Website www.aceiot.ur.ac.rw
Mail: aceiot@ur.ac.rw

COLLEGE OF SCIENCE AND TECHNOLOGY

AFRICAN CENTER OF EXCELLENCE IN INTERNET OF THINGS (ACEIoT)

Design and Development of a TinyML– IoT- based Vehicle Crash Detection and Emergency Alert System

A dissertation submitted in partial fulfillment of the requirements for the award of Masters of Science degree in Internet of Things: Embedded Computing Systems

Submitted By:

Name: Henry Eric Kapalamula (Ref. No:222023010)

December, 2024

AFRICAN CENTRE OF EXCELLENCE IN INTERNET OF THINGS

Design and Development of a TinyML– IoT-based Vehicle Crash Detection and Emergency Alert System

A dissertation submitted in partial fulfilment of the requirements for the award of Masters of Science degree in Internet of Things: Embedded Computing Systems

Submitted By

HENRY ERIC KAPALAMULA (REF: 222023010)

Supervised by:

ASSOCIATE PROF. CHOMORA MIKEKA
DR. EVARISTE TWAHIRWA

December, 2024

DECLARATION

I **HENRY ERIC KAPALAMULA**, Masters' student from African Center of Excellence in Internet of Things, at University of Rwanda. I declare that this research thesis is my own original work, and it has never been presented before anywhere in the world.

Names: **HENRY ERIC KAPALAMULA**

Ref: **222023010**

Signed:

Date: **12/12/2024**

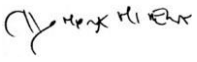
BONAFIDE CERTIFICATE

This is to certify that this submitted Research Thesis work report is a record of the original work done by **HENRY ERIC KAPALAMULA (Ref. No: 222023010)**, MSc. IoT-ECS Student at the University of Rwanda / College of Science and Technology / African Center of Excellence in Internet of Things, the Academic year 2022/2024.

This work has been submitted under the supervision of **Associate Prof. CHOMORA MIKEKA** and **Dr. EVARISTE TWAHIRWA**.

Main Supervisor:
Associate Prof. CHOMORA MIKEKA

Date: **11/12/2024**

Signature: 

Co-Supervisor:
Dr. EVARISTE TWAHIRWA

Date: 12/12/2024

Signature: 

The Head of Masters and Training

Dr. JAMES RWIGEMA

Signature:

Date:

Signature……

ACKNOWLEDGEMENT

I would like to sincerely thank Associate Prof. Chomora Mikeka, my main supervisor, for his unwavering dedication, guidance, and invaluable expertise throughout the course of this thesis. His commitment, coordination, and supervision has greatly contributed to the quality and focus of this work. His mentorship and insightful contributions have been a consistent source of encouragement and inspiration.

I extend my deepest gratitude to Dr. Evariste Twahirwa, my co-supervisor, for his invaluable input, support, and encouragement throughout this research. His insights and suggestions have greatly enhanced the depth and scope of this thesis.

I wish to acknowledge the management of the African Centre of Excellence in Internet of Things at the University of Rwanda for their support, resources, and the conducive research environment they provided. Their dedication to promoting academic excellence has been an invaluable foundation for my academic endeavors.

I am deeply grateful to my family for their unwavering support, understanding, and encouragement, which have been the cornerstone of my academic journey. Their love and belief in me have continuously motivated me to strive for excellence. More importantly, Patience Mhone, for her emotional support, sacrifice and encouragement throughout my studies.

I would also like to express my appreciation to my colleague, Ipyana Issah Mwaisekwa, for his encouragement and inspiration, which have been a constant source of strength throughout this endeavor.

Finally, I would like to extend my heartfelt thanks to all those who have contributed to this thesis in various ways, whether through their time, knowledge, or support.

ABSTRACT

This thesis aimed at employing IoT and TinyML techniques to develop a system that accurately detects vehicle crash accidents using acoustic data and triggers an emergency alert in real time to the nearest police unit. The methodology involved collecting secondary data on vehicle crashes, training a machine learning model, and deploying it on the Arduino Nano 33 BLE microcontroller, which supports TinyML. This technology enables low-power, on-device machine learning for embedded systems, facilitating real-time data processing and decision-making while extending battery life without requiring cloud connectivity. To detect a vehicle crash, the model uses acoustic data to identify an accident and transmits the GPS location via a LoRaWAN communication model. The system successfully alerts and provides the exact location of the accident by showing it on google map. Results demonstrate high accuracy, with 99.3% for training and 98.4% for testing, indicating effective differentiation between accident and non-accident scenarios. When an accident is detected, the GPS location is sent to the Arduino Cloud, plotted on a map, and an alert sound is produced. Further improvements can be made by employing energy harvesting techniques, developing a dedicated mapping system, and creating a model capable of classifying and detecting various forms of accidents beyond vehicle crashes.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DBm	Decibel milliwatts
WAN	Wide Area Network
MCU	Microcontroller Unit
TinyML	Tiny Machine Learning
GPS	Global Positioning System
LoRa	Long Range
IoT	Internet of Things
WHO	World Health Organization
RESNET	Residual Neural Network
DWI	Driving While Intoxicated
GDP	Gross Domestic Product
GSM	Global System for Mobile Communication
CCTV	Closed Circuit Television
CNN	Convolutional Neural Network
VGG	Visual Geometry Group
SMS	Short Message Service
SBC	Single Board Computer
UART	Universal Asynchronous Receiver Transmitter
ML	Machine Learning
MFCC	Mel-Frequency Cepstral Coefficients
MFE	Mel-Filterbank Energy

Design and Development of a Transfer Learning based TinyML– IoT Vehicle Crash Detection and Emergency Alert System

CHAPTER 1: INTRODUCTION

In this chapter, we outline the motivation for the thesis research, followed by an introduction to the problem statement. We then propose the hypothesis, summarize the research contributions, and conclude with an overview of the thesis structure.

1.1. Motivation

As the world population continues to grow, there is an increased demand for vehicles and this has resulted into traffic congestion and road accidents [1]. According to the report released by World Health Organization, as of 2014, the ratio of road accidents fatal injuries to the total number of deaths worldwide had increased by 2.2% [2]. Recently, there's approximately 1.19 million deaths per year that are due to road accidents and it's the leading cause of deaths among children and young people [3]. Statistics prove that road accidents is the leading cause of deaths due to injury [3]. In addition, about 20-50 million people suffer from deadly injuries due to road accidents that result in disability [4].

Some of the causes of road accidents include driver drowsiness, where a drive is sleeping while driving [5]. Another cause of road accident is driving while intoxicated (DWI), which happens when a person is driving while drunk [6]. Over speeding is another cause of road accident [7] [8] that is resulting in increased number of deaths and fatal injuries. Not only that, but weather conditions can also lead to road accidents, where fog, rain, snow and high winds can influence the car to deviate in direction or hit an obstacle such as a tree that has fallen due to high winds [9].

In most cases, the delay in access to medical care treatment is likely to cause death [3]. This is where there is a time delay in detecting and providing medical care treatment to road accident victims that leads to fatal injuries and even death. Therefore, reducing the time delay to access medical care during the golden hour, which is the time after the injury, can reduce the probability of death by one – third on average [10] [11].

Apart from deaths and fatal injuries, road accidents can also cost countries 3% of their GDP and about 92% of the world's fatal accidents occur in low- and middle-income countries [3]. The losses are a result of cost of medical treatment, lack of productivity as family and relatives have to stay away from work and attend their loved ones. In addition, there is a lack of skilled personnel due to injury and death in general. Due to the negative impacts it brings, there's a need to come up with an efficient approach in order to effectively curb the negative impacts of road accidents.

To address the challenges posed by road accidents, there is a need to develop a solution that can accurately detect vehicle crashes and trigger emergency alerts to the nearest police station in real time. Achieving this requires a low-power solution, real-time data processing, on-device decision-making, cost efficiency, and scalability for deployment in remote or battery-operated environments—all benefits provided by TinyML.

1.2. Background

This chapter presents the foundational concepts of the research in three sections. Section 1.2.1 provides an overview of IoT, emphasizing its importance and applications. Section 1.2.2 explores Machine Learning, discussing various algorithms and their practical uses. Lastly, Section 1.2.3 introduces TinyML, outlining its advantages and concluding the background concepts.

1.2.1. Internet of Things (IoT)

The Internet of Things refers to the interconnection of multiple devices through the internet, enabling them to communicate and share data without the need for human intervention [12].

It’s a combination of hardware devices, connectivity, data processing, user interface and cloud services [13]. The devices are made of sensors and actuators which are used for detecting changes in the environment and proving response to the change, respectively [14]. In IoT systems, different sensors are used to collect data about the environment [15]. Temperature sensors check heat levels, while humidity sensors measure moisture in the air. Pressure sensors keep track of air or liquid pressure, and motion sensors notice movement in a space. Gas sensors detect specific gases present in the environment. Each type of sensor plays an important role in providing real-time information, which helps in making informed decisions across various applications, from smart homes to industrial settings, improving efficiency and response in different situations.

Actuators are important parts of IoT systems that perform specific tasks based on data from sensors. These components include different types like motors, switches, and solenoid devices, which change electrical signals into physical actions or movements [16]. By automating processes, actuators help devices respond quickly to changes in the environment or commands from users. They are also integrated into IoT setups, which boosts overall performance and efficiency in various applications. Understanding how actuators work is essential for creating responsive IoT solutions that can adjust to changing conditions, ultimately leading to a better experience for users. Some of the communication modules used in IoT include Wi-Fi, LoRaWAN, Bluetooth, Zigbee, and cellular network [17] as shown in figure 1.

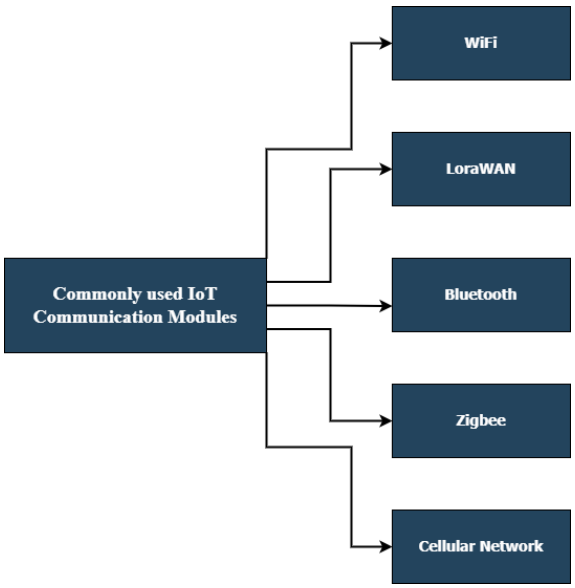


Figure 1: Commonly used IoT modules

Wi-Fi is a crucial technology for connecting IoT devices, known for being widely available and providing fast internet speeds [18]. It enables devices to work together easily in different places, from homes to factories. Although there are some challenges, like higher energy use and security concerns, the authors suggest using energy-saving methods and strong security practices. Overall, Wi-Fi is seen as a key part of the growing IoT world, helping connect devices in our daily lives.

LoRaWAN (Low Power Wide Area Network Protocol for IoT) is a vital technology for connecting numerous devices in the Internet of Things [19]. It facilitates long-range communication with low power use and low data rates, allowing devices to connect easily to gateways. While other technologies are gaining traction, LoRaWAN excels in supporting large sensor networks and addressing challenges like energy efficiency, latency, coverage, and bandwidth.

Bluetooth is a wireless technology that allows devices to communicate over short distances, usually up to 100 meters, using radio waves in the 2.4 GHz frequency band [20]. It can connect devices for audio and sensor data and is known for being energy-efficient and easy to pair. Many devices are compatible with Bluetooth, making it widely used. However, it has some downsides, including a shorter range than Wi-Fi, slower data transfer speeds, possible interference from other devices, and security risks if not set up correctly. Despite these issues, Bluetooth is great for IoT, M2M communications, and everyday gadgets [21].

ZigBee is a common wireless technology used in Internet of Things (IoT) applications, especially in networks that include sensor nodes with cameras, microphones, and other sensors [22]. It is lightweight, low-cost, and energy-efficient. ZigBee often uses a tree structure for its networks, which helps send data packets easily through root, intermediate, and leaf nodes.
However, sometimes network issues can cause orphan nodes, which are nodes that lose their connection to the network.

Cellular networks are important for the Internet of Things (IoT), connecting billions of devices through technologies like LTE, NB-IoT, and 5G [23]. They offer wide coverage, can support many devices at once, are reliable, and use low power, making them great for different IoT applications.

However, they also have some downsides, like higher costs compared to other options, possible delays, congestion in crowded areas, and a need for existing infrastructure, which may not be available in some places.

The Internet of Things (IoT) brings many benefits by connecting devices, sensors, and objects. It allows large amounts of data to be collected and shared, helping organizations make better decisions and work more efficiently [24]. In factories, IoT connects smart devices to improve processes and increase productivity [25]. Businesses can use IoT to innovate and grow, making services better and improving customer experiences [26]. IoT is scalable, meaning it can grow with the needs of the market, and its automation features help manage resources more effectively. In general, IoT changes industries by making systems smarter and more responsive, leading to better results for businesses and customers alike.

As a new and emerging field, it has proven to be effective in enhancing various aspects of life including health [27], where it is being used to allow remote patient monitoring and better health care service provision. Smart Cities is another application, in which IoT is being used to come up with solutions that allow smart parking, roads and IoT integrated infrastructures to solve issues related to traffic and congestion [28]. In addition, IoT has been applied in agriculture, smart homes, industries [29], etc. Figure 2 summarizes some of the applications of Internet of Things;

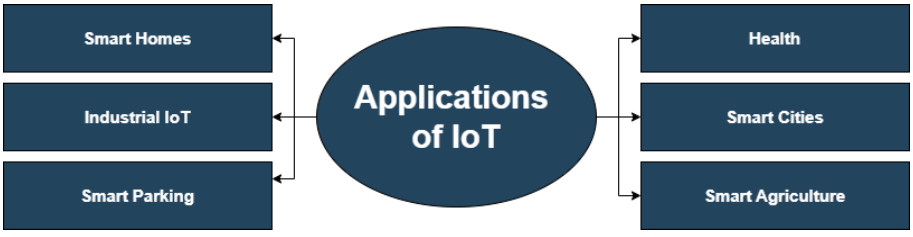


Figure 2: Some Common applications of IoT

1.2.2. Machine Learning

Machine learning is a division of artificial intelligence (AI) that concentrates on developing algorithms and models that enable computers to recognize patterns and make decisions or forecasts without explicit programming [30]. Types of machine learning includes; supervised, unsupervised and reinforcement learning [31].

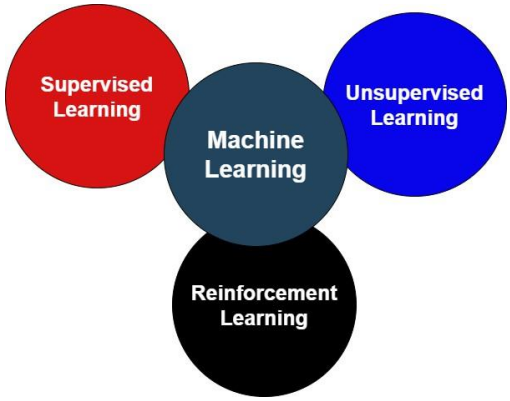


Figure 3 : Types of Machine Learning

In **supervised learning**, a model learns on a dataset that includes labels, meaning the input data corresponds to the correct responses [32]. The model learns to forecast future outcomes by identifying trends in the relationship between inputs and outputs. Examples of supervised learning algorithms include linear regression, logistic regression, decision trees, random forests, support vector machines (SVM), K-nearest neighbors (KNN), naive Bayes, and gradient boosting machines (GBM) [33].

On the other hand, **unsupervised learning** involves working with unlabeled data, aiming to uncover hidden structures or trends within the data [34]. Techniques include clustering (like K-Means and Hierarchical Clustering), dimensionality reduction (such as PCA and t-SNE), anomaly detection (using methods like Isolation Forest), association rule learning (including the Apriori algorithm), topic modeling (like Latent Dirichlet Allocation), and feature learning (through Deep Belief Networks) [35].

Reinforcement learning is a method of teaching artificial systems to improve their performance by learning from trial-and-error experiences rather than relying solely on predefined examples from a knowledgeable teacher [36]. For example, Q-Learning is a model-free algorithm that learns the value of action-state pairs to determine optimal actions by maximizing expected future rewards. Secondly, Deep Q-Networks (DQN) enhance Q-learning by using deep neural networks for high-dimensional state spaces, often in video games. SARSA (State-Action-Reward-State- Action) is an on-policy algorithm that updates Q-values based on the action taken by the agent instead of the optimal action [37].

1.2.2.1 Neural Networks

Neural networks are computer systems inspired by how the brain processes information. These systems focus on brain-style computation, using many simple units called neurons that work together in parallel, unlike traditional computers that handle tasks one step at a time [38]. Each neuron has an activity level and connections to other neurons, receiving inputs that are adjusted by weights (similar to brain connections). These weights determine how much influence each input has on the neuron's activity.

The first layer is the input layer, which takes in the raw data. The hidden layers, found between the input and output layers, change the data by using weights and activation functions. The last layer is the output layer, which gives the final result or prediction. Each layer learns patterns from the data, and as the network gets deeper, it can understand more complex features, making it better at tasks like recognizing images, understanding language, or making predictions [39].

To model acoustic audio with a neural network, sound is converted into features (like spectrograms or MFCCs), which the network processes to learn patterns and provide results, such as classifying or detecting sounds.

1.2.2.2 Deep Learning

Deep learning is a subset of machine learning that employs computational frameworks consisting of multiple processing layers to autonomously learn and represent data across various levels of abstraction [40]. This approach has notably enhanced domains like speech recognition, visual object identification, and object detection, as well as finding utility in areas such as drug discovery

and genomics.

Deep learning applications encompass visual, audio, and text processing, as well as social network analysis and natural language processing, leveraging algorithms like convolutional neural networks and generative adversarial networks to transform data abstraction and enable advancements in information processing across diverse domains, despite challenges like unsupervised learning and black-box models [41].

1.2.3 Tiny Machine Learning

TinyML (Tiny Machine Learning) is a new and emerging field under the umbrella of machine learning where models can be trained and deployed in tiny embedded devices or microcontrollers [42]. TinyML is important because it enables low-power, on-device machine learning, providing low latency, improved data safety and privacy, cost-effectiveness, and reliable performance for IoT applications, particularly in environments with poor connectivity [43].

1.2.3.1 Benefits of TinyML

Some of the benefits of TinyML are highlighted in Figure 4 which includes Low Power consumption, On-device processing, reduced latency, privacy and security, cost efficiency, and real time decision making.

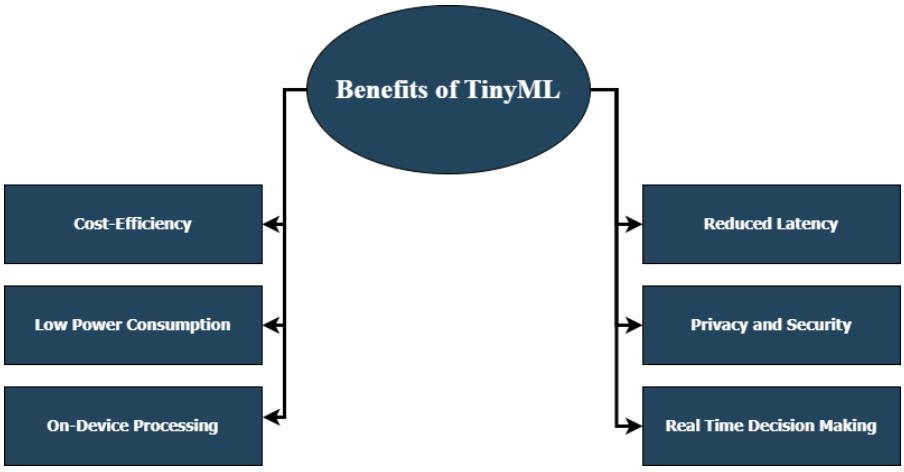


Figure 4 : Some of the Benefits of TinyML

Low power consumption: TinyML runs on devices that use very little energy, which helps extend battery life. This is important for devices that need to operate for long periods without being recharged. By using less power, TinyML makes it easier to place devices in remote areas where power sources may be limited.

On-Device Processing: TinyML processes data directly on the device instead of sending it to the cloud. This means that the device can quickly analyze information and act without waiting for a response from the internet. This is especially useful in places where internet connections are weak or unreliable.

Reduced Latency: TinyML reduces the time it takes to get results by analyzing data on the device immediately. This quick processing is important for applications that need fast responses, like health monitoring or safety systems. By cutting down on delays, TinyML improves user experience and makes systems more efficient.

Privacy and Security: TinyML keeps sensitive information on the device, which helps protect user data. By not sending data to the cloud, the risk of data breaches is lower. This feature gives users more confidence that their private information is safe, making TinyML a good choice for handling personal data.

Cost Efficiency: Using TinyML can save money by lowering the need for cloud services and reducing data transmission costs. This cost-saving aspect is especially helpful for businesses that want to implement many IoT devices. By cutting down on expenses while still providing useful functions, TinyML is a smart option for budget-conscious organizations.

Real-Time Decision Making: TinyML enables devices to make quick decisions by analyzing data right away. This ability to act fast is important for applications that need immediate responses, such as in safety or home automation systems. Quick decision-making improves the overall effectiveness and reliability of IoT solutions.

1.2.3.2. TinyML Key Technologies

This section describes some of the key technologies that have contributed to the advancement and possibility of Tiny machine learning.

1.2.3.2.1. Microcontrollers

Microcontrollers (MCUs) play a crucial role in TinyML, particularly low-power MCUs like the Arm Cortex-M series (e.g., Cortex-M0, M3, M4, M7), which are favored for their low power consumption and adequate processing capabilities [44]. Additionally, certain microcontrollers are specifically designed for AI applications, such as the Google Coral Edge TPU, which accelerates machine learning inference, enhancing the performance of ML tasks in resource-constrained environments [45].

1.2.3.2.2 Smart Sensors

Smart sensors are advanced devices that combine several functions into traditional sensors. They process and convert signals from the environment into digital data, allowing for easy communication and analysis. Made using CMOS technology, these sensors are important for future systems that need precise control and monitoring. They can be used in many areas, including cars, healthcare, industrial processes, and consumer electronics, improving their performance and addressing various challenges [46]. Therefore, smart sensors have enhanced TinyML by enabling real-time data processing and analysis on low-power devices, facilitating intelligent, autonomous solutions across various applications.

1.2.3.2.3 Neural Processing Units

Neural Processing Units (NPUs) are specialized hardware designed to accelerate deep neural network (DNN) computations, particularly in cloud services [47]. They offer significant advantages, such as cost efficiency through resource sharing via consolidation and virtualization, and the ability to manage high-priority tasks effectively with "preemptible" capabilities. This allows NPUs to temporarily pause lower-priority tasks to meet stringent latency requirements for critical inference tasks.

1.2.3.2.4 Memory

Microcontrollers (MCUs) with onboard RAM, ROM, and flash memory are essential for TinyML, enabling the storage and execution of lightweight machine learning algorithms on resource-constrained devices. This efficient memory usage allows for real-time inference, reduced latency, lower power consumption, and enhanced privacy, facilitating effective deployment in various applications.

1.2.3.2.5 Development Boards

Development boards for TinyML, like the Arduino Nano 33 BLE Sense [48], Raspberry Pi Pico [49], and Google Coral Dev Board, combine small computers with enough memory and sensors to run machine learning models directly on the device. These boards often have good power management, options for Wi-Fi and Bluetooth, and work with popular machine learning tools.

1.2.3.3 Model Compression Techniques

Model compression means using methods to lower the resource needs of machine learning models, especially for use on low-power, limited-resource devices in edge computing and the Internet of Things (IoT) [50]. The aim of model compression is to keep the models accurate while making sure they can work well on devices with less processing power and energy. This is important in the TinyML approach, which focuses on making machine learning possible on small edge devices.

1.2.3.3.1 Quantization

Quantization is a method that reduces the precision of weights in deep neural networks (DNNs), allowing them to use fewer bits [51]. This makes the model smaller and needs less power, which is great for devices with limited resources. By improving how weights are represented, quantization speeds up processing and helps run machine learning applications on small devices like those used in IoT and edge computing.

1.2.3.3.2 Pruning

Pruning is a method used in neural networks that reduces the number of parameters to lower computing needs, often sacrificing some accuracy [52]. It is especially useful in very limited settings, like those in TinyML applications. However, pruning has challenges, such as needing special hardware and not enough research on its effectiveness for small models, which makes it harder to use widely.

1.2.3.4 Applications of TinyML

Recently, TinyML has successfully been deployed on a number of applications and more research is being done on the same.

In [53], TinyML is being applied in health to develop elder people's health status wearable devices. These devices, powered by small machine learning models, can track vital signs like heart rate, movement, and body temperature in real time. TinyML's ability to process data locally on the device, without needing a constant internet connection, ensures faster responses and improved privacy. This helps detect early signs of health issues, allowing for quick intervention, while also providing continuous monitoring to support the independence and safety of elderly individuals.

Air pollution monitoring based on TinyML is proposed in [54], where the model running locally on Raspberry pi Pico and the system is able to predict air quality. Running air quality prediction models on a Raspberry Pi Pico allows for real-time monitoring and faster responses while keeping data private. It is energy-efficient and cost-effective for long-term use, and since it works without the internet, it can track air quality continuously, making it great for remote areas.

In agriculture, a TinyML based smart sensor system for precision agriculture and predicting fruits production is proposed in [55]. The benefits include increased crop yields, reduced resource waste, and improved decision-making, allowing farmers to predict fruit production more effectively and sustainably. By leveraging local processing, the system ensures real-time data analysis and enhances data privacy, making it a valuable tool for modern farming practices.

A TinyML based gas leakage detection system is proposed in [56], where an edge-based model is deployed into Arduino Nano 33 BLE Sense Microcontroller which supports Machine learning. The system offers real-time monitoring for immediate leak detection, enhancing safety. It processes data locally, which speeds up responses and keeps information private. Its compact and energy-efficient design allows for continuous use, making it a cost-effective solution for homes and businesses to ensure safer environments. These and more are some of the applications of TinyML and more research is being done.

1.3. Problem Statement

Internet of Things (IoT) based solutions for road accident detection and emergency alert systems have presented a limited approach. The use of accelerometer sensors can lead to triggering false alerts when a vehicle goes through a pothole or when the driver makes sudden stops [58]. Since accelerometer sensors measure a car's speed and acceleration, they can sometimes make mistakes. When a car hits a pothole, the sensor feels a sudden change in acceleration, which can seem like a crash, leading to a false alert. The same thing happens when a driver makes a sudden stop—the sensor detects the rapid change in acceleration and thinks it's an accident. Because the accelerometer only senses changes in acceleration, it can't tell the difference between a pothole, a sharp stop, or a real crash, causing confusion in accident detection.

Additionally, the alerts generated are GSM-based, a technology with drawbacks such as bandwidth lag and limited data transfer rates [59] [60]. Bandwidth lag in GSM occurs due to limited frequency allocation and high network traffic, causing delays in message delivery. Limited data transfer rates restrict the amount of information transmitted at once, resulting in slower communication and potential missed alerts in critical situations.

Using cameras or visual data is also unreliable and insufficient for understanding vehicle activities or spotting dangerous situations. For example, a camera might fail to detect a vehicle crash if it's out of view, and cameras depend on illumination. In contrast, audio-based detection systems work well in any lighting conditions, both day and night, and can detect a crash regardless of the angle of focus as long as it is within their detection range [61].

Moreover, machine learning can be effectively applied to detect accidents if the solutions are based on TinyML. TinyML presents an efficient approach that uses edge-based, low-power devices capable of running data analytics and algorithms that can be deployed anywhere and in various applications [62].

Since the time delay in accessing medical care contributes to worse injuries and higher mortality rates, there is a critical need to effectively use machine learning and TinyML to accurately detect vehicle crashes and reduce the time delay in accessing emergency and medical services to save lives.

1.4. Study Objectives

1.4.1. General Objective

The main objective of this research is to develop a TinyML- and IoT-based Vehicle Crash Detection and Emergency Alert System that accurately detects accidents and triggers a LoRaWAN-based communication module to send the GPS location. Finally, the system will alert emergency services in the most effective and efficient way.

1.4.2. Specific Objectives

i. To collect vehicle crash acoustic secondary data;
ii. To Train a Model that will detect a Vehicle Crash accident using acoustic data;
iii. To develop a GPS Location transmitter using LoRaWAN communication, and
iv. To develop a real-time Accident Alerting System.

1.5 Hypothesis

The main hypothesis of this study is that the implementation of a TinyML and IoT – based Vehicle crash system using acoustic data and Lora technology will significantly contribute in curbing the devastating effects of road accidents including deaths.

In addition, that a TinyML based model will offer low power consumption, real-time data processing, and the ability to run machine learning models on edge devices, enabling efficient and responsive applications in resource-constrained environments.

Finally, it is hypothesized that a Lora based communication module will allow sending of alerts over long distances which is essential in accident detection.

1.6 Study Scope

The study scope of this research includes;

i. Collecting and processing of acoustic based vehicle crash dataset which will be required for training a machine learning model.
ii. Training an acoustic based TinyML model that will accurately classify accident and no accident scenarios.
iii. Configuration of a LoRaWAN based communication system that will send GPS coordinates to the receiving end.
iv. Development of an alert system and mapping the actual GPS location of the accident in Arduino Cloud platform.

CHAPTER 2: LITERATURE REVIEW

This chapter provides an overview of the current advancements in this field. It consists of two sections: Section 2.1 will focus on the application of Internet of Things to detect accidents and trigger emergency alerts. Section 2.2 will explore some of the related works that uses machine learning and TinyML based techniques to detect road accidents.

2.1 IoT -Based Accident Detection Systems

In designing and implementing solutions relating to detecting accidents, emergency alert warning systems are crucial in order to provide a real time communication to the hospital emergency sections or control rooms for medical support. Some researchers have efficiently applied IoT in accident detection and sending alerts for emergency medical support.

In [63], a Raspberry pi-based accident detection system using accelerometer and vibration sensor is proposed. Using an android application for server service, a GSM and GPS module is being used to send alerts about the accident to the relevant authorities.

Authors in [64], propose an automatic accident detection and notification system based on a smart phone. An external pressure sensor is being used to measure vehicle parameters and the data is being sent the smart where data analytics and final decision is made. Finally, the system sends a notification message regarding the accident.

An ESP32 MCU based intelligent accident detection system for emergency medical services is proposed in [65]. Using an Impact Switch Collision Sensor, the ESP32 MCU is able to track real time parameters and it is able to detect an accident. After detecting the accident, the ESP32 MCU triggers an alert signal which is in form of audio message and GPS location. Not only have that, but the MCU triggers the camera module connected to its input pinned to take pictures that are sent together with the alert message.

An automated Vehicle accident detection and alert system is proposed by authors in [66]. It's a NodeMCU based system that uses an accelerometer sensor in order to detect sudden deceleration and alert selected contacts and emergency services.

The research in [67] aims to create a system that helps prevent and detect bike accidents, given the high number of two-wheeler crashes. The system uses sensors, GPS, and GSM to detect accidents and send alerts to hospitals and authorities. It also prevents further accidents and checks if the driver is in a safe condition to drive. Additionally, it ensures that only people with valid driving licenses can operate the bike by using an RFID reader. The goal is to improve safety, save lives, and respond quickly in emergencies.

In [68], the authors developed a system to detect vehicle crashes using Arduino, GPS, GSM, and an accelerometer. The system detects sudden vehicle movements and sends a message with the accident location through a Google Maps link, along with the vehicle's speed. It can also be modified for tracking and other purposes with minimal changes to the hardware and software.

The research in [69] focused on the development of a system that automatically detects accidents using Vehicular Adhoc Network (VANET) and Internet of Things (IoT). It employs sensors in vehicles to identify accidents and assess their severity. When an accident occurs, a message is sent to a hospital, which determines the location of the accident and the nearest medical center. An ambulance is then dispatched, and alert messages are generated to clear the way for the ambulance to reach the accident site.

In [70], researchers focused to improve how quickly emergency services respond to car accidents by developing a smart unit installed in vehicles. This device collects and sends vehicle information using different sensors safely stored in a strong black box, even after an accident. The project hopes to allow fast accident alerts, help track stolen cars, remotely turn off vehicles, and quickly send accident details to insurance companies. In the end, the study introduces a prototype that enhances vehicle safety features and the reliability of accident detection and reporting systems, making it useful for car manufacturers.

These and more are some of the applications of IoT in the development of road accident detection and emergency alert systems.

2.2 Machine Learning and TinyML-Based Systems

In [71], authors propose a deep learning based Convolutional Neural Network approach to detecting road accidents. A machine learning model is trained using 4000 images to detect accidents and out of 2000 images, the model correctly classifies 1702 images giving them an accuracy of 85%. The system uses a camera module which is deployed on accident prone areas and the model classifies images from the video streams. After detecting the accident, the system triggers an alert signal using GSM communication model. The problem in the system is that cameras are deployed on accident prone areas only, of which accidents can occur anywhere even on the least prone areas. In this case, two vehicles can crash into each other or hit a pedestrian and since the cameras are not deployed in this area, the accident will not be detected.

An accident detection system using Convolutional Neural Network is proposed in [72]. The proposed model is a fusion of Convolutional Neural Network and Long Short-Term Memory layers to classify video feeds from CCTV which are installed in highways. InceptionV3 was found to be the best after comparing it with other models and it was fast, with an accuracy of more than 95%. Finally, the model was deployed in Raspberry Pi 3 model B+ and the GSM was used as an alert communication model. Since the system uses CCTV in highways, the detection location is biased because an accident can happen anywhere, not only in highways.

A Convolution Neural Network and TensorFlow based accident detection and report system is proposed in [73]. The authors trained a model to detect accidents and got an accuracy of 84%. When the model successfully detects an accident using images, it reports or sends an alert through email. In as far as alerting the emergency service for an accident is concerned, sending an alert through Email is the least effective way. Even if the model successfully classifies and detects an accident, sending the alert through Email is not an effective approach. An Email is internet based and there's very low probability to read it in real time. In other words, it will depend on the control room or emergency service to check their emails regularly. Finally, the trained model is not deployed on any Machine learning supporting microcontroller but runs on a personal computer which cannot be deployed effectively.

In [74], an accident detection and severity analysis using Convolutional Neural Network is proposed. The authors present an image processing and deep learning based real time image

classification. The trained model is deployed on the internet and has a training and testing accuracy of 84.5% and 96.8%, respectively. When the model successfully detects an accident, it sends an SMS to the Hospitals for emergency medical support provision. The researchers compared the results of using CNN and VGG16 pre-trained model which proved to be the best. In addition, the model is able to predict the accident and cost required for insurance companies. The trained model is running on the web, which can only be accessed using a personal computer. As such, it was not deployed for effective detection of road accidents. In addition, the use of GSM is not an effective mode of communicating the occurrence of an accident.

A machine learning model that detects accidents and is hosted in the cloud is proposed in [75]. The model collects various sensor data such as position, pressure, gravitational force, speed, etc. Once the data is sent to the cloud, the model classifies and detects an accident according to the data received. The authors used InceptionResNetV2 which gave them an accuracy of 98% for testing. All sensors are connected to ESP8266 and Raspberry Pi which has a camera attached to it that also takes image data. Finally, when the model successfully detects an accident, it sends a signal that triggers GSM module to send an alert. With the model being hosted in the cloud, there are issues to do with bandwidth since data about various parameters has to be sent to the cloud. In addition, there is high latency as the system is based on cloud computing. In addition, an alert sent through GSM cannot present an effective communication module to the receiving end, since it's just a message.

In [76], authors propose an accident detection using machine learning and artificial intelligence techniques. Their proposed system is based on Computer Vision accident detection using YOLOV3 algorithm. When the trained model successfully detects an accident, it sends an alert to emergency team through an android application. The alert sent contains a GPS link, pictures and video of that accident. At the same time, the model uses CCTV footage in order to classify video frames into accident or no-accident classes. Like the other proposed systems, the use of CCTV is not effective in detecting accidents because an accident can happen anywhere, even in places where there are no CCTV cameras. In addition, the use of Android application for alerts depends on internet connection and it's not an effective approach to producing an alert that will result in real time response from the emergency service.

Authors propose an accident occurrence detection system that classifies images into accident or no accident classes [77]. The proposed approach to accident detection is a combination of Internet of Things and machine learning. Using K-Means clustering, an accuracy of 96% is achieved and once an accident is successfully detected, they system sends a message to the registered individuals. The system uses sensors like accelerometer, gyroscope, camera, etc., to send data to a Raspberry pi MCU that is running the trained machine learning model. As pointed out earlier, using accelerometer does not guarantee accuracy since it can lead to false alarms

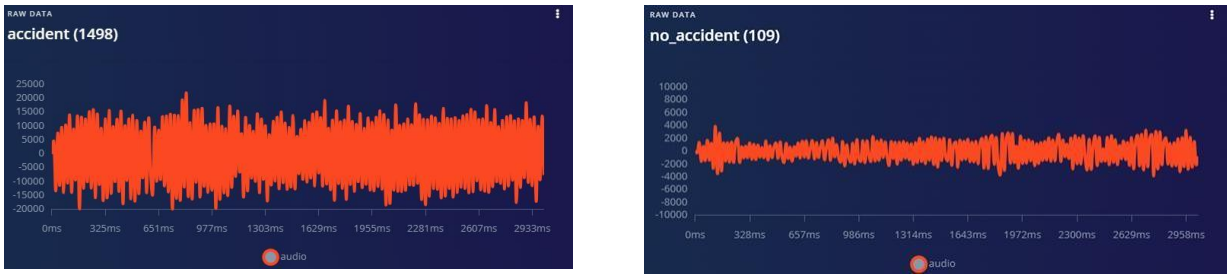
CHAPTER 3: RESEARCH METHODOLOGY

This chapter comprises of a detailed description on how we achieved the objectives of this research.

3.1. Vehicle Crash Acoustic Dataset Collection and Processing

After downloading the dataset from MIVIA's website, the dataset was analyzed and recorded using Arduino Nano 33 BLE's in-built sound sensor. Hence, the audio was played and the Arduino Nano 33 BLE was recording these data and sending it to the serial where the python script was recording them into 3 seconds chunks. Likewise, another class of acoustic data was collected where now there was a normal driving vehicle sound with no vehicle crash and skidding sounds.

Then, the dataset was uploaded into EdgeImpulse platform for further processing and model training. The figure 5 shows some samples of the dataset for accident_detected and no_accident_detected classes.



(a) (b)

Figure 5 : Accident-detected sample (a) and no-accident data sample (b)

Overall, the amount of data collected was amounting to 8 hours, 32 minutes and 46 seconds, a total of 15,370 audio files or chunks, 7,685 files from each class.

The dataset collected was split into 80% and 20% for training and testing split, respectively. As such, there were 12,300 files in the training and 3083 files in the testing split as shown in figure 6.

Parameter	Details
Data Collected	8 hours 32 minutes 46 seconds
Train/Test Split	80% / 20%
Training Samples	12,300
Test Samples	3,083

Figure 6 : Data Acquisition and Split

3.2. Impulse Design

After the data was uploaded into EdgeImpulse as described in the previous sub-section, an impulse was designed that included the processing block and the learning block.

Window Size which refers to the length of the audio segment (or frame) that is analyzed at one time during feature extraction. It defines how much audio data is captured for processing to extract features like Mel-frequency cepstral coefficients (MFCCs) or spectrograms or MFE. In this case the window size was fixed to 3 seconds. **Window increase** refers to the practice of gradually increasing the size of the analysis window in acoustic modeling to capture more context or frequency information over time was set to 500 and finally the frequency was 16KHz as shown in figure 7.

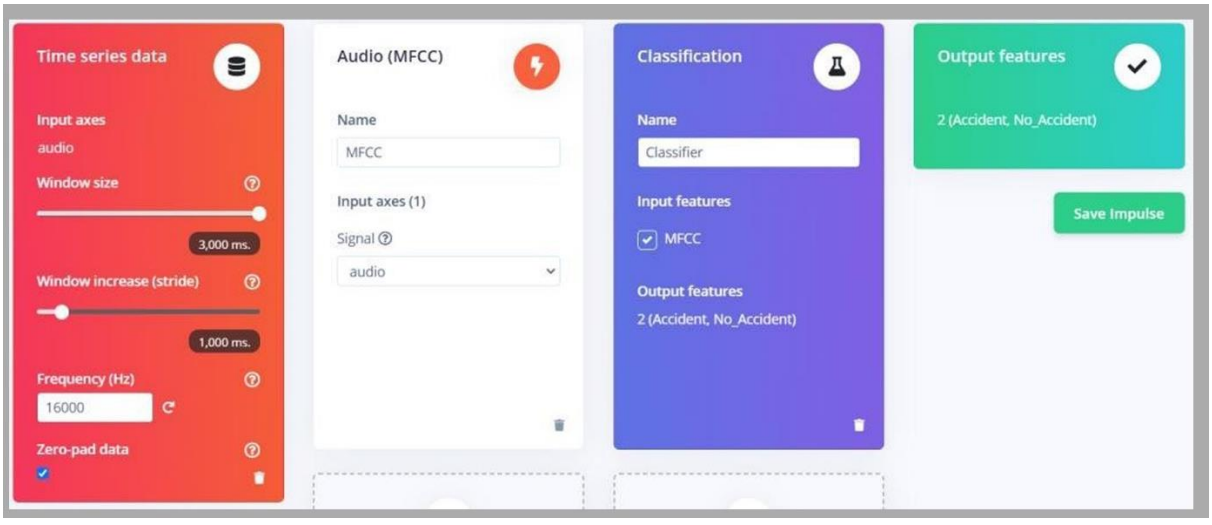


Figure 7 : Impulse Design in EdgeImpulse

As shown in figure 7, the first audio processing block used was MFCC which is a method to represent audio signals, especially for recognizing speech. The process starts by recording the audio and

breaking it into small, overlapping pieces. Each piece is changed from a time signal to a frequency signal using a method called Fast Fourier Transform (FFT), which shows the frequencies present [88]. These frequencies are then adjusted to the Mel scale, which reflects how humans hear sounds. Finally, the Discrete Cosine Transform (DCT) is used on the Mel-scaled frequencies to create MFCCs. This method helps computers understand audio signals better, making it useful for speech recognition. Likewise, the process was done for other audio processing techniques; MFE and Spectral.

After extracting the meaning features from the acoustic data, a model was trained using the classifier learning block as shown in figure 14 above. This meant that the output would be classified into tow class; **accident-detected** and **no-accident**. Finally, the impulse was saved, ready for the next step.

3.3. Generating Features

The next step was to generate features for the training data that was upload. This means, the processing block of choice, MFCC, MFE and Spectral, were used separately and compared to extract meaningful information from the dataset that was uploaded and, in the training, split so that it's in a form that can be used to train a model. The results obtained by comparing the three audio processing techniques were compared to choose the best that will be used to train the model.

3.4. Neural Network Settings

Next step was to train the model using the neural network settings described below;

Number of training cycles is how many times the training algorithm goes through all the training data. During each pass, it adjusts the model's settings using back-propagation. Each full pass is called an epoch or training cycle. In this case, the number of training cycles or epochs was set to 100.

Learning rate which decides how much the model's settings are changed with each step of training or how fast the neural network learns. In our case, the learning rate was set to 0.005 for effective learning which is recommended.

Validation set size is the percentage of your training data that is set aside for validation. In our case, the validation size was set to 20% as explained earlier during data acquisition.

The batch size is the number of samples used in each training step and we used the default value which is 32.

3.5. Neural Network Architecture

In the neural network architecture, the extracted features from your data serve as inputs and are processed through multiple layers. In our case, the neural network architecture is shown in figure 11;

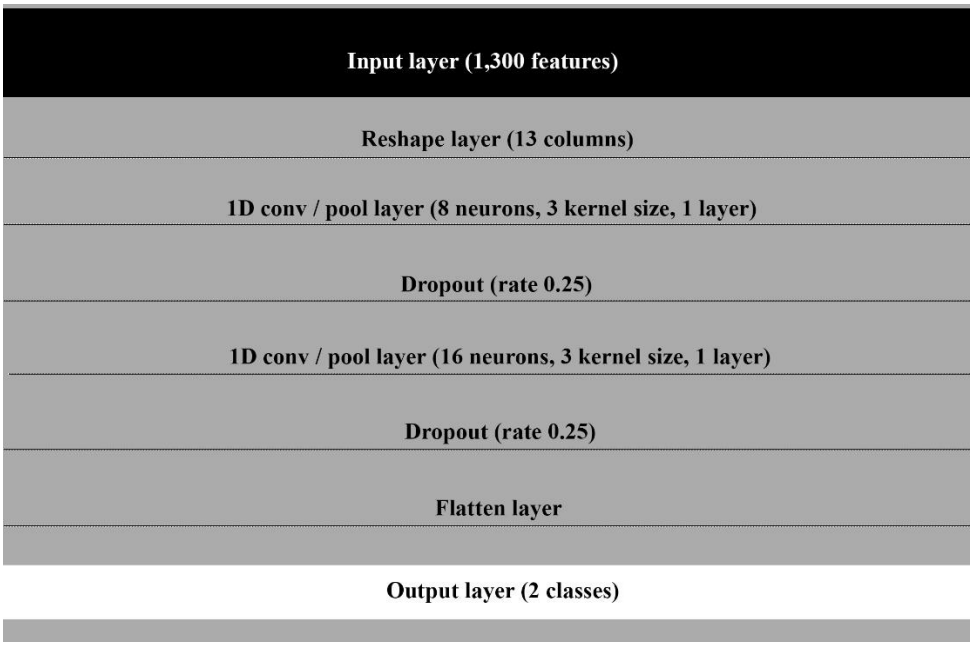


Figure 8 : Neural network architecture

The neural network was composed of several key layers, each serving a specific purpose in the architecture. The input layer accepted 1,300 features, which were then reshaped into 13 columns by a reshape layer. This reshaped input was processed through two 1D convolutional/pooling layers: the first containing 8 neurons and the second 16 neurons, both utilizing a kernel size of 3.

Each convolutional layer was followed by a dropout layer with a rate of 0.25, implementing a regularization technique to mitigate overfitting. After the second dropout layer, a flatten layer converted the multi-dimensional output to a one-dimensional array.

The network ended with an output layer designed for binary classification, producing 2 classes as the final output. The whole architecture is shown in figure 12.

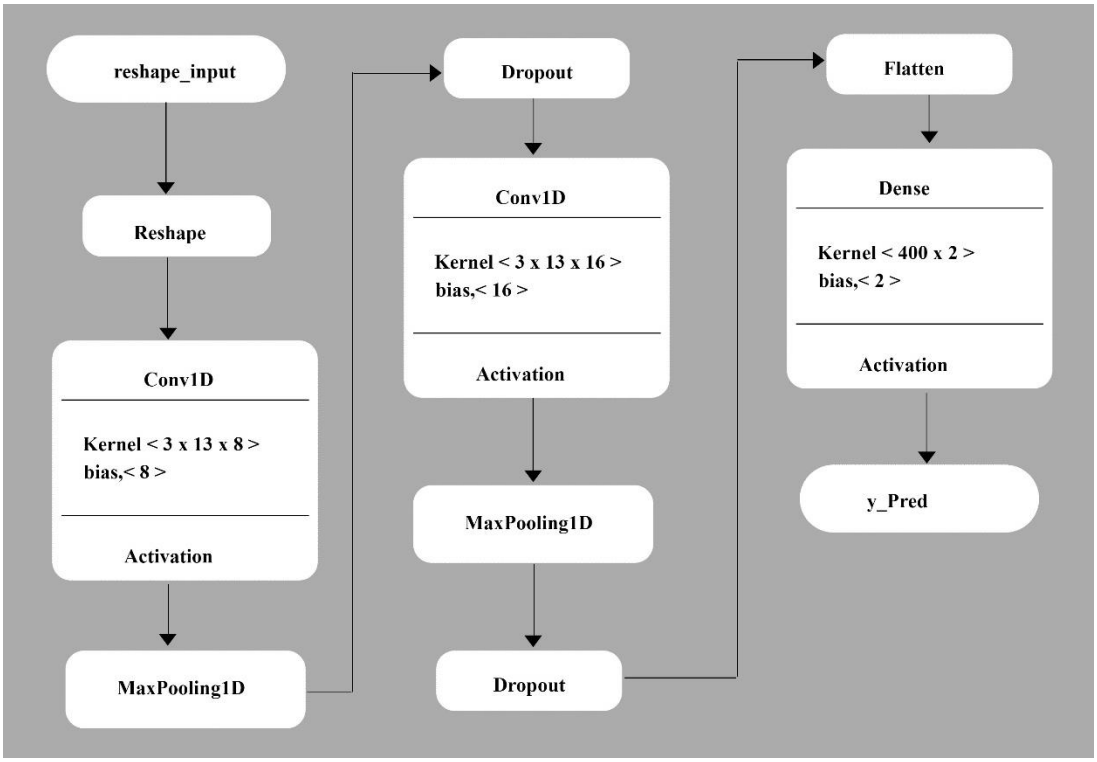


Figure 9 : Neural Network Architecture overview

The model was trained and results were obtained which will be discussed later.

3.6. Hardware Implementation

This section will describe how the hardware was developed in order to achieve the objectives of this research. Section 3.6.1 will focus on the implementation of the vehicle crash detection and transmitter system. Later on, section 3.6.2 will describe how the Lora receiver and the alert warning system was developed.

3.6.1. Vehicle Crash Detection and Lora Transmission System

This involved the connection of the GN-801 GPS receiver, the OLED Display and the Lora SX1278 module to Arduino Nano 33 BLE Sense board.

The Lora SX1278 was interfaced to the Arduino Nano 33 BLE Sense by connecting its pins as shown in table 1. The SX1278 module features several pins, each with a distinct function.

The **VCC** and **GND** pins provide power and establish ground for the module. **SCK**, **MOSI**, and **MISO** pins handle data transfer between the microcontroller and the module using **SPI communication**. The **NSS** pin selects the module during communication, while the **DIO** pin manage interrupts and events like data transmission. Lastly, the **RESET** pin resets the module to its default state. These pins enable efficient communication and control, supporting long-range, low-power IoT applications.

Lora SX1278 Transmitter	**Arduino Nano 33 BLE Sense**
GND	GND
VCC	3.3V
RST	D9
NSS	D10
SCK	D13
MOSI	D11
MISO	D12
D100	D2

Table 1 : Lora SX1278 and Arduino Nano 33 BLE Sense connection

Next, the GN-801 GPS Receiver was also connected to the Arduino Nano 33 BLE Sense with the pin configuration shown in table 2. For the GN-801, only four pins were connected to the Arduino Nano 33 BLE, the VCC, GND, TXD and RXT Pins.

GN-801 GPS Receiver	**Arduino Nano 33 BLE Sense**
VCC	5V
GND	GND
TXD	RX
RXD	TX

Table 2 : GN-801 GPS Receiver to Arduino Nano 33 BLE Sense configuration

In addition, the OLED Display was also connected to the Arduino Nano 33 BLE Sense board so that we are able to show inferences as they run on the board in real time.

For the OLED Display, the connections were made as shown in table 3.

OLED Display	**Arduino Nano 33 BLE Sense**
VCC	5V
GND	GND
SDA	SDA
SCL	SCL

Table 3 : GN-801 GPS Receiver to Arduino Nano 33 BLE Sense configuration

Finally, all components were soldered on the prototype PCB, with power being supplied by the Li-on batteries as shown in the figure 13.

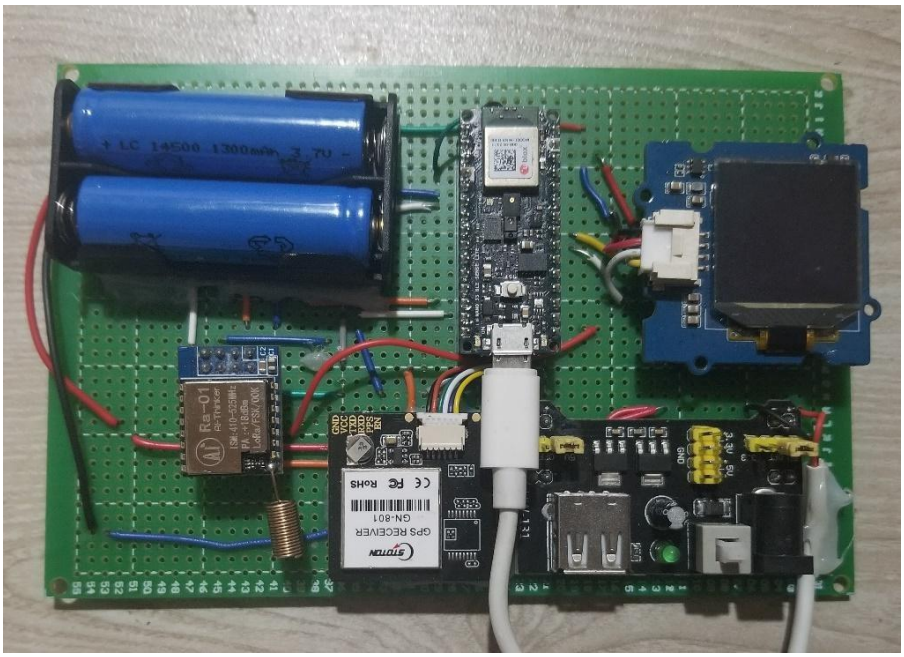


Figure 10 : Vehicle Crash Detection and Lora Transmission system prototype

3.6.2 Lora Receiver and Alert Warning System

The pin configuration for the LoRa receiver was set up as shown in table 4, with the SX1278 connected to the Arduino Nano 33 IoT board, as illustrated.

Lora SX1278 Receiver	**Arduino Nano 33 IoT**
GND	GND
VCC	3.3V
RST	D9
NSS	D10
SCK	D13
MOSI	D11
MISO	D12
D100	D2

Table 4 : Lora Receiver to Arduino Nano 33 IoT configuration

In addition, the buzzer was connected to D7, Red LED was connected to D6 and the Green LED was connected to D4 of the Arduino Nano 33 IoT board. Likewise, the battery pack and prototyping breadboard regulator was soldered together with the rest on a PCB as shown in figure 14.

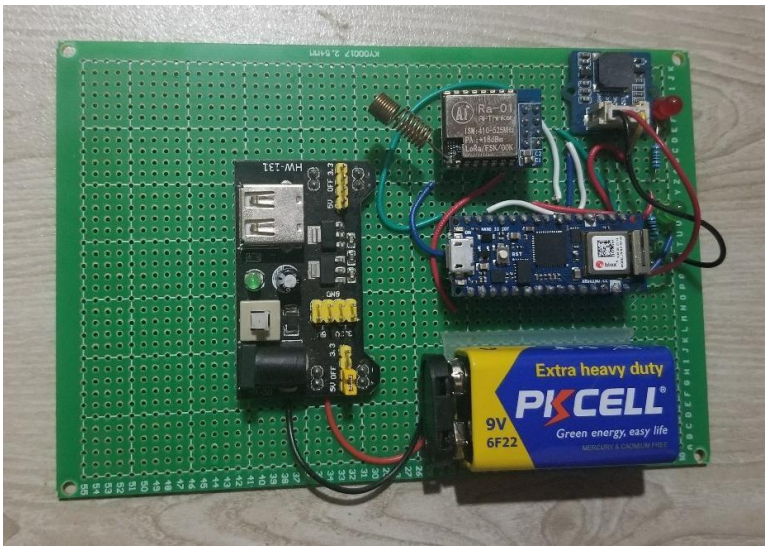


Figure 11 : Lora Receiver and Alert system Prototype

3.7. Model Deployment and Optimization

This section focuses on the deployment and model optimization steps that were taken in order to have the model deployed on Arduino Nano 33 BLE Sense.

3.7.1. Model Deployment

After setting the hardware, the trained model was then deployed in the Arduino Nano 33 BLE Sense microcontroller so that it can run inference within the device. EdgeImpulse provides multiple deployment options and a model can be deployed using any of these available options depending on the task and preference. In this case, the option we used to deploy our model was through the Arduino Library.

3.7.2. Model Optimization

For model optimization, the EON Compiler was used, which is like another layer that is added to the impulse for the sake of optimization. The EON Compiler allows you to run neural networks using less RAM and other limited resources on small microcontrollers while maintaining the accuracy. The available options for the EON compiler included Quantized (int8) and unoptimized (float32). These metrics were considered in order to choose the model that was deployed in our device.

	Quantized(int8)	**Unoptimized(float32)**
Accuracy	Quantization usually results in a slight accuracy reduction due to the lower precision, particularly in complex models or tasks that are highly sensitive to numerical accuracy.	Model maintain greater accuracy because of their complete precision, which is especially advantageous for tasks that demand detailed numerical accuracy.

Model Size	By using 8 bits to represent each value, the model size is about four times smaller than its float32 equivalent.	These models are larger because each parameter is represented using 32-bit values, leading to higher memory usage.
Inference Time	Offer faster inference since 8-bit integer arithmetic is more efficient on low-power hardware. This increase in speed is especially crucial for real-time TinyML applications.	Models typically take longer to execute because 32-bit floating-point arithmetic is more resource-intensive on most low- power devices.
Energy Consumption	Lower precision arithmetic (int8) results in decreased power consumption, making int8 models more energy-efficient, which is ideal for battery-operated devices.	Float32 models require more energy because of their greater computational complexity.

Table 5 : Model comparison for Quantized and Unoptimized Models

3.8. Arduino Cloud Platform Setup

This step involved opening account on the platform, setting the device by adding it to the platform. Since it's the receiving end, the device added was the Arduino Nano 33 IoT microcontroller.

Then we created a “thing” which in this case would be representing the GPD data and we created a variable with the type Location since it will be handling latitude and longitude data as shown in Figure 15.

Figure 12 : Setting up the GPS mapping in Arduino Cloud IoT

After this, the whole setup was complete and then next thing was to test and see how the proposed solution was performing.

CHAPTER 4: SYSTEM DESIGN

This chapter provides a detailed description of the proposed solution in terms of the hardware, software, tools and all platforms used in order to achieve the main objective of this research.

4.1. Overview of the Proposed Solution

The proposed solution is shown in figure 16 which is made up of two systems, the vehicle crash detection and Lora transmitter system, and the Lora receiver and emergency alert system.

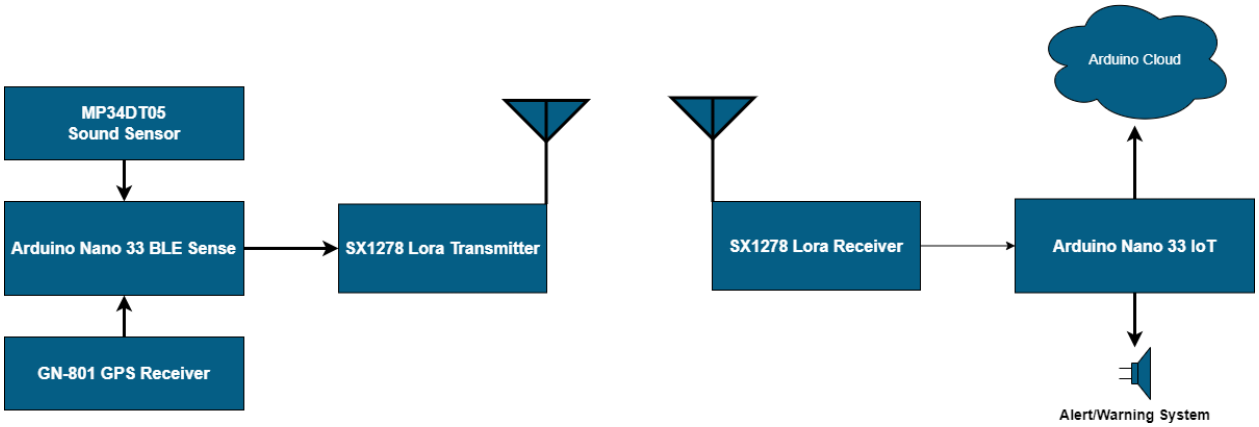


Figure 13 : Proposed solution system design

4.2. System Block Diagram

The vehicle crash detection and Lora transmitter, and the Lora receiver and alert system block diagram is shown in figure 17 and figure 18, respectively;

As shown in figure 17, the MP34DT05, which is Arduino Nano 33 BLE's in built sensor, is used as the sound sensor. The OLED display is connected to Arduino Nano 33 BLE and is used to show inferences that will be running within the device. This will display the live classifications as either accident-detected or no accident detected.

The GN-801 GPS receiver is used to read the GPS location and these coordinates are transmitted via Lora technology using SX1278 Lora Transmitter module.

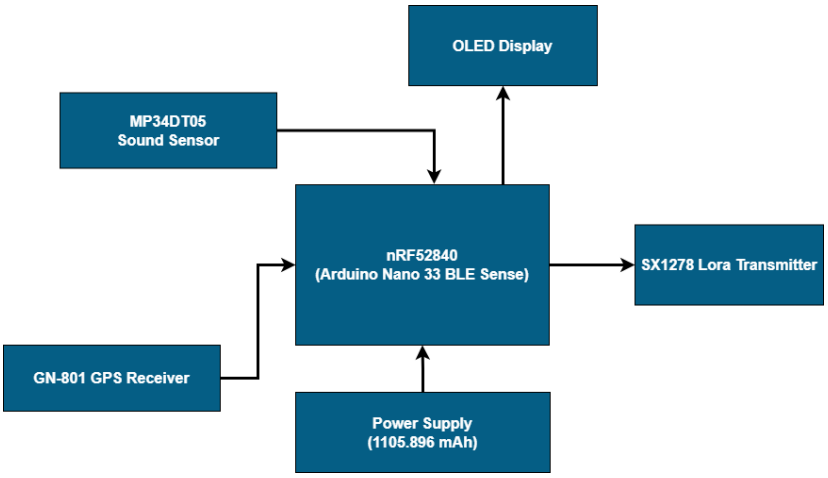


Figure 14 : Vehicle Crash detection and Lora Transmission system

On the Lora reception and emergency alert system, the SX1278 Lora receiver is used to receive the Lora transmitted coordinates, which are then sent to Arduino Cloud that maps the location. As soon as the coordinates are received, an alert signal is triggered to alert the relevant authorities about the detected accident so that they can respond in time.

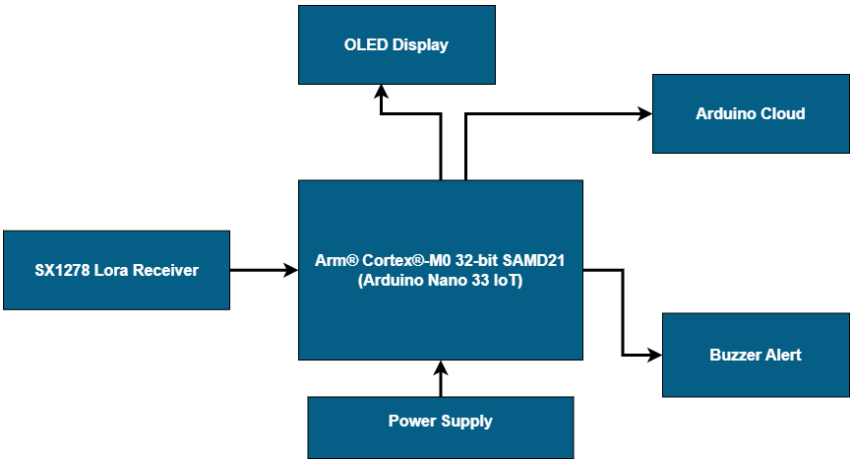


Figure 15 : Lora Reception and Alert Warning System

4.3. Dataset Collection

The dataset that was used in this research was audio-based secondary data called MIVIA_ROAD_DB1 that was collected from MIVIA's website [78]. Mivia is a research laboratory at the University of Salerno, specializing in Pattern Recognition and Computer Vision. The lab works on fundamental topics such as pattern classification, graph matching, and enhancing classification dependability. They also engage in applied research, including real-time video monitoring for surveillance, analyzing human behavior, interpreting biomedical images for automated diagnosis, and developing vision technology for robotics.

The dataset includes a total of 400 events for road surveillance purposes, such as tire skidding and vehicle collisions. These sounds were recorded using an Axis P8221 Audio Module and an Axis T83 omnidirectional microphone, with a sampling rate of 32,000 Hz and 16-bit PCM encoding. The audio recordings are available in WAV format. Each fold contains several audio files, each lasting around 1 minute, where a sequence of hazardous events is overlaid on typical road background noise. Each audio file has a distinct background sound, simulating a variety of real-life situations.

4.4. Software

This section explores the software and platforms that were used in implementing the proposed solution including EdgeImpulse and Arduino Cloud platform.

4.4.1. Edge Impulse

EdgeImpulse [79], is a web-based platform that allows you to create datasets, train models, and fine-tune libraries to run directly on devices, ranging from the smallest microcontrollers to gateways equipped with cutting-edge neural accelerators. In this context, an acoustic-based TinyML model would require uploading the dataset into the platform through a section called data acquisition. Secondly, impulse design follows which refers to the full pipeline of how raw data is processed and transformed into a machine learning model that can make predictions. It defines the steps needed to turn sensor data (like sound, accelerometer readings, or images) into something that can be classified by a machine learning algorithm. It starts with input, which involves capturing raw sensor data such as audio, accelerometer, or image data.

The next step is preprocessing, where the raw data is cleaned and structured through techniques like feature extraction or signal processing. This processed data is then fed into a learning block, which is a machine learning model, such as a neural network, trained to recognize patterns. Finally, the output provides a classification or regression result based on the input data.

4.4.2. Arduino Cloud Platform

The Arduino Cloud platform [80] is an online service that helps users connect, manage, and monitor their Arduino devices over the internet. It offers an easy-to-use interface for creating and running IoT (Internet of Things) applications, allowing users to see data from their devices, control them from afar, and automate different tasks. With features like remote access, data tracking, and support for various sensors and actuators, the Arduino Cloud makes it easier to develop IoT projects. Its benefits include simple device management, real-time data viewing, and the ability to work with others on projects, which speeds up development and makes it easier to expand IoT solutions.

In this context the platform will be used to plot the actual location where the accident has been detected using latitude and longitude.

4.5. Hardware Components

This section dives deep into the hardware components that were used in implementing the proposed solution. Firstly, we explore at the hardware components used to deploy the vehicle crash detection model and the Lora transmission system. After that, a Lora receiver and emergency warning system is also analyzed in terms of the hardware components used.

4.5.1. Arduino Nano 33 BLE

The Arduino Nano BLE Sense [81], shown in Figure 8, is a small board that comes with various environmental sensors and can run machine learning models using TinyML and TensorFlow Lite. It uses the nRF52840 microcontroller and operates on Arm Mbed OS. This board can connect via Bluetooth Low Energy and has sensors to detect things like proximity, motion, temperature, and humidity.

It includes 13 LED pins, 14 digital input/output (I/O) pins, 8 analog input pins, and 5 pulse-width modulation (PWM) pins, with all digital pins able to handle external interrupts. The board has a built-in Bluetooth module for wireless connections and features sensors for sound, gestures, light, barometric pressure, temperature, and humidity.

For memory, it has 256 KB of SRAM and 1 MB of flash memory. The Arduino Nano BLE Sense also supports various communication methods, including UART, I2C, and SPI, as shown in Figure 19. This makes it a powerful tool for building Internet of Things (IoT) projects and experimenting with machine learning applications.

Figure 16 : Arduino Nano 33 BLE [82]

4.5.2. Lora SX1278 Transceiver

The Lora SX1278 is a wireless module that uses SEMTECH's SX1278 transceiver and advanced LoRa technology, allowing communication over distances of up to 10,000 meters [83]. It has strong resistance to interference and an air wake-up feature, making it perfect for use in crowded areas, such as for meter reading, smart home devices, and security alarms.

Figure 17 : Lora SX1278 Communication module [83]

As shown in figure 20, the communication module works at a frequency of 433 MHz and can send signals up to 15 km away, with a sensitivity of -148 dBm. It can be set to different bit rates, reaching up to 300 kbps, and has a received signal strength indicator (RSSI) dynamic range of 127 dB.

The module uses LoRa spread spectrum technology and keeps a steady RF output power of +20 dBm (100 mW) while operating between 1.8 and 3.7 V. It can function in temperatures from -40 to +80 °C and supports several modulation types, including FSK, GFSK, MSK, GMSK, LoRa, and OOK. It also features half-duplex SPI communication, automatic RF signal detection, a packet engine with a maximum of 256 bytes CRC, and comes in a small dual-row stamp-hole package with a shielded case and spring antenna.

The Lora SX1278 is a transceiver module that can both transmit and receive. In this case, we used two modules, one was configured as the transmitter and another as a receiver to send and receive GPS location data once the model detect that an accident has happened.

4.5.3. GN-801 GPS Receiver

The GN-801 is a simple-to-use, powerful, and energy-efficient GNSS smart antenna module. It uses the UBLOX8030 GNSS chip design and has built-in SAW and LNA amplifiers [84]. It can switch between GPS + GLONASS and GPS + BEI dual positioning modes. As shown in figure 10, The GN-801 GPS receiver offers several benefits, including high sensitivity, quick startup times, and the ability to connect to multiple GNSS systems, which enhances positioning accuracy. With a protection rating of IPX6, it is durable and suitable for outdoor use. Its compact size and low power consumption make it ideal for a range of applications, such as smart agriculture, vehicle tracking, and robotics, where precise location data is essential.

In this context, the GN-801 GPS receiver was used to read GPD coordinates of the location once the accident is detected. This is intended to be sent to the nearest police station receiver using LoRaWAN technology.

Figure 18 : GN – 801 GPS Receiver [84]

The rest of the components that were used to develop the vehicle crash detection and transmission system are summarized in table 6

S/N	Component	Description	Function
1	0.96-inch OLED Display	The device has a Grove-compatible interface and communicates using the I2C protocol. It features low power consumption and displays in white. It operates effectively within a wide temperature range of -20°C to 70°C, and uses the I2C address 0x3C [85].	It was used to display inferences that were running on the Arduino Nano 33 BLE Sense microcontroller.
2.	HW-131 Breadboard power Regulator	This product supplies independent power for breadboard projects, featuring adjustable output voltage of 3.3V or 5V, an on/off button, LED indicator, USB output, and a DC input for various power supplies. [86].	It was used to supply a steady and regulated power for the prototype.

Table 6 : Other components used at the Lora Transmitter side

4.5.4. Arduino Nano 33 IoT

The Arduino Nano 33 IoT is a low-cost and easy-to-use platform for connecting existing devices to the Internet of Things (IoT) and for building small network applications [87]. It is great for setting up sensor networks linked to home or office routers and for creating Bluetooth® Low Energy devices that send data to smartphones. The board has a low-power Arm® Cortex®-M0 32-bit SAMD21 processor and uses the NINA-W10 module from u-blox for WiFi and Bluetooth® connectivity in the 2.4GHz range. It also has a Microchip® ECC608 crypto chip for secure communication to protect data. Additionally, the Nano 33 IoT comes with a 6-axis IMU, making it suitable for applications like vibration alarm systems, pedometers, and robot positioning. With

these features, the Arduino Nano 33 IoT is a flexible and affordable solution for many basic IoT projects, allowing users to improve their devices or create new applications easily.

Figure 19 : Arduino Nano 33 IoT [87]

The Arduino Nano 33 IoT will be used to receive GPS coordinates and plot them in the Arduino Cloud. It connects to the internet and send the latitude and longitude to map the actual location where the accident has happened.

4.5.5. OLED Display

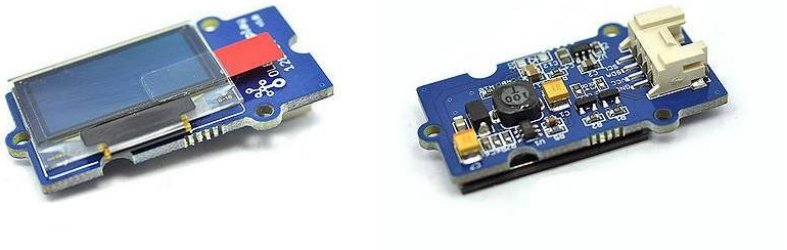


Figure 20 : 0.96-inch Grove OLED Display [88]

The Grove - OLED Display 0.96" is a small black-and-white screen with a 128×64 dot matrix and uses a 4-pin I2C interface. OLED screens are better than LCDs because they are brighter, have higher contrast, use less power, are thinner, and can be viewed from different angles. This display is larger than the 96×96 OLED, so it can show more information. It will be used to display the model's classification and the percentages.

The OLED 128×64 display module with I2C interface offers high brightness, contrast, and low power consumption. It works with Arduino and is easy to use with the SH1106_I2C library. Its applications include smartwatches, MP3 players, and DIY projects. It operates on 3.3-5V and supports temperatures from -40°C to 70°C. In this case, tit will be used at the receiver end to display the status of the system.

Figure 21 : 128X64 OLED Display [89]

CHAPTER 5: RESULTS AND ANALYSIS

In this section, we present the outcomes of our work along with a thorough explanation of our discoveries.

5.1. Model Training Results

The model demonstrates outstanding performance with a remarkable accuracy of 99.3% on the validation set, indicating its effectiveness in correctly classifying data. The low loss value of 0.02 further supports this, showing that the model's predictions closely align with actual labels as shown in figure 25.

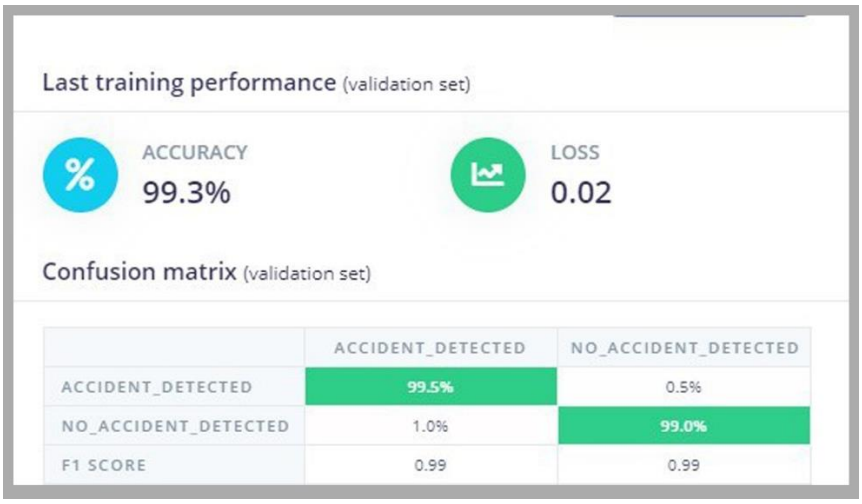


Figure 22 : Model Training Results

According to the confusion matrix, for cases where an accident was detected, the true positive rate is 99.5%, meaning that the model correctly identified 99.5% of actual accidents. However, there is a false negative rate of 0.5%, indicating that 0.5% of accidents were wrongly labeled as non-accidents. On the other hand, for cases where no accident was detected, the true negative rate is 99.0%, which shows that the model correctly identified 99.0% of non-accident situations. However, there is a false positive rate of 1.0%, meaning that 1.0% of non-accidents were incorrectly labeled as accidents.

The F1 score for both classes is 0.99, reflecting an excellent balance between precision and recall in detecting accidents and non-accidents.

In general, these findings indicate that the model is highly effective in differentiating between accident and non-accident events based on acoustic data, showcasing its robustness and reliability.

5.2. Model Testing Results

This section discusses on the results obtained by testing the model to classify new and unseen data. The figure 26 shows the results obtained for further evaluation.

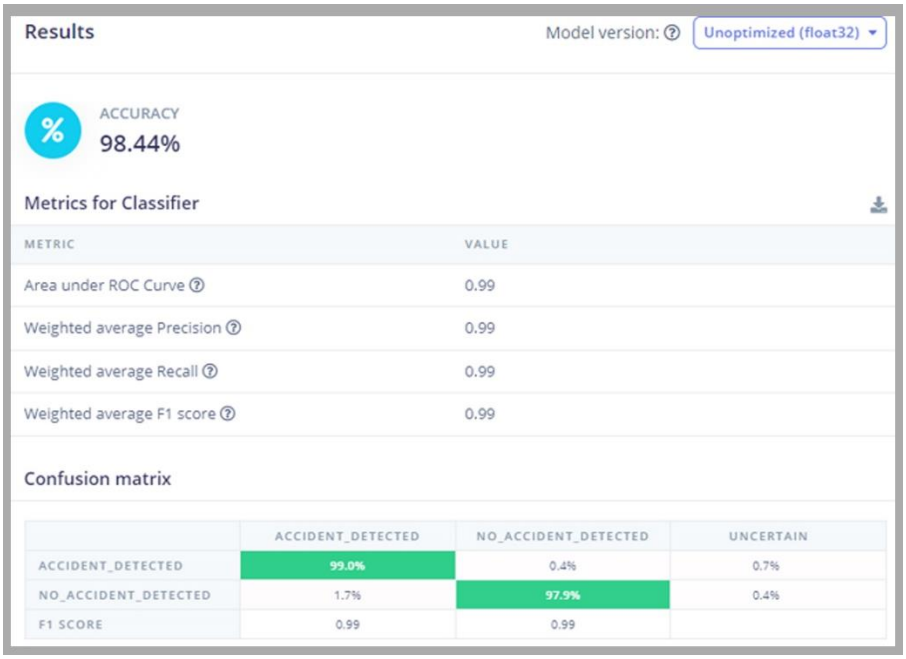


Figure 23 : Model Testing Results

The acoustic accident detection model shows strong performance in both training and testing stages. During training, the quantized (int8) model achieved a high accuracy of 99.3%. When tested on new, unseen data with the unoptimized (float32) version, it maintained a high accuracy of 98.44%, which indicates it generalizes well with very little overfitting. Both versions consistently scored 0.99 on the F1 metric, showing a great balance between precision and recall. The test results show strong performance across different metrics, including a 0.99 score for the area under the ROC curve, weighted average precision, and weighted average recall.

The confusion matrix in figure 25 show slight differences in classification performance between training and testing. Notably, the test model introduced an "Uncertain" category, which made up 0.7% of actual accident cases and 0.4% of non-accident cases. This feature could improve real-world use by flagging borderline cases for human review. Even though there was a small drop in accuracy of 0.86% from training to testing, the model's overall performance remains very high, suggesting it is effective in telling apart accident and non-accident events based on sound data.

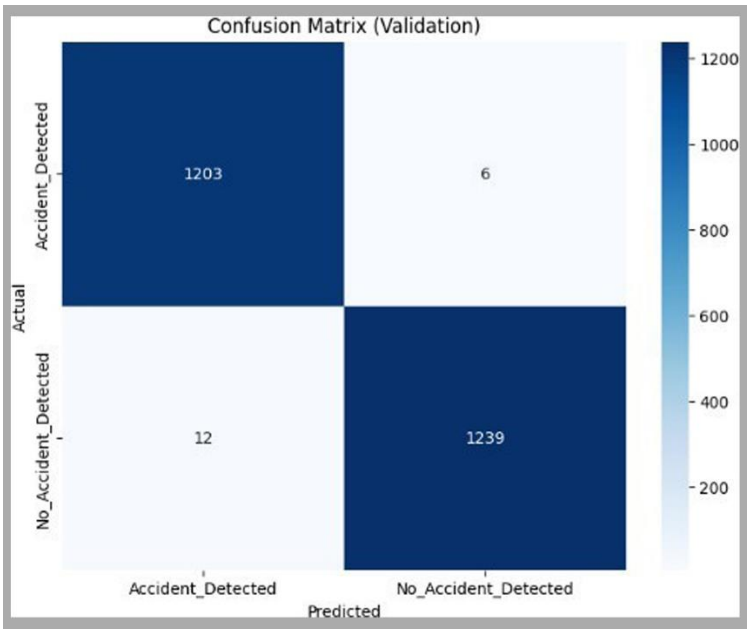


Figure 24 : Confusion Matrix for model testing results

The model achieves a remarkable accuracy rate, successfully differentiating between "Accident Detected" and "No Accident Detected" cases. With a considerable number of true positives (1203) and true negatives (1239), it showcases solid overall performance.

The model demonstrates exceptionally high precision in detecting accidents, with only 6 false positives. This suggests that it infrequently misclassifies non-accident situations as accidents, which is vital for reducing unnecessary alerts.

The model also exhibits a robust recall rate, identifying nearly all actual accident cases while recording only 12 false negatives. This indicates that the model is dependable for recognizing incidents, ensuring that most accidents are reported accurately.

5.3 Quantization Model

Before deploying the model, the figure 28 illustrated the difference in the models using the following metrics;

Processing blocks optimize DSP components to make the model run faster and use less power for tasks like MFCC and FFT.

Learn blocks show how long the model takes to make predictions, which is important for real-time applications.

Latency is the total time it takes for the model to process input and give output, important for quick responses.

RAM usage shows how much memory the model needs while running, which matters for devices with limited memory.

Flash memory usage tells how much storage space the model takes up, important for devices with limited storage capacity.

Accuracy measures how often the model makes correct predictions, which is key to knowing how well it works in practice.

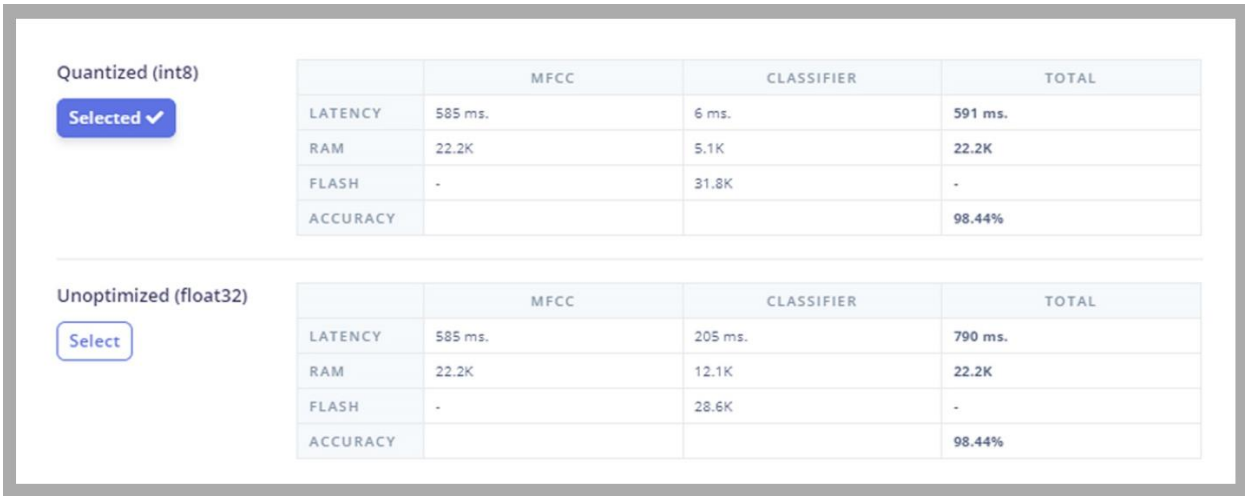


Figure 25 : Quantized vs unoptimized model results

5.4 Overall Device Performance

The Quantized model was selected as shown in figure 29 and the downloaded model in form of Arduino library was customized to configure other components and finally, deployed into Arduino Nano 33 BLE Sense. The model, running on Arduino Nano 33 BLE Sense, was able to classify new and unseen acoustic data as **accident and no accident** depending on the data it is

receiving as shown in figure 29.

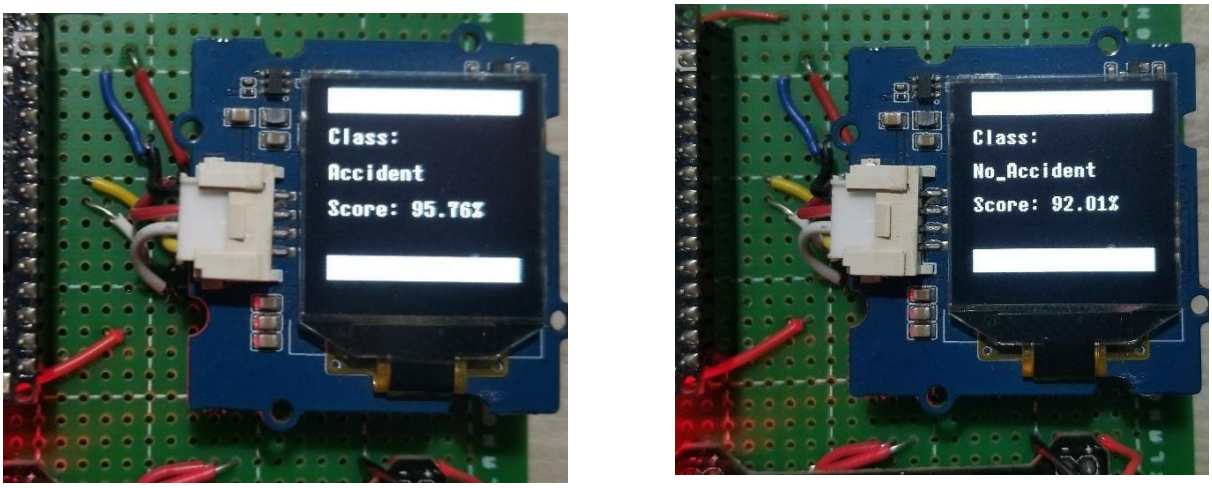


Figure 26 : Audio data classification

When the model detects an accident and when Lora receiver gets the data, it automatically produces an alert or warning by blinking the red LED and turning on the buzzer. On the other hand, when no accident is detected, only the green LED turns ON, indicating normal performance.

5.5 Arduino Could GPS Location Mapping

After an accident is detected, Arduino Nano 33 IoT, which is at the Lora receiving end and connected to the Arduino Cloud, plotted the GPS location which it received from the Lora transmitter.

The figure 30 shows the GPS location which was plotted using latitude and longitude data transmitted through Lora after the model detected an accident.

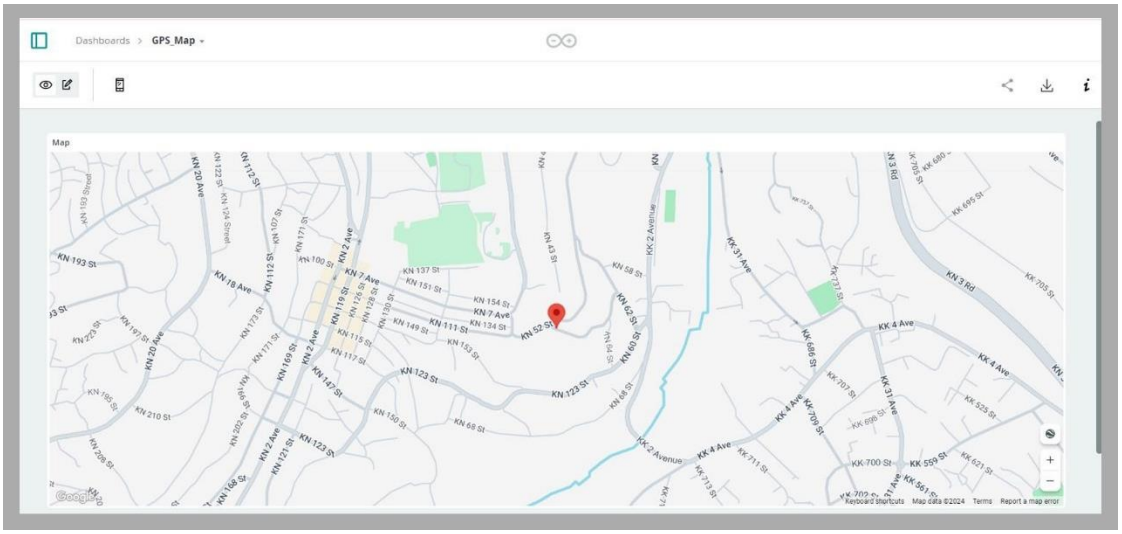


Figure 27 : GPS Location in map view

Figure 31 shows the GPS map in satellite view which is different to figure 30 which is just a GPS map

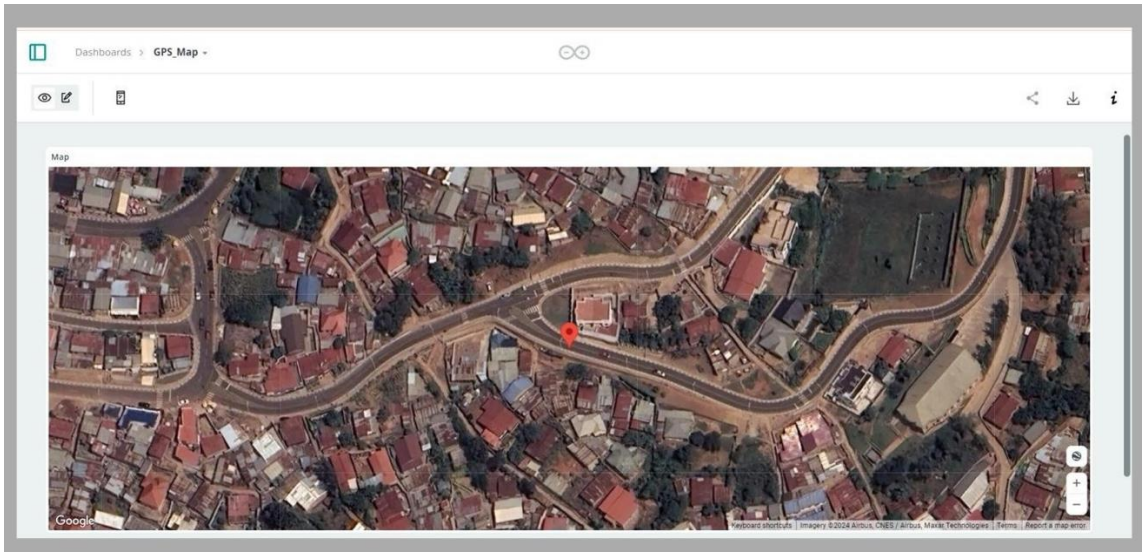


Figure 28 : GPS location in satellite view

view. As shown in both figures, the pinned location is the actual place where the accident was detected during testing.

5.6 Discussion of Results

The MFCC processing was chosen to be used because of its inference time and overall on-device performance metrics. This is in contrast with the MFE and spectrogram whose performance was less than MFCC. The figure 32 summarizes the results obtained.

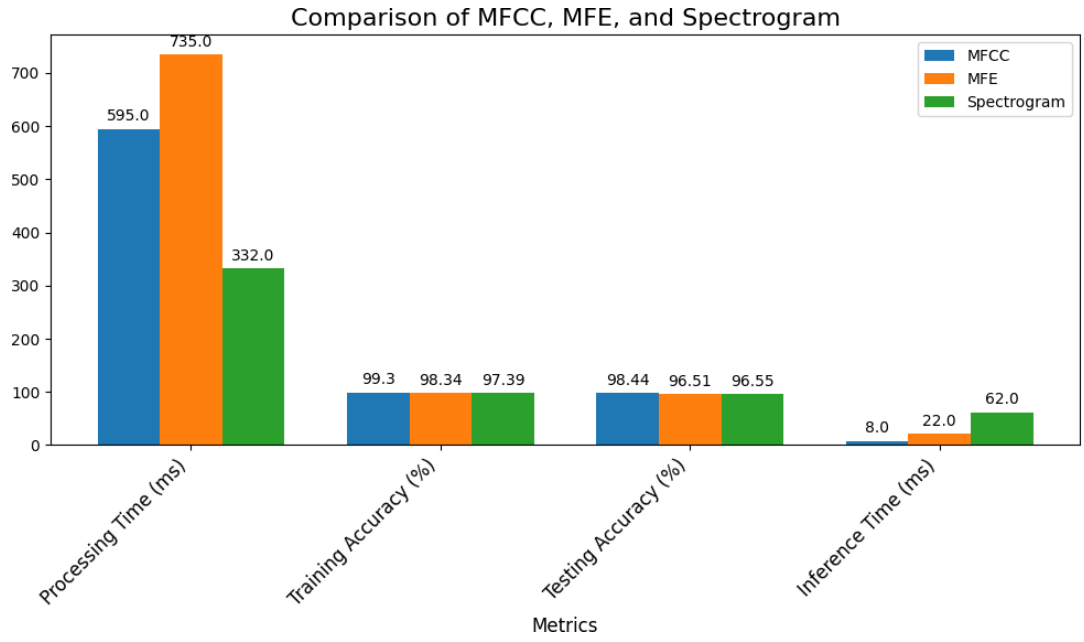


Figure 29 : Comparison of MFCC, MFE, and Spectrogram

The results show that Mel Frequency Cepstral Coefficients (MFCC) perform better than both Mel-Frequency Envelope (MFE) and Spectrogram [51] in training and testing accuracy, inference time, and memory usage for detecting acoustic events. MFCC achieved the highest accuracy (99.3% during training and 98.4% during testing) with the lowest inference time (8 ms). This is because MFCC captures the key features of sound that are most important to human hearing, which helps the model learn and generalize better [52] [53] [54]. On the other hand, MFE is efficient but loses some of the important details and accuracy because it focuses more on the signal's envelope [55]. Spectrograms, while very detailed, require more computation and led to lower accuracy and higher resource usage, as seen in other studies [56]. These results confirm that MFCC is effective for detecting acoustic events, especially in systems with limited computing power.

For the model training results, the accuracy and F1 score results indicate that the acoustic detection approach can be effective when it comes to detecting road accidents. Likewise, during testing, the accuracy of 98.4% of new and unseen data is also an indication of how well the model will perform in real world scenarios since it is expected that it will accurately classify the two distinct scenarios.

According to table 6 in section 3.6.2, the metrics were analyzed in order to deploy a model that will be able to run on a resource constrained device such as Arduino Nano 33 BLE Sense board. By considering the accuracy, model size, power consumption and inference time, a quantized model was selected and it's the one that was deployed.

Finally, the 3D casing was developed for the prototypes as shown in figure 33 and 34 below;

Figure 30 : Accident Detection and Lora Transmission System Prototype

The prototype casing was designed to ensure that there is space for the LCD, LEDs and the two switches for power on/off for both the systems.

As for the accident detection system, the LCD was displaying model classification, as Accident or No accident detected. On the other hand, the Lora reception and emergency alert LCD was showing the system status. When it has not received the signal for accident detection it was printing “System Status: No alerts”. Once the accident detection signal is received, it was printing “Accident Detected, Triggering Alert!”.

Figure 34 : Lora Reception and Emergency Alert System Prototype

5.7 Validation of Hypothesis

Initially we made a hypothesis that our proposed solution will present an effective approaching to accident detection and contribute inn curbing the devasting effects of road accident by effectively minimizing the time it takes to detect an accident and alert relevant authorities. By analyzing the accuracy of accident detection, with an inference time of approximately 600ms, and the accuracy of 98.4% on new and unseen data, the results indicate that our proposed approach presents an effective and real time accident detection.

Likewise, we claimed that our solution will provide a low power and limited resource -based model that will run with low power consumption. According to the results obtained, the model is able to run on a resource constrained device that uses low power.

Lastly, we claimed that the Lora transmission will be effective in the case of transmitting GPS data and over long distance. As such, its cheap and effective when it comes to location data transmission according to the results obtained since we are able to plot the actual location in real time.

CHAPTER 6: CONCLUSION AND FUTURE WORK

In this section, we wrap up the research by outlining the primary goal and the actions carried out to accomplish it. Afterward, we will go into further detail on potential future developments.

6.1 Summary

With road accident being a global issue that affects all countries, there is a great need to come up with solutions that are aimed at curbing the devastating effects such as fatal injuries and even death. This thesis was focused on the development of a TinyML and IoT-based vehicle crash detection using acoustic data. In order to achieve the main objective of this research, an acoustic dataset was collected and processed using MFCC, MFE and Spectral techniques separately in EdgeImpulse platform and MFCC was the best according the results obtained. Then, a machine learning model was trained using MFCC with Arduino Nano 33 BLE Sense as the targeted device. The model had a training accuracy of 99.3% and 98.4% model testing accuracy on new and unseen data. This result indicate that the model is performing exceptionally well in classifying the two different scenarios which are; accident and no accident detected cases. After that the model was quantized and deployed an Arduino library into Arduino Nano 33 BLE Sense microcontroller which supports Tiny machine learning. When the model detects an accident, it was commanding the SX1278 to send the GPS location to the Lora receiver circuit which was connected to Arduino Nano 33 IoT. Finally, the location data was used to plot GPS location in Arduino Cloud platform.

According to the results obtained, this research project has presented an effective approach to accident detection and will contribute in curbing the devastating effects of accident such as fatal injuries and loss of life. Since it's a TinyML-based system, the solution is able to perform well without demanding more resources and without being costly.

6.2 Future Work

Despite having results that confirms that our proposed system presents an effective and efficient approach to road accidents detection and real time emergency alert, there are some improvements that can be made for future work.

Energy Harvesting technique can be employed to make sure that the system is able to capture from various energy sources including RF, wind energy and the rest. This will help to keep the device charging whenever necessary so that its able to work through out with power not being a hindrance.

Secondly, GPS Transmission needs to be improved in the sense that, there should be a way to send the GPS location directly from the Lora Transmitter to the Arduino cloud. This is necessary because it will help to reduce chances of errors. For example, if the Lora receiver devices turns off due to power outage or if there is no internet connection, it means, the system will not be able to send GPS location to cloud and even alert the authorities. As such, coming up with a way to send GPS location data directly from the Lora transmitter to the cloud would be effective.

Development of a GPS mapping platform specifically designed for this system will also essential as opposed to using third party software like Arduino cloud which was used in this case.

Finally, the model developed in this research is specifically for vehicle crash detection only. Therefore, there is a need to develop a model that will be able to detect accidents in general including motor bikes and the rest.

REFERENCES

[1] U. Alvi, M. A. K. Khattak, B. Shabir, A. W. Malik and S. R. Muhammad, "A Comprehensive Study on IoT Based Accident Detection Systems for Smart Vehicles," in IEEE Access, vol. 8, pp. 122480-122497, 2020, doi: 10.1109/ACCESS.2020.3006887.

[2] WHO, “World Health Statistics 2014: A Wealth of Information of Global Public Health,” Geneva, 2014. [Online]. Available: https://apps.who.int/iris/handle/10665/112739. Accessed on: 27/02/2024

[3] F. Bhatti, M. A. Shah, C. Maple, and S. U. Islam, ``A novel Internet of Things-enabled accident detection and reporting system for smart city environments," Sensors, vol. 19, no. 9, p. 2071, May 2019.

[4] Road Accident Injuries: WHO; Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries. Accessed on: 27/02/2023

[5] S. Nanda, H. Joshi, and S. Khairnar, ``An IOT based smart system for acci-dent prevention and detection," in Proc. 4th Int. Conf. Comput. Commun. Control Autom. (ICCUBEA), Aug. 2018, pp.1_6.

[6] T. V. N. Rao and K. R. Yellu, ``Preventing drunken driving accidents using IoT," Int. J. Adv. Res. Comput. Sci., vol. 8, no. 3, pp. 397_400, 2017.

[7] S. Chandran, S. Chandrasekar, and N. E. Elizabeth, ``Konnect: An Internet of Things(IoT) based smart helmet for accident detection and noti_cation," in Proc. IEEE Annu. India Conf. (INDICON), Dec. 2016, pp. 1_4.

[8] M. Syedul Amin, J. Jalil, and M. B. I. Reaz, ``Accident detection and reporting system using

GPS, GPRS and GSM technology," in Proc. Int. Conf. Informat., Electron. Vis. (ICIEV), May 2012, pp. 640_643.

[9] J. B. Edwards, ``The relationship between road accident severity and recorded weather," J. Saf. Res., vol. 29, no. 4, pp. 249_262, Dec. 1998.

[10] E. B. Lerner and R. M. Moscati, “The Golden Hour: Scientific Fact or Medical ‘Urban Legend’?,” Acad. Emerg. Med., 2001, doi: 10.1111/j.1553-2712.2001.tb00201.x.

[11] R. Sánchez-Mangas, A. García-Ferrrer, A. De Juan, and A. M. Arroyo, “The probability of death in road traffic accidents. How important is a quick medical response?,” Accid. Anal. Prev., 2010, doi: 10.1016/j.aap.2009.12.012.

[12] Gokhale, Pradyumna, Omkar Bhat, and Sagar Bhat. "Introduction to IOT." International Advanced Research Journal in Science, Engineering and Technology 5.1 (2018): 41-44.

[13] Atzori, Luigi, Antonio Iera, and Giacomo Morabito. "The internet of things: A survey." *Computer networks* 54.15 (2010): 2787-2805.

[14] Singh, Dhananjay, Gaurav Tripathi, and Antonio J. Jara. "A survey of Internet-of-Things: Future vision, architecture, challenges and services." *2014 IEEE world forum on Internet of Things (WF-IoT)*. IEEE, 2014.

[15] Kazeem, Olaide O., Olubiyi O. Akintade, and Lawrence O. Kehinde. "Comparative study of communication interfaces for sensors and actuators in the cloud of internet of things." Int. J. Internet Things 6.1 (2017): 9-13.

[16] Khanna, Abhishek, and Sanmeet Kaur. "Internet of things (IoT), applications and challenges: a comprehensive review." *Wireless Personal Communications* 114 (2020): 1687-1762.

[17] Al-Fuqaha, Ala, et al. "Internet of things: A survey on enabling technologies, protocols, and applications." *IEEE communications surveys & tutorials* 17.4 (2015): 2347-2376.

[18] Montori, Federico, Riccardo Contigiani, and Luca Bedogni. "Is WiFi suitable for energy efficient IoT deployments? A performance study." *2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI)*. IEEE, 2017.

[19] Haxhibeqiri, J.; De Poorter, E.; Moerman, I.; Hoebeke, J. A Survey of LoRaWAN for IoT: From Technology to Application. *Sensors* **2018**, *18*, 3995. https://doi.org/10.3390/s18113995

[20] Chang, Kuor-Hsin. "Bluetooth: a viable solution for IoT?[Industry Perspectives]." *IEEE Wireless Communications* 21.6 (2014): 6-7.

[21] Raza, Shahid, et al. "Building the Internet of Things with bluetooth smart." *Ad Hoc Networks* 57 (2017): 19-31.

[22] Chang, Hong-Yi. "A connectivity-increasing mechanism of ZigBee-based IoT devices for wireless multimedia sensor networks." *Multimedia Tools and Applications* 78.5 (2019): 5137-5154.

[23] R. Ratasuk, N. Mangalvedhe and A. Ghosh, "Overview of LTE enhancements for cellular IoT," 2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), Hong Kong, China, 2015, pp. 2293-2297, doi: 10.1109/PIMRC.2015.7343680.

[24] L. Fetahu, A. Maraj and A. Havolli, "Internet of Things (IoT) benefits, future perspective, and implementation challenges," 2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO), Opatija, Croatia, 2022, pp. 399-404, doi: 10.23919/MIPRO55190.2022.9803487

[25] Soori, Mohsen, Behrooz Arezoo, and Roza Dastres. "Internet of things for smart factories in industry 4.0, a review." *Internet of Things and Cyber-Physical Systems* 3 (2023): 192-204.

[26] Aboltins, Ugis, Jevgenijs Novickis, and Andrejs Romanovs. "Iot impact on business opportunities." *2020 61st International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS)*. IEEE, 2020.

[27] Mathew, Prabha Susy, Anitha S. Pillai, and Vasile Palade. "Applications of IoT in healthcare." Cognitive Computing for Big Data Systems Over IoT: Frameworks, Tools and Applications (2018): 263-288.

[28] Zanella, Andrea, et al. "Internet of things for smart cities." IEEE Internet of Things journal 1.1 (2014): 22-32.

[29] Chataut, Robin, and Alex Phoummalayvane. "Unleashing the Power of IoT: A Comprehensive Review of IoT Applications, Advancements, and Future Prospects in Healthcare, Agriculture, Smart Homes, Smart Cities, and Industry 4.0." (2023).

[30] F. Ben Abdelaziz, H. Kunze, D. La Torre, and B. Sinclair-Desgagné, “Guest Editorial: Artificial Intelligence and Machine Learning in Business and Management,” J. Model. Manag., 2022.

[31] M. Utmal, “Machine Learning Its Applications, Challenges &Amp; Tools: A Review,” Int. J. Comput. Sci. Mob. Comput., 2021.

[32] Cunningham, Pádraig, Matthieu Cord, and Sarah Jane Delany. "Supervised learning." *Machine learning techniques for multimedia: case studies on organization and retrieval*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. 21-49.

[33] Caruana, Rich, and Alexandru Niculescu-Mizil. "An empirical comparison of supervised learning algorithms." *Proceedings of the 23rd international conference on Machine learning*. 2006.

[34] Barlow, Horace B. "Unsupervised learning." *Neural computation* 1.3 (1989): 295-311.

[35] Celebi, M. Emre, and Kemal Aydin, eds. *Unsupervised learning algorithms*. Vol. 9. Cham: Springer, 2016.

[36] Barto, Andrew G. "Reinforcement learning." *Neural systems for control*. Academic Press, 1997. 7-30.

[37] Arulkumaran, Kai, et al. "Deep reinforcement learning: A brief survey." *IEEE Signal Processing Magazine* 34.6 (2017): 26-38.

[38] Rumelhart, David E., Bernard Widrow, and Michael A. Lehr. "The basic ideas in neural networks." *Communications of the ACM* 37.3 (1994): 87-93.

[39] Uzair, Muhammad, and Noreen Jamil. "Effects of hidden layers on the efficiency of neural networks." *2020 IEEE 23rd international multitopic conference (INMIC)*. IEEE, 2020.

[40] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *nature* 521.7553 (2015): 436-444.

[41] Pouyanfar, Samira, et al. "A survey on deep learning: Algorithms, techniques, and applications." *ACM computing surveys (CSUR)* 51.5 (2018): 1-36.

[42] M. Shafique, T. Theocharides, V. J. Reddy and B. Murmann, "TinyML: Current Progress, Research Challenges, and Future Roadmap," 2021 58th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 2021, pp. 1303-1306, doi: 10.1109/DAC18074.2021.9586232.

[43] Dutta, Lachit, and Swapna Bharali. "Tinyml meets iot: A comprehensive survey." *Internet of Things* 16 (2021): 100461.

[44] Schizas, N.; Karras, A.; Karras, C.; Sioutas, S. TinyML for Ultra-Low Power AI and Large Scale IoT Deployments: A Systematic Review. *Future Internet* **2022**, *14*, 363. https://doi.org/10.3390/fi14120363

[45] Coral Products; Available : https://coral.ai/products/ . Accessed on : 24/09/2024

[46] Giachino, Joseph M. "Smart sensors." *Sensors and actuators* 10.3-4 (1986): 239-248.

[47] Y. Choi and M. Rhu, "PREMA: A Predictive Multi-Task Scheduling Algorithm For Preemptible Neural Processing Units," 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA), San Diego, CA, USA, 2020, pp. 220-233, doi: 10.1109/HPCA47549.2020.00027

[48] Arduino Nano 33 BLE; Available: https://docs.arduino.cc/hardware/nano-33-ble/ . Accessed on : 24/09/2024

[49] TinyML Implementation using Raspberry Pi Pico: Geometry Gesture Detection ;Available: https://medium.com/@subirmaity/tinyml-implementation-using-raspberry-pi-pico-geometry-gesture-detection-part-i-3f0717677561 . Accessed on : 24/09/2024

[50] Alajlan, Norah N., and Dina M. Ibrahim. "TinyML: Enabling of inference deep learning models on ultra-low-power IoT edge devices for AI applications." *Micromachines* 13.6 (2022): 851.

[51] Ghamari, Sedigh, et al. "Quantization-guided training for compact tinyml models." *arXiv preprint arXiv:2103.06231* (2021).

[52] De Leon, Josen Daniel, and Rowel Atienza. "Depth pruning with auxiliary networks for tinyml." ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2022

[53] M. Shafique, T. Theocharides, V. J. Reddy and B. Murmann, "TinyML: Current Progress, Research Challenges, and Future Roadmap," 2021 58th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 2021, pp. 1303-1306, doi: 10.1109/DAC18074.2021.9586232

[54] S. O. Ooko, M. Muyonga Ogore, J. Nsenga and M. Zennaro, "TinyML in Africa: Opportunities and Challenges," 2021 IEEE Globecom Workshops (GC Wkshps), Madrid, Spain, 2021, pp. 1-6, doi: 10.1109/GCWkshps52748.2021.9682107.

[55] I. N. K. Wardana, S. A. Fahmy and J. W. Gardner, "TinyML Models for a Low-Cost Air Quality Monitoring Device," in IEEE Sensors Letters, vol. 7, no. 11, pp. 1-4, Nov. 2023, Art no. 6007804, doi: 10.1109/LSENS.2023.3315249.

[56] C. Nicolas, B. Naila and R. -C. Amar, "TinyML Smart Sensor for Energy Saving in Internet of Things Precision Agriculture platform," 2022 Thirteenth International Conference on Ubiquitous and Future Networks (ICUFN), Barcelona, Spain, 2022, pp. 256-259, doi: 10.1109/LSENS.2023.3315249.

[57] I. N. K. Wardana, S. A. Fahmy and J. W. Gardner, "TinyML Models for a Low-Cost Air Quality Monitoring Device," in IEEE Sensors Letters, vol. 7, no. 11, pp. 1-4, Nov. 2023, Art no. 6007804, doi: 10.1109/LSENS.2023.3315249.

[58] [29] Baballe, Muhammad. (2022). Accident Detection and Alerting Systems: A Review. 2. 24-29. 10.5281/zenodo.7063008.

[59] ECSTUFF4U for Electronics Engineers; Available : https://www.ecstuff4u.com/2018/04/advantages-and-disadvantages-of-gsm.html. Accessed on: 3/02/2024

[60] GSM(Global System for Mobile Communication) by Solomon Ndungu and Erica Mixon . Available on: https://www.techtarget.com/searchmobilecomputing/definition/GSM. Accessed on: 3/02/2024.

[61] Foggia, Pasquale, et al. "Audio surveillance of roads: A system for detecting anomalous sounds." IEEE transactions on intelligent transportation systems 17.1 (2015): 279-288.

[62] Wagner, Marco. "Why TinyML? Exploring the reasons why TinyML is used for real-world problems." *CS & IT Conference Proceedings*. Vol. 14. No. 9. CS & IT Conference Proceedings, 2024.

[63] [25] G. T. Selvi, S. R. K. V, S. A and J. R, "Advanced Accident Detection System Using Sensor Networks and IoT," 2022 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS), Chennai, India, 2022, pp. 1-5, doi: 10.1109/ICPECTS56089.2022.10047538.

[64] C. DEVI and S. GOWRI, "An automatic Smart Phone with IoT based Accident detection and alerting System," 2021 5th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2021, pp. 426-432, doi: 10.1109/ICECA52323.2021.9676093.

[65] [27] G. R, J. V. Ranagatti, L. S. M, L. Sangamad and N. J, "An Intelligent IoT Accident Detection System For Emergency Medical Services," 2022 Fourth International Conference on Cognitive Computing and Information Processing (CCIP), Bengaluru, India, 2022, pp. 1-4, doi: 10.1109/CCIP57447.2022.10058635.

[66] P. S. Priyadharshini, A. L. H. B, A. S and A. S, "Automated Vehicle Accident Alert System using IoT," 2023 7th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Kirtipur, Nepal, 2023, pp. 1040-1046, doi: 10.1109/I-SMAC58438.2023.10290672.

[67] Nanda, Sayanee, Harshada Joshi, and Smita Khairnar. "An IOT based smart system for accident prevention and detection." *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*. IEEE, 2018.

[68] L Vijayaraja et al 2021 IOP Conf. Ser.: Mater. Sci. Eng. 1055 012061

[69] Khaliq, Kishwer Abdul, Amir Qayyum, and Jürgen Pannek. "Prototype of automatic accident detection and management in vehicular environment using VANET and IoT." *2017 11th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*. IEEE, 2017.

[70] Kathiravan, M., et al. "IoT-based Vehicle Surveillance and Crash Detection System." *2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*. IEEE, 2022.

[71] G. Rajesh, A. R. Benny, A. Harikrishnan, J. Jacob Abraham and N. P. John, "A Deep Learning based Accident Detection System," 2020 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 2020, pp. 1322-1325, doi: 10.1109/ICCSP48568.2020.9182224.

[72] S. Ghosh, S. J. Sunny and R. Roney, "Accident Detection Using Convolutional Neural Networks," 2019 International Conference on Data Science and Communication (IconDSC), Bangalore, India, 2019, pp. 1-6, doi: 10.1109/IconDSC.2019.8816881.

[73] D. A. Machhi, A. S. Pillai, S. S. More and P. Thirumalaikumar, "Accident Detection and Reporting using Convolution Neural Network," 2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC), Salem, India, 2022, pp. 207-212, doi: 10.1109/ICAAIC53929.2022.9793063.

[74] D. A. Machhi, A. S. Pillai, S. S. More and P. Thirumalaikumar, "Accident Detection and Reporting using Convolution Neural Network," 2022 International Conference on Applied Artificial Intelligence and Computing

[75] Pathik, N.; Gupta, R.K.; Sahu, Y.; Sharma, A.; Masud, M.; Baz, M. AI Enabled Accident Detection and Alert System Using IoT and Deep Learning for Smart Cities. *Sustainability* **2022,** *14, 7701*. https://doi.org/10.3390/su14137701

[76] R.Desai, A. Jadhav,S.Sawant ,N.Thakur, " Accident Detection Using ML and AI Techniques," engpaper journal

[77] A. Bhakat, N. Chahar and V. Vijayasherly, "Vehicle Accident Detection & Alert System using IoT and Artificial Intelligence," 2021 Asian Conference on Innovation in Technology (ASIANCON), PUNE, India, 2021, pp. 1-7, doi: 10.1109/ASIANCON51346.2021.9544940.

[78] Mivia website. Available: https://mivia.unisa.it/ Accessed on 23/06/2024

[79] EdgeImpulse Platform. Available: https://edgeimpulse.com/ . Accessed on : 23/09/2024

[80] Arduino Cloud platform. Available: https://cloud.arduino.cc/. Accessed on: 24/07/2024

[81] Arduino Nano 33 BLE. Available: https://docs.arduino.cc/hardware/nano-33-iot/. Accessed on:25/08/2024

[82] Arduino Nano 33 BLE Sense. Available: https://market.samm.com/arduino-nano-33-ble-with-headers-en . Accessed on: 23/08/2024

[83] SX1278 Lora Module. Available: https://www.elecrow.com/sx1278-lora-module. Accessed on: 21/07/2024

[84] GN-801 GPS receiver. Available: https://gpswebshop.com/products/gn-801-gnss-receiver. Accessed on: 23/08/2024

[85] 0.96 inch oled display. Available: https://wiki.seeedstudio.com/Grove-OLED_Display_0.96inch/ . Accessed on : 17/09/2024

[86] HW-131 Power Module for Arduino Breadboards PCB - 3.3V / 5V. Available: https://gleanntronics.ie/en/products/hw-131-power-module-for-arduino-breadboards-pcb-3-3v-5v-2141.html . Accessed on:05/08/2024

[87] Arduino Nano 33 IoT. Available : https://store.arduino.cc/products/arduino-nano-33-iot . Accessed on: 9/08/2024.

[88] Ittichaichareon, Chadawan, Siwat Suksri, and Thaweesak Yingthawornsuk. "Speech recognition using MFCC." *International conference on computer graphics, simulation and modeling*. Vol. 9. 2012.

[89] Cirkit Designer, *How to Use 128x64 OLED Display (I2C IIC SPI Serial): Examples, Pinouts, and Specs.* Available: https://docs.cirkitdesigner.com/component/ebabf69f-6411-44a1-b085-f993a135b2d7/128x64-oled-display-i2c-iic-spi-serial . Accessed on: 09/08/2024.

[90] Islam, Zubayer, and Mohamed Abdel-Aty. "Deep Convolutional Neural Network for Roadway Incident Surveillance Using Audio Data." arXiv preprint arXiv:2203.06059 (2022).

[91] Tiwari, Vibha. "MFCC and its applications in speaker recognition." International journal on emerging technologies 1.1 (2010): 19-22.

[92] Majeed, Sayf A., et al. " Mel frequency cepstral coefficients (MFCC) feature extraction enhancement in the application of speech recognition: A comparison study." Journal of Theoretical & Applied Information Technology 79.1 (2015).

[93] Muda, Lindasalwa, Mumtaj Begam, and Irraivan Elamvazuthi. "Voice recognition algorithms using mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques." arXiv preprint arXiv:1003.4083 (2010).

[94] Abdul, Zrar Kh, and Abdulbasit K. Al-Talabani. "Mel frequency cepstral coefficient and its applications: A review." IEEE Access 10 (2022): 122136-122158.

[95] M. Mulimani and S. G. Koolagudi, "Acoustic Event Classification Using Spectrogram Features," TENCON 2018 - 2018 IEEE Region 10 Conference, Jeju, Korea (South), 2018, pp. 1460-1464, doi: 10.1109/TENCON.2018.8650444.