TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A Minor Project Report

On

GPS-Based Vehicle Tracking and Accelerometer-driven Crash Detection

Submitted By:

Anish Timsina	(EXAM ROLL No. 33555)
Kapur Pant	(EXAM ROLL No. 33569)
Saugat Neupane	(EXAM ROLL No. 33588)
Sugam Khatiwada	(EXAM ROLL No. 33591)

Submitted To:

Department of Electronics and Computer Engineering,

Thapathali Campus

Kathmandu, Nepal

April, 2024

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A Minor Project Report

On

GPS-Based Vehicle Tracking and Accelerometer-driven Crash Detection

Submitted By:

Anish Timsina	(EXAM ROLL No. 33555)
Kapur Pant	(EXAM ROLL No. 33569)
Saugat Neupane	(EXAM ROLL No. 33588)
Sugam Khatiwada	(EXAM ROLL No. 33591)

Submitted To:

Department of Electronics and Computer Engineering,

Thapathali Campus

Kathmandu, Nepal

In Partial fulfillment for the award of the Bachelor's Degree in

Electronics,Communication and Information Engineering.

Under the Supervision of

Er. Khetraphal Bohara

April, 2024

DECLARATION

We hereby declare that the report of the project entitled "**GPS-Based Vehicle Tracking and Accelerometer-driven Crash Detection**" which is being submitted to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in **Electronics, Communication, and Information Engineering**, is a bonafide report of the work carried out by us. The materials contained in this report have not been submitted to any University or Institution for the award of any degree and we are the only author of this complete work and no sources other than the listed here have been used in this work.

Anish Timsina (THA077BEI007) _____

Kapur Pant (THA077BEI021) _____

Saugat Neupane (THA077BEI043) _____

Sugam Khatiwada (THA077BEI046) _____

Date: April, 2024

CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, a minor project work entitled "**GPS-Based Vehicle Tracking and Accelerometer-driven Crash Detection**" submitted by **Anish Timsina, Kapur Pant, Saugat Neupane and Sugam Khatiwada** in partial fulfillment for the award of Bachelor's Degree in Electronics, Communication, and Information Engineering. The Project was carried out under special supervision and within the time frame prescribed by the syllabus.

We found the students to be hardworking, skilled and ready to undertake any related work to their field of study and hence we recommend the award of partial fulfillment of Bachelor's degree of Electronics, Communication, and Information Engineering.

Project Supervisor
Mr. Kshetraphal Bohara
Telecom Engineer, Nepal Telecom

External Examiner
Mr. Yogesh Aryal
Ministry of Education, Science and Technology

Project Co-ordinator
Mr. Umesh Kanta Ghimire
Department of Electronics and Computer Engineering, Thapathali Campus

Mr. Kiran Chandra Dahal
Head of the Department,
Department of Electronics and Computer Engineering, Thapathali Campus

April, 2024

COPYRIGHT

ACKNOWLEDGEMENT

We would like to thank the **Institute of Engineering, Tribhuvan University** for providing us with such a wonderful opportunity to learn and implement our knowledge in the form of a minor project.

We appreciate the dedication of the **Department of Electronics and Computer Engineering, Thapathali Campus** and express our gratitude for the approval of our project proposal. We are also very thankful to our supervisor **Er. Kshetraphal Bohara** for his support and guidance.

In addition, we would like to thank all teaching as well as non-teaching staff and classmates for their useful advice and support.

Sincerely,

Anish Timsina (THA077BEI007)

Kapur Pant (THA077BEI021)

Saugat Neupane (THA077BEI043)

Sugam Khatiwada (THA077BEI046)

ABSTRACT

GPS-Based Vehicle Tracking and Accelerometer-driven Crash Detection presents a solution for enhancing road safety through real-time vehicle monitoring and crash detection using a Raspberry Pi-based system. Using components such as GPS and an accelerometer, the system tracks vehicle location and detects potential accidents. Deep Learning techniques Gated Recurrent Units (GRUs) and Autoencoders are used to train the model using the dataset of accelerometer under normal driving condition instead of training the model on crash dataset which are comparatively more difficult to obtain. By training on normal driving scenarios, the system detects the anomalous data obtained from accelerometer while driving. The time series data obtained from accelerometer is reconstructed by the model according to the pattern it learnt from the training dataset. The anomaly is detected based on the error between input time series data and the reconstructed data. A threshold Mean Absolute Error is set above which the data is considered to be an anomaly. Keeping the threshold error 2.5, the model achieved an F1 score of 0.835 and with threshold error 3, F1 score was 0.837. The system utilizes internet connection for emergency alerts and location sharing. Overall, this project aims to contribute to a safer driving environment by providing a reliable and efficient solution.

Keywords: Accelerometer, Autoencoders, Crash Detection, Deep Learning, GRU, Raspberry Pi

Table of Contents

List of Figures

List of Tables

List of Equations

List of Abbreviations

ADC	Analog to Digital Convertor
AI	Artificial Intelligence
ANSI	American National Standards Institute
API	Application Programming Interface
CCTV	Closed Circuit Television
CIFAR	Canadian Institute for Advanced Research
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSV	Comma-separated values
DL	Deep Learning
ESP	Embedded System Processor
FN	False Negatives
FP	False Positives
GND	Ground
GNSS	Global Navigation Satellite Systems
GPIO	General Purpose Input/Output
GPRS	General Packet Radio Service
GPS	Global Positioning System
GRU	Gated Recurrent Unit
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IoT	Internet of Things
LSTM	Long Short-term Memory

MAE	Mean Average Error
ML	Machine Learning
MNIST	Modified National Institute of Standards and Technology
MPU	Motion Processing Unit
MSE	Mean Squared Error
PWM	Pulse Width Modulation
R/C	Remote Control
RNN	Recurrent Neural Network
Rx	Receive
SCL	Serial Clock Line
SDA	Serial Data Line
SIM	Subscriber Identity Module
SMS	Short Message Service
SoC	System-on-Chip
SOM	Self-Organizing Maps
SSID	Service Set Identifier
TCP/IP	Transmission Control Protocol/ Internet Protocol
TN	True Negatives
TP	True Positives
Tx	Transmit
UART	Universal Asynchronous Receiver/Transmitter
UML	Unified Modeling Language
VCC	Voltage at Common Collector
Wi-Fi	Wireless Fidelity

1. INTRODUCTION

1.1 Background Information

Majority of Nepal is covered by hills and mountains. Due to difficult geography, most parts lack well developed road infrastructure because of which road safety is a major concern. Predicting accidents might be a little out of our hands but the best we can do to save lives is act immediately after the accident. So, we wanted to develop a system that is basically concerned with acting as quickly as possible post-accident. Likewise, nowadays tracking devices have grown to be common. GPS is used commonly nowadays in a lot of devices including smartwatches, smartphones and some vehicles even have GPS tracking system integrated which can be monitored through mobile applications. Live tracking can be important in case of theft, especially motorcycles, which are easier to steal than automobiles.

Many studies have confirmed that timely emergency rescue to crashes result in higher survival rates. We can take an example of China in 2016 where two-thirds of traffic accidents victims died because they did not receive timely rescue and emergency treatment services [1]. Moreover, when a traffic accident occurs in the rural or the driver is the only person in the vehicle and the crash results in loss of consciousness, no person is available to notify the related authorities within the “golden window period” for medical service, and this delay decreases the survival rate.

Thus, to overcome these problems, our system detects accidents using accelerometer trained using Deep Learning and alert the family members (or chosen emergency contact) after which they can act accordingly. Moreover, the use of GPS for live tracking also provides an added benefit as the location of the vehicle can be monitored for enhanced safety measures. For live tracking, a server is connected to Raspberry Pi which will continuously send data through internet.

All the components (e.g. GPS, accelerometer, buttons) will be integrated into RaspberryPi including the deep learning model developed for crash detection.

1.2 Motivation

Road accidents, unfortunately, are common in countries with difficult terrain like Nepal. So, road safety is a major concern for a country like ours. Our project targets the tracking of vehicles and their crash detection. Generally, there are no problems with tracking the vehicle as GPS can be used effectively to track the mobile phone of the driver to know the location of the vehicle. But what our project focuses on is a device which can be used for both tracking as well as crash detection.

In short, we wanted to make an affordable product, also focusing on simplicity, dedicated to vehicle tracking and crash detection, which is why we decided to do this project.

1.3 Problem Statement

The increased use of automobiles has increased the traffic risks and road accidents. Due to lack of emergency services right after an accident, lives are lost, and injuries become more serious than they would be if proper emergency services were available. Similarly, in case of vehicle theft, especially motorcycles, without any means to track the stolen vehicle, it is difficult to recover it. There is a shortage of devices that integrate both crash detection and real-time vehicle tracking within a single product. So, this design is a system that automatically detects accident alerts the emergency contact chosen by the driver. It also provides capability of live tracking of the vehicle using the same device which can be used for monitoring the location of the vehicle which comes in handy in situations like theft. Along with crash detection, the feature of tracking live location of a vehicle is also added. The crash detection devices using cameras tend to be a little expensive, require more computational power and are susceptible to visual obstructions. It can be applied across a range of vehicles, including motorcycles and bicycles, and is not constrained by the need for a direct line of sight unlike camera-based crash detection devices.

1.4 Objectives

- To design and develop a system to track and monitor vehicle’s location usingGPS.
- To detect accidents and alert the emergency contact as quickly as possible.

1.5 Scope and Application

The system has a wide range of functionalities that go beyond accident detection, most notably the ability to track a vehicle's current location in real-time. This feature not only improves security procedures but also comes in very handy when there is a car theft or loss. The capacity to consistently track and monitor a vehicle's location provides a strong means of providing a speedy recovery, which adds significantly to the general security and comfort of owners.

1.5.1 Project Scopes and Limitations

- The system is designed to detect crashes and notify designated contacts in realtime.
- The system continuously monitors the location and movements of vehicles equipped with the tracking system.
- While it detects crashes, the system does not currently assess the severity of accidents.

1.5.2 Project Applications

- **Increased Safety in Vehicles**

  Our system significantly increases the chances of survival for drivers and passengers by providing real time crash detection. In the event of an accident, immediate alerts are sent enabling timely assistance and potentially saving lives.

- **Vehicle Management**

  Vehicle management companies can use our system to track and monitor their vehicles. This helps them gain better operational control, find the best routes, and use resources more efficiently which can help them in increasing productivity and cutting down on operational expenses.

- **Emergency Vehicle Tracking**

  By utilizing our system on emergency vehicles such as ambulances, fire trucks, police cars, people can monitor them in real-time. Using our website, one can see where these vehicles are going and anticipate when they'll arrive. This helps everyone coordinate better during emergencies and ensures help arrives critical situation.

- **Stolen vehicle recovery**

  In cases of vehicle theft, the ability to track the stolen vehicle in real-time greatly improves the chances of recovery. Both law enforcement agencies and owners themselves can use our system to pinpoint the exact location of the stolen vehicle, intervene quickly to retrieve it and reduce losses.

1.6 Report Organization

This report is divided into ten chapters for clear and concise presentation of the project. Chapter one is the Introduction which covers the background, motivation, problem definition, objective, scope, and application of the project. Second chapter, which is Literature Review, provides an overview of the relevant literature and works related to the project.Chapter three, Requirement Analysis, analyzes the hardware and software requirementof the project. Chapter four, The System Architecture and Methodology, provides a detailed description of the system architecture, deep learning architecture, and flow ofthe project. Chapter five, Dataset Analysis, contains detailed description of the datasetcollection, data preparation, data visualization, and data splitting. Chapter six, Implementation Details, describes how the hardware and software tools have been implemented for the project along with use cases, activity, and sequence diagrams. Chapter Seven, Results and Analysis, includes the outcome of the project and various curves used for result analysis. Chapter Eight, Future Enhancement, provides insight about features that can be added in the future to improve the project. Chapter Nine, Conclusion, concludes the report summarizing the main findings. Lastly, Chapter Ten, Appendices, includes the project schedule, budgets, module specifications, and code snippets.

2. LITERATURE REVIEW

There has been some research conducted related to crash detection and vehicle location tracking. The use of airbags is also triggered by employing a crash detection system in automobiles. Here are a few research papers and products that share similar interests to ours, from which we have sought assistance to develop our system.

The “Accelerometer-based Data-driven Hazard Detection and Classification for Motorcycles” by D. Selmanaj et al. (2014) [2] also uses accelerometers to detect accidents. There are four accelerometers, two on the body and two on the wheels, which sense the acceleration along the three axes, comprising twelve acceleration measures. The ML algorithm used to tune the model for detecting accidents is SOM. The tuning is done using data of accelerometer under normal driving conditions. However, this article claims to validate the proposed method using data obtained from a simulation which might sometimes fail to capture the real-world situations.

Luca Kubin et al.’s “Deep Crash Detection from Vehicular Sensor Data with Multimodal Self-Supervision” (2021) [3] uses Deep Learning approach. The researchers used crash, non-crash and severe crash data from accelerometer to train the model to detect accidents. This system incorporates a specialized neural architecture, a self-supervised pre-training technique, and various data augmentation methods. The research focuses on the classification of accidents and theirseverity using data from inertial measurement units and GPS devices. Through extensive testing on a large dataset comprising over 200,000 time series from US vehicles, the proposed method achieves notable results, attaining an average-precisionscore of 0.9 for crash detection and 0.76 for severe crash detection. The study emphasizes the potential applicability of the developed approach to similar challenges involving diverse input data. Future research directions are suggested, including exploring video data for a more comprehensive understanding of crash scenes, and optimizing the system for smaller and faster deployment on embedded devices.

The incorporation of audio and video data from dashboard cameras is suggested in "Car crash detection using ensemble deep learning and multimodal data from dashboard cameras" by Jae Gyeong Choi et al. (2021) [4]. For both audio and video data, a base

classifier based on CNN and GRU is created. The suggested vehicle crash detection system is developed using deep learning techniques (gated recurrent units, or GRUs and convolutional neural networks, or CNNs), with the usage of a weighted average ensemble as an ensemble method. As this article suggests using both visual and aural data, a sophisticated CPU with many resources is needed to implement the system, which makes it bulky.

Recently, Apple launched a device, Apple Watch, in 2022 which had crash detection feature embedded in it [5]. This system uses accelerometers and gyroscopes to detect accidents. The apple watch is worn on the wrist of the driver. After the accident is detected, it will try to call 911 through regular carrier network. There have been some successful cases of accident detection using Apple watch where some lives were saved.[6] But Apple products are not affordable to everyone. Also, as the device is worn on the wrist, false alarms might be triggered in case of the user riding rollercoasters.

The "Accident Detection in Autonomous Vehicles Using Modified Restricted Boltzmann Machine" by Dr. C. K. Gomathy et al. (2022) [7] uses generic CCTV to identify the road accidents in real time. Deep learning approach modified restricted Boltzmann machine is used to identify the images captured in CCTV camera and separate accidents and non-accident conditions. The problem with using CCTV cameras to detect accidents is that the cameras have limited coverage which implies that a lot of cameras need to be installed.

Hanif Hakimi Azlan and Suriana Salimin's "Vehicle Accident Detection System using Accelerometer" (2023) [8] uses accelerometer, to detect crashes but ML approach is not used. Instead, sudden changes in data of accelerometer are detected using some logic in code, or by keeping a threshold value and when the value is crossed, the alert system is triggered, and the location of the vehicle is sent to chosen emergency contact. In this approach, as ML is not used, and a certain threshold is set for the acceleration of the three axes measured by accelerometer, it can be very difficult to detect accidents correctly. Using ML can be helpful under diverse and dynamic situations. Identifying appropriate thresholds for acceleration values to signify an accident might be subjective and may not cover all possible scenarios.

The article "Classification of time series by shapelet transformation" by Jon Hills et al.[9] underscores the superiority of deep learning (DL) approaches over classical machinelearning (ML) methods in handling time-series data, such as that derived from accelerometers. This finding supports our choice to utilize DL techniques in our project, aligning with the demands of processing accelerometer data and enhancing our capacity to classify and interpret the intricate patterns inherent in time-series information.

Marzieh Farahani's work on "Anomaly detection in gas turbines with deep LSTM-autoencoders" [10] serves as an insightful reference for accident detection system utilizing accelerometers. Farahani's methodology demonstrates the potential of LSTM-autoencoders in accurately identifying anomalies within time-series data, thereby enhancing predictive maintenance and operational efficiency in gas turbine systems. The project underscores the versatility of deep learning frameworks in handling complex data patterns and anomalies, laying a foundation for future research in fault detection and condition monitoring across various domains.

Similarly, in the research paper “Deep Learning for Anomaly Detection in Time-Series Data: Review, Analysis, and Guidelines” by Kukjin Choi et al. [11], LSTM architecture is recommended for achieving early anomaly warnings due to its capacity to capture long-term dependencies. LSTM's proficiency in detecting subtle changes in data patterns enhances its suitability for early anomaly detection objectives. However, in our case where we plan to achieve real-time anomaly detection, GRU architecture has notable benefits. GRUs offers a much simpler design compared to LSTMs resulting in enhanced computational efficiency and faster training. This streamlined architecture makes GRUs more suitable for applications where real-time processing and response are critical.

understanding. Our project, however, diverges by centering on accelerometer data from normal driving conditions for accident detection. We utilize GRU-Autoencoders for time series data processing, integrating domain-specific features to enhance accident detection precision in real-world scenarios. This contrast shows the uniqueness of our approach in anomaly detection systems.

The effectiveness of autoencoder-based anomaly detection techniques in detecting

semantic anomalies, like vehicles, in highway driving scenarios recorded by front-facing cameras is examined in the research article "Comparing autoencoder-based approaches for anomaly detection in highway driving scenario images," authored by Vasilii Mosin et al. [13]. The study uses reconstruction errors and bottleneck values to examine two autoencoder-based methods. Experiments are conducted using datasets of simulated driving scenarios to evaluate performance with different autoencoders, training parameters, and color effects. Although some encouraging results are noted, the paper highlights that before safe autoencoder-based algorithms are applied in automobile perception systems, more research including a variety of use-cases and real-world datasets are required.

3. REQUIREMENT ANALYSIS

The system is set up to track the real-time location of vehicles and detect crashes using a deep learning model. This requires specific hardware components like microcontrollers, a GPS module, and an accelerometer sensor, along with software such as Flask, TensorFlow and so on. To ensure the connectivity between the Raspberry Pi and the attached sensors, a circuit schematic has been designed and implemented for convenient integration via GPIO pins.

3.1 Hardware Requirements

3.1.1 Raspberry Pi

The Raspberry Pi is a series of small single-board computers developed by the Raspberry Pi Foundation. Key features of Raspberry Pi include compact size, versatility, GPIO pins, OS support, expansion features and so on. In our project, the Raspberry Pi serves as the central component, integrating with all other hardware components and hosting the necessary software required for operation.

Figure 3-1: Raspberry Pi 4 Model B

3.1.2 GPS Module

A GPS module is an electronic device that receives signals from satellites to determine its own geographical location, including latitude, longitude, altitude, and speed. This module is widely used in navigation systems, vehicle tracking, geocaching, mapping, surveying, and various IoT applications where precise location information is crucial.

Figure 3-2: GPS NEO 7M

3.1.3 Accelerometer

An accelerometer is a sensor that measures acceleration, which is the rate of change of velocity of an object. It detects changes in motion and is commonly used to determine the orientation, tilt, and movement of devices.

Figure 3-3: MPU 6050

3.1.4 Piezo Buzzer

A piezo buzzer is a type of electronic sound-producing device that uses the piezoelectric effect to generate sound waves. It consists of a piezoelectric ceramic disc sandwiched between two metal plates or electrodes.

Figure 3-4: Piezo Buzzer

3.2 Software Requirements

3.2.1 Twilio

Twilio is a cloud communications platform that allows software developers to programmatically make and receive phone calls, send, and receive text messages, and perform other communication functions using its APIs. With Twilio, developers can integrate voice, messaging, and video capabilities into their applications, whether web or mobile.

3.2.2 Visual Studio Code

Visual Studio Code is an open and versatile source code editor from Microsoft that stands out for its speed and broad language support. Visual Studio Code offers a highly customizable experience through extensions, with features like intelligent code completion, debugging tools, an integrated terminal, and Git integration. Its cross-platform compatibility has led to widespread adoption among developers.

3.2.3 Jupyter Notebook

Jupyter Notebook is an open-source interactive web application that facilitates the creation and sharing of documents containing live code, equations, visualizations, and narrative text. It supports Python and various other programming languages as well. Users can produce dynamic reports, conduct data analysis, and create machine learning models within a single, user-friendly interface.

3.2.4 Library Packages

- **TensorFlow**

  TensorFlow is an open-source AI framework developed by Google and is a Python library customized for numerical calculations and machine learning tasks. It combines a wide variety of classification, regression, and deep learning algorithms, providing users with versatile tools for AI-based application models.

- **Keras**

  Keras serves as a user-friendly interface that sits atop open-source machine learning frameworks like TensorFlow. It is tailored with researchers and data scientists in mind, enabling them to efficiently construct deep learning models and easily add multiple layers to neural networks for improved performance which helps in simplifying complex deep learning tasks.

- **Matplotlib**

  Matplotlib is a Python library for creating two-dimensional plots, ideal for producing professional-looking figures in different formats and interactive environments. It is handy for visualizing various datasets and utilizing NumPy for handling large arrays of data. With Matplotlib, we can easily plot complex datasets to gain insights and make informed decisions.

- **Pandas**

  Pandas is a powerful open-source Python library designed for data manipulation and analysis. It provides easy-to-use data structures and tools for working with structured data, making it a fundamental tool for data scientists, analysts, and developers.

- **Sklearn**

  Scikit-learn, often abbreviated as Sklearn, is a popular open-source machine-learning library for Python. It provides a wide range of tools for machine learning tasks such as classification, regression, clustering, dimensionality reduction, and model selection.

- **Numpy**

  NumPy, abbreviated for Numerical Python, serves as a crucial library for numerical operations in Python. It helps us work with multi-dimensional arrays and matrices, along with special math tools for easy handling. It is widely used in various fields like scientific computing, engineering, data analysis, and machine learning for computational tasks.

- **Flask**

Flask is like a toolkit for building websites and web apps with Python. It is known for being easy to use and flexible. It can be used to create small or medium-sized projects, like websites or APIs. It helps with things like figuring out where to send requests, managing user sessions, and working with databases. Flask falls into the category of microframeworks because it does not demand specific tools or libraries.

4. METHODOLOGY

4.1 System Architecture

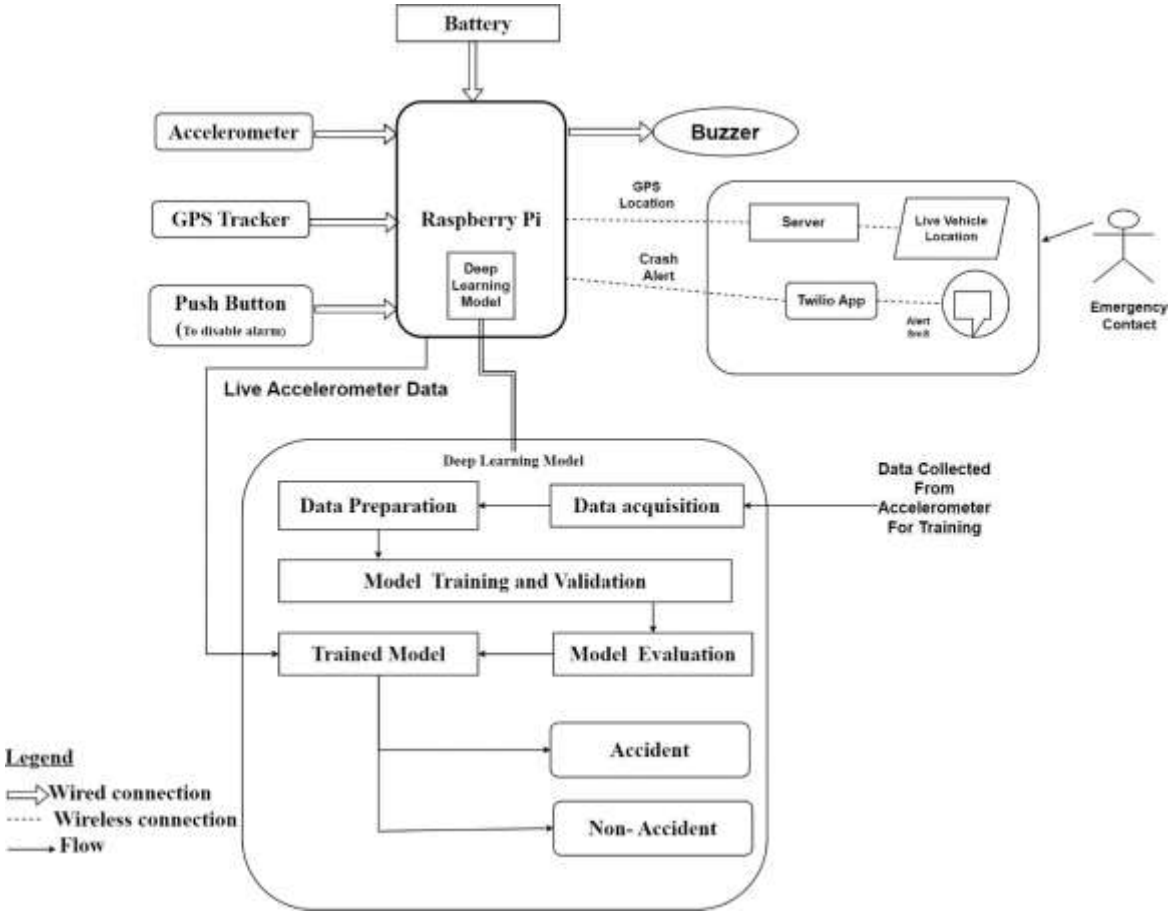


Figure 4-1: System Block Diagram

4.2 Working Principle

The overall system workflow is shown in Figure 4-1. The project's workflow begins with connecting the Raspberry Pi to an accelerometer, GPS tracker, and a push button. The GPS tracker continuously monitors the device's location, and this information is transmitted to a website for remote access and navigation. For crash detection, the accelerometer continuously sends data to the Raspberry Pi. An embedded deep learning model in the Raspberry Pi analyzes the accelerometer data using anomaly detection algorithms to identify potential crashes. Upon detecting outlier data, indicating a potential crash, an emergency buzzer is activated. But before taking any further actions, the buzzer gives the driver ten seconds to respond by pressing a safety button. This pre-alert phase enables the driver to indicate their safety in case of a false alarm by pressing a push button manually incorporated into the setup. If the driver presses the safety button, indicating they are unharmed, no emergency message is sent. However, if the driver does not press the safety button within ten seconds time frame, the system initiates an alert message to the emergency contact through the Twilio app. After sending an alert message, the current location of the vehicle can be viewed on the website.

This crash detection and alert system ensures the minimization of false alarms through driver intervention while providing timely assistance in case of emergencies.

4.3 Deep Learning Architecture

Deep learning is a method in Artificial Intelligence (AI) that teaches computers to process data in a way that is inspired by the human brain. Deep learning models can recognize complex patterns in pictures, text, sounds, and other data to produce accurate insights and predictions. Deep learning algorithms are neural networks that are modeled after the human brain. For example, a human brain contains millions of interconnected neurons that work together to learn and process information. Similarly, deep learning neural networks, or artificial neural networks, are made of many layers of artificial neurons that work together inside the computer. The components of a deep neural network are the Input layer, Hidden layers, and Output layer.

Building upon the foundational understanding of deep learning, our crash detection

system employs specialized architectures known as Recurrent Neural Networks (RNN) and Gated Recurrent Units (GRU).

4.4 Recurrent Neural Network

Recurrent Neural Networks (RNN) represent a type of Neural Network that introduces a unique approach by feeding the output from the preceding step as input to the current step. Unlike traditional neural networks, where inputs and outputs operate independently, RNNs prove beneficial in scenarios requiring the prediction of the next element in a sequence. This is particularly evident in tasks like predicting the next word in a sentence, where the context of previous words is crucial. RNNs address this requirement by incorporating a Hidden Layer, a key component that retains information about the sequence. Often referred to as the Memory State, this hidden state memorizes past inputs, contributing to the network's ability to remember and process sequential information. Notably, RNNs utilize the same set of parameters across all inputs, simplifying the complexity of parameters when compared to other neural networks.

In the context of our crash detection system, the use of Recurrent Neural Networks (RNN) is crucial. Traditional neural networks treat inputs and outputs independently, overlooking the importance of sequential information. However, when predicting potential crashes based on accelerometer data, the sequence of previous readings becomes crucial for accurate analysis.

By integrating the RNN architecture into our system, we enable it to analyze past accelerometer readings, essentially creating a memory of the events leading up to the present moment. This sequential memory, housed in the hidden state of the RNN, proves invaluable for identifying patterns and anomalies in the accelerometer data. Consequently, the RNN's capacity to utilize this contextual information significantly improves the accuracy of crash predictions within our project.

In the image below, we can see a basic diagram that illustrates the basic architecture of an RNN:

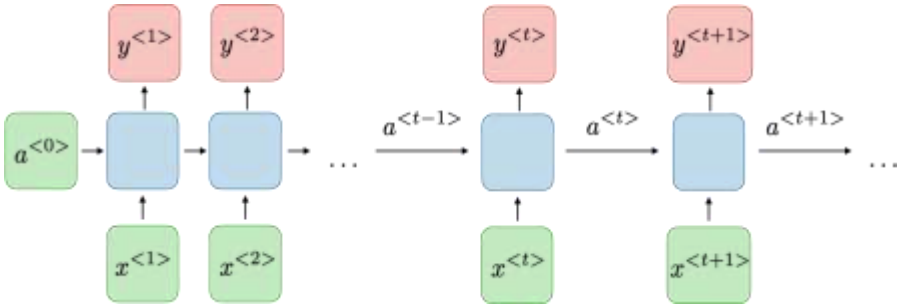


Figure 4-2: Basic Architecture of RNN[14]

Every RNN consists of a series of repeating modules that are called cells and process the input data sequentially. That means that the first cell takes as input the first sample of the sequence, the second cell takes the second sample, and so on. Each cell i takes the input vector $x^{<i>}$ and, after some processing, generates a vector known as a hidden state $a^{<i>}$ that is passed to the next cell i +1. That means that each time the hiddenstate $a^{<i>}$ captures all the information given so far, enabling the network to have some memory.

4.5 Gated Recurrent Unit

The GRU architecture is particularly well-suited for the temporal nature of accelerometer data. Its ability to capture sequential dependencies and retain long-term information is crucial for discerning the subtle patterns that precede a vehicular crash. In our project, the GRU network is employed as a sequence model to analyze the time-series data generated by the accelerometers.

Alongside GRUs, Long Short-Term Memory (LSTM) networks stand out as a prominent architecture in the domain of sequential data processing. Similar to GRUs, LSTMs are designed to address the challenges posed by vanishing gradients in traditional RNNs and to capture long-term dependencies in sequential data.

Gated Recurrent Units's gated structure comprises of the update and reset gates which are explained below.

Update Gate:

The update gate governs the flow of information from the previous time step,

determining how much of the past information should be retained. In our context, this helps the model in adapting to changing dynamics in the accelerometer data. It allows the model to selectively update its memory, emphasizing relevant information for accurate event prediction. The equation of the update gate is shown below.

$$ut = \sigma(xt * Uu + Ht - 1 * Wu) \quad 4.1$$

Reset Gate:

The reset gate complements the update gate by deciding how much of the past information should be forgotten or reset. This functionality is crucial forpreventing the model from relying too heavily on outdated patterns. The reset gate ensures a dynamic adaptation to evolving conditions, allowing the GRU network to stay responsive to the real-time dynamics of the vehicle. It is responsible for the short-term memory of the network i.e., the hidden state (H_t).Here is the equation of the Reset gate.

$$rt = \sigma(xt * Ur + Ht - 1 * Wr) \quad 4.2$$

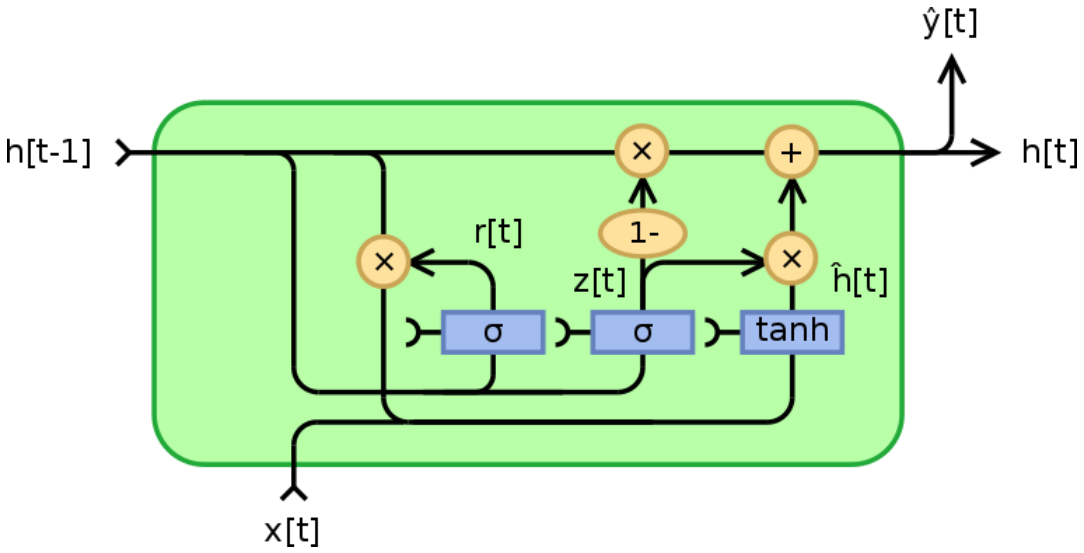


Figure 4-3: GRU model[14]

A two-step procedure is employed to derive the hidden state (H_t). The first state known as candidate hidden state involves taking the input and the hidden state from the preceding timestamp (t-1), which is then multiplied by the output of the reset gate (r_t). Subsequently, this information is fed into the hyperbolic tangent (tanh) function.

$$\widehat{H}_t = \tanh(x_t * U_g + (r_t \circ H_{t-1}) * W_g \quad 4.3$$

If the value of rt is equal to 1 then it means the entire information from the preceding hidden state H_{t-1} is being considered. Conversely, if rt equals 0, it implies that the information from the prior hidden state is completely ignored.

After obtaining the candidate state, it is used to formulate the present hidden state, H_t.

$$H_t = u_t \circ H_{t-1} + (1 - u_t) \circ \widehat{H}_t \quad 4.4$$

Consider a case where the value of u_t is close to zero. The first term in the equation disappears in this instance, suggesting that not much information from the previous hidden state will be taken into the current hidden state. On the other hand, the second component is almost equal to one, indicating that data from the candidate state will make up the majority of the hidden state at this timestamp.

Conversely, if u_t is approximately 1, the second term becomes zero. As a result, the information from the hidden state at the prior timestamp (t-1) will be the primary factor that determines the present hidden state, as it will only depend on the first term.

4.6 Autoencoder

Autoencoders are a fundamental concept in machine learning, particularly in the realm of unsupervised learning and data compression. They serve as powerful tools for learning compact representations of data, enabling tasks such as dimensionality reduction, data denoising, and anomaly detection. An autoencoder is a type of neural network consisting of two main components: an encoder and a decoder. The encoder compresses the input data into a lower-dimensional representation, while the decoder reconstructs the original input from this compressed representation. By training the autoencoder to minimize the reconstruction error, it learns to capture meaningful features from the input data.

The encoder typically consists of several layers of neurons that progressively reduce the dimensionality of the input data. The decoder mirrors the architecture of the encoder, with layers that progressively expand the compressed representation back to the original input dimensions. Common activation functions like sigmoid are used in the layers, and the choice of architecture can vary depending on the specific task and data characteristics.

During training, the autoencoder is fed with input data, and the encoder produces a compressed representation. The decoder then reconstructs the input data from this representation. The difference between the input and the reconstructed output is measured using a loss function, such as mean squared error or binary cross-entropy, which guides the training process via backpropagation.

In summary, autoencoders are versatile tools with numerous applications in machine learning and data analysis. By understanding their principles and best practices, we can leverage autoencoders to extract valuable insights from our data and tackle a wide range of real-world problems.

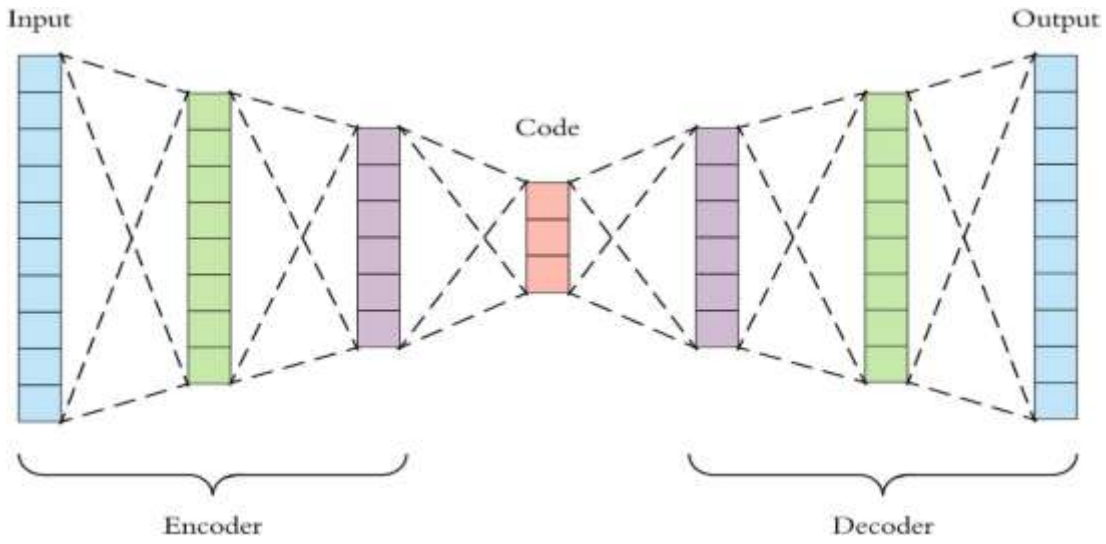


Figure 4-4: Working of Autoencoder

4.7 Autoencoder for Anomaly Detection

Autoencoders are commonly used for anomaly detection due to their ability to learn a compact representation of normal data. The general structure of an autoencoder is based on mapping an input vector x to an output vector r = g(h) (a reconstruction) through an internal representation - a latent space vector h = f(x). The autoencoder has two components: the encoder f (mapping x to h) and the decoder g (mapping h to r). The learning process is described simply as minimizing a loss function:

$$L(x, g(f(x))) \qquad 4.5$$

Various steps included in anomaly detection are described below:

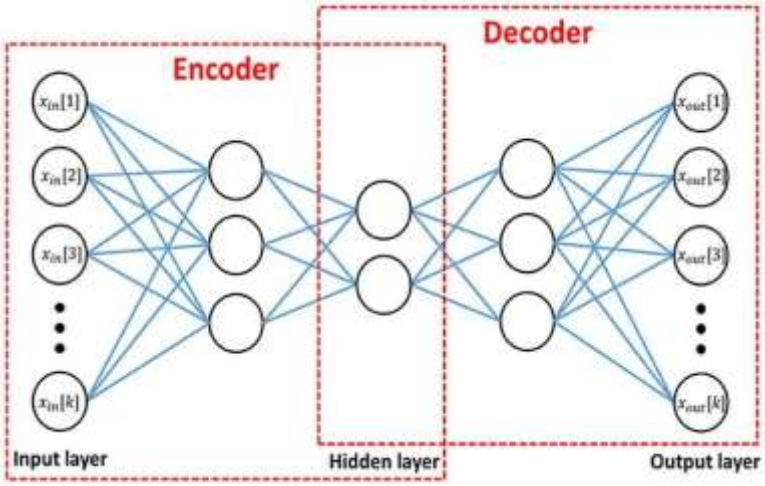


Figure 4-5: Scheme for an Autoencoder[15]

- Firstly, train the model with normal data (non-anomalous data).
- Use a loss function that measures the reconstruction error between the input and the output of the autoencoder.
- During training, the model learns to reconstruct normal data accurately,capturing its underlying patterns and structure.
- After training, use the trained autoencoder to reconstruct both normal and anomalous data.
- Calculate the reconstruction error for each data point by comparing the input with its reconstructed output.
- The reconstruction error represents how well the autoencoder can reconstructthe

input data.

- Set a threshold on the reconstruction error to distinguish between normal and anomalous data.
- Data points with reconstruction errors above the threshold are classified as anomalies, while those below the threshold are considered normal.
- Evaluate the performance of the autoencoder-based anomaly detection system on a separate validation or test dataset.

4.8 Deep Learning Pipeline

A deep learning pipeline is a way to codify and automate the workflow it takes to produce a learning model. A deep learning pipeline contains multiple sequential steps that do everything from data acquisition to model training. It takes raw data and transforms it into a model that makes intelligent decisions. The various steps of deep learning pipeline are described below.

4.8.1 Data Acquisition

Data acquisition involves gathering the data necessary to train a model, encompassing tasks like identifying sources, collection methods, and data formats. In our scenario, we've acquired a dataset using a remote-controlled car, focusing solely on normal driving conditions.

4.8.2 Data Preparation

Data preparation entails organizing the datasets needed for our project. In this case, we've extracted three parameters, namely acc_x, acc_y, and acc_z, from the collected datasets in .csv format.

4.8.3 Model Training and Validation

Selecting the appropriate model for training is crucial. In our case, we're utilizing GRU and autoencoders for training our model. This phase involves multiple iterations to effectively train the model. Validating the model is a crucial stage in deep learning model development, ensuring that it performs as expected and can generalize well to unseen data. Essentially, it is the process of confirming that the model accomplishes its

intended objectives.

4.8.4 Model Evaluation

Model evaluation compromise of using various evaluation metrices to understand the deep learning model performance. It includes the checking of strength and weakness of a model. Some of the evaluation metrices are:

- Accuracy
- Precision
- F1 score
- Recall
- Mean Average Error (MAE).

4.8.5 Trained Model

It is the ultimate product of the deep learning pipeline that is used to detect the crash comparing accelerometer data which determines accidental or non-accidental cases in our system. It is an outcome of deep learning model after performing multiple steps mentioned above.

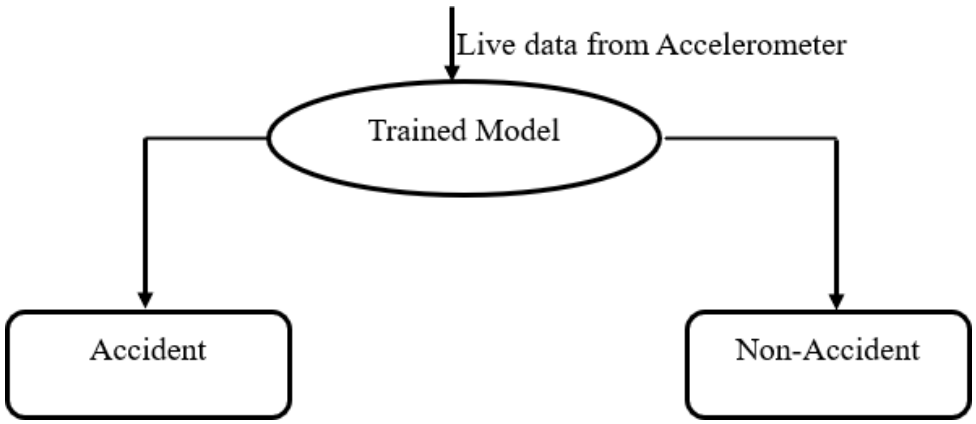


Figure 4-6: Accident or Non-accident Detection

4.9 Performance Metrices

4.9.1 Confusion Matrix

Confusion matrix is a performance measurement for machine learning classification problem where output can be two or more classes. It a table with 4 different combinations of predicted and actual values. A confusion matrix is a grid of information that shows the number of True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN) returned when applying a test set of data to a classification algorithm.

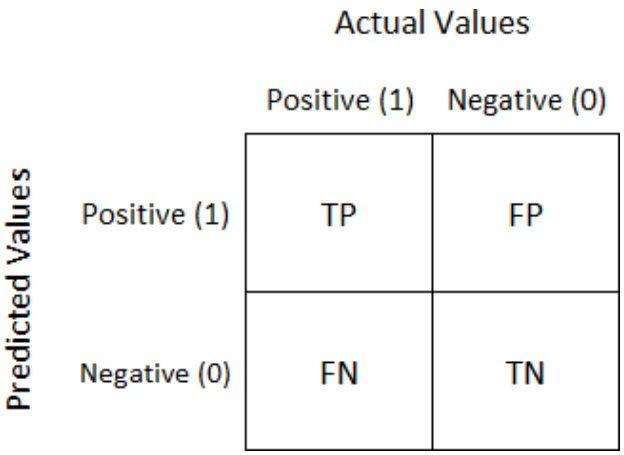


Figure 4-7: Confusion Matrix for Binary Classification

In the context of time series data, a confusion matrix is a tool used to evaluate the performance of a predictive model by comparing its predictions against the actual values over time. It is typically used in classification tasks where the model predicts the class or label of each data point in the time series.

Here's how a confusion matrix can be adapted for time series data:

True Positives (TP): These are the instances where the model correctly predicts a positive class or label, and the actual value is also positive.

True Negatives (TN): These are the instances where the model correctly predicts a negative class or label, and the actual value is also negative.

False Positives (FP): These are the instances where the model incorrectly predicts a

positive class or label, but the actual value is negative.

False Negatives (FN): These are the instances where the model incorrectly predicts a negative class or label, but the actual value is positive.

- **Accuracy**

  Accuracy is a fundamental performance metric used to evaluate the overall effectiveness of a predictive model in binary classification tasks. It measures the proportion of correctly classified instances, regardless of their class, over time series data. Accuracy is calculated as the number of all correct predictions divided by the total number of the dataset. The best accuracy is 1.0, whereas the worst is 0.0.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \quad 4.6$$

- **Precision**

  Precision is another important performance metric used in binary classification tasks to evaluate the quality of a predictive model, especially in situations where the positive class is of particular interest. In the context of time series data, precision measures the accuracy of the model's positive predictions over time. Precision tells us how many of the correctly predicted cases actually turned out to be positive.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad 4.7$$

  This would determine whether our model is reliable or not. The best precision is 1.0, whereas the worst is 0.0.

- **Recall**

  Recall, also known as sensitivity or true positive rate, is a performance metric used to evaluate the effectiveness of a predictive model, particularly in binary classification tasks. In the context of time series data, recall measures the ability of the model to correctly identify all positive instances over time. Recall tells us how many of the actual positive cases we were able to predict correctly with our model. Recall should be high as possible.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (4.8)$$

- **F1-Score**

  F1-Score is a harmonic mean of Precision and Recall, and so it gives acombined idea about these two metrics. It is maximum when Precision is equal to Recall. The interpretability of the F1-score is poor. This means that we don't know what our classifier is maximizing – precision or recall. So, we use it in combination with other evaluation metrics, giving us a complete picture of the result.

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}} \quad (4.9)$$

4.9.2 Mean Absolute Error

Mean Absolute Error (MAE) is a metric used to evaluate the performance of a regression model. It measures the average absolute difference between the predicted values and the actual values.

Here's how MAE is calculated:

For each data point, calculate the absolute difference between the predicted value (y_pred) and the actual value (y_true). Take the average of all these absolute differences to get the Mean Absolute Error. Mathematically, the formula for MAE is:

$$\text{MAE} = \sum_{\text{i=1}}^{\text{n}} \frac{|\text{y}_{\text{true,i}} - \text{y}_{\text{pred,i}}|}{\text{n}} \quad (4.10)$$

Where:

- n is the number of data points.
- $y_{true,i}$ is the actual value of the i^{th} data point.
- $y_{pred,i}$ is the predicted value of the i^{th} data point.

MAE provides a straightforward interpretation on average, how far off are the predictions from the actual values, without considering the direction of the errors. A lower MAE indicates better model performance, as it means the model's predictions are closer to the actual values.

4.10 Flowchart

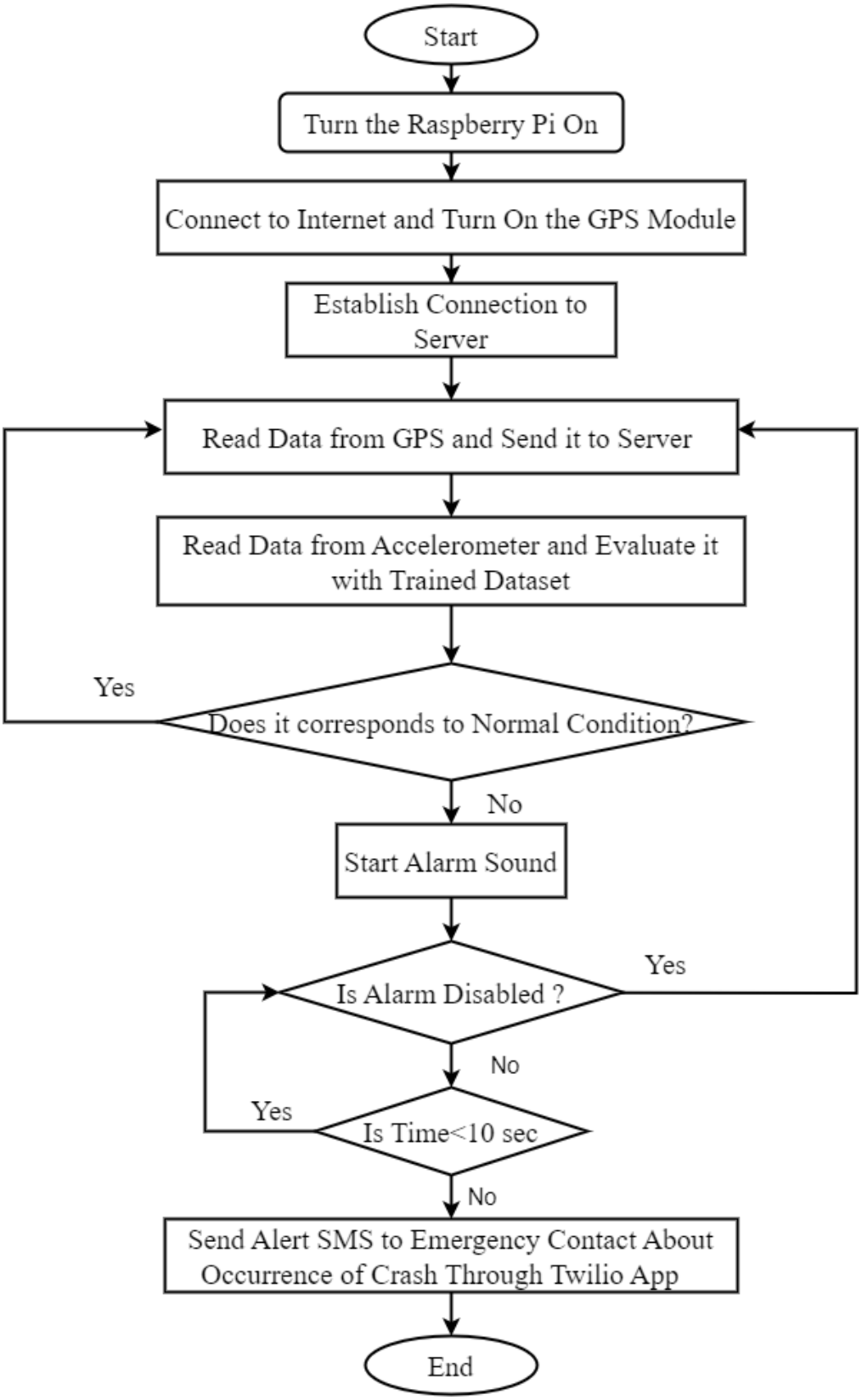


Figure 4-8: Flowchart of System Architecture

The system begins by powering up the Raspberry Pi with a battery and initializing the GPS module. Subsequently, it establishes a connection to a website. Live location data from the GPS module is then transmitted to the website.

The next phase involves reading data from the accelerometer and comparing it with a pre-trained dataset. If the comparison indicates an accident, a buzzer is activated. The user has the option to deactivate the buzzer by pressing a push button in non-accident scenarios. In cases where the push button remains unpressed for 10 seconds after an accident detection, the system automatically forwards the live location and a crash alert to an emergency contact through the Twilio app.

This multi-phase setup ensures a comprehensive approach to accident detection and emergency response, enhancing user safety.

5. DATASET ANALYSIS

To train our deep learning model, we chose not to rely on accelerometer data from crash conditions, as finding such datasets online is challenging and collecting them ourselves would be even more difficult. Instead, we collected our own data using an R/C car under normal driving conditions, considering it to be a more practical approach.

5.1 Dataset Collection

The setup had a Raspberry Pi and an accelerometer powered by a constant 5V power supply attached to the module. We collected data from the accelerometer while the car was driven including the conditions like braking, turning left or right, and going reverse.

Figure 5-1: Setup for Dataset Collection

5.2 Dataset Preprocessing

All the data collected were then filtered manually to retrieve appropriate data and after filtering, 44,723 instances of data were used for training and testing. The data were transformed by standard scaling to ensure that all features have same scales. A snippet of collected dataset is shown in Figure 5-2.

acc_x	acc_y	acc_z
0.37	0.17	9.79
0.39	0.18	9.77
0.41	0.18	9.73
0.37	0.16	9.75
0.39	0.16	9.87
0.34	0.19	9.8
0.33	0.15	9.75
0.36	0.16	9.75
0.39	0.16	9.76
0.36	0.14	9.81
0.34	0.15	9.77
0.39	0.15	9.78
0.39	0.16	9.8
0.34	0.17	9.74
0.34	0.16	9.79
0.35	0.16	9.76
0.37	0.12	9.8
0.39	0.17	9.83
0.34	0.18	9.79
0.37	0.14	9.77
0.36	0.14	9.82
0.35	0.21	9.8
0.35	0.19	9.79
0.36	0.16	9.82

Figure 5-2: Portion of Collected Dataset

5.3 Dataset Visualization

The dataset was visualized after completing preprocessing by plotting dataset in different graphs using *'matplotlib.pyplot'* library of TensorFlow.

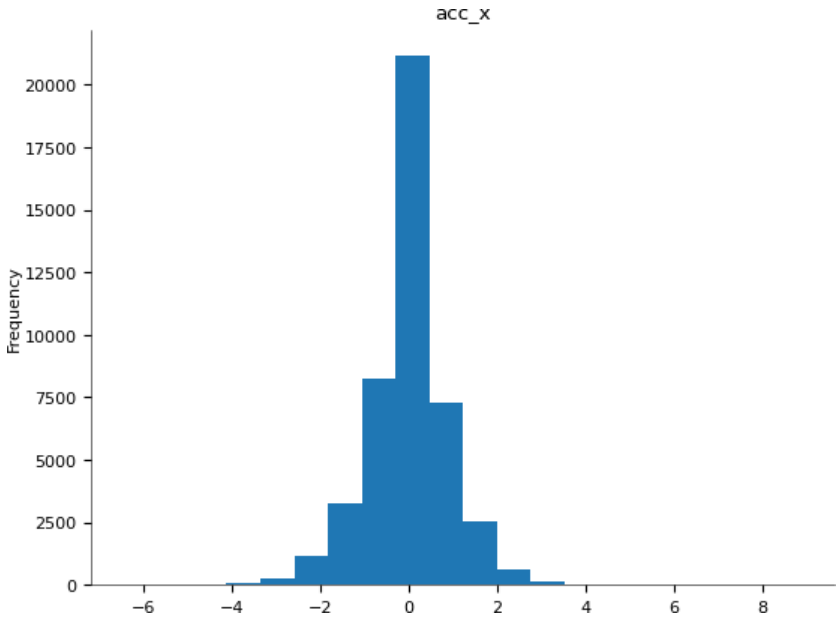


Figure 5-3: Distribution of X-axis of Accelerometer

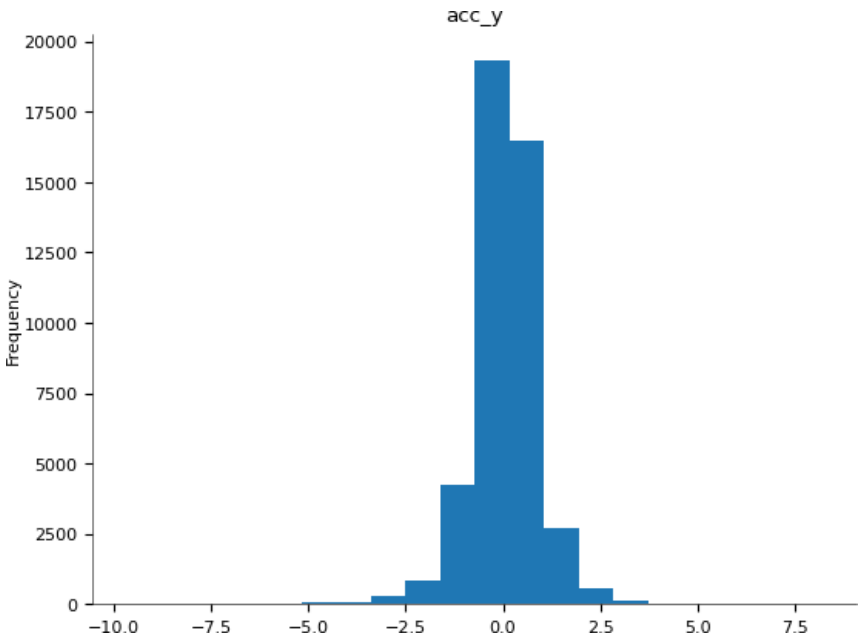


Figure 5-4: Distribution of Y-axis of Accelerometer

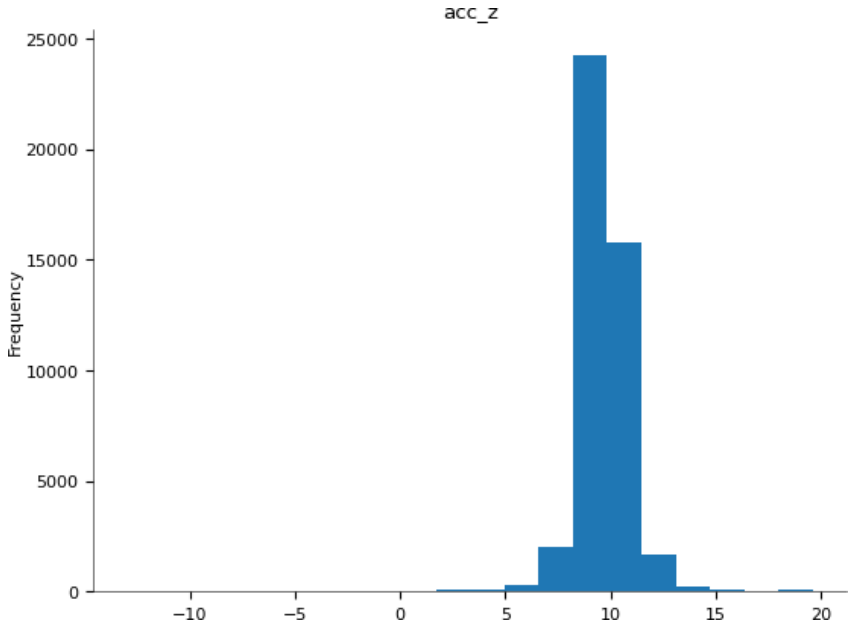


Figure 5-5: Distribution of Z-axis of Accelerometer

The graph below displays the accelerometer data for the X, Y, and Z axes, limited tothe first 1000 data points.

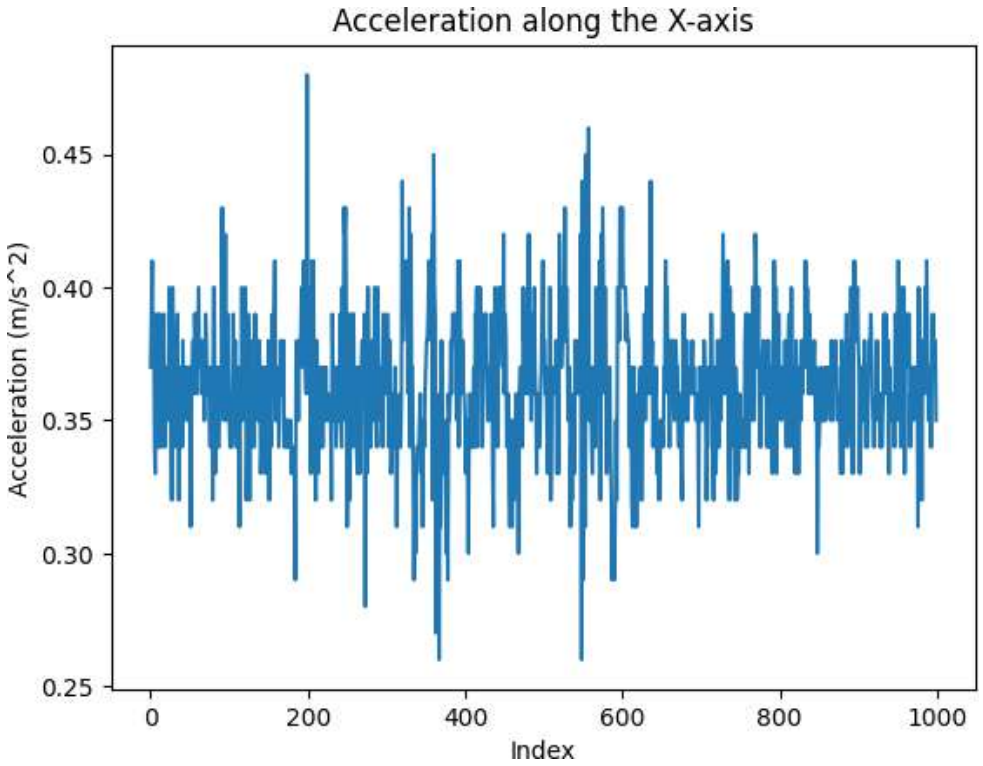


Figure 5-6: Acceleration along X-axis

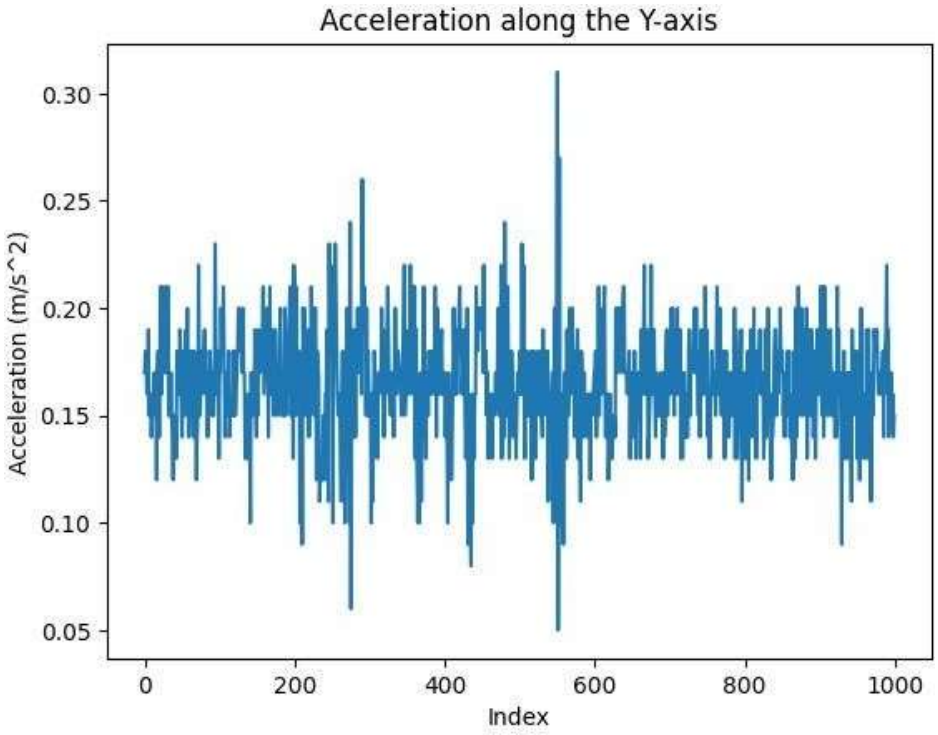


Figure 5-7: Acceleration along Y-axis

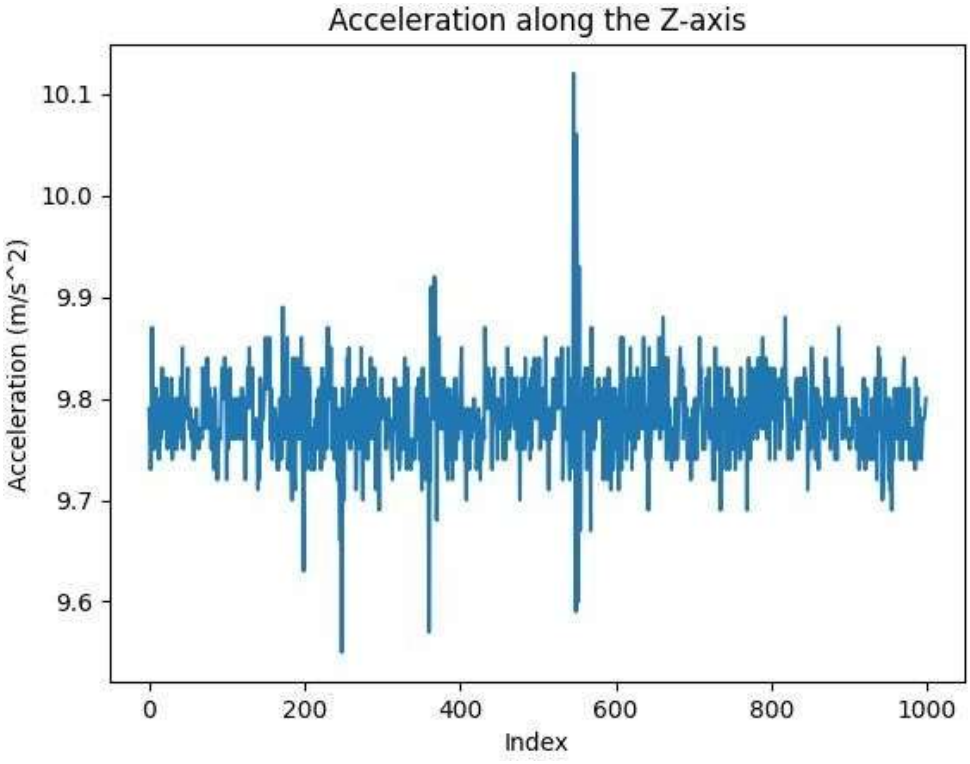


Figure 5-8: Acceleration along Z-axis

5.4 Dataset Splitting

In this phase, the collected data were split into training and testing sets. 85% of the dataset was taken for training and the remaining dataset was used for testing.

$$split\ index = int(0.85 * len(df)) \quad 5.1$$

5.5 Dataset Scaling and Shaping

As GRU works better with a scaled data, Standard Scaling was used to scale the data before feeding it to the GRU-Autoencoder model. Standard Scaling standardizes features by removing the mean and scaling to unit variance. The formula for Standard Scaling is:

$$x = \frac{acc_x - \overline{a_x}}{\sigma(acc_x)} \quad 5.2$$

$$y = \frac{acc_y - \overline{a_y}}{\sigma(acc_y)} \quad 5.3$$

$$z = \frac{acc_z - \overline{a_z}}{\sigma(acc_z)} \quad 5.4$$

Where:

- acc_x, acc_y, and acc_z are the original data points in x, y and z axes respectively.
- $\overline{a_x}$, $\overline{a_y}$ and $\overline{a_z}$ are mean of the data points in the dataset.
- $\sigma(acc_x)$, $\sigma(acc_y)$, and $\sigma(acc_z)$ are the standard deviation of the data points in the dataset.
- x, y, and z are the scaled values of the data points acc_x, acc_y, and acc_z respectively.

Similarly, a GRU layer expects the input data in the shape of 3D array [batch, timesteps, features]. Here, 'batch' is the number of samples or data points, 'timesteps' is the sequence length in each input which denotes the number of data points taken as one time series data and 'features' is the number of variables in each timestep, which is 3 in this case as accelerometer measures the values along 3 axes. We chose a timestep value of 5 for our GRU model because accelerometer data is dynamic, and patterns in this data typically don't persist for long periods. Setting a higher timestep value could lead to overfitting, as it might capture short-lived fluctuations as significant patterns.

6. IMPLEMENTATION DETAILS

6.1 Raspberry Pi

The project begins with integrating a Raspberry Pi with various components, including an accelerometer, GPS tracker, and a push button. The Raspberry Pi utilizes its 2.4 GHz dual-mode Wi-Fi and Bluetooth capabilities for connectivity. Continuously, the GPS tracker records the device's location, which is then transmitted to a server via Wi-Fi. In addition to real-time navigation, the Raspberry Pi includes functionality for emergency contact notifications. In case of emergencies, the Raspberry Pi sends an alert to designated emergency contacts using the Twilio app.

6.2 Interfacing MPU6050 With Raspberry Pi

To connect the MPU6050 sensor to the Raspberry Pi, we match voltage levels (3.3V) and link the VCC and GND pins to their counterparts. This establishes the necessary electrical connection. Next, we initiate the I2C (Inter-Integrated Circuit) bus and accelerometer module using the Circuit Python library. The *'busio.I2C'* module is used to initialize the I2C bus, with board. The SDA (Serial Data Line) and SCL (Serial Clock Line) pins of the MPU6050 are connected to corresponding GPIO pins. Using the I2C protocol, the Raspberry Pi sends commands and requests motion data from the sensor. Through this communication, the Raspberry Pi receives motion data from the MPU6050, which measures movement along three axes.

6.3 Interfacing GPS NEO 7M With Raspberry Pi

Interfacing the GPS NEO-7M module with the Raspberry Pi enables location tracking and navigation capabilities within our project. The GPS NEO-7M module, equipped with a receiver, provides positioning data using signals from Global Navigation Satellite Systems (GNSS) such as GPS.

To establish the connection between the GPS NEO-7M and the Raspberry Pi, we utilize UART communication. The NEO-7M module typically communicates via serial communication, making it compatible with the UART (Universal Asynchronous Receiver-Transmitter) interface present on the Raspberry Pi. The Raspberry Pi is programmed using Python, with libraries installed for the GPS NEO-7M module. The

code initializes UART communication with the GPS module, extracts location data, and utilizes functions to establish internet connectivity and transmit the data to the designated server. On the server side, we set up a server application to receive the GPS location data from the Raspberry Pi and process it accordingly.

6.4 Interfacing Wi-Fi with Raspberry Pi

Configuring a Raspberry Pi to connect to a Wi-Fi network follows a straight process. The device requires configuration with the essential Wi-Fi credentials, including the network name (SSID) and password. Raspberry Pi's operating system typically includes Wi-Fi management tools that streamline this connection process. In the code written using Python, Wi-Fi credentials are set using libraries such as '*wpa_supplicant'* or *'netplan'*. The Raspberry Pi then initiates a connection attempt to the designated Wi-Fi network.

6.5 Interfacing Push Button and Piezo Buzzer with Raspberry Pi

To interface a push button and a buzzer with a Raspberry Pi, the push button can be connected to a GPIO pin configured as an input and ground, while the buzzer can be connected to another GPIO pin configured as an output and a voltage source using a current-limiting resistor (10 ohms). The push button state can be detected using '*GPIO.input()*' and the buzzer state can be toggled accordingly using '*GPIO.output()*'.

6.6 GRU Autoencoder Model for Crash Detection

In our crash detection project, we utilized a GRU (Gated Recurrent Unit) autoencoder model. Initially, we gathered accelerometer data under normal conditions. Following data cleaning and preprocessing, we devised the architecture of the GRU autoencoder, specifying input, encoding, decoding, and output layers. Subsequently, we trained the model to reconstruct input data while minimizing reconstruction error.

After completing the training, we assessed the model's performance using a separate testing dataset. With a threshold error of 3 established, the model was trained to detect outlier data in real time, alerting potential accidents when the error exceeds the threshold.

6.7 Circuit Diagram

The circuit diagram below outlines the setup for acquiring accelerometer data via the MPU6050 sensor and concurrently tracking the real-time location of the vehicle using the GPS NEO 7M module. Additionally, the circuit includes a buzzer that activates upon accident detection.

The driver has the option to disable the buzzer if it's a false alarm. However, if the buzzer remains enabled for more than 10 seconds, the system identifies it as an accident and proceeds to notify the emergency contact with the SMS and location details.

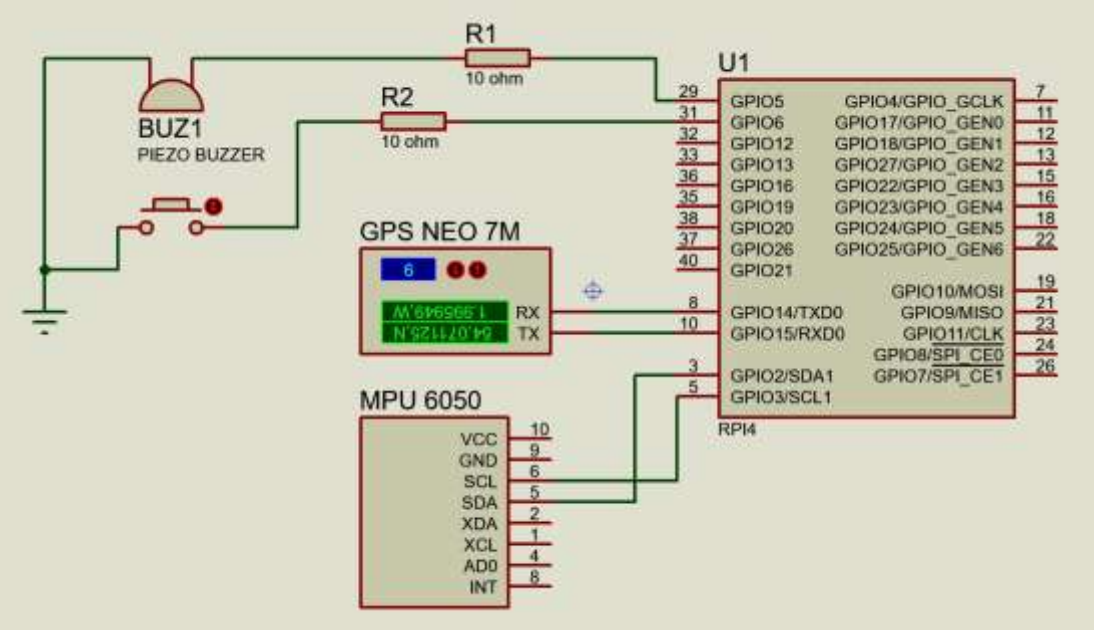


Figure 6-1: Circuit Diagram

6.8 Interfacing Website for Location Tracking

The GPS continuously transmits its location data to the Raspberry Pi, which in turn communicates this information to the server through HTTP (Hypertext Transfer Protocol) and sends the data to the website via Wi-Fi. The website then displays the real-time location of the vehicle on an open map based on the longitude and latitude coordinates sent by the Raspberry

6.9 Use Case Diagram

A use case diagram is used primarily in software engineering to represent the functionality of a system from a user's perspective. It serves as a graphical representation of functional requirements. Here is a simplified use case diagram for the GPS-based vehicle tracking and accelerometer-driven crash detection system: In this use case diagram, Driver represents the people who are using this crash detectionsystem. This system tracks the vehicle location as well as alerts emergency contact about the accident in case any accident is detected. The system also contains a push button to control false alert which gives drivers some time to disable the alarm.

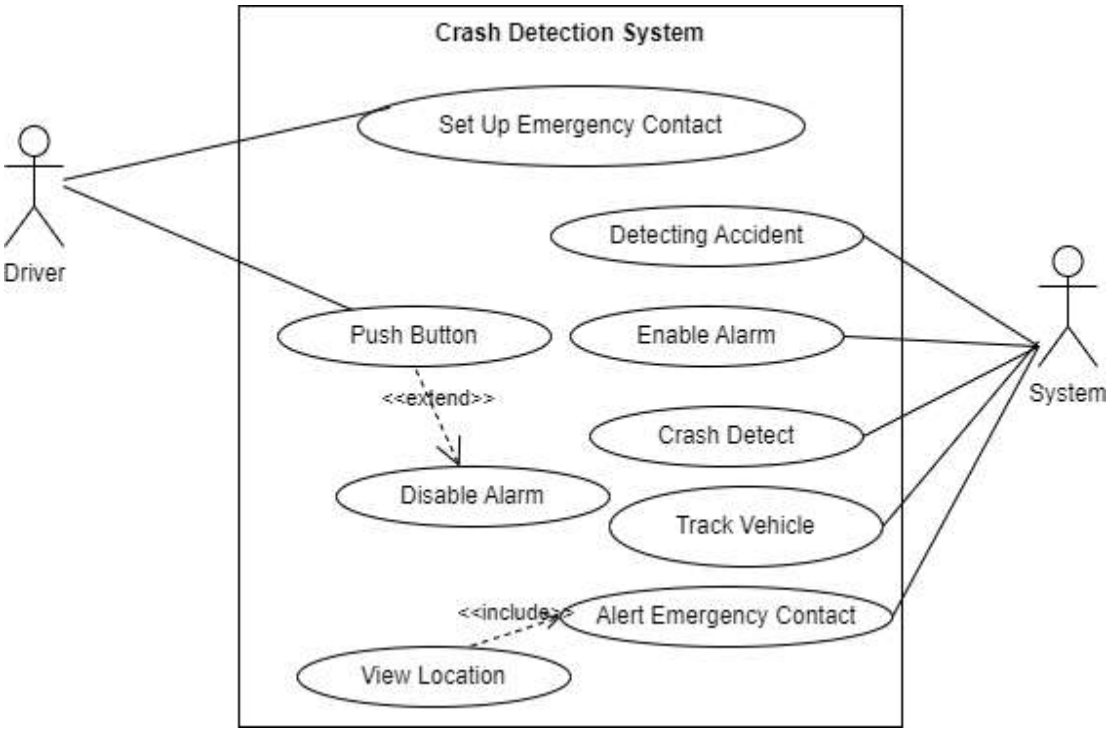


Figure 6-2: Use Case Diagram

6.10 Sequence Diagram

A sequence diagram is a type of interaction diagram that illustrates how objects interact in a particular scenario of a use case. It depicts the sequence of messages exchanged between the objects or participants involved in the scenario over time. In this Sequence Diagram, the driver initiates a crash detection system. When the system finds the case of the crash from accelerometer readings, the alarm goes off. If itis not a crash, the alarm is disabled by the user using the push button. When the buttonis not pressed for a certain allocated time, the system notifies the emergency contact about the accident with the location of the vehicle.

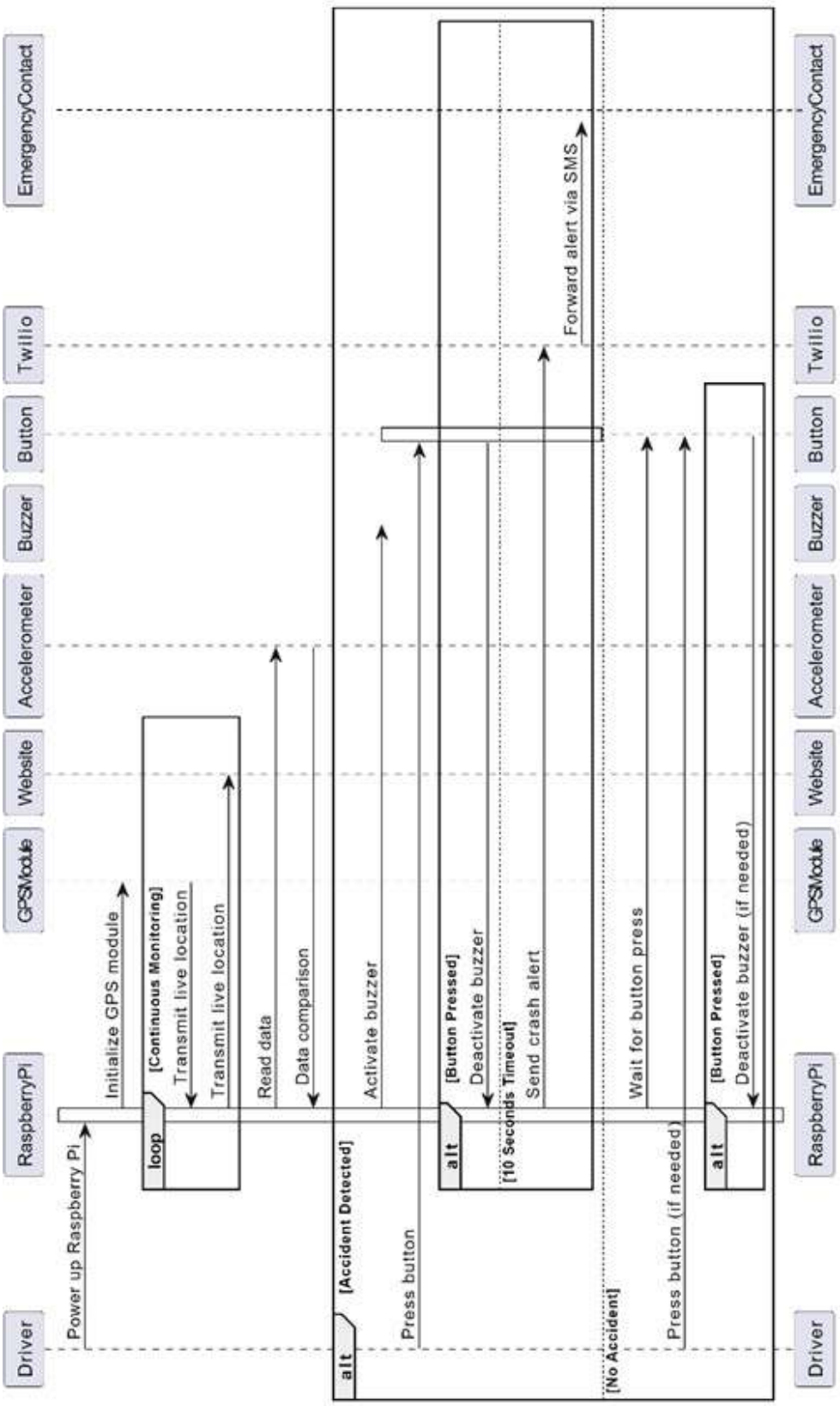


Figure 6-3: Sequence Diagram

6.11 Activity Diagram

An activity diagram is a type of behavioral diagram in the Unified Modeling Language (UML) that illustrates the flow of activities within a system or process. It visually represents the workflow or sequence of actions and decisions in a system from start to finish. Activity diagrams provide a structured way to visualize the flow of activities and decisions within a system or process, facilitating analysis, design, and communication among project stakeholders.

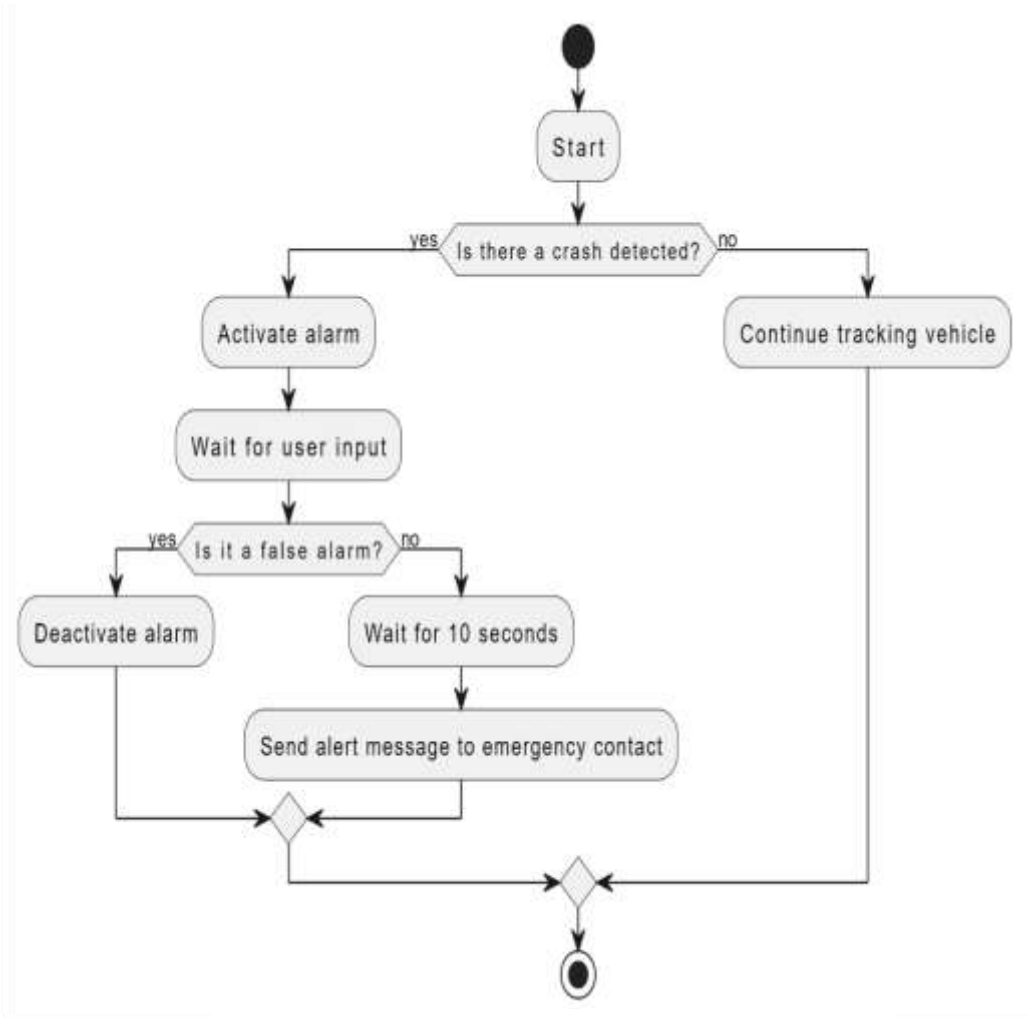


Figure 6-4: Activity Diagram

This activity diagram outlines the steps involved in the crash detection process. It begins with the initialization of the process and then proceeds with acquiring accelerometer data, preprocessing it, and applying the autoencoder algorithm. If a crash is detected, the system checks if the false alarm button is pressed. If not, it alerts the emergency services; otherwise, it concludes that no crash is detected.

7. RESULTS AND ANALYSIS

7.1 Real-Time GPS tracking With IoT Integration

The data obtained from the GPS tracker is sent to the Website and can be seen by the emergency contact whenever the device is turned on.

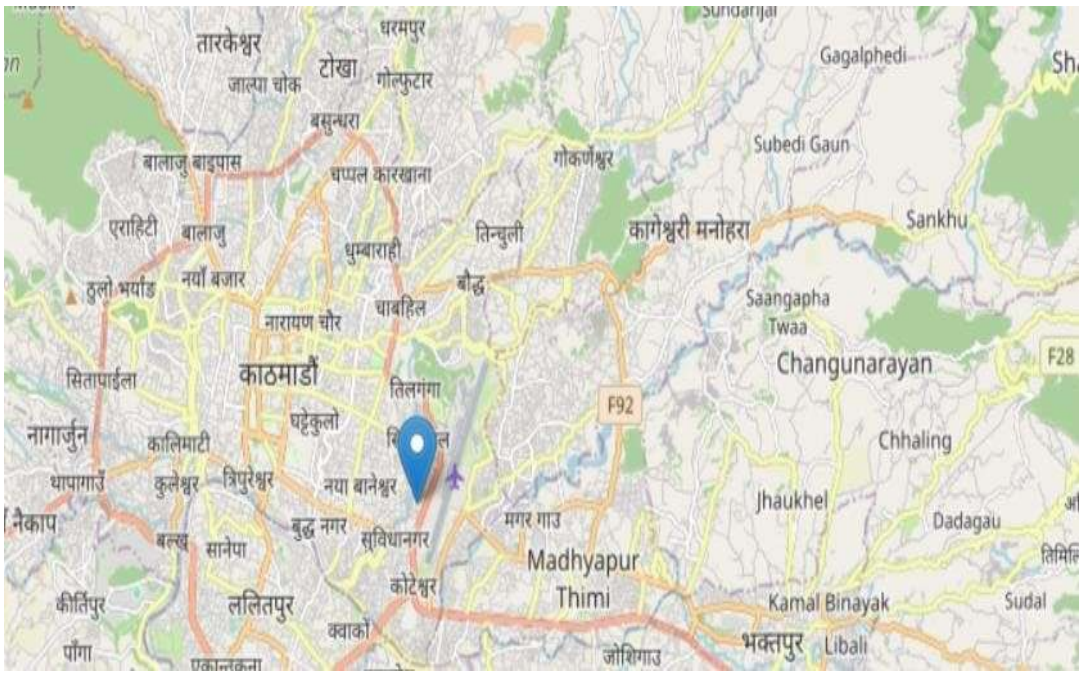


Figure 7-1: Real-Time Tracking of Vehicle

7.2 Hardware Setup for Vehicle Tracking and Crash Detection

All the hardware components are integrated into a matrix board with Raspberry Pi as the central component. The components GPS tracker, buzzer, accelerometer, and push button have been soldered in the matrix board along with Raspberry Pi attached and connected to it. This is the hardware part that will be kept in the vehicle. Raspberry Pi is connected to the network through which the data from GPS is seen as shown in the figure. Also, an alert SMS will be sent to an emergency contact using Twilio API. It takes the accelerometer reading to detect an accident if it occurs.

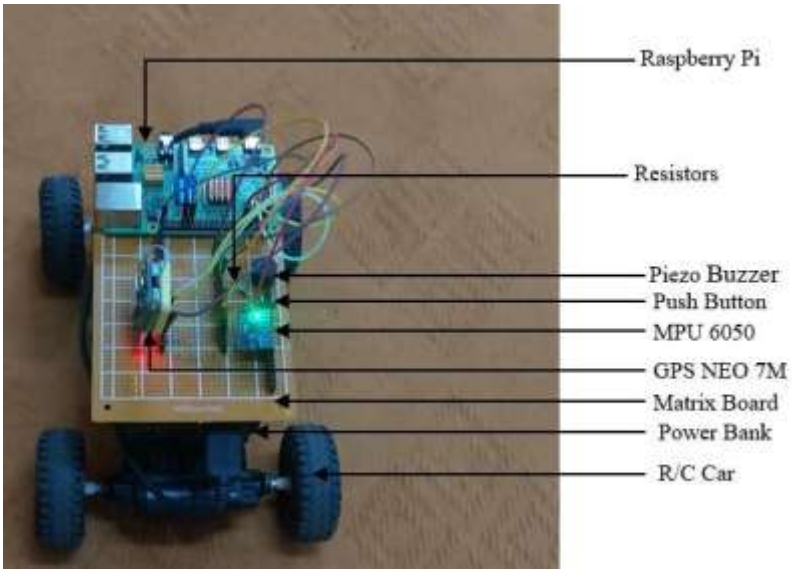


Figure 7-2: Hardware Setup

7.3 Alert SMS To Emergency Contact

If the buzzer is not disabled by pressing the push button, the system identifies it as an accident and proceeds to send an alert SMS to the emergency contact. The format of the alert SMS is as in the provided figure.

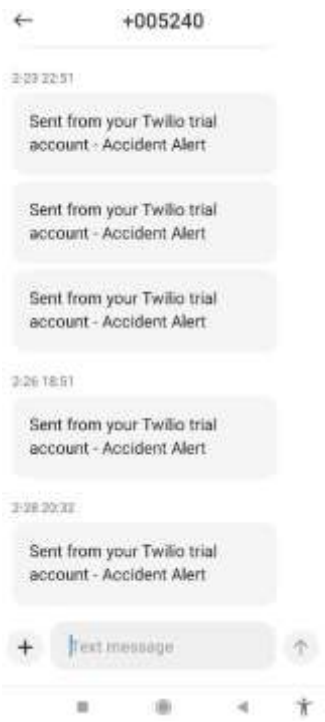


Figure 7-3: Alert SMS sent to Emergency Contact

7.4 Loss Curve

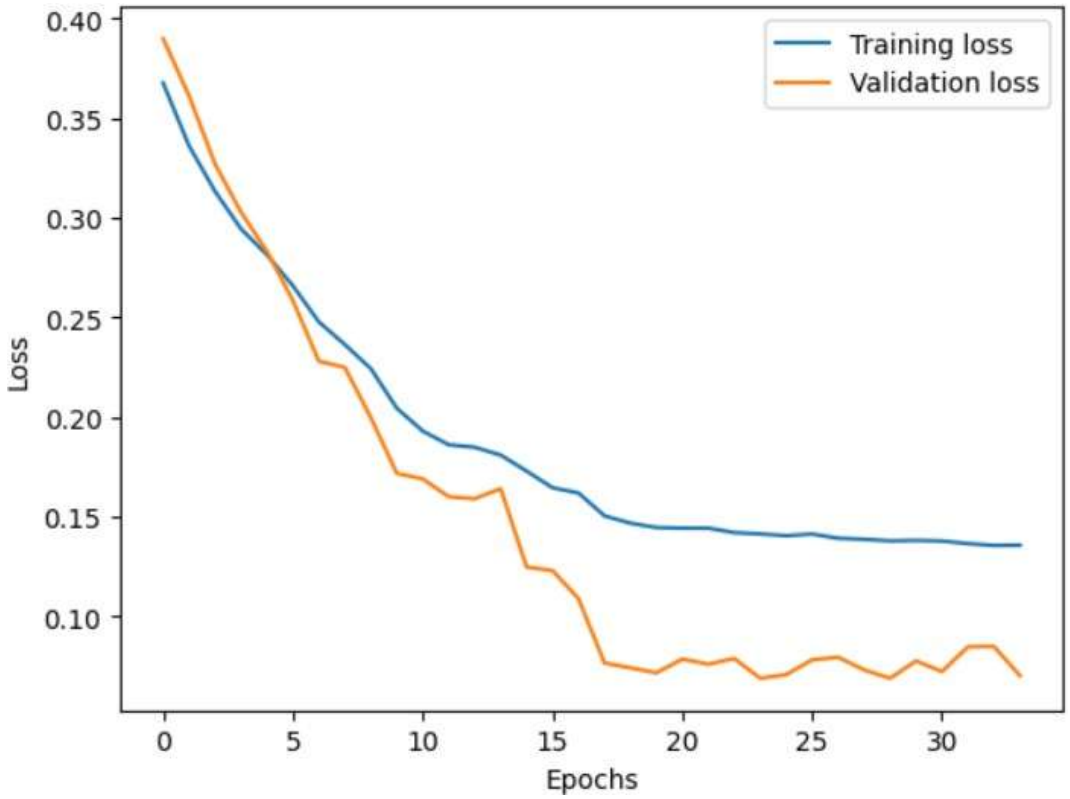


Figure 7-4: Loss Curve

The loss curve, also known as the training loss curve, depicts the trend of the loss function's value over epochs or iterations during the training process. The validation curve, also called the validation loss curve or performance curve, illustrates the model's performance on a separate validation dataset during training. The above result is based on 40 epochs training with 32 batch size with shuffle set false. Early Stopping Callback is also enabled which stops the model training if there is no improvement in validation loss over five epochs. In our model, the validation loss cross training loss at about 5 epochs. The validation loss starts to be constant at 20 epochs.

7.5 Anomaly Detection

We trained our model using regular driving data, treating any outliers as potential accidents. So, when a new data is input to the model, it reconstructs the data given to it based on the patterns learnt by the model from the training data. An anomalous dataset yields a high reconstruction MAE. By setting a threshold error, we flag readings surpassing this threshold as accidents. Therefore, we define any deviations from typical

driving conditions as crashes or anomalies. The below-depicted histogram shows a reconstruction error in the validation dataset. For validation dataset, we collected about 1700 data by driving the RC car in different location from the one where it was trained.

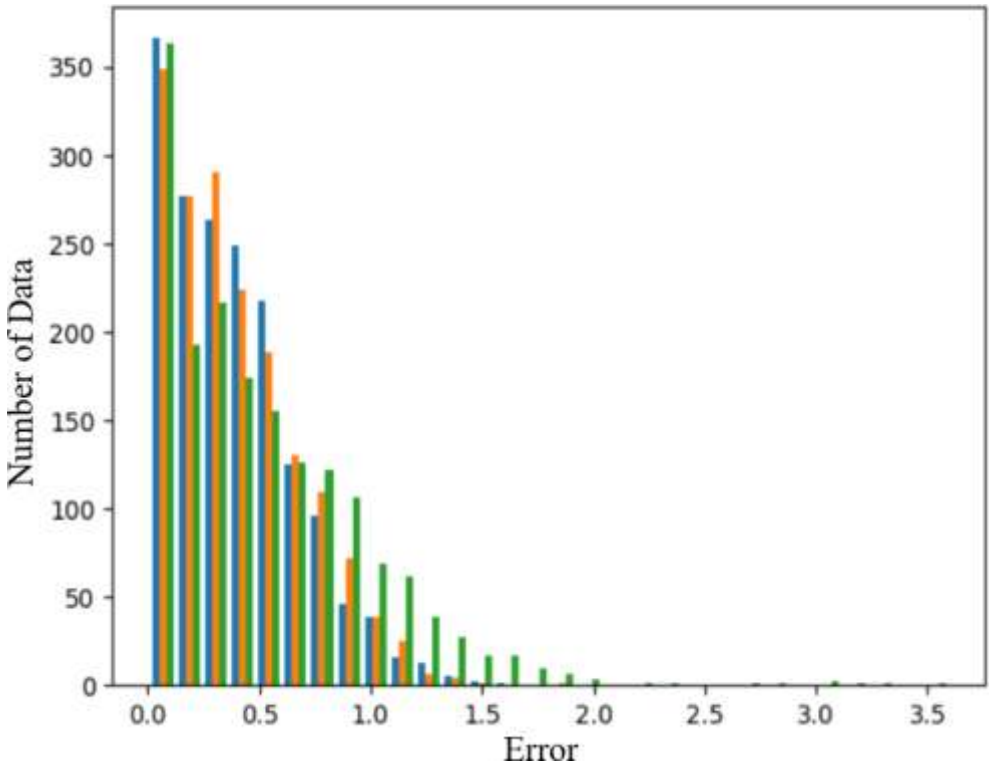


Figure 7-5: Reconstruction Error in Validation Dataset

7.6 Confusion Matrix

To evaluate the performance of our model for crash detection, we constructed confusion matrices at various threshold values. The confusion matrices were generated based on the number of instances for both non-accidents and accidents, considering the threshold values of 1, 2, 2.5, 3, 4, and 5. We conducted our evaluation using a sample size of 20 instances for both accidents and non-accidents. Below are the confusion matrices at different threshold values.

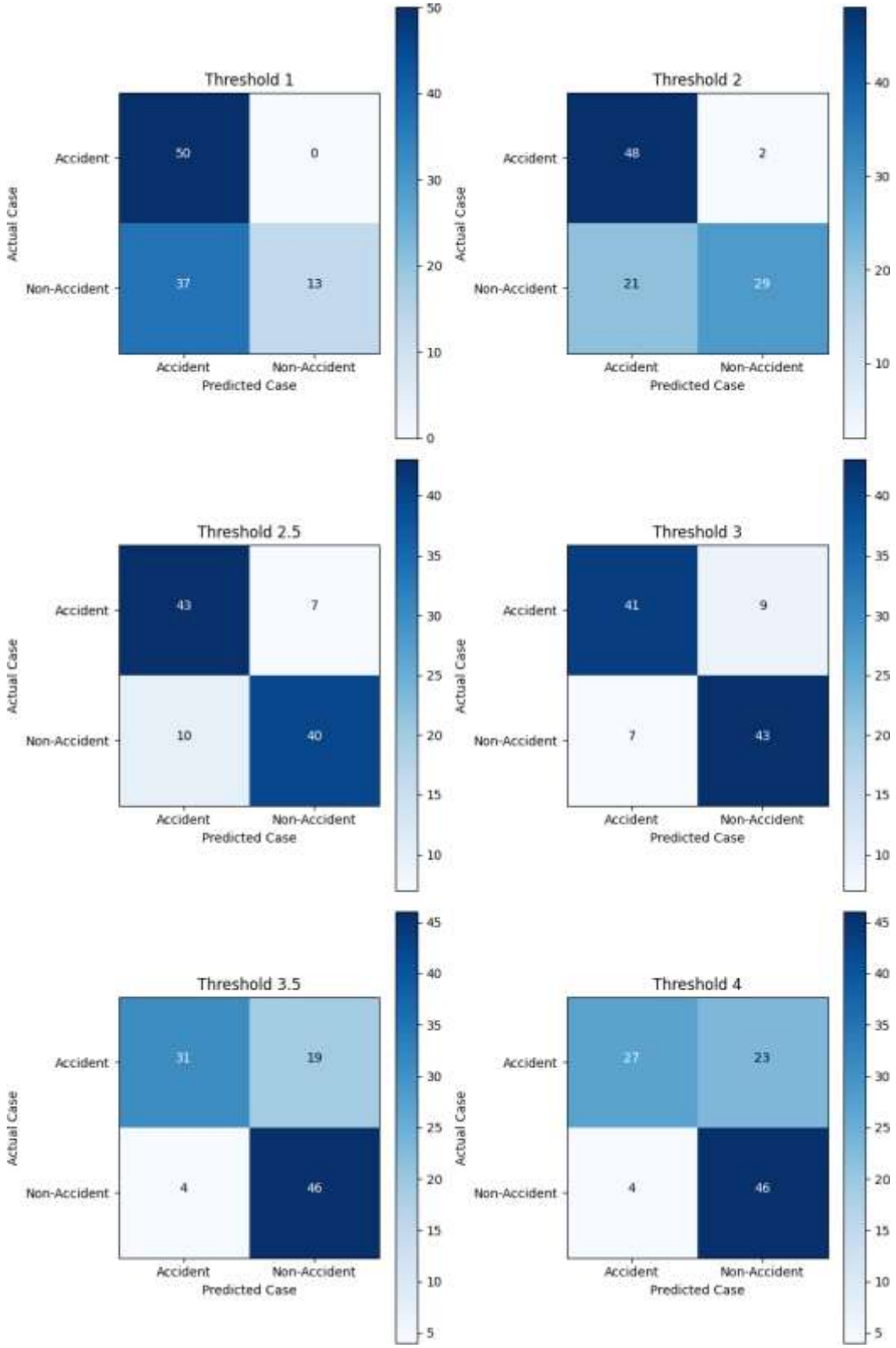


Figure 7-6: Confusion Matrix

After analyzing the confusion matrices, we found that the model achieved the highest accuracy and minimum false predictions at threshold value 3 as this value provided the least number of false positives and false negatives.

7.7 Accuracy

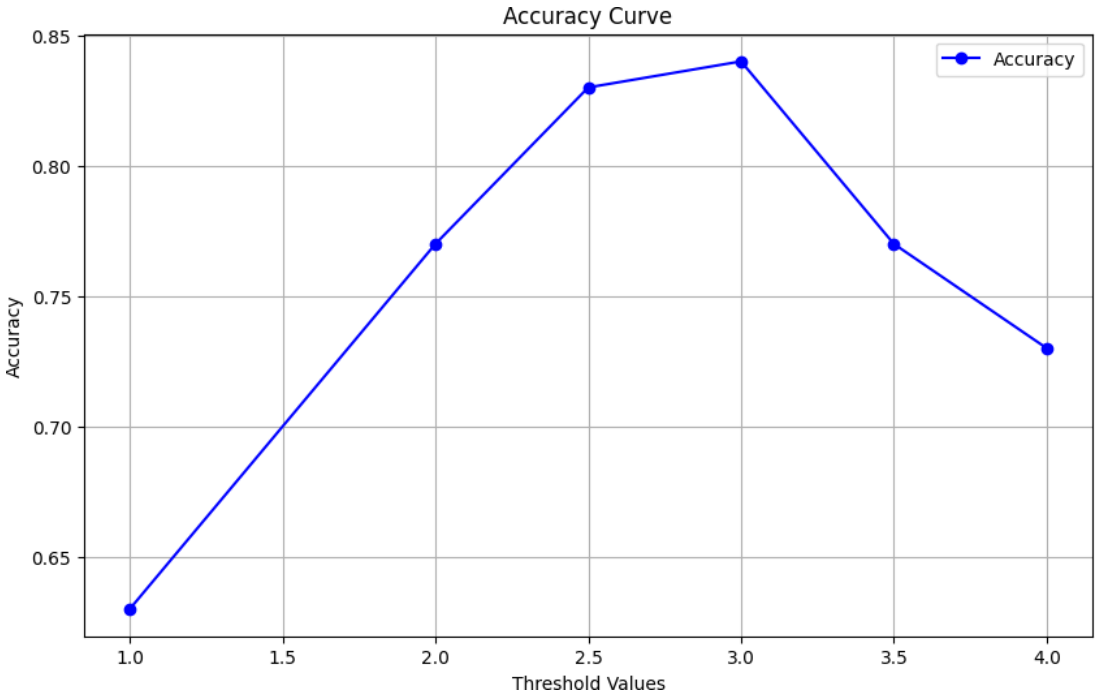


Figure 7-7: Accuracy Curve

The accuracy curve shows how accurate the model's predictions are at different threshold values. We tested threshold values from 1.0 to 4.0. The curve helped us see how accuracy changed with each threshold. We found that the model is most accurate when the threshold is between 2.5 and 3.0.

7.8 Precision

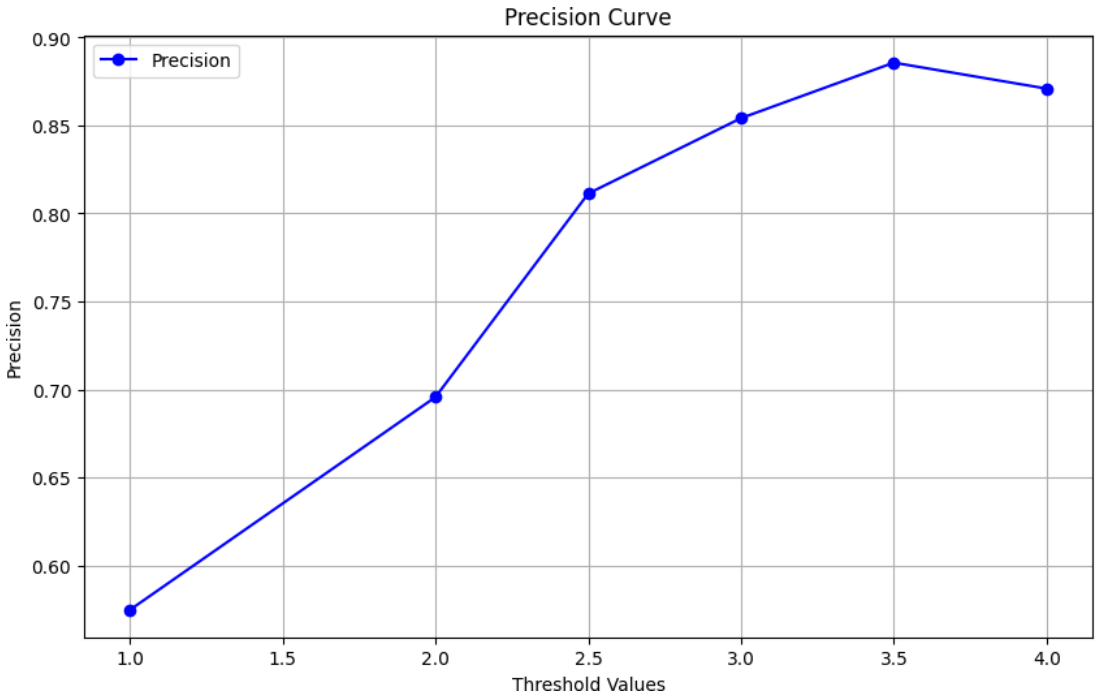


Figure 7-8: Precision Curve

The precision curve shown in the figure was constructed using six different threshold values: 1.0, 2.0, 2.5, 3.0, 3.5, and 4.0. Among these thresholds, the highest precision was achieved at a threshold value of 3.5. Therefore, the model makes most precise predictions when taking a threshold value of 3.5, as indicated by the precision plot.

7.9 Recall

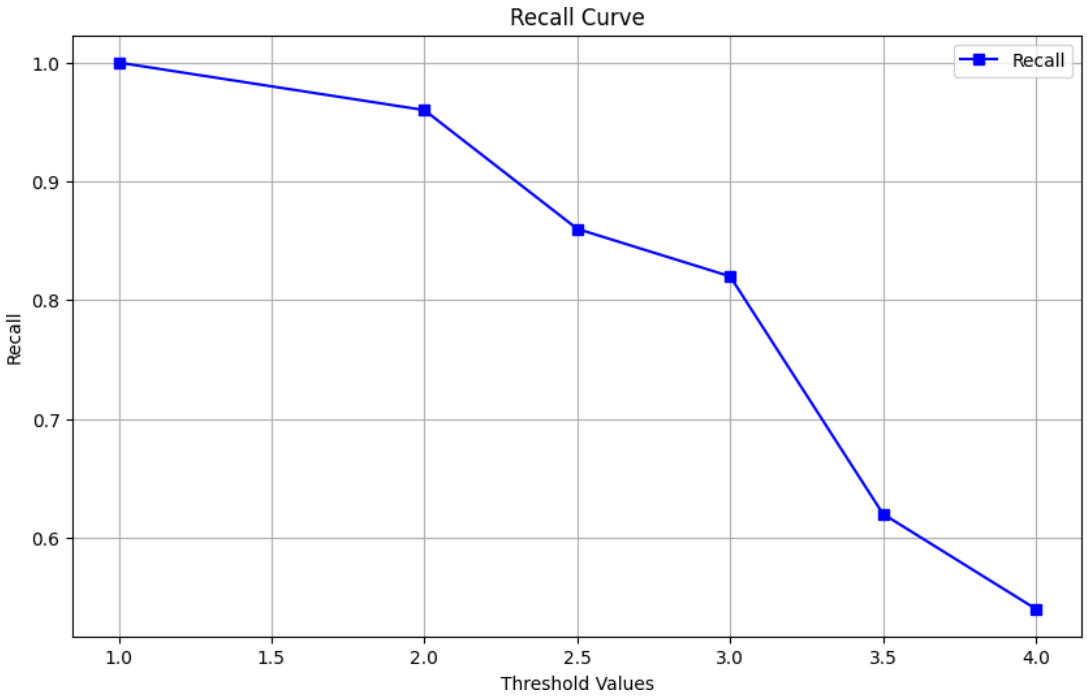


Figure 7-9: Recall Curve

The Recall curve was generated by plotting the recall values obtained at different threshold levels: 1.0, 2.0, 2.5, 3.0, 3.5, and 4.0. The analysis gave the highest recall score at a threshold of 1.0, gradually decreasing to the lowest value observed at 4.0. This shows how well the model can find positive instances, with the best performance seen at a threshold of 1.0.

7.10 Precision vs Recall

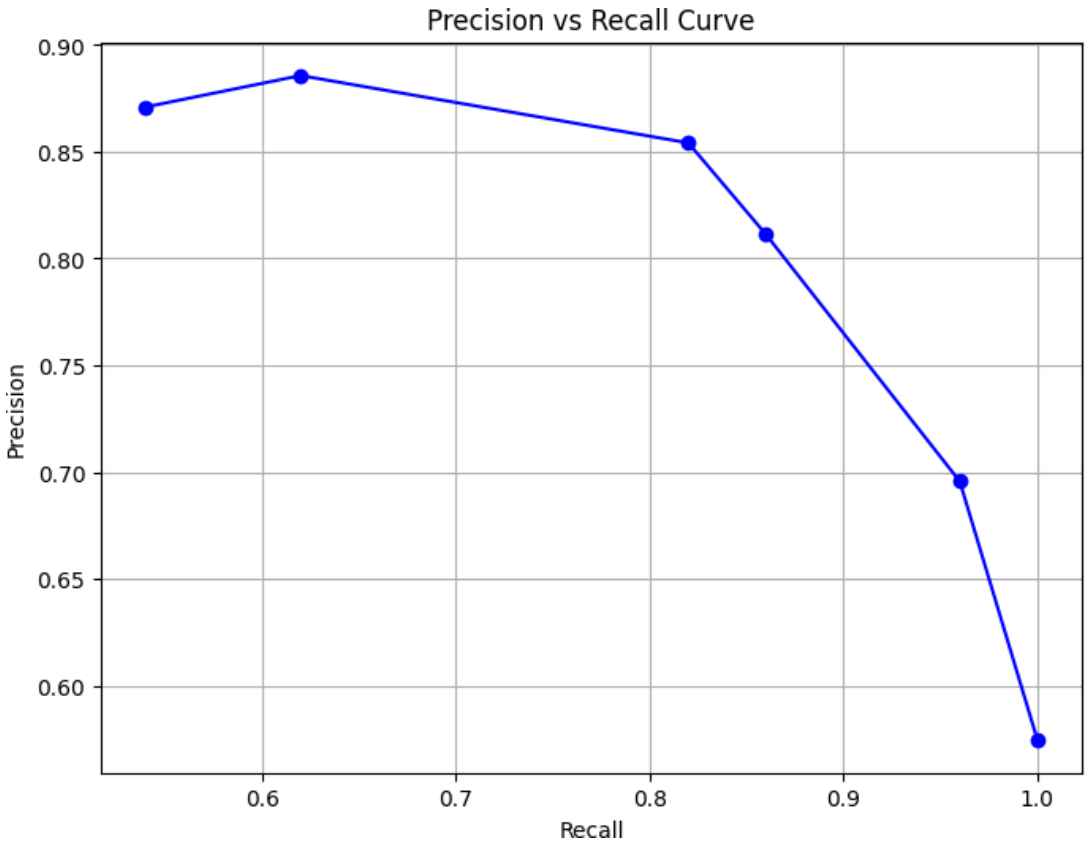


Figure 7-10: Precision vs Recall Curve

Precision-Recall curve illustrated above is generated for six threshold values: 1.0, 2.0, 2.5, 3.0, 3.5, and 4.0. A higher precision value indicates fewer false positive predictions, while a higher recall value indicates fewer false negative predictions. The trade-off between precision and recall is often illustrated by the Precision-Recall curve, where higher precision typically comes at the cost of lower recall, and vice versa. The Precision-Recall curve shows that model performs best at threshold value 2.5 which means at this threshold model achieves a balance between precision and recall.

7.11 F1 Score

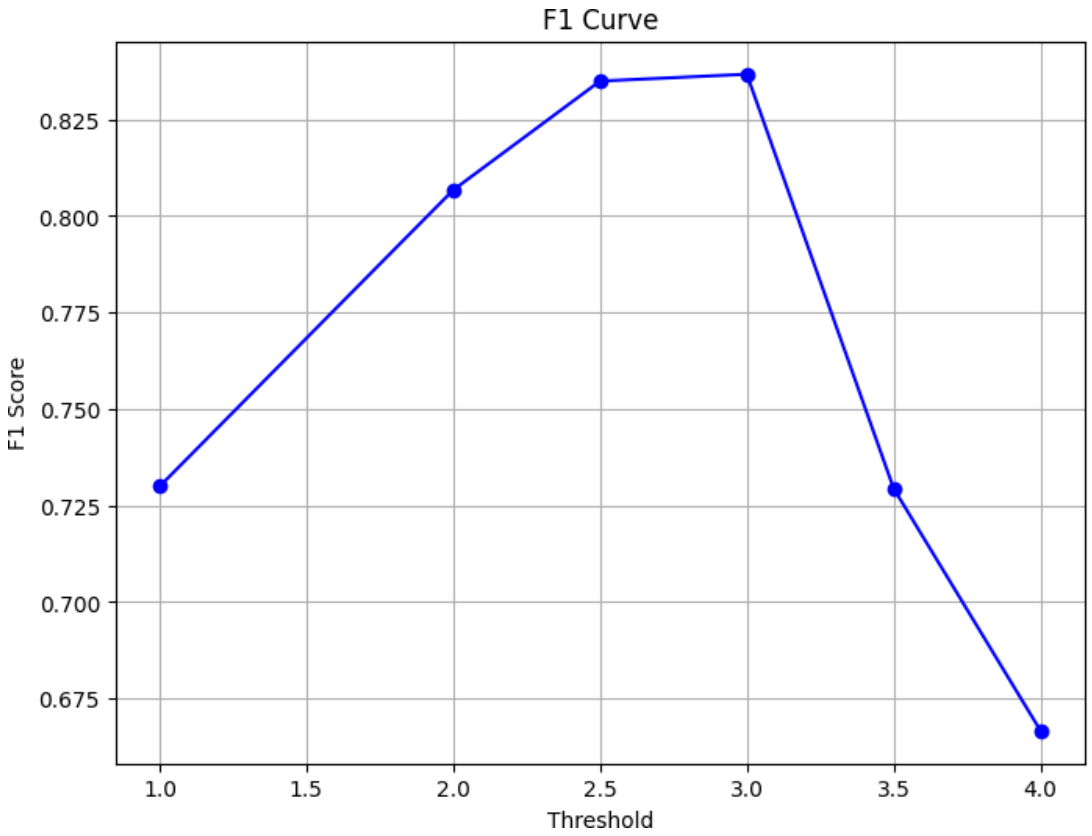


Figure 7-11: F1 Score Curve

The F1 score curve is a graphical representation of the F1 score across different threshold values in a classification model. In above figure, F1 curve is generated by plotting F1 score against different threshold values: 1.0, 2.0, 2.5, 3.0, 3.5, and 4.0. Each point on the curve corresponds to specific threshold value. The threshold value ranging from 2.5 to 3.0 maximizes the F1 score on the curve and indicate the optimal threshold for achieving a balance between precision and recall.

8. FUTURE ENHANCEMENT

The improvements that can be worked upon in the future includes:

8.1 Dataset Collection from Real Vehicle

The dataset we've gathered is from a remote-controlled car. To enhance real-world applicability, it's essential to collect data from actual vehicles. Additionally, the dataset's quality significantly influences the accuracy of trained models. Therefore, it's important to gather data under various conditions, including sudden braking, lane changes,bumps, and other scenarios. This diverse data set improves the system's ability to accurately detect and respond to potential incidents in real-world driving conditions.

8.2 Severity Classification

The system does not assess the severity of crashes. However, future enhancements could involve developing methodologies to classify the severity of crashes by considering factors such as vehicle deformation, impact force, and the severity of injuries sustained. This advancement would enable a more comprehensive understanding of the crash scenario, allowing for more informed response measures and potentially improving overall safety outcomes.

9. CONCLUSION

In conclusion, the project has successfully achieved both its objectives: vehicle tracking and crash detection. The system relies on the Raspberry Pi as its core component. The GPS module monitors the vehicle's location, sending the data to Raspberry Pi and from Raspberry Pi to the server. Similarly, the accelerometer sensor transmits its readings to the Raspberry Pi. If abnormal data suggests a crash, a buzzer alarm is triggered through a deep learning algorithm. Users have the option to deactivate false alarms via a push button, otherwise, an alert message is sent to an emergency contact via the Twilio Application after a 10-second delay. To apply this system in real-life scenarios, data collection from the original vehicle is necessary, as data are currently collected on a Remote-Control Car.

10. APPENDICES

Appendix A: Project Schedule

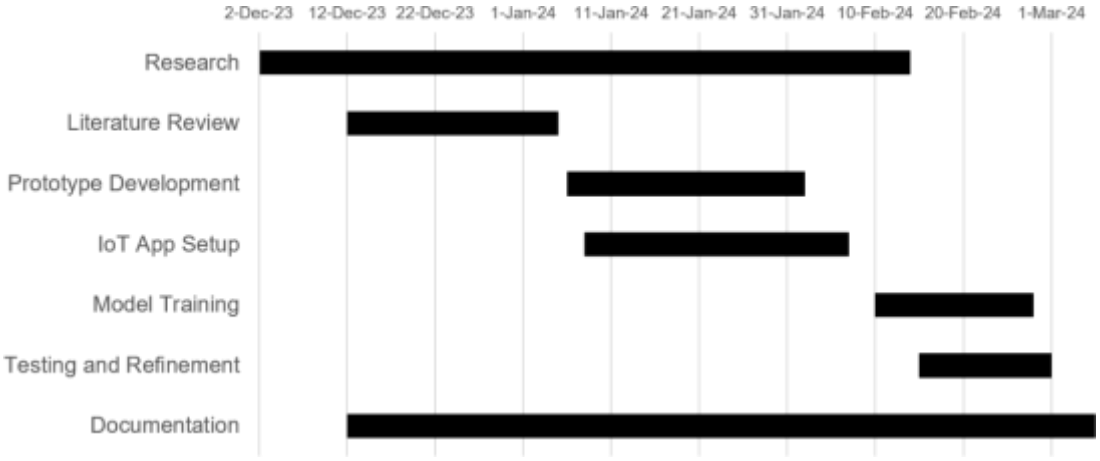


Figure 10-1: Gantt Chart

Appendix B: Project Budget

Table 10-1: Project Budget Description

S.N.	Component	Price
1	Raspberry Pi	10000
2	GPS Module	1000
3	Accelerometer	500
4	Power Bank	1000
5	RC Car	2500
6	Miscellaneous	700
	Total	**15,700**

Appendix C: Module Specifications

Table 10-2: Raspberry Pi

Specification	Value
Microcontroller	Raspberry Pi 4.0
Operating Voltage	5V
Input Voltage (recommended)	5V DC
GPIO Pins	40
Flash Memory	microSD Card
SDRAM	2 GB
Clock Speed	1.5 GHz

Table 10-3: NEO 7M GPS Module

Specification	Value
Chipset	u-blox NEO-7M
Operating Voltage	3.3V
Input Voltage	3.3V to 5V
Communication	UART (TTL)
Baud Rate	Up to 115200 bps
Update Rate	Up to 10 Hz
Tracking Sensitivity	-162 dBm
Time to First Fix	Cold: 26s, Hot: 1s

Table 10-4: Piezo Buzzer

Specification	Value
Voltage	5V
Diameter	12mm
Foot spacing	6.5mm
Height	8.5mm
Current	< 250mA

Appendix D: Code Snippets

```
import time
import serial
import board
import busio
from adafruit_mpu6050 import MPU6050
import numpy as np
from tensorflow.keras.models import load_model
import requests
from twilio.rest import Client
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setup(5, GPIO.OUT)          #buzzer
GPIO.output(5, GPIO.LOW)
GPIO.setup(6, GPIO.IN, pull_up_down=GPIO.PUD_UP) #btn

account_sid = 'AC4b402edaaf2fa7e7cdf9e8a31293ca54'
auth_token = '328632e984e2d4932bb8ca8fa7c17f37'
client = Client(account_sid, auth_token)

url = 'http://192.168.18.206:5000/update_map/'

model = load_model('./models/gkpmy.h5')
#model = load_model('./models/crashdetection2.h5')
#model = load_model('./models/newdata.h5')
#model = load_model('./models/newdata64nss10.h5') #ss=20
batch_size = 10
data_lines = []

seq_size=5
def to_sequences(x, seq_size=1):
    x_values = []
    for i in range(len(x)-seq_size+1):
        x_values.append(x[i:i+seq_size])
    return x_values
```

Figure 10-2: Python Script for Importing Libraries and Pin Configuration

```python
# Serial comm for gps
ser = serial.Serial('/dev/serial0', 9600, timeout=1)
# Initialize I2C bus and accelerometer.
i2c = busio.I2C(board.SCL, board.SDA)
accelerometer = MPU6050(i2c)

# Main loop to read accelerometer data.
while True:
    pushed = 0
    # Read accelerometer data.
    x, y, z = accelerometer.acceleration

    x = (x - 0.01252124) / 0.86023712
    y = (y - 0.08577487) / 0.86885321
    z = (z - 9.73643464) / 1.34532993

    data=[x,y,z]
    prediction_data = []
    data_lines.append(data)
    #print(data_lines)
    if len(data_lines) >= batch_size:
        #print(data_lines) # 10 data per array

        for index in range(len(data_lines)-seq_size+1):
            prediction_data.append(data_lines[index:index+seq_size])
        prediction_data = to_sequences(data_lines, seq_size)

        #print(prediction_data)
        prediction = model.predict(prediction_data)
        errorMAE = np.mean(np.abs(prediction - prediction_data), axis=1)
        #print(errorMAE)
        anomalies = errorMAE > 2.2
        print(np.sum(anomalies))
```

Figure 10-3: Python Code for Accelerometer Configuration

```python
    if  np.sum(anomalies)>=1:
        GPIO.output(5, GPIO.HIGH)
        start_time = time.time()
        while time.time() - start_time < 5:
            print("checking")
            # Check if the button is pressed
            if GPIO.input(6) == GPIO.LOW:
                pushed = 1
                GPIO.output(5, GPIO.LOW)
                print("pressed button")
                break

        # If the button was not pressed, send the message
        if pushed == 0:
            print(anomalies)
            print('accident detected')
            message = client.messages.create(
                from_='+14157671713',
                body='Accident Alert',
                to='+9779864063381'
            )
            GPIO.output(5,GPIO.LOW)
            GPIO.cleanup()
            break
    data_lines = []
```

Figure 10-4: Python Code for sending Accident Alert

```
try:
    line = ser.readline().decode('utf-8',errors='ignore').strip()
    if line.startswith('$GPGGA'):
        # Split the line into fields
        fields = line.split(',')
        # Extract latitude, longitude, and altitude
        latitude = fields[2]
        degree = latitude[:2]
        minute = latitude[2:]
        longitude = fields[4]
        degreel = longitude[:3]
        minutel = longitude[3:]
        # Convert latitude and longitude to decimal format
        if degree:
            latitude_decimal = int(degree) + float(minute) / 60
            longitude_decimal = int(degreel) + float(minutel) / 60

        # Prepare data for POST request
            data = {'latitude': latitude_decimal, 'longitude': longitude_decimal}
            print(data)
            try:
                response = requests.post(url, json=data,timeout=1)
            except:
                continue
            time.sleep(0.01)
        # Print the data
        #print(f"Latitude: {degree}.{minute/60}', Longitude: {degreel}.{minutel/60}'")
except serial.SerialException:
    print("GPS module disconnected. Proceeding without GPS data..."
```

Figure 10-5: Python Code for reading GPS Data and sending it to the Server

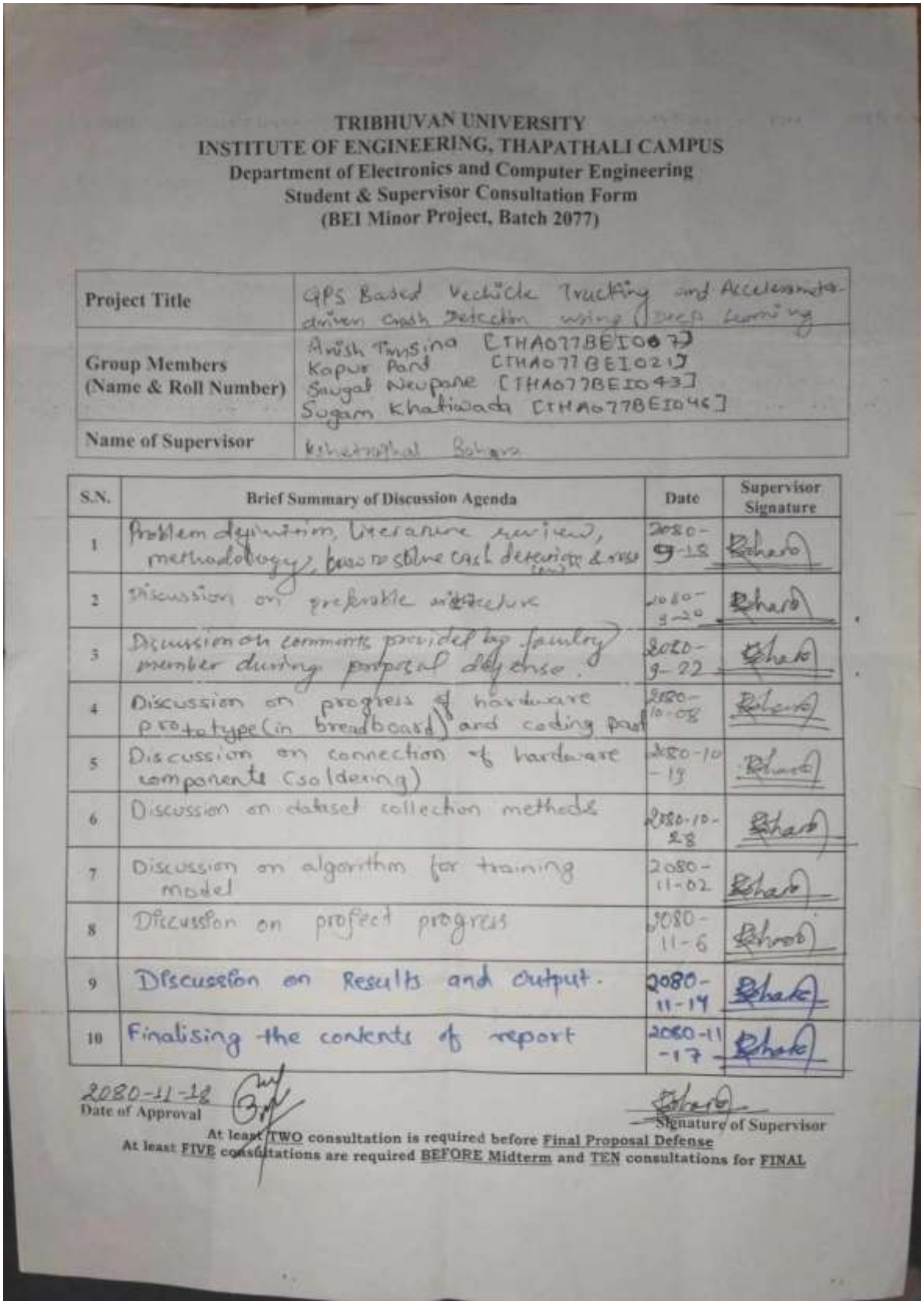
```
model = Sequential()
model.add(GRU(64, input_shape=(trainX.shape[1], trainX.shape[2])))

model.add(Dropout(rate=0.2))

model.add(RepeatVector(trainX.shape[1]))
model.add(GRU(64, return_sequences=True))
model.add(Dropout(rate=0.1))
model.add(TimeDistributed(Dense(trainX.shape[2])))
model.compile(optimizer='adam', loss='mae')
```

Figure 10-6: GRU Autoencoder Model Development

Appendix E: Supervisor Consultation Form



TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING, THAPATHALI CAMPUS
Department of Electronics and Computer Engineering
Student & Supervisor Consultation Form
(BEI Minor Project, Batch 2077)

Project Title	GPS Based Vechicle Tracking and Accelerometer driven crash Detection using Deep Learning
Group Members (Name & Roll Number)	Anish Timsina [THA077BEI007] Kapur Pant [THA077BEI021] Saugat Neupane [THA077BEI043] Sugam Khatiwada [THA077BEI046]
Name of Supervisor	Khetraphal Bohara

S.N.	Brief Summary of Discussion Agenda	Date	Supervisor Signature
1	Problem definition, literature review, methodology, how to solve crash detection & rescue	2080-9-18	Bohara
2	Discussion on preferable architecture	2080-9-20	Bohara
3	Discussion on comments provided by faculty member during proposal defense.	2080-9-22	Bohara
4	Discussion on progress of hardware prototype(in breadboard) and coding part	2080-10-08	Bohara
5	Discussion on connection of hardware components (soldering)	2080-10-19	Bohara
6	Discussion on dataset collection methods	2080-10-28	Bohara
7	Discussion on algorithm for training model	2080-11-02	Bohara
8	Discussion on project progress	2080-11-6	Bohara
9	Discussion on Results and output.	2080-11-14	Bohara
10	Finalising the contents of report	2080-11-17	Bohara

2080-11-18
Date of Approval

Bohara
Signature of Supervisor

At least TWO consultation is required before Final Proposal Defense
At least FIVE consultations are required BEFORE Midterm and TEN consultations for FINAL

Figure 10-7: Student & Supervisor Consultation form

Appendix F: Orginality Report

GPS-Based Vehicle Tracking and Accelerometer-driven Crash Detection

ORIGINALITY REPORT

23%
SIMILARITY INDEX

PRIMARY SOURCES

1	www.researchgate.net Internet	343 words — 3%
2	www.analyticsvidhya.com Internet	143 words — 1%
3	it.overleaf.com Internet	138 words — 1%
4	aws.amazon.com Internet	112 words — 1%
5	fastercapital.com Internet	112 words — 1%
6	www.baeldung.com Internet	107 words — 1%
7	Wan-Jung Chang, Liang-Bi Chen, Ke-Yu Su. "DeepCrash: A Deep Learning-Based Internet of Vehicles System for Head-On and Single-Vehicle Accident Detection With Emergency Notification", IEEE Access, 2019 Crossref	85 words — 1%
8	Jaroslav Kopčan, Ondrej Škvarek, Martin Klimo. "Anomaly detection using Autoencoders and Deep	65 words — 1%

Convolution Generative Adversarial Networks", Transportation Research Procedia, 2021
Crossref

9	elibrary.tucl.edu.np Internet	64 words — 1%
10	www.coursehero.com Internet	64 words — 1%
11	dokumen.pub Internet	58 words — < 1%
12	open-innovation-projects.org Internet	54 words — < 1%
13	Polonowski, Christopher J. "Accelerometer based measurements of combustion in an automotive turbocharged diesel engine", Proquest, 20111108 ProQuest	52 words — < 1%
14	medium.com Internet	43 words — < 1%
15	Ali Al Bataineh, Devinder Kaur. "Immunocomputing-Based Approach for Optimizing the Topologies of LSTM Networks", IEEE Access, 2021 Crossref	39 words — < 1%
16	blog.roboflow.com Internet	38 words — < 1%
17	2019carscomingout.com Internet	34 words — < 1%
18	dev.to Internet	33 words — < 1%

19	Dhwaani Parikh, Vineet Menon. "Machine Learning Applied to Cervical Cancer Data", International Journal of Mathematical Sciences and Computing, 2019 Crossref	32 words — < 1%
20	Krish Sethi, Simrit Kaul, Ishan Patel, Sujatha R.. "FaceLock Homes: A Contactless Smart Home Security System to Prevent COVID Transmission", 2021 Sixth International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), 2021 Crossref	31 words — < 1%
21	thesai.org Internet	31 words — < 1%
22	eprints.utem.edu.my Internet	30 words — < 1%
23	umu.diva-portal.org Internet	30 words — < 1%
24	lup.lub.lu.se Internet	26 words — < 1%
25	www.geeksforgeeks.org Internet	26 words — < 1%
26	dspace.plymouth.ac.uk Internet	23 words — < 1%
27	earsiv.cankaya.edu.tr:8080 Internet	23 words — < 1%
28	fabacademy.org Internet	22 words — < 1%

29	www.azkurs.org Internet	22 words — < 1%
30	Isaac Kega Mwangi, Lawrence Nderu, Ronald Waweru Mwangi, Dennis Gitari Njagi. "Hybrid Interpretable Model Using Roughset Theory and Association Rule Mining to Detect Interaction Terms in a Generalized LINEAR Model", Expert Systems with Applications, 2023 Crossref	21 words — < 1%
31	research.chalmers.se Internet	21 words — < 1%
32	arxiv.org Internet	20 words — < 1%
33	ijsrset.com Internet	20 words — < 1%
34	link.springer.com Internet	20 words — < 1%
35	usermanual.wiki Internet	20 words — < 1%
36	Luca Kubin, Tommaso Bianconcini, Douglas Coimbra de Andrade, Matteo Simoncini, Leonardo Taccari, Francesco Sambo. "Deep Crash Detection From Vehicular Sensor Data With Multimodal Self-Supervision", IEEE Transactions on Intelligent Transportation Systems, 2022 Crossref	19 words — < 1%
37	www.scoop.it Internet	19 words — < 1%
38	LaChance, Julienne Marie. "Machine Learning and Statistical Analysis of the Collective Behaviors of	18 words — < 1%

Large Tissues", Princeton University, 2021
ProQuest

39	Shezeena Qureshi, Faheemullah Shaikh, Laveet Kumar, Farooque Ali, Muhammad Awais, Ali Etem Gürel. "Short-term Forecasting of Wind Power Generation using Artificial Intelligence", Environmental Challenges, 2023 Crossref	17 words — < 1%
40	ruor.uottawa.ca Internet	17 words — < 1%
41	www.mdpi.com Internet	17 words — < 1%
42	www.process.st Internet	17 words — < 1%
43	Santiago Felipe Yepes Chamorro, Juan Jose Paredes Rosero, Ricardo Salazar-Cabrera, Álvaro Pachón de la Cruz et al. "Design, Development and Validation of an Intelligent Collision Risk Detection System to Improve Transportation Safety: The Case of the City of Popayán, Colombia", Sustainability, 2022 Crossref	16 words — < 1%
44	repository.iaa.ac.tz:8080 Internet	16 words — < 1%
45	www.bartleby.com Internet	16 words — < 1%
46	www.theseus.fi Internet	16 words — < 1%
47	Fernandes, Bruno de Sa Moreira. "Sensor Integration for Smart Cities Using Multi-Hop	15 words — < 1%

Networks", Instituto Politecnico do Porto (Portugal), 2022
ProQuest

48 tel.archives-ouvertes.fr
Internet
15 words — < 1%

49 Supriya M S, Sahana P Shankar, Himanshu Jain B J, Lisha L Narayana, Nikhita Gumalla. "Car Crash Detection System using Machine Learning and Deep Learning Algorithm", 2022 IEEE International Conference on Data Science and Information System (ICDSIS), 2022
Crossref
14 words — < 1%

50 rei.iteso.mx
Internet
14 words — < 1%

51 statistikhessen-blog.de
Internet
14 words — < 1%

52 www.navilock.com
Internet
14 words — < 1%

53 Hlophe, Mduduzi Comfort. "A Model-Based Deep Learning Approach to Spectrum Management in Distributed Cognitive Radio Networks", University of Pretoria (South Africa), 2023
ProQuest
13 words — < 1%

54 Liqiong Wang, Yan Huang, Fanrong Kong. "Chapter 16 Multi-scale Texture Network for Industrial Surface Defect Detection", Springer Science and Business Media LLC, 2024
Crossref
13 words — < 1%

55 m.moam.info
Internet
13 words — < 1%

mdpi.com

56	Internet	13 words — < 1%
57	www.sciencebuddies.org Internet	13 words — < 1%
58	"Improving Productivity through Automation and Computing", 2019 25th International Conference on Automation and Computing (ICAC), 2019 Crossref	12 words — < 1%
59	Wallat, Eric M.. "Improving the Efficacy of Functional Lung Avoidance Radiation Therapy", The University of Wisconsin - Madison, 2023 ProQuest	12 words — < 1%
60	cs.appstate.edu Internet	12 words — < 1%
61	liu.diva-portal.org Internet	12 words — < 1%
62	www.renesas.com Internet	12 words — < 1%
63	essay.utwente.nl Internet	11 words — < 1%
64	summit.sfu.ca Internet	11 words — < 1%
65	theses.liacs.nl Internet	11 words — < 1%
66	Ahmed Alzahrani, Muhammad Zubair Asghar. "Intelligent Risk Prediction System in IoT-Based	10 words — < 1%

Supply Chain Management in Logistics Sector", Electronics, 2023
Crossref

67	dspace.auk.edu.kw Internet	10 words — < 1%
68	services.phaidra.univie.ac.at Internet	10 words — < 1%
69	www.ijser.in Internet	10 words — < 1%
70	dergipark.org.tr Internet	9 words — < 1%
71	kylo.tv Internet	9 words — < 1%
72	megaeshop.pk Internet	9 words — < 1%
73	sono-nincsenek.com Internet	9 words — < 1%
74	st.robu.in Internet	9 words — < 1%
75	www.eurchembull.com Internet	9 words — < 1%
76	www.matec-conferences.org Internet	9 words — < 1%
77	"Social Computing and Social Media. Design, Human Behavior and Analytics", Springer Science and Business Media LLC, 2019 Crossref	8 words — < 1%

78	Dezfooli, Seyed Ali Rokni. "Dynamic Adaptation of Recognition Algorithms on Wearables with Minimal Human Supervision.", Washington State University, 2018 ProQuest	8 words — < 1%
79	Fue, Kadeghe Goodluck. "Development of the Autonomous Cotton Harvesting Robot.", University of Georgia, 2020 ProQuest	8 words — < 1%
80	Kadve, Ritu. "Federated Learning to Build Sentiment Analysis Models for Amazon Review Datasets Without Labels", University of Houston-Clear Lake, 2023 ProQuest	8 words — < 1%
81	acikarsiv.atilim.edu.tr Internet	8 words — < 1%
82	de Almeida, Joao Carlos Canto. "Curve Classification Using Computational Geometry Concepts", University of Massachusetts Lowell, 2024 ProQuest	8 words — < 1%
83	dspace.univ-ouargla.dz Internet	8 words — < 1%
84	flipkarma.com Internet	8 words — < 1%
85	ierjournal.org Internet	8 words — < 1%
86	octai.readme.io Internet	8 words — < 1%
	pyimagesearch.com	

87	Internet	8 words — < 1%
88	repository.ju.edu.et Internet	8 words — < 1%
89	theses.hal.science Internet	8 words — < 1%
90	vivekrai1011.medium.com Internet	8 words — < 1%
91	www.irjmets.com Internet	8 words — < 1%
92	www.repo.uni-hannover.de Internet	8 words — < 1%
93	Abhiramy Ajith, Abin Oommen Philip, Sreela Sreedhar, M U Sreeja. "Road Accident Detection from CCTV Footages using Deep Learning", 2022 3rd International Conference on Smart Electronics and Communication (ICOSEC), 2022 Crossref	7 words — < 1%
94	Jae Gyeong Choi, Chan Woo Kong, Gyeongho Kim, Sunghoon Lim. "Car crash detection using ensemble deep learning and multimodal data from dashboard cameras", Expert Systems with Applications, 2021 Crossref	7 words — < 1%
95	Michelle Viscaino, Matias Talamilla, Juan Cristóbal Maass, Pablo Henríquez et al. "Color Dependence Analysis in a CNN-Based Computer-Aided Diagnosis System for Middle and External Ear Diseases", Diagnostics, 2022 Crossref	7 words — < 1%

96 Mohamed Amine Mahjoubi, Soufiane Hamida, Bouchaib Cherradi, Oussama El Gannour, Ahmed El Abbassi, Abdelhadi Raihani. "Classifying Brain Tumors using Convolutional Neural Networks on MRI Scans", Proceedings of the 6th International Conference on Networking, Intelligent Systems & Security, 2023
Crossref
7 words — < 1%

97 Sougata Sheet, Ranjan Ghosh, Anupam Ghosh. "Recognition of cancer mediating genes using MLP-SDAE model", Systems and Soft Computing, 2024
Crossref
7 words — < 1%

98 Donald J. Norris. "Machine Learning with the Raspberry Pi", Springer Science and Business Media LLC, 2020
Crossref
6 words — < 1%

99 Gattani, Vishal. "Experimental Design Using Bayesian Network Simulation-Based Assurance Cases", University of Maryland, College Park, 2023
ProQuest
6 words — < 1%

100 Sulove Bhattarai, Sudip Bhujel, Santosh Adhikari, Shanta Maharjan. "Design and Implementation of a Portable ECG Device", Journal of Innovations in Engineering Education, 2020
Crossref
6 words — < 1%

101 Vasilii Mosin, Miroslaw Staron, Yury Tarakanov, Darko Durisic. "Comparing autoencoder-based approaches for anomaly detection in highway driving scenario images", SN Applied Sciences, 2022
Crossref
6 words — < 1%

102 ds.inflibnet.ac.in
Internet
5 words — < 1%

103	publisher.uthm.edu.my Internet	5 words — < 1%
104	docs.neu.edu.tr Internet	4 words — < 1%
105	lucris.lub.lu.se Internet	4 words — < 1%

EXCLUDE QUOTES	ON	EXCLUDE SOURCES	OFF
EXCLUDE BIBLIOGRAPHY	ON	EXCLUDE MATCHES	OFF

References

[1] "National Health Commission of People's Republic of China," [Online]. Available: http://en.nhc.gov.cn/infographics.html.

[2] S.M. Savaresi, D. Selmanaj and M. Corno, "Accelerometer-based Data-driven Hazard Detection and Classification Accelerometer-based Data-driven Hazard Detection and Classification," in *2014 European Control Conference (ECC)*, Strasbourg, France, 2014.

[3] Luca Kubin, Tommaso Bianconcini, Douglas Coimbra de Andrade, Matteo Simoncini, Leonardo Laccari, and Francesco Sambo, "Deep Crash Detection From Vehicular Sensor Data With Multimodal Self-Supervision," *IEEE Transactions on Intelligent Transportation Systems,* vol. 23, 2022.

[4] Jae Gyeong Choi, Chan Woo Kong, Gyeongho Kim, Sunghoon Lim, "Car crash detection using ensemble deep learning and multimodal data from dashboard cameras," *Expert Systems with Applications,* vol. 183, 2021.

[5] Apple Inc., [Online]. Available: https://support.apple.com/en-us/104959.

[6] Fast Company & INC, 2023. [Online]. Available: https://www.fastcompany.com/90904197/apple-ios-17-craig-federighi-privacy-check-in-lockdown-mode-safari.

[7] Roohullah Fazli Wahid, Sikandar Ali, Irshad Ahmed Abbasi, Samad Baseer, and Habib Ullah Khan, "Accident Detection in Autonomous Vehicles Using Modified Restricted Boltzmann Machine".

[8] Hanif Hakimi Azlan, Suriana Salimin, "Vehicle Accident Detection System using Accelerometer," *Evolution in Electrical and Electronic Engineering Vol. 4No. 1(2023),* vol. 4, 2023.

[9] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall, "Classification of time series by shapelet transformation," *Data Mining and Knowledge Discovery,* vol. 28, 2013.

[10] M. Farahani, "Anomaly Detection on Gas Turbine Time-series' Data Using Deep LSTM-Autoencoder," Umeå University, 2020.

[11] Kukjin Choi, Jihun Yi. Changhwa Park, and Sungroh Yoon, "Deep Learning for Anomaly Detection in Time-Series Data: Review, Analysis, and Guidelines," *IEEE Access,* vol. 9, 2021.

[12] Jaroslav Kopčan, Ondrej Škvarek, and Martin Klimo, "Anomaly detection using Autoencoders and Deep Convolution Generative Adversarial Networks," *Transportation Research Procedia,* vol. 55, pp. 1296-1303, 2021.

[13] V. Mosin, M. Staron, Y. Tarakanov and D. Durisic, "Comparing autoencoder-based approaches for anomaly detection," *SN Applied Sciences,* vol.4, no. 334, 2022.

[14] A. Amidi and S. Amidi, "Vip cheatsheet: Recurrent neural networks.," Kaggle, 2018.

[15] Y. K. Saheed, A. A. Usman, F. D. Sukat and M. Abdulrahman, "A novel hybrid autoencoder and modified particle swarm optimization feature selection for intrusion detection in the internet of things network.," Frontiers in Computer Science, 2023.