneni Naga Srinivasu, Paolo Barsocchi, and Akash Kumar Bhoi. "Statistical analysis of design aspects of various YOLO-based deep learning models for object detection." International Journal of Computational Intelligence Systems 16, no. 1 (2023): 126.

10. Sherimon, Vinu, Sherimon PC, Alaa Ismaeel, Alex Babu, Sajina Rose Wilson, Sarin Abraham, and Johnsymol Joy. "An Overview of Different Deep Learning Techniques Used in Road Accident Detection." International Journal of Advanced Computer Science & Applications 14, no. 11 (2023).

11. Yawovi, Helton Agbewonou, Masato Kikuchi, and Tadachika Ozono. "Who was wrong? an object detection based responsibility assessment system for crossroad vehicle collisions." AI 3, no. 4 (2022): 844-862.

12. Lee, Sungjae, and Yong-Gu Lee. "Split liability assessment in car accident using 3D convolutional neural network." Journal of Computational Design and Engineering 10, no. 4 (2023): 1579-1601.

13. Gu, Yang, and Bingfeng Si. "A novel lightweight real-time traffic sign detection integration framework based on YOLOv4." Entropy 24, no. 4 (2022): 487.

14. Olugbade, Samuel, Stephen Ojo, Agbotiname Lucky Imoize, Joseph Isabona, and Mathew O. Alaba. "A review of artificial intelligence and machine learning for incident detectors in road transport systems." Mathematical and Computational Applications 27, no. 5 (2022): 77.

15. Balasubramanian, Saravana Balaji, Prasanalakshmi Balaji, Asmaa Munshi, Wafa Almukadi, T. N. Prabhu, K. Venkatachalam, and Mohamed Abouhawwash. "Machine learning based IoT system for secure traffic management and accident detection in smart cities." PeerJ Computer Science 9 (2023): e1259.

16. Basheer Ahmed, Mohammed Imran, Rim Zaghdoud, Mohammed Salih Ahmed, Razan Sendi, Sarah Alsharif, Jomana Alabdulkarim, Bashayr Adnan Albin Saad, Reema Alsabt, Atta Rahman, and Gomathi Krishnasamy. "A real-time computer vision based approach to detection and classification of traffic incidents." Big data and cognitive computing 7, no. 1 (2023): 22.

17. Tian, Daxin, Chuang Zhang, Xuting Duan, and Xixian Wang. "An automatic car accident detection method based on cooperative vehicle infrastructure systems." IEEE Access 7 (2019): 127453-127463.

18. Pan, Shin-Hung, and Shu-Ching Wang. "Identifying Vehicles Dynamically on Freeway CCTV Images through the YOLO Deep Learning Model." Sensors & Materials 33 (2021).

19. Chung, Yao-Liang, and Chuan-Kai Lin. "Application of a model that combines the YOLOv3 object detection algorithm and canny edge detection algorithm to detect highway accidents." Symmetry 12, no. 11 (2020): 1875.

20. Karim, Abdul, Muhammad Amir Raza, Yahya Z. Alharthi, Ghulam Abbas, Salwa Othmen, Md Shouquat Hossain, Afroza Nahar, and Paolo Mercorelli. "Visual Detection of Traffic Incident through Automatic Monitoring of Vehicle Activities." World Electric Vehicle Journal 15, no. 9 (2024): 382.

APPENDICES

A1 SCREENSHOTS

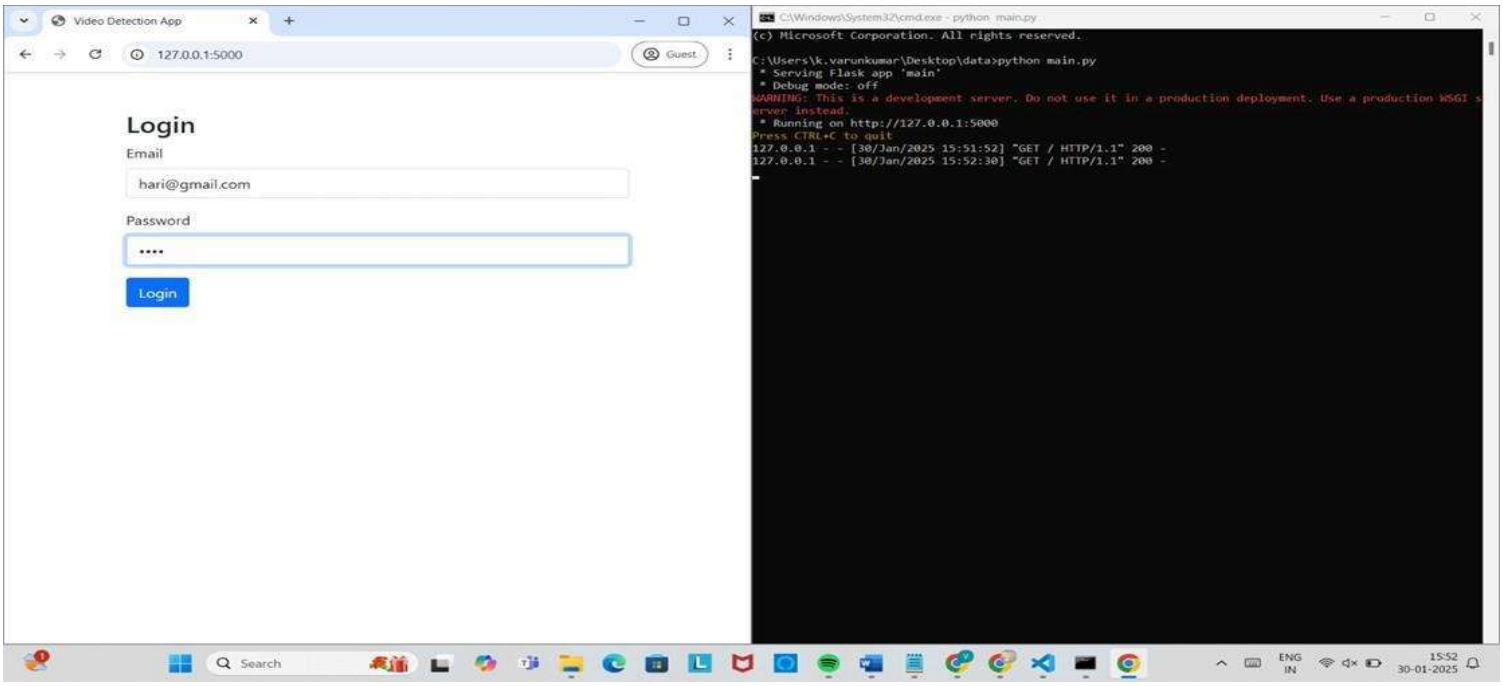

Fig A1.1 Login page

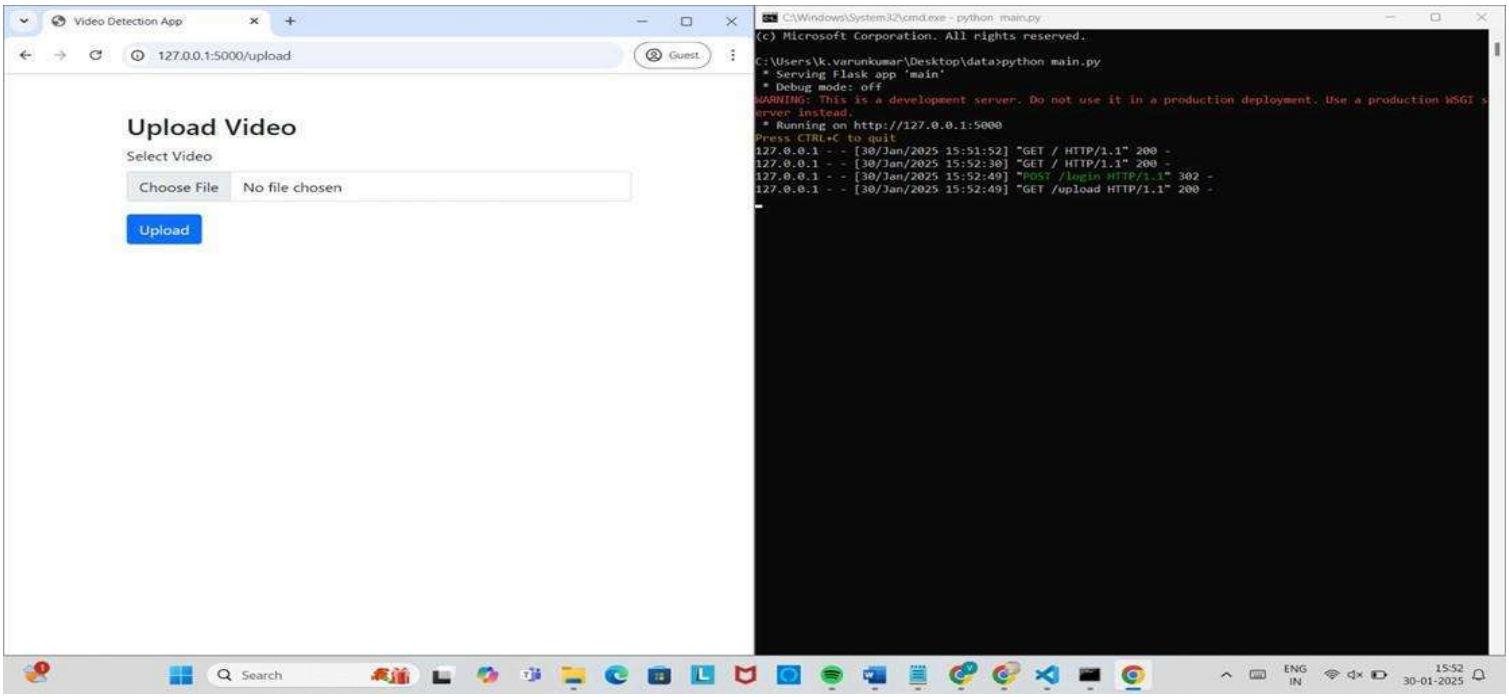

Fig A1.2 Video Upload Page

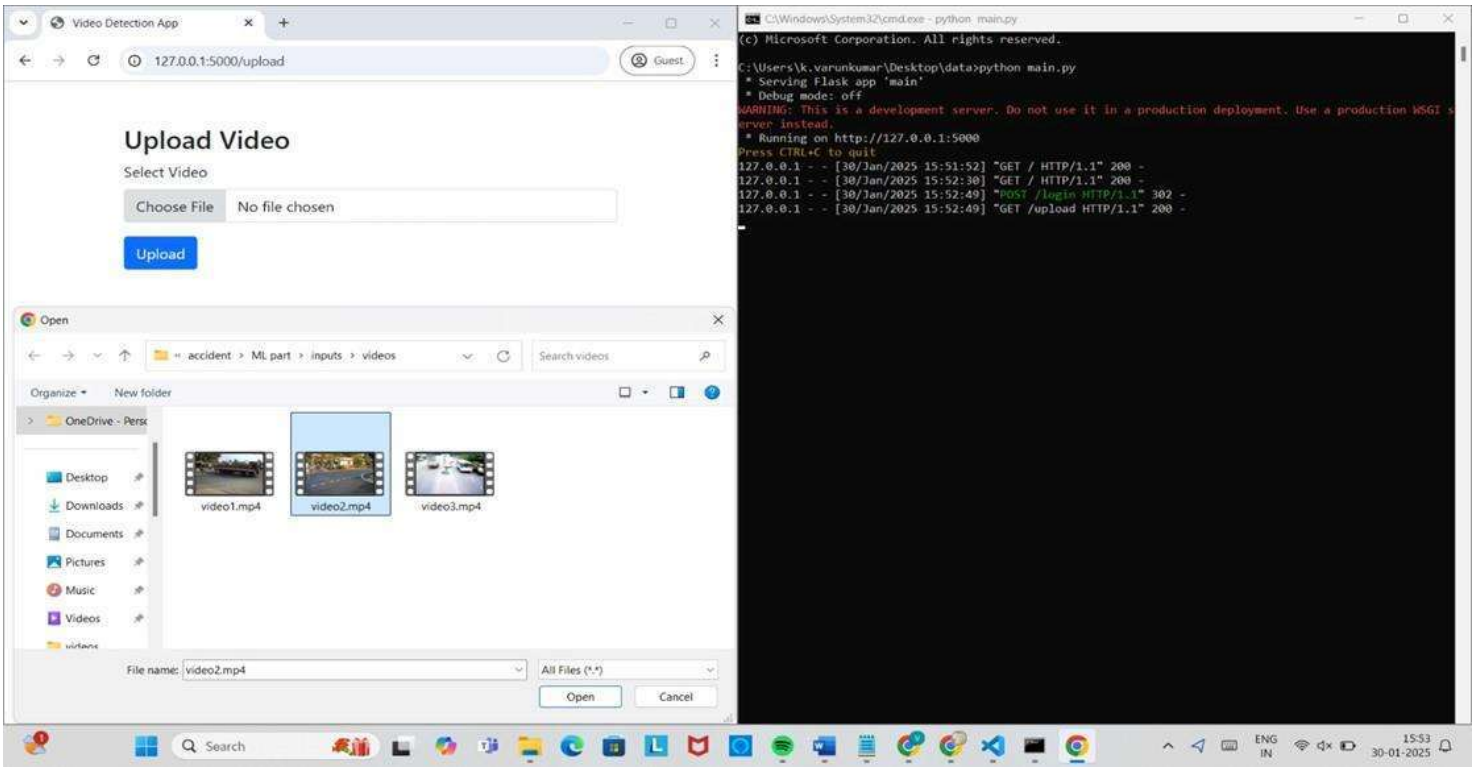

Fig A1.3 Video Selection

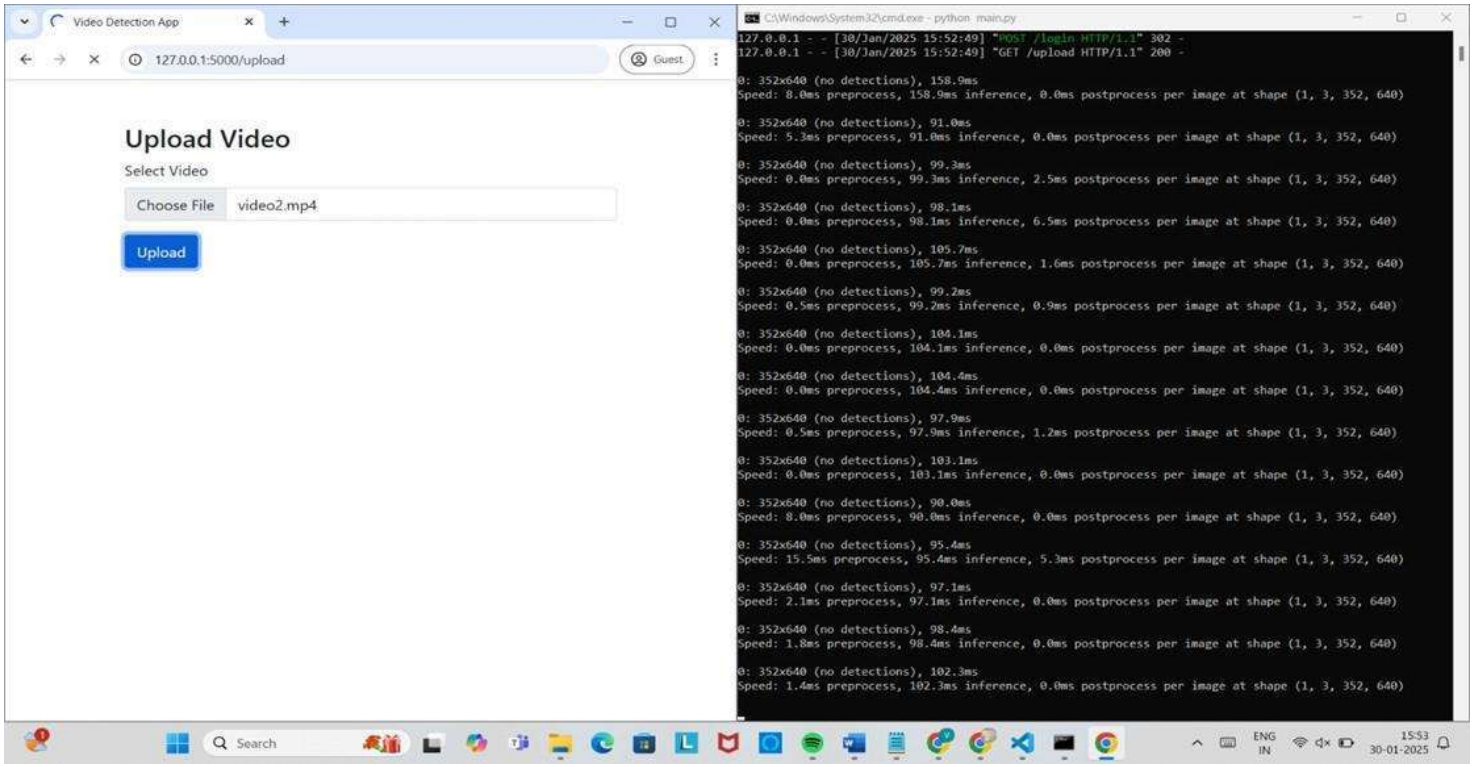

Fig A1.4 Detection Process

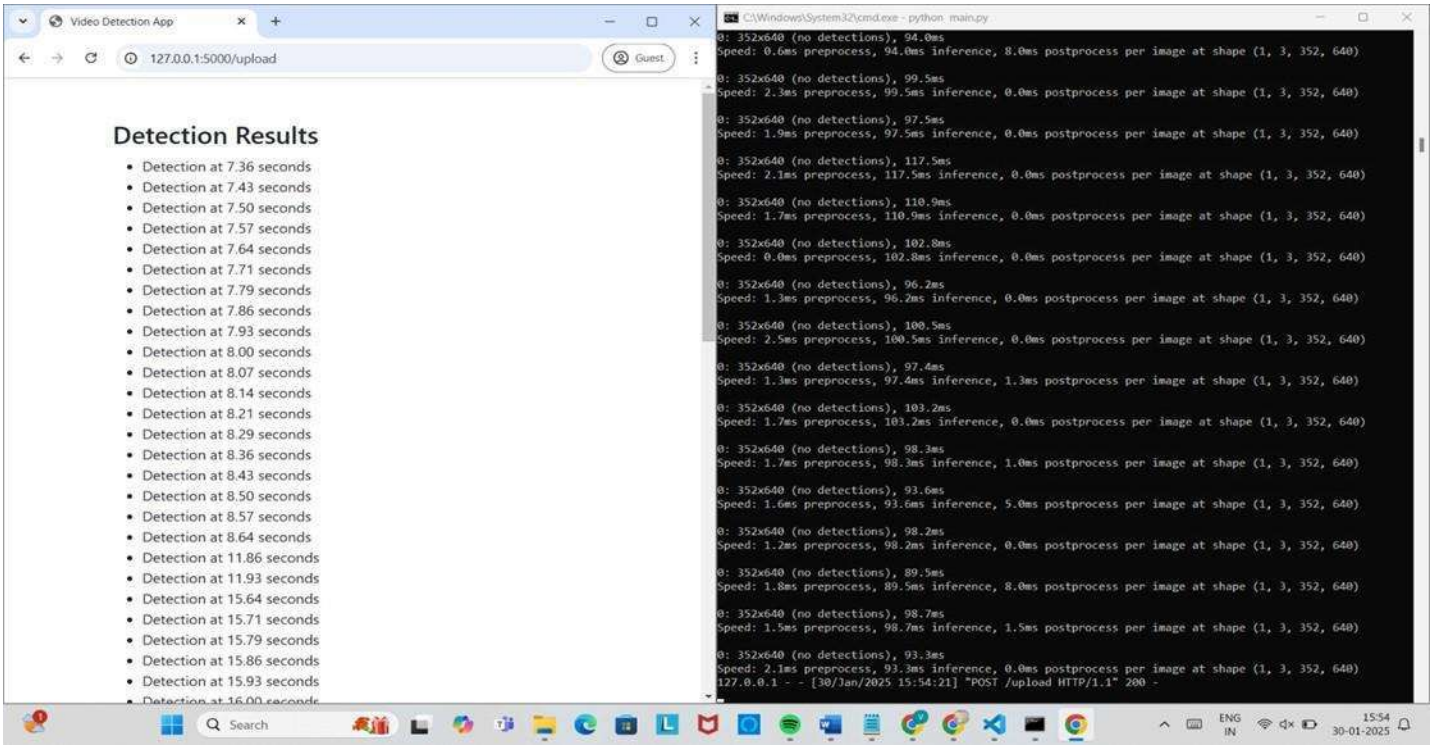

Fig A1.5 Detection Results

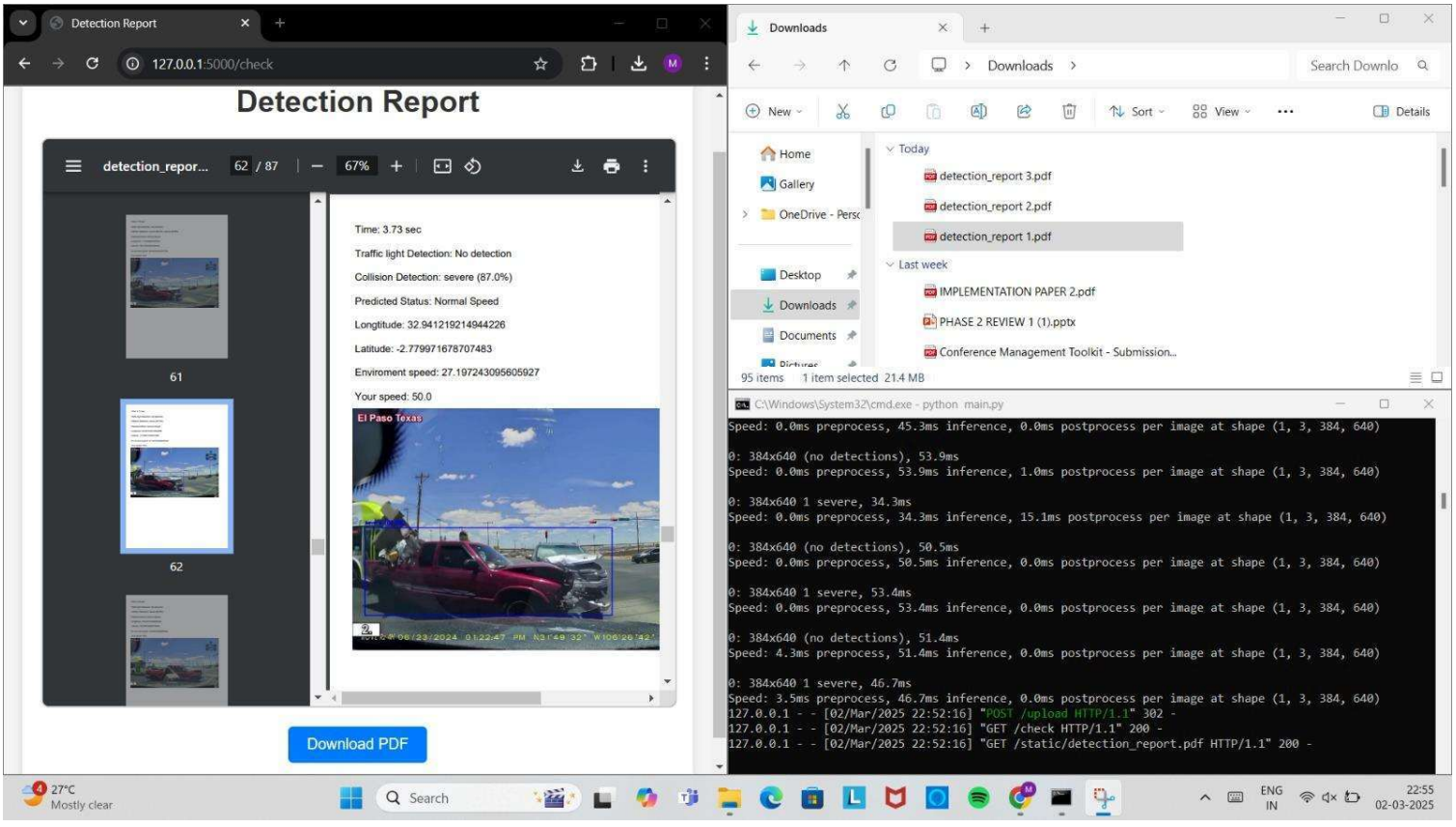

Fig A1.6 Detection Report 1

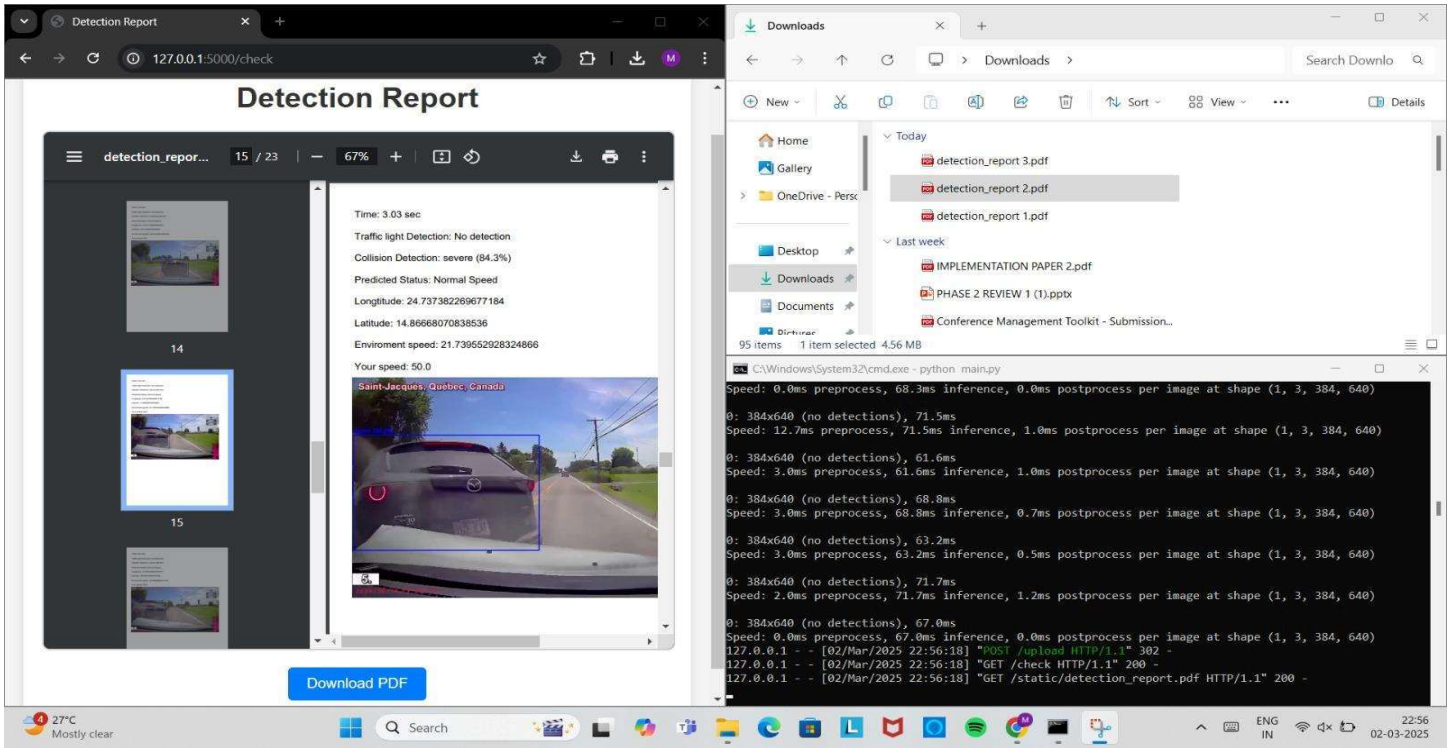

Fig A1.7 Detection Report 2

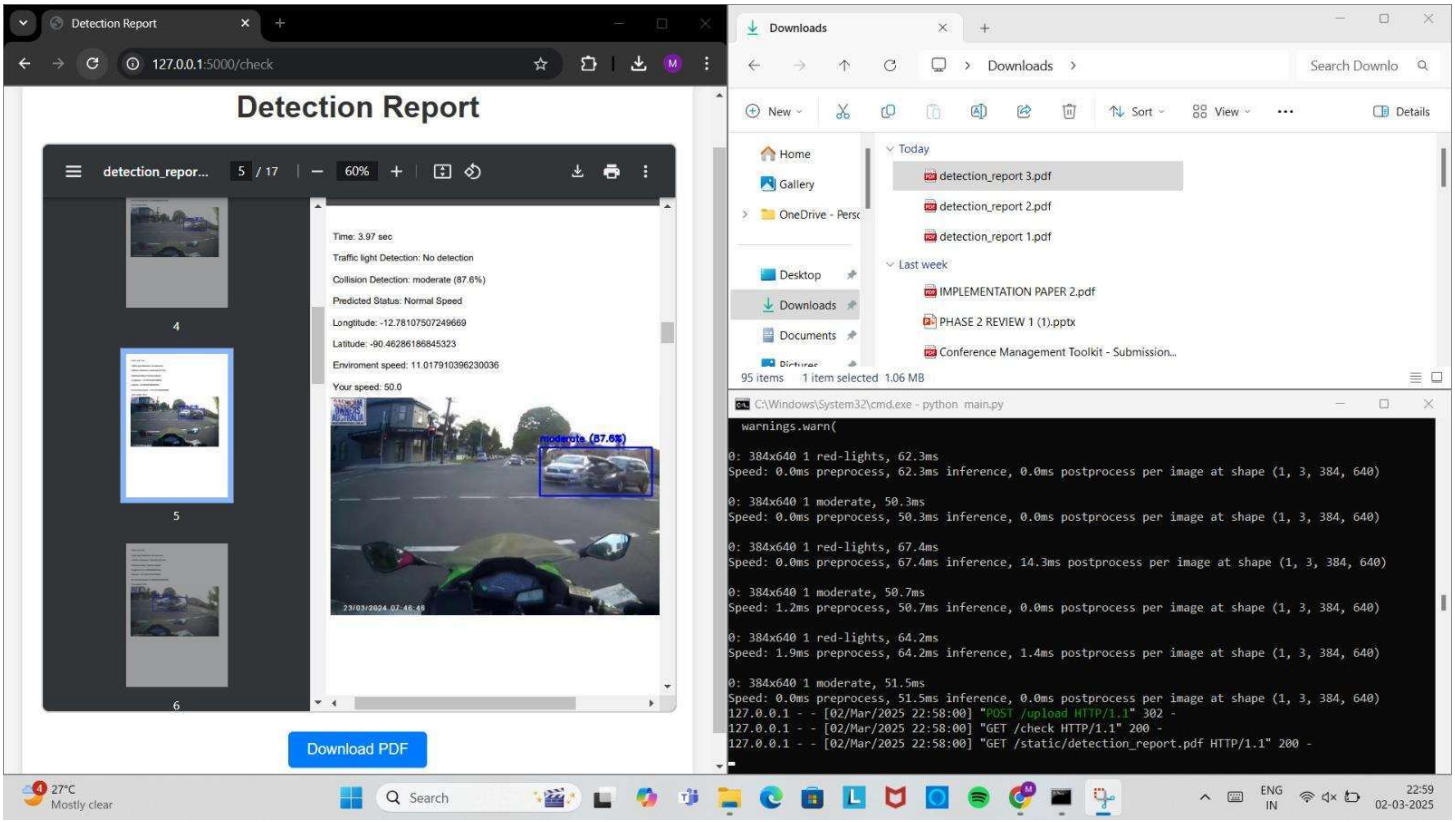

Fig A1.8 Detection Report 3

A2 CODE

DATA COLLECTION:

```
#include "esp_camera.h"
#include <WiFi.h>
#include <HTTPClient.h>
#include <TinyGPS++.h>
#include <HardwareSerial.h>

// ==== WiFi Config ====
const char* ssid = "project1234";
const char* password = "project1234";
const char* serverURL = "http://192.168.228.177:5000/data";  // Flask endpoint

// ==== Ultrasonic Pins ====
#define TRIG_PIN 14
#define ECHO_PIN 15

// ==== GPS Pins ====
#define GPS_RX 3  // U0RXD (connect to GPS TX)
#define GPS_TX 1  // Not used for GPS

// ==== Camera Config ====
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   -1
#define XCLK_GPIO_NUM     0
#define SIOD_GPIO_NUM    26
#define SIOC_GPIO_NUM    27
#define Y9_GPIO_NUM      35
#define Y8_GPIO_NUM      34
#define Y7_GPIO_NUM      39
#define Y6_GPIO_NUM      36
#define Y5_GPIO_NUM      21
#define Y4_GPIO_NUM      19
#define Y3_GPIO_NUM      18
#define Y2_GPIO_NUM       5
#define VSYNC_GPIO_NUM   25
#define HREF_GPIO_NUM    23
#define PCLK_GPIO_NUM    22

TinyGPSPlus gps;
HardwareSerial GPSSerial(1);

// ==== CAMERA SERVER FUNCTION ====
#include <esp_http_server.h>

httpd_handle_t stream_httpd = NULL;
```

```
esp_err_t stream_handler(httpd_req_t *req) {
  camera_fb_t * fb = NULL;
  esp_err_t res = ESP_OK;
  char * part_buf[64];

  res = httpd_resp_set_type(req, "multipart/x-mixed-replace; boundary=frame");
  if(res != ESP_OK) return res;

  while(true){
    fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Camera capture failed");
      return ESP_FAIL;
    }

    res = httpd_resp_send_chunk(req, "--frame\r\n", strlen("--frame\r\n"));
    if(res == ESP_OK){
      snprintf((char *)part_buf, 64, "Content-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n", fb->len);
      res = httpd_resp_send_chunk(req, (const char *)part_buf, strlen((char *)part_buf));
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, (const char *)fb->buf, fb->len);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, "\r\n", strlen("\r\n"));
    }

    esp_camera_fb_return(fb);
    if(res != ESP_OK){
      break;
    }

    delay(100); // slight delay for stream
  }

  return res;
}

void startCameraServer() {
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  config.server_port = 80;

  httpd_uri_t stream_uri = {
    .uri       = "/",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
```

```
  };

  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &stream_uri);
  }
}

void setup() {
  Serial.begin(115200);
  GPSSerial.begin(9600, SERIAL_8N1, GPS_RX, GPS_TX);

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println(" Connected!");

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer   = LEDC_TIMER_0;
  config.pin_d0       = Y2_GPIO_NUM;
  config.pin_d1       = Y3_GPIO_NUM;
  config.pin_d2       = Y4_GPIO_NUM;
  config.pin_d3       = Y5_GPIO_NUM;
  config.pin_d4       = Y6_GPIO_NUM;
  config.pin_d5       = Y7_GPIO_NUM;
  config.pin_d6       = Y8_GPIO_NUM;
  config.pin_d7       = Y9_GPIO_NUM;
  config.pin_xclk     = XCLK_GPIO_NUM;
  config.pin_pclk     = PCLK_GPIO_NUM;
  config.pin_vsync    = VSYNC_GPIO_NUM;
  config.pin_href     = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn     = PWDN_GPIO_NUM;
  config.pin_reset    = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  config.frame_size   = FRAMESIZE_QVGA;
  config.fb_count     = 1;

  if (esp_camera_init(&config) != ESP_OK) {
    Serial.println("Camera init failed");
```

```
    return;
  }

  startCameraServer();
  Serial.println("Camera stream ready!");
  Serial.println("View stream at: http://" + WiFi.localIP().toString());
}

float measureDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH, 30000);
  if (duration == 0) return -1;
  return duration * 0.034 / 2.0;
}

void loop() {
  while (GPSSerial.available()) {
    gps.encode(GPSSerial.read());
  }

  float d1 = measureDistance();
  delay(500);
  float d2 = measureDistance();

  if (d1 < 0 || d2 < 0) {
    Serial.println("Ultrasonic failed.");
    delay(2000);
    return;
  }

  float speed = abs((d1 - d2) / 0.5); // cm/s
  float lat = gps.location.isValid() ? gps.location.lat() : 11.9147;
  float lon = gps.location.isValid() ? gps.location.lng() : 79.6345;

  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverURL);
    http.addHeader("Content-Type", "application/json");

    String payload = "{\"speed\":" + String(speed, 2) +
                     ",\"latitude\":" + String(lat, 6) +
                     ",\"longitude\":" + String(lon, 6) + "}";
```

```
    int httpResponseCode = http.POST(payload);
    Serial.println("POST Response: " + String(httpResponseCode));
    Serial.println("Payload: " + payload);
    http.end();
  }

  delay(2000);

}
```

DETECTION OF VIDEO STREAMING:

```
from ultralytics import YOLO
import cv2

# Load a model
model = YOLO("best.pt")

# Set the dimensions for captured frames
frame_width = 640
frame_height = 480

# known freely available urls with traffic stream:
# http://204.106.237.68:88/mjpg/1/video.mjpg

# URL of the video stream - preferably use motion jpg urls
#stream_url = "http://kamera.mikulov.cz:8888/mjpg/video.mjpg"

stream_url = "inputs/videos/video1.mp4"
print(stream_url)


# start capturing
cap = cv2.VideoCapture(stream_url)

while cap.isOpened():
    # boolean success flag and the video frame
    # if the video has ended, the success flag is False
    is_frame, frame = cap.read()
    if not is_frame:
        break

    resized_frame = cv2.resize(frame, (frame_width, frame_height))

    # Save the resized frame as an image in a temporary directory
    temp_image_path = "temp/temp.jpg"
    cv2.imwrite(temp_image_path, resized_frame)
```

```
    # Perform object detection on the image and show the results
    model.predict(source=temp_image_path, show=True)

    # Check for the 'q' key press to quit
    if cv2.waitKey(1) == 0xff & ord('q'):
        break

# Release the video capture and close any OpenCV windows
cap.release()
cv2.destroyAllWindows()
```

PROCESSING DATA AND REPORT GENERATION:

```
from flask import *
from flask_sqlalchemy import SQLAlchemy
from flask_bcrypt import Bcrypt
from ultralytics import YOLO
import cv2
import numpy as np
import os
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from fpdf import FPDF


app = Flask(_name_)
app.secret_key = "your_secret_key"

# Configuration
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///users.db'
app.config['UPLOAD_FOLDER'] = './static/uploads'
os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)

UPLOAD_FOLDER = './static/uploads'
PREDICTIONS_FOLDER = './static/predictions'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
os.makedirs(PREDICTIONS_FOLDER, exist_ok=True)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['PREDICTIONS_FOLDER'] = PREDICTIONS_FOLDER


# Load YOLO models
model1 = YOLO("C:/Users/k.varunkumar/Desktop/Accident-Detection-and-Notification-
main/traffic/runs/detect/train3/weights/best.pt")
model2 = YOLO("C:/Users/k.varunkumar/Desktop/Accident-Detection-and-Notification-main/ML part/best.pt")
```

```
# Generate synthetic dataset
np.random.seed(42)
num_samples = 1000
time_data = np.round(np.linspace(0, 10, num_samples), 2)
latitude = np.random.uniform(-90, 90, num_samples)
longitude = np.random.uniform(-180, 180, num_samples)
placespeed = np.random.uniform(10, 50, num_samples)
your_speed = 50
status = np.where(placespeed >= your_speed, 1, 0)
df = pd.DataFrame({"Time": time_data, "Latitude": latitude, "Longitude": longitude, "PlaceSpeed": placespeed,
"YourSpeed": your_speed, "Status": status})
X = df[["Time", "Latitude", "Longitude", "PlaceSpeed", "YourSpeed"]]
y = df["Status"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

def predict_status(input_time):
    nearest_row = df.iloc[(df["Time"] - input_time).abs().argsort()[:1]]
    input_data = nearest_row[["Time", "Latitude", "Longitude", "PlaceSpeed", "YourSpeed"]].values
    prediction = rf_model.predict(input_data)[0]
    return {
        "speed": "Overspeed" if prediction == 1 else "Normal Speed",
        "sec": float(input_data[0][0]), "long": float(input_data[0][1]),
        "lat": float(input_data[0][2]), "placespeed": float(input_data[0][3]), "yourspeed": float(input_data[0][4])
    }

# Initialize extensions
db = SQLAlchemy(app)
bcrypt = Bcrypt(app)

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    email = db.Column(db.String(100), unique=True, nullable=False)
    password = db.Column(db.String(255), nullable=False)
    mobile = db.Column(db.String(15), nullable=False)

# Initialize YOLO model
model = YOLO("best.pt")


@app.route('/')
def home():
    return render_template('login.html')


@app.route('/register', methods=['GET', 'POST'])
```

```
def register():
    """User Registration"""
    if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']
        password = request.form['password']
        mobile = request.form['mobile']

        if not name or not email or not password or not mobile:
            flash("All fields are required!", "danger")
            return redirect(url_for('register'))

        hashed_password = bcrypt.generate_password_hash(password).decode('utf-8')
        user = User(name=name, email=email, password=hashed_password, mobile=mobile)
        db.session.add(user)
        db.session.commit()

        flash("Registration successful. Please login.", "success")
        return redirect(url_for('home'))

    return render_template('register.html')


@app.route('/login', methods=['POST'])
def login():
    """User Login"""
    email = request.form['email']
    password = request.form['password']

    user = User.query.filter_by(email=email).first()
    if user and bcrypt.check_password_hash(user.password, password):
        return redirect(url_for('upload_video'))

    flash("Invalid email or password", "danger")
    return redirect(url_for('home'))


@app.route('/upload', methods=['GET', 'POST'])
def upload_video():
    """Video Upload and Object Detection"""
    if request.method == 'POST':
        file = request.files.get('file')

        if not file or file.filename == '':
            flash("No file selected", "danger")
            return redirect(url_for('upload_video'))

        file_path = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
```

```
        file.save(file_path)

        # Run detection
        detection_results =  process_video(file_path)

        pdf_path = generate_pdf_report(detection_results)
        return redirect("/check")


    return render_template('upload.html')


@app.route('/check', methods=['GET', 'POST'])
def check_video():
    pdf_path="static/"
    return render_template('results.html',pdf_path=pdf_path)


def process_video(video_path):
    cap = cv2.VideoCapture(video_path)
    frame_rate = int(cap.get(cv2.CAP_PROP_FPS))
    frame_count = 0
    detections = []

    os.makedirs("detected_frames", exist_ok=True)

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        frame_count += 1
        timestamp = frame_count / frame_rate
        results1 = model1.predict(source=frame, show=False)
        results2 = model2.predict(source=frame, show=False)

        def draw_boxes(image, results, color, model):
            detected_classes = []
            for result in results:
                for box in result.boxes:
                    confidence = float(box.conf[0]) * 100
                    if confidence < 79:
                        continue
                    x1, y1, x2, y2 = map(int, box.xyxy[0])
                    class_id = int(box.cls[0])
                    class_name = model.names[class_id]
                    detected_classes.append(f"{class_name} ({confidence:.1f}%)")
                    cv2.rectangle(image, (x1, y1), (x2, y2), color, 2)
```

```
                cv2.putText(image, f"{class_name} ({confidence:.1f}%)", (x1, y1 - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
            return detected_classes

        detected1 = draw_boxes(frame, results1, (0, 255, 0), model1)
        detected2 = draw_boxes(frame, results2, (255, 0, 0), model2)

        if detected1 or detected2:
            frame_filename = f"detected_frames/frame_{frame_count}.jpg"
            cv2.imwrite(frame_filename, frame)
            pr = predict_status(timestamp)
            detections.append({"time": timestamp, "model1": detected1, "model2": detected2, "image":
frame_filename, "pr": pr})

    cap.release()
    return detections
def generate_pdf_report(detections):
    pdf = FPDF()
    pdf.add_page()
    pdf.set_font("Arial", "B", 16)
    pdf.cell(200, 10, "YOLO Detection Report", ln=True, align="C")
    pdf.set_font("Arial", size=12)
    for detection in detections:
        pdf.add_page()
        pdf.cell(200, 10, f"Time: {detection['time']:.2f} sec", ln=True)
        pdf.cell(200, 10, f"Traffic light Detection: {', '.join(detection['model1']) if detection['model1'] else 'No
detection'}", ln=True)
        pdf.cell(200, 10, f"Collision Detection: {', '.join(detection['model2']) if detection['model2'] else 'No
detection'}", ln=True)
        pdf.cell(200, 10, f"Predicted Status: {detection['pr']['speed']}", ln=True)
        pdf.cell(200, 10, f"Longtitude: {detection['pr']['long']}", ln=True)
        pdf.cell(200, 10, f"Latitude: {detection['pr']['lat']}", ln=True)
        pdf.cell(200, 10, f"Enviroment speed: {detection['pr']['placespeed']}", ln=True)
        pdf.cell(200, 10, f"Your speed: {detection['pr']['yourspeed']}", ln=True)
        pdf.image(detection["image"], x=10, w=180)
        pdf.ln(10)
    pdf_path = "detection_report.pdf"
    pdf.output("static/"+pdf_path)
    return pdf_path
@app.route('/prediction_images/<filename>', methods=['GET'])
def get_prediction_image(filename):
    return send_from_directory(app.config['PREDICTIONS_FOLDER'], filename)
if _name_ == '_main_':
    with app.app_context():
        db.create_all()  # Create the database tables
    app.run()
```