MAKERERE UNIVERSITY

COLLEGE OF ENGINEERING, DESIGN, ART AND TECHNOLOGY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Bachelor of Science in Computer Engineering

LUGANDA SPEECH INTENT CLASSIFICATION FOR IOT APPLICATIONS

Submitted by:
JOHN TREVOR KASULE 19/U/14669/PS

Main supervisor: Dr. Andrew Katumba
Co-supervisor: Dr. Ronald Kizito

A Final Year Full Project Report Submitted to the Department of Electrical and Computer Engineering, Makerere University in partial fulfilment of the requirements for the award of Degree of Bachelor of Science in Computer Engineering.

July 2023

Declaration

I, KASULE JOHN TREVOR, do hereby declare that to the best of my knowledge, the information presented in this Project Report is original and has not been published and/or submitted for any other degree award to this or any other university before.

[illegible]
Signature:

07.07.2023
Date:

Approval

This Project Report has been submitted for examination with the approval of the following supervisors.

Main Supervisor
Dr.Andrew Katumba
Lecturer
Department of Electrical and Computer Engineering
Makerere University

Signature:

Date:

Co Supervisor
Dr. Ronald Kizito
Lecturer
Department of Electrical and Computer Engineering
Makerere University

Signature: [illegible]

Date:

Dedication

I dedicate this report to my dear mother, Ms Namwanga Edwige, whose unwavering support, love, and encouragement have been my pillar of strength throughout my academic pursuit. Her selflessness and sacrifices have paved the way for my success, and I am deeply grateful for her presence in my life.
I also dedicate this report to the Computer Engineering class of 2019 for all the great moments and hard times we have successfully passed through together.
Furthermore, I dedicate the report to all my friends, whose unwavering encouragement and support have been a source of motivation and inspiration. Their belief in my abilities has propelled me forward and made this journey all the more meaningful.

Acknowledgement

I would like to express my deepest gratitude to the following individuals who have played a significant role in the completion of this project:
I am sincerely thankful to my project partner, Elvis Mugume, for his dedication, collaboration, and shared experiences. Our teamwork and combined efforts have greatly contributed to the success of this project. I am grateful for the knowledge we have gained together and for his unwavering support throughout the journey.
I extend my deepest appreciation to my supervisor, Dr Andrew Katumba, for his guidance, expertise, and continuous support. His valuable insights, constructive feedback, and encouragement have significantly contributed to the successful completion of this project. I would also like to express my gratitude to my co-supervisor, Dr Ronald Kizito, for his guidance and support throughout this project. His expertise, feedback, and valuable suggestions have been instrumental in shaping the direction and outcomes of this research.
I would also like to express my gratitude to Sudi Murindanyi and Kwezi Louis for their guidance throughout the project.
I would like to acknowledge and express my gratitude to my friends for their unwavering encouragement and support. Their presence has brought joy, laughter, and motivation during challenging times.
Lastly, I would like to express my deep appreciation to all those who have believed in me, encouraged me, and supported me in various ways. Your belief in my abilities and your unwavering support have been pivotal in my journey.

Abstract

The rise of Internet of Things (IoT) technology has sparked interest in voice-controlled smart homes. This research project focuses on developing a Luganda speech intent classification system for IoT applications, aiming to integrate local languages into smart home environments. The project utilizes a combination of hardware components, including a Raspberry Pi, Wio Terminal, and ESP32 nodes as microcontrollers. The Raspberry Pi acts as the central hub for processing and interpreting Luganda voice commands, while the Wio Terminal serves as a display device, providing a user-friendly interface for interacting with the smart home system. The ESP32 nodes are utilized for controlling and coordinating the IoT devices within the smart home setup. The IoT devices within the smart home system are implemented using the MQTT (Message Queuing Telemetry Transport) protocol. The main objective of the project is to enable voice control using Luganda, a widely spoken local language in Uganda. To achieve this, a natural language processing (NLP) model is developed and deployed on the Raspberry Pi. The model incorporates Mel Frequency Cepstral Coefficients (MFCCs) as acoustic features and utilizes a Convolutional Neural Network (Conv2D) architecture for speech intent classification. A dataset of Luganda voice commands was collected. By incorporating Luganda as the spoken language for voice commands, this project addresses the localization challenges and linguistic diversity in IoT applications. The proposed Luganda speech intent classification system allows users to interact seamlessly and naturally with their smart home devices, eliminating the requirement for English language proficiency. The research findings will contribute to the advancement of IoT applications by promoting inclusive and accessible voice control interfaces for smart home environments, particularly in regions where local languages are predominant.

List of Acronyms

AI Artificial Intelligence

ANN Artificial Neural Network

API Application Programming Interface

ASR Automatic Speech Recognition

CNN Convolutional Neural Network

DOA Direction of Arrival

DT Decision Tree

HDF5 Hierarchical Data Format

IoT Internet of Things

I2S Inter-IC Sound

KWS Keyword Spotting

ML Machine Learning

MFCCs Mel-Frequency Cepstral Coefficients

NLP Natural Language Processing

NLU Natural Language Understanding

NN Neural Network

PCM Pulse-Code Modulation

RNN Recurrent Neural Network

SLU	Spoken Language Understanding
SOTA	State Of The Art
SVM	Support Vector Machine
TDM	Time-Division Multiplexing
VAD	Voice Activity Detection
WiFi	Wireless Fidelity
.tflite	TensorFlow Lite
.wav	Waveform Audio File Format

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The rapid advancements in technology have led to the proliferation of smart devices and the Internet of Things (IoT) in various domains. Smart homes, in particular, have gained significant popularity, offering convenience, efficiency, and enhanced control over home appliances and systems[1]. Voice-controlled smart home systems have emerged as a convenient and intuitive way to interact with these devices, allowing users to control various functions using voice commands[2].These systems respond to voice commands, whether they are for playing music, turning on the lights, or answering questions. In order to be effective, these systems must be able to extract the intent of the user from the voice command. This is known as intent recognition. The intent is something the user wants to do. Intent classification is a sub-task of Spoken Language Understanding and it aims to extract the aim of the user in a given voice command. However, the majority of existing voice-controlled smart home systems are primarily designed to support major languages such as English, leaving users of low-resource languages at a disadvantage. In countries like Uganda, where English is not widely spoken, the existing systems fail to cater to the needs and preferences of the local population[3].
This project aims to address this language barrier by developing a speech intent classification system for Luganda, a prominent local language in Uganda. By leveraging the power of artificial intelligence and machine learning techniques, the project seeks to enable Ugandan users to interact with smart home devices using their native language, enhancing user experience and accessibility.
The project recognizes the importance of data collection, preprocessing, and model development to achieve accurate and reliable speech intent classification. By collecting a substantial amount of data from diverse participants and employing state-of-the-art techniques such as convolutional neural networks (CNNs), the project aims to develop a robust and efficient model capable of accurately classifying spoken commands related to various smart home devices and functions.The project explores post training model quantisation techniques to choose an appropriate model size for deployment on edge devices.
Furthermore, the project extends its scope to include the integration of IoT devices,

allowing the classified speech intents to be translated into actionable commands that control the corresponding smart devices. This integration involves the use of the MQTT protocol, which enables seamless communication between the central device (Raspberry Pi) and other IoT devices such as Wio terminals and ESP boards.
Overall, this project combines advancements in speech recognition, machine learning, and IoT technologies to bridge the language gap in voice-controlled smart home systems. By empowering users to interact with their smart devices in their native language, the project aims to improve accessibility, usability, and overall user satisfaction in the context of smart home automation.

1.2 Problem Statement

There is increasing demand for voice-interfaced systems to control devices in low-resource languages. Most of the existing work is in foreign languages and limited work in native languages (low-resource). **There is a big gap between research on low-resource languages like Luganda and high-resource languages in voice control and other Speech Related areas.** There is a need to target research and build systems that can receive and interpret voice commands in these languages like Luganda so as to reduce the gap.

1.3 Justification

Less than 30% of Ugandans can speak English, the official language of the country, as reported by the Uganda Bureau of Statistics (UBOS)[4]. This indicates a significant language barrier that limits the accessibility and usability of existing systems designed for English and other foreign languages.
The low proficiency in English among Ugandans suggests that the majority of the population faces challenges in using the current voice-controlled systems, which are primarily designed for English speakers. Consequently, there is a pressing need to develop alternative solutions that cater to the linguistic capabilities of the local population.
Studies have revealed a notable under-representation of low-resource African languages in Speech Intent Recognition systems. Uganda has over 40 indigenous languages and all these are classified as low-resource. This highlights the lack of focus on addressing the linguistic diversity and specific needs of African communities, including those in Uganda.
However, it is evident that a significant portion of Ugandans can effectively express themselves in their native languages. For instance, Luganda, spoken by 11 million people in Central Uganda and understood by approximately 20 million people (44% of the population), serves as a prominent means of communication[5].
By focusing on Luganda, this research project takes a proactive step towards breaking down language barriers and promoting inclusivity in smart home systems for Ugandan users. It aims to provide a voice-controlled solution that aligns with the linguistic capabilities of the majority of the population, enabling a more natural and intuitive user experience.

Additionally, this project contributes to the field of low-resource language research in the speech domain. By developing a Luganda speech intent classification system, it offers valuable insights into the challenges and opportunities associated with developing Natural Language Processing (NLP) models for low-resource languages. Furthermore, the project generates a comprehensive voice command dataset that can facilitate future advancements in this area.

1.4 Main Objective

The main objective of this project is to design, develop, and evaluate a Luganda speech intent recognition system for IoT applications.

1.4.1 Specific Objectives

The specific objectives of the study were:

- To collect and prepare a dataset of Luganda speech commands for training a machine learning model.
- To train and optimize an NLP model for recognizing the intent behind different Luganda speech commands.
- To integrate the trained model into a Raspberry Pi-based system for controlling IoT devices through voice commands in Luganda.
- To evaluate the system

1.5 Scope of the study

This project limited itself to the following scope:

- Implementation and development of a Luganda speech intent classification system.
- Utilization of Convolutional Neural Networks (CNNs) as the primary architecture for speech intent classification.
- Classification of 20 specific intents (classes) within the Luganda language.
- Integration of the Internet of Things (IoT) technology, specifically MQTT protocol, for communication and coordination between devices.
- Focus on edge devices for processing and interpreting voice commands within the smart home environment.
- Utilisation of the direct speech-to-intent approach in performing speech intent classification.

1.6 Organisation of The Report

The report is organized into the following chapters:
Chapter 1: Introduction

This chapter provides an introduction to the project, presenting the background, objectives, problem statement, scope and significance of the study. It establishes the context and rationale for the project, outlines the research questions, and provides an overview of the report's structure.

Chapter 2: Literature Review

In this chapter, a comprehensive review of the existing literature and research related to the project topic is presented. It explores relevant studies, frameworks, and theories, highlighting their strengths, weaknesses, and gaps. The chapter serves as a foundation for understanding the current state of the field and identifying the research gap addressed by the project.

Chapter 3: Methodology

This chapter details the methodology employed in the project. It explains the research design, data collection methods, tools, and techniques used for data analysis. The chapter provides a step-by-step explanation of the procedures followed to achieve the project objectives. It discusses the selection of the dataset, feature extraction techniques, model development approaches, and evaluation metrics employed.

Chapter 4: Results and Discussion

The fourth chapter presents the results obtained from the experiments conducted in the project. It provides a detailed analysis and interpretation of the findings, comparing them with the existing literature and research objectives. The chapter engages in critical discussions, drawing connections between the results and the research questions. It highlights the implications of the results, and discusses their strengths, limitations, and areas for further improvement.

Chapter 5: Limitations, Recommendations, and Conclusion

Chapter 5 focuses on the limitations of the project, discussing the constraints and challenges encountered during the research process. It also offers recommendations for future improvements or further research in the field. The chapter concludes with a summary of the key findings, a restatement of the research objectives, and a concise conclusion that addresses the research questions and reflects on the overall significance and contributions of the project.

Chapter 2

Literature Review

Introduction

This chapter provides a comprehensive overview of existing research and scholarly work in the field of Luganda speech intent classification for IoT applications. This chapter aims to critically examine and analyze relevant studies, theories, and methodologies that have contributed to the understanding of speech intent classification and its applications. By reviewing the existing literature, we gain valuable insights into the advancements, challenges, and gaps in the field.

2.1 Machine Learning

Machine Learning is a subset of Artificial intelligence (AI) that gives computers the ability to learn without being programmed[6]. Machine learning uses technologies where the rules or algorithms are created from the input data, that is; the data on which the system is trained. In machine learning, the computer is allowed to come up with its solution based on the data it is given instead of telling it what to do.
Machine learning involves giving machines to learn from raw data so that the computer program can change when exposed to new data (learning from experience).[7], [8] There are many different types of machine learning which include; unsupervised learning, supervised learning and reinforcement learning. In supervised learning, a system is trained using labelled data and the correct output is provided for each input. In unsupervised learning, the system is trained using unlabelled data and it is tasked to learn to identify patterns and relationships in the data on its own. In reinforcement learning, the system is trained by learning from its actions and receiving feedback in the form of rewards or penalties[9].

2.2 Deep Learning

Deep learning is a type of machine learning that involves using neural networks with multiple layers to learn complex patterns and relationships in data. It involves a hierarchy of nonlinear transformation of input that can be used to generate a statistical model as output.[9–11]

2.2.1 Neural Networks

Artificial Neural Networks are normally called Neural Networks (NN). A neural network is a type of machine-learning model that is inspired by the structure and function of the brain. It is composed of a large number of interconnected artificial neurons, which are computational units that are designed to mimic the behaviour of biological neurons.
In a neural network, the neurons are organized into layers, with each layer receiving input from the previous layer and passing its output to the next layer. This allows the network to learn and represent increasingly complex and abstract patterns in the data. A large amount of labelled data is provided for training a neural network and the network uses this data to adjust the strengths of the connections between the neurons. A trained neural network can be used to make predictions on new data. This involves providing the input data to the network and running the computational graph to generate the output.[9, 12, 13] From the image above, it can be seen that a neural

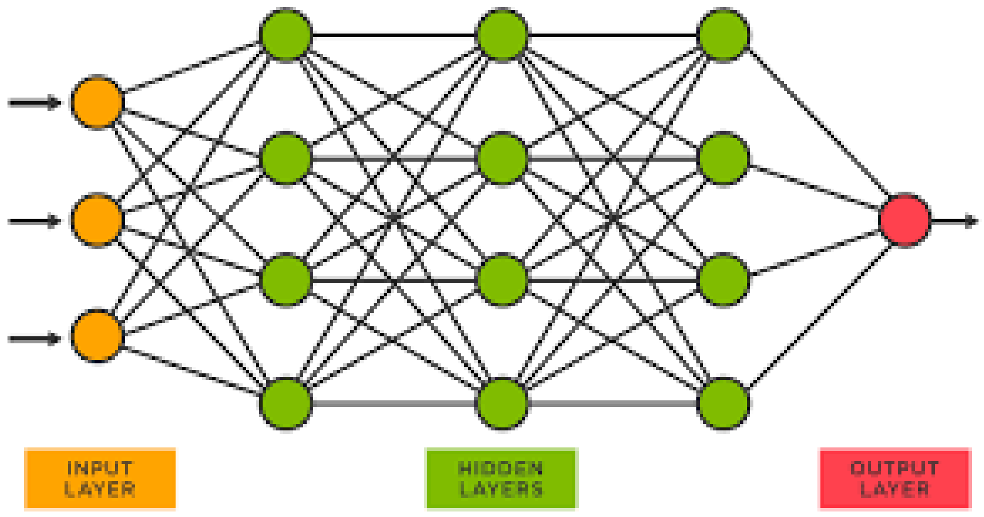


Figure 2.1: Neural Network architecture

network is made of interconnected neurons. Each of them is characterised by its weight, bias, and activation function. Here are other elements of this network[9].

- Input Layer: The input layer takes raw input from the domain. No computation is performed at this layer. Nodes here just pass on the information (features) to the hidden layer.
- Hidden Layer: As the name suggests, the nodes of this layer are not exposed. They provide an abstraction to the neural network. The hidden layer performs all kinds of computation on the features entered through the input layer and transfers the result to the output layer.

- Output Layer It's the final layer of the network that brings the information learned through the hidden layer and delivers the final value as a result.

All hidden layers usually use the same activation function. However, the output layer will typically use a different activation function from the hidden layers. The choice depends on the goal or type of prediction made by the model.

2.2.2 Deep Learning Frameworks

Deep learning frameworks provide the necessary infrastructure and libraries for building, training, and deploying deep neural networks. In this subsection, we will discuss three of the most prominent deep learning frameworks: TensorFlow, PyTorch, and Keras.

TensorFlow

TensorFlow, developed by Google Brain, is widely regarded as one of the most popular deep learning frameworks. It offers a comprehensive ecosystem for machine learning, featuring a high-level API called Keras, as well as a lower-level computational graph API for increased flexibility. TensorFlow supports distributed computing and allows seamless deployment on various platforms, including CPUs, GPUs, and TPUs. It is known for its scalability, performance, and extensive community support.

PyTorch

PyTorch, developed by Facebook's AI Research lab, has gained significant popularity for its simplicity and dynamic computation graph. It provides a user-friendly and intuitive interface, making it easy to learn and use. PyTorch's dynamic graph allows for efficient debugging and flexible network architectures. The framework excels in GPU acceleration and includes tools for model deployment and production. It has a strong community and is widely adopted by researchers and practitioners.

Keras

Keras is a high-level neural networks API that runs on top of TensorFlow, Theano, or Microsoft Cognitive Toolkit (CNTK). It has gained popularity for its simplicity and ease of use, particularly for beginners in deep learning. With Keras, users can rapidly prototype and build deep learning models using a user-friendly and intuitive interface. It supports both convolutional and recurrent networks and provides seamless integration with TensorFlow for efficient training and deployment.
These three deep learning frameworks—TensorFlow, PyTorch, and Keras—offer a range of features and capabilities, making them popular choices for researchers and practitioners in the field of deep learning.

2.2.3 Natural Language Processing

Natural Language Processing is a term used in the field of computer science to describe the study of human natural languages. Natural Language Processing (NLP)

is a subfield of semantics, software engineering, and artificial intelligence dealing with the coordination between computers and human language, specifically how to program computers to be able to process and investigate a lot of natural language information. The objective is to program a computer to understand the written texts or spoken utterances, including the context of the language inside them.
To understand the structure of the language, knowledge of various units of language is required. The major units of language are phonemes, morphemes, lexemes, syntax and context. When syntax, Semantics, and Pragmatics are conceptualised along with these units, they help to obtain an understanding of the language and communicate among the people. The smallest unit of sound is the phoneme. It brings changes in the meaning of a language but it does not hold any meaning when used alone. A morpheme is the smallest unit of words. A lexeme is the arrangement of all the inflected forms of a single word. The syntax is the arrangement of rules by which an individual builds full sentences while the context conveys a particular meaning of a sentence based on the interaction of all the units of language[14].

Most of the challenges in NLP are listed as follows: human language understanding, enabling computers to derive meaning from the voice commands given to them through human (natural) language, and others involve natural language interaction between computers and humans.[15]

2.2.4 Spoken language Understanding

Spoken Language Understanding (SLU) is a subfield of Natural Language Processing (NLP) that focuses on developing algorithms and models to enable machines to comprehend and interpret human language in spoken form.
It refers to the ability of a machine to extract semantic information from speech signals in a form that is amenable to further processing including dialogue, voice commands, information retrieval, etc.[16]
SLU involves the analysis of the acoustic signal of speech to identify phonemes and words and then to extract meaning from the identified words. This includes tasks such as automatic speech recognition, intent detection, slot filling, and dialogue management[17].
Automatic Speech Recognition (ASR) is a critical component of SLU, as it converts spoken words into text that can be analyzed by NLP algorithms[14] [15]. Intent detection identifies the user's intention behind the spoken words, while slot filling extracts relevant information from the user's utterances. Dialogue management involves managing the flow of conversation, generating appropriate responses, and handling errors and uncertainties in the user's input.[17]
SLU is used in a variety of applications, including virtual assistants, chatbots, and voice-enabled devices. By enabling machines to understand and respond to spoken language, SLU improves the user experience, making it more intuitive and natural.

Slot Filling

Slot filling is a task in Natural Language Processing (NLP) that involves extracting specific pieces of information, known as slots, from natural language input(speech or text)[18]. It is usually performed in conjunction with intent detection, which identifies the user's intention in the input. Once the intent is detected, slot filling can be used to extract the relevant information needed to fulfil the intent[17].

Speech Intent Classification

Speech intent classification is a branch of Spoken Language Understanding that enables systems to extract or classify the intent of the user from the speech in a particular domain. The intent is something the user wants to do. Speech intent classification can be done either by converting speech to text followed by doing a text classification or directly using a classification algorithm for extracted features of speech.
The classical approach to Speech Intent Classification and Spoken Language Understanding has been a pipeline of Automatic Speech Recognition (ASR) to transcribe speech signals into a textual representation that feeds a Natural Language Understanding (NLU) module[19]. This method is widely used in many generic ASR systems, whereas the second method is more applicable to specific domains. For a specific domain, the speech vocabulary is limited and well-defined[20]. Speech command classification is a part of speech intent classification that covers research on the use of voice commands.
Speech Intent classification works along with slot filling which helps to extract specific parts of information from a speech command or utterance.
For example, if a user provides an utterance/command saying ”turn on the lights in the kitchen,” the system can extract the following slots: ”activate” (action), ”lights” (the object), and ”kitchen” (the location). These slots can then be used to complete the intent of activating the lights in the kitchen.
Speech intent classification has led to progress in the development of the use of voice commands to control devices. The recognition of the intent from a voice command is what is known as voice command recognition. Voice command recognition has been implemented in home automation using the help of IoT[15].

2.2.5 Model Quantization

Model quantization is a vital technique for optimizing deep learning models to be deployed on resource-constrained devices. It involves reducing the precision of model parameters and activations to improve computational efficiency and reduce memory footprint without significant performance degradation. There are two main approaches to model quantization: post-training quantization and quantization-aware training.

Post-training Quantization

Post-training quantization is a widely used technique where a pre-trained model is quantized after the training process. It allows for the utilization of existing models without retraining. This technique involves quantizing weights, activations, or both, by reducing the number of bits used to represent these values. This reduction in precision significantly reduces the memory requirements of the model. However, post-training quantization may introduce some quantization errors that can affect the model's accuracy. Despite this, it remains a convenient and effective method for optimizing models for deployment on resource-constrained devices.

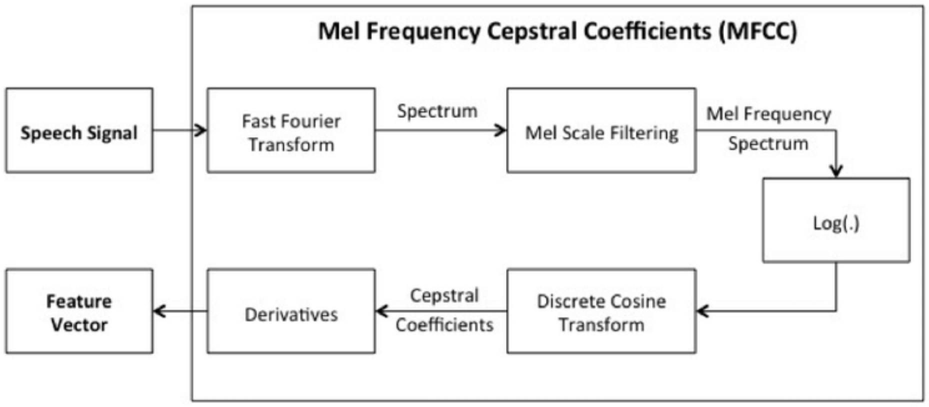
Quantization-Aware Training

Quantization-aware training combines training and quantization to achieve efficient and accurate models. Unlike post-training quantization, quantization-aware training incorporates the effects of quantization during the training process. By considering quantization early on, the model learns to adapt and maintain performance even with reduced precision. This approach often involves incorporating quantization constraints or specific loss functions that encourage the model to be robust to quantization effects. Quantization-aware training can result in models with improved accuracy compared to post-training quantization, making it a valuable technique for optimizing models for deployment on resource-constrained devices.
The selection of an appropriate quantization technique depends on the specific requirements and constraints of the deployment scenario. Both post-training quantization and quantization-aware training can be employed to optimize models and ensure their efficient performance in resource-limited environments.

2.3 Mel Frequency Cepstral Coefficients

Mel Frequency Cepstral Coefficients (MFCCs) are widely used in speech and audio signal processing as a feature representation. They capture the spectral characteristics of an audio signal and have been proven to be effective in various applications such as speech recognition, speaker identification, and audio classification.
MFCCs are derived through a series of steps that mimic the human auditory system. The process involves dividing the audio spectrum into Mel-scale frequency bands, which are spaced according to the perceptual characteristics of human hearing. The power spectrum of the signal is then calculated using techniques like the Fast Fourier Transform (FFT). Next, the Mel Filterbank is applied to emphasize the important spectral components and compress the spectrum. Finally, the Discrete Cosine Transform (DCT) is used to obtain the cepstral coefficients that represent the MFCCs.

In this project, MFCCs played a vital role in speech recognition for our IoT system. These features served as the input to our deep learning models, enabling them to learn the discriminative patterns in speech during training and perform accurate command recognition at deployment.
The choice of the number of MFCC coefficients is an important consideration. A higher number of coefficients can capture more detailed spectral information but may also increase computational complexity. In our experiments, we explored different configurations, including 10 and 13 MFCC coefficients, striking a balance between information richness and computational efficiency.



Source: https://www.researchgate.net/profile/Alexandra-Trujillo-J/publication/322578191/figure/fig1

Figure 2.2: MFCC feature extraction process

2.4 IoT

The Internet of Things (IoT) refers to the connection of devices such as appliances, vehicles, and other items embedded with electronics, software, sensors, and network connectivity which enable these objects to collect and exchange data. This technology allows for the automation of various processes and the ability to monitor and control devices remotely. The IoT has the potential to greatly improve efficiency and convenience in many areas of our lives, such as healthcare, transportation, and energy management. The IoT market is growing at a rapidly growing technology market and has many different concepts. IoT technology is most often associated with ”smart home” related products, including gadgets and appliances. The devices include; lighting fixtures, thermostats, home security systems, cameras, and other home appliances that support one or more common ecosystems, and can be controlled via devices associated with that ecosystem, such as smartphones and speakers[15]
The relation within the IoT network is usually people-to-people, people-to-things, and things-to-things. This concept has greatly changed the way people interact with these devices. This is because people are able to control a number of interconnected devices at the same time. The connection can be set up using modules such as Bluetooth, Wi-Fi, and Zigbee among others.[8] The interconnection set up using these modules is what is known as the Internet.[17] The traditional approach to controlling devices has always been by the use of remotes but recently, the use of voice has sparked up and improved the use of IoT.[15], [21], [22]
The use of voice commands to control devices is efficient as it enables users to control devices without raising a hand or leaving a position. With IoT, it is achievable to manage a number of devices using voice with the help of speech intent recognition which helps to determine the intent of the voice command utterance. When the intent is obtained, a particular action occurs on a device(s). The use of such commands can vary from “turn off the lights in the kitchen” to “close all doors”.

2.4.1 WIO Terminal

The WIO Terminal shown in Figure 2.3is a device built with a powerful ATSAMD51 microcontroller and wireless connectivity via Realtek RTL8720DN that runs at 120MHz (Boost up to 200MHz), 4MB External Flash and 192KB RAM. It supports both Bluetooth and Wi-Fi providing the backbone for IoT projects. The Wio Terminal itself is equipped with a 2.4" LCD Screen, onboard IMU(LIS3DHTR), Microphone, Buzzer, microSD card slot, Light sensor, and Infrared Emitter(IR 940nm). It also has two multifunctional Grove ports for Grove Ecosystem and 40 Raspberry Pi compatible pin GPIO for more add-ons. In this project, it was used to establish an IoT connection along with 2 ESP32 nodes and a Raspberry Pi where it acted as the display device.[23].

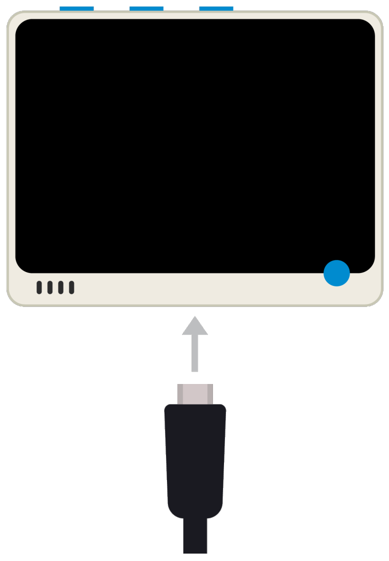


Figure 2.3: The Wio Terminal

2.4.2 ESP32

The ESP32 is a highly popular and versatile microcontroller unit (MCU) that has gained significant popularity in the field of Internet of Things (IoT) development. With its powerful features and comprehensive set of functionalities, the ESP32 MCU serves as an excellent choice for a wide range of IoT applications.
For this project, the ESP32 MCU was chosen as the preferred microcontroller platform for several compelling reasons:
Wireless Connectivity: The built-in Wi-Fi and Bluetooth capabilities of the ESP32 MCU perfectly align with the project's requirement for seamless communication and integration with other IoT devices and networks.
Processing Power: The dual-core architecture of the ESP32 MCU ensures sufficient processing power to handle the computational demands of the project. It enables the

efficient execution of complex algorithms and allows for the simultaneous handling of multiple tasks.
Peripheral Interfaces: The availability of versatile peripheral interfaces in the ESP32 MCU grants flexibility for interfacing with various sensors, actuators, and external devices. This facilitates easy integration and control of the project's hardware components.
Development Ecosystem: The well-established and extensive development ecosystem surrounding the ESP32, including support for popular programming platforms such as Arduino and MicroPython, ensures access to a wealth of resources, libraries, and community support. This simplifies development and makes project implementation easy.
Cost-Effectiveness: The ESP32 MCU provides an optimal balance between performance and affordability, making it a cost-effective choice for IoT projects with budget considerations.
Considering these factors, the ESP32 MCU emerged as the ideal choice for this project, offering a robust and reliable platform that meets the project's requirements for wireless connectivity, processing power, versatility, and community support[24]. Additionally, the ESP32's dual-core processor, with a clock speed of up to 240MHz, enables it to perform complex tasks efficiently[24].

2.4.3 Raspberry Pi

The Raspberry Pi is a credit card-sized single-board computer that offers a cost-effective and versatile solution for various computing applications. It has gained popularity due to its compact size, low power consumption, and GPIO (General Purpose Input/Output) pins that allow for easy integration with external devices and sensors. In this project, the Raspberry Pi Model 3b+ was utilized as the main computing platform.
The Raspberry Pi Model 3b+ is an enhanced version of its predecessor, providing improved processing power, enhanced connectivity options, and expanded memory capacity. It features a 1.4GHz quad-core ARM Cortex-A53 processor, 1GB RAM, and onboard wireless connectivity (Wi-Fi and Bluetooth). The presence of multiple USB ports, Ethernet, HDMI, and a 40-pin GPIO header makes it suitable for a wide range of applications.
In this project, the Raspberry Pi Model 3b+ served as the central control unit for the IoT integration. It facilitated the communication between the NLP model, MQTT broker, and ESP32 boards. The Raspberry Pi, equipped with the necessary software libraries and frameworks, executed the NLP model and managed the incoming and outgoing MQTT messages. Additionally, its GPIO pins were utilized to connect external devices, such as the button for initiating recording and the APA LEDs for visual feedback.
The Raspberry Pi Model 3b+ provided a reliable and capable platform for executing the project tasks, demonstrating its effectiveness in enabling IoT integration and serving as a bridge between the NLP model and the IoT devices. Its versatility, computational capabilities, and GPIO interfacing capabilities made it a suitable choice for the project's

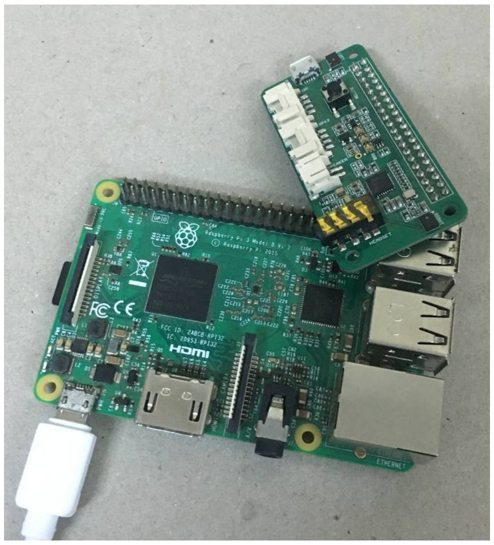


Figure 2.4: Raspberry pi Model 3b+ and reSpeaker 2-Mics Pi HAT

requirements. However, the Raspberry Pi Model 3b+ does not come with an inbuilt microphone and a reSpeaker 2-Mics Pi Hat was used for audio capture. This device is discussed in the next part.

reSpeaker 2-Mics Pi HAT: Voice User Interface for Raspberry Pi Series

The reSpeaker 2-Mics Pi HAT[1] is a highly capable voice user interface designed specifically for the Raspberry Pi Series. It was designed by Seeed Studio, the manufacturer of the Wio terminal. It features two analogue microphones and the WM8960 Audio Codec, enabling high-definition voice capture and processing. With the integration of NLU software algorithms, it offers advanced functionalities such as Voice Activity Detection (VAD), Direction of Arrival (DOA), and Key Word Spotting (KWS), making it suitable for a wide range of Voice Interaction applications. This device serves as an excellent solution for local Natural Language Understanding.
The reSpeaker is equipped with analogue microphones positioned at each corner, allowing it to capture audio data from the surrounding environment. These microphones transmit the data to the integrated codec, which includes synchronized ADCs with microphone boost amplifiers. The codec then delivers valid channel data to the Raspberry Pi or Pi Zero through I2S ports in standard I2S or PCM format, along with a single port in TDM format.
Once the audio data arrives at the Raspberry Pi, NLU algorithms are applied. These algorithms encompass VAD, DOA, and KWS, enabling various Voice Interaction applications such as voice-based remote control cars, voice user interfaces, and keyword wake-up systems.
To enhance audio quality and mitigate noise interference, the reSpeaker incorporates functions like Auto Echo Cancellation (AEC), Beamforming, Webrtc Noise Suppression (NS), and Automatic Gain Control (AGC). These features ensure improved audio data

[1]https://www.seeedstudio.com/ReSpeaker-2-Mics-Pi-HAT.html

transmission to the Raspberry Pi.
The reSpeaker 2-Mics Pi HAT also includes several additional features that enhance its functionality. It provides three programmable APA102 RGB LEDs, which offer a versatile way to display information and create visually appealing interfaces. These LEDs can be controlled programmatically to indicate different states or provide visual feedback in Voice Interaction applications. Furthermore, the reSpeaker 2-Mics Pi HAT incorporates a user button, which is connected to GPIO pin 17 on the Raspberry Pi. This button can be utilized to trigger specific actions or functions within the application.
In our project, we used the button to initiate the recording process. Additionally, the APA LEDs were employed to create an engaging interface display that visually indicated when the recording had started. These features, including the programmable LEDs and the user button, provide convenient options for interaction and customization, enhancing the overall user experience and functionality of the reSpeaker 2-Mics Pi HAT in Voice Interaction applications.

2.4.4 Message Queuing Telemetry Transport

The MQTT [2] protocol was chosen as the primary communication protocol for this project. MQTT, or Message Queuing Telemetry Transport, is a lightweight messaging protocol that is well-suited for IoT applications due to its low power and bandwidth requirements. It allows for reliable communication between devices, even in low-bandwidth or unreliable network conditions.
MQTT follows a publish-subscribe messaging pattern, where devices communicate by publishing messages to topics or subscribing to topics to receive messages. The devices involved in the connection are called MQTT clients and use the pub/sub model. This asynchronous communication model enables efficient and scalable data exchange between devices in a distributed system.
MQTT clients encompass both publishers and subscribers, which determine whether the client is responsible for sending or receiving messages. It is possible for an MQTT client to fulfil both roles simultaneously. In MQTT, the act of sending data from a device (or client) to a server (or broker) is referred to as publishing, while the reverse operation, receiving data, is known as subscribing.
MQTT clients are usually very small, and require minimal resources so can be used on small micro-controllers. MQTT message headers are small to optimize network bandwidth. MQTT allows for messaging between device-to-cloud and cloud-to-device. This makes for easy broadcasting messages to groups of things[25].
Neither the publishers nor the subscribers contact each other directly and therefore third parties,the brokers take care of the connections between the clients[26]. In this project, we use a popular broker called mosquitto [3] which was installed on the Raspberry Pi. Mosquitto is lightweight and is suitable for use on all devices from low power single board computers to full servers[27].

[2]https://mqtt.org/
[3]https://mosquitto.org/

2.5 Low-resource languages

Low-resource languages are those which have limited amounts of data available for building Artificial Intelligence (AI) systems such as Conversational AI and machine translation. These languages are primarily found in Africa and Asia, where data storage is limited. As a result, it becomes challenging to develop natural language processing (NLP) solutions like conversational AI systems. Oral languages with few written resources are particularly challenging to work with[28].
In order to develop language-based solutions, access to a significant amount of raw text data from various sources such as books, emails, social media content, and scientific papers is crucial. Task-specific resources like parallel corpora for machine translations, various kinds of annotated text to be used in part-of-speech tags, and dictionaries for developing named entity recognition systems are also necessary. The lack of working solutions and available data makes it difficult to fine-tune models for downstream tasks, limiting the range of possible tasks that can be solved with low-resource NLP tools[3]. Auxiliary data such as labelled data in different languages, as well as lexical, syntactic, and semantic resources like dictionaries, dependency tree corpora, and semantic databases like WordNet, are also essential for developing AI solutions for low-resource languages. It is important to note that the absence of adequate resources and data for low-resource languages is a significant obstacle in the development of NLP tools. As a result, there is a need for concerted efforts toward creating solutions for low-resource languages to promote linguistic diversity and inclusivity in the field of AI.

2.5.1 Luganda

Luganda is the official spoken language of Buganda. Buganda is the central region of Uganda where the largest number of native speakers are found in the country[29]. After English which is the official language of Uganda, Luganda is the most spoken language in the country (even more so than the second official language, Kiswahili)[30]. Luganda is spoken by about 11 million people in the country and understood by over 20 million people beyond[5]. Luganda and all the native languages in Uganda are classified as low-resource languages in the conversational AI field as there are not enough resources available to work with the languages in AI tasks, especially Natural Language processing[31] [3]. In this project, our focus was put on building a Luganda Speech Intent Classification System.

2.6 Related work

This section discusses work in relation to speech intent classification for IoT applications.

2.6.1 Voice commands datasets

Fluent Speech Commands dataset

The Fluent Speech Commands dataset is a collection of 30,043 spoken commands that are designed to train and test speech recognition systems for smart home scenarios. The dataset was collected through crowd-sourcing, which involved recruiting multiple individuals to record the utterances. This approach has several benefits, including the ability to include a diverse range of voices and speech patterns in the dataset, which makes it more representative of real-world usage scenarios[32].
The dataset includes 97 speakers, and each utterance is recorded as a 16 kHz single-channel .wav file. The dataset is labelled with three slots - action, object, and location - and each slot can take on multiple values. The combination of slot values represents the intent of the spoken command. For example, a spoken command might have the intent of ”turn on the lights in the kitchen,” where ”turn on” is the action, ”lights” is the object, and ”kitchen” is the location.
There are multiple possible wordings for each intent, which makes the dataset more challenging and realistic for speech recognition systems to recognize. In total, the dataset has 248 phrasing mappings to 31 unique intents. The demographic information about the speakers, including age range, gender, and speaking ability, is included in the dataset.
The dataset is randomly divided into train, valid, and test splits, with no speaker appearing in more than one split. Each split includes all possible wordings for each intent, although there is an option to include data for only certain wordings for different sets to test the model's ability to recognize wordings not seen during training.
The Fluent Speech Commands dataset is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International license. This license allows the dataset to be shared, copied, and redistributed as long as it is not used for commercial purposes, and any adaptations or modifications of the dataset are not allowed. This license ensures that the dataset is freely available to the research community for non-commercial research purposes while also protecting the rights of the dataset creators.
The Fluent Speech Commands dataset is therefore a valuable resource for researchers and developers working on speech recognition systems for smart home scenarios. It was collected through crowd-sourcing which makes it a more representative and diverse dataset that can improve the performance of speech recognition systems in real-world scenarios[32, 33].

Google Speech Commands Dataset

The Google Speech Commands dataset[34] is a valuable resource for researchers and practitioners interested in developing and testing speech recognition models. It was created by the TensorFlow and AIY teams to showcase speech recognition examples using the TensorFlow API and contains 65,000 clips of one-second-long duration. Each clip contains one of 30 different words spoken by thousands of different subjects. The clips were recorded in realistic environments with phones and laptops. The number of clips for each command in the dataset is shown in figure 2.5 below. One notable feature

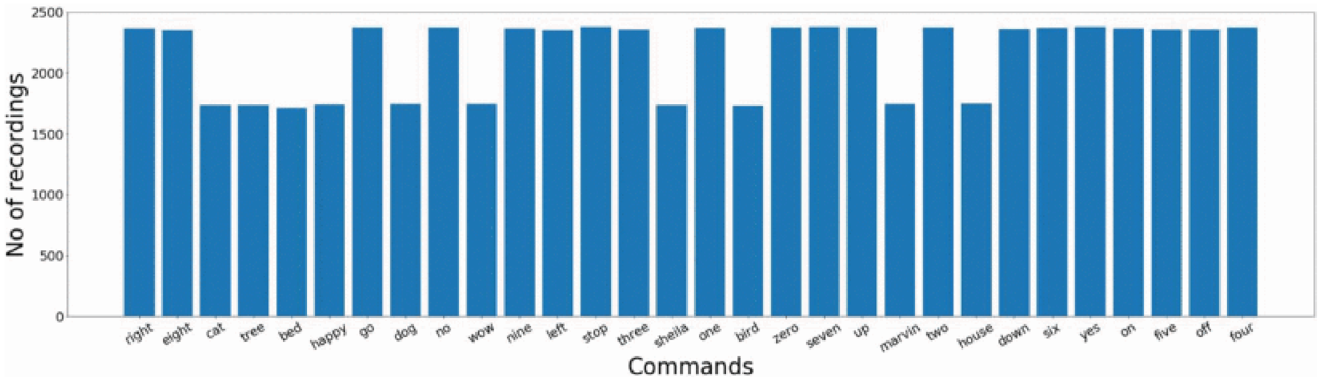


Figure 2.5: Commands in Google Speech Commands Dataset

of the Google Speech Commands dataset is that it includes noise words, which add complexity to the task of recognizing specific words in noisy or cluttered environments. This is particularly relevant for speech recognition in industry settings, where background noise or interference may be common. By including noise words, the dataset provides a more realistic and challenging task for speech recognition models to solve.
The Google Speech Commands dataset is available and can be accessed through the TensorFlow API and other platforms which makes it a valuable resource for further research and experimentation in speech recognition. Its use can help researchers and practitioners develop and test new speech recognition models, and can also facilitate the comparison and evaluation of different approaches[34].

2.6.2 State-of-the-art: Speech Intent Recognition

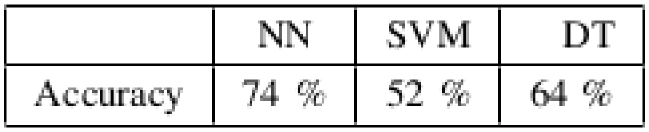
There are **two** primary approaches to speech intent Recognition: speech-to-intent directly and speech-to-text-to-intent.

A. Direct Speech-to-intent

The speech-to-intent directly approach aims to map spoken language directly to its intent without first transcribing the speech into text. These features can include MFCCs, spectrograms, or other time-frequency representations that capture relevant information about the audio signal. Machine learning models or deep learning models are then trained on these features to classify the user's intent. This approach has the advantage of not relying on intermediate text transcription, which can introduce errors and reduce the accuracy of the final prediction[20, 35]. This approach is discussed in the following papers.

Domain Specific Intent Classification of Sinhala Speech data

In [36], Buddhika et al. proposes a domain-specific intent classification system for the Sinhala[4] language using a feed-forward neural network with backpropagation. In this approach, Buddhika et al. extracted Mel Frequency Cepstral Coefficients (MFCC) from a Sinhala speech corpus of 10 hours to train the neural network. The performance of the system was evaluated using the recognition accuracy of the speech queries. A comparison of the intent classification accuracy achieved by three different approaches: Neural Network (NN), Support Vector Machine (SVM), and Decision Tree (DT) was conducted. The performance of these approaches was evaluated and compared in figure 2.6 below, which shows the accuracy achieved by each approach. From the comparison, the Neural



	NN	SVM	DT
Accuracy	74 %	52 %	64 %

Figure 2.6: Comparison of NN, SVM and DT

Network approach outperformed both SVM and DT approaches. This suggests that the proposed approach using a feed-forward neural network with backpropagation is a promising method for intent classification. Buddhika et al. also investigated the use of raw audio recordings and MFCC features. The summary of these results is shown in the table below. From the results, it was observed that the accuracy of the model

	Raw audio	**MFCC features**
Accuracy	66%	73%

Table 2.1: Model accuracy when trained on raw audio vs mfccs

did not exceed 66% when the neural network was trained with raw audio. Then after preprocessing the corpus and removing over-recorded samples, recordings that did not contain the full inflection, and speech samples with heavy noise, the model accuracy improved to 73%. This demonstrates the importance of having a preprocessed corpus for the proposed system. The work was done on only 6 intents which represents a small scope.

Speech Recognition of Command words based on Convolutional Neural Network

In [33], Yang et al. presents a method of speech recognition using convolutional neural networks (CNNs) for recognizing command words in industry settings. The authors compare the proposed CNN-based method with traditional deep neural network (DNN)

[4]Sinhala is a low-resource language primarily spoken in Sri Lanka, a country located in South Asia.

and recurrent neural network (RNN) training methods and report that the CNN-based method significantly improves the accuracy of recognizing command words. The features extracted from the speech are MFCCs which are fed to the different model architectures.The results are shown in the table 2.2 below; The experimental results clearly

Approach	Accuracy
DNN	82.46%
RNN	89.45%
CNN	92.88%

Table 2.2: Comparison of accuracies for model approaches

show that the CNN-based method outperforms DNN and RNN models on the Google Speech Commands dataset, which suggests that CNNs could be a promising solution for speech recognition tasks in industry settings. This finding is significant for our project which has a similar problem concerning voice commands. The proposed approach is well-supported by the literature, which suggests that CNNs are adequate for speech recognition tasks. The authors also provide some insights into the architecture and hyperparameters used in the CNN model, which can be helpful for researchers and practitioners who want to apply this method to other speech recognition tasks. However, it would be interesting to see how this proposed CNN-based method performs on other datasets.

B. Speech-to-text-to-intent

The speech-to-text-to-intent approach, on the other hand, involves first transcribing the spoken language into text using automatic speech recognition (ASR) and then extracting intent from the text using natural language understanding (NLU) techniques. There are also two primary methods within this approach: the pipeline approach and the end-to-end approach[18]. The approaches are shown in Figure 2.7.

The pipeline approach represented with *Conventional SLU* involves separate modules for ASR and NLU. The ASR module converts speech into text, and then the NLU module extracts intent and slots from the transcribed text. The major limitation of this approach is the potential propagation of errors from the ASR module throughout the pipeline, which can adversely impact the performance of the SLU system. The end-to-end approach combines the Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU) modules into a single model. The model is trained directly on speech and is trained to learn patterns rather than relying on word-by-word recognition. This approach improves performance by avoiding errors introduced by separating ASR and NLU modules. However, this requires large amounts of domain-specific data for training, which can be difficult to acquire for new domains.

The following papers discuss the pipeline and end-to-end approach:

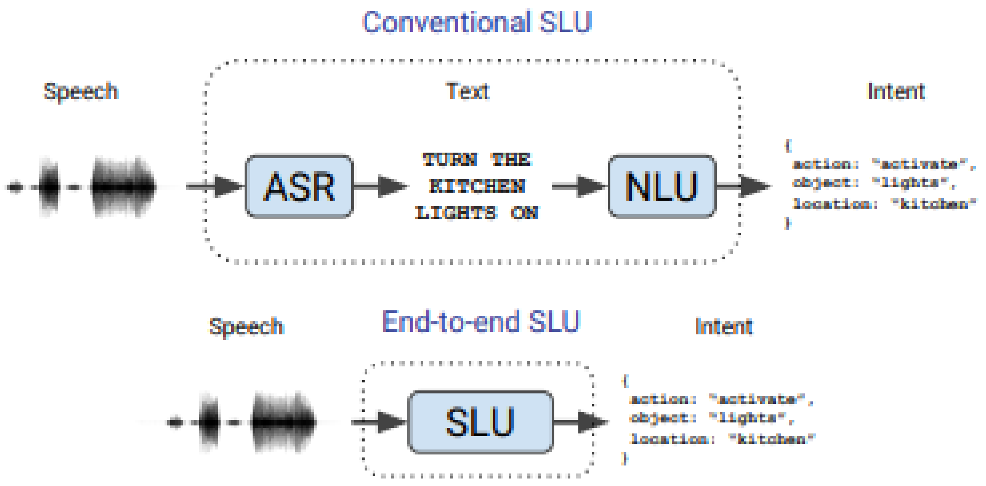
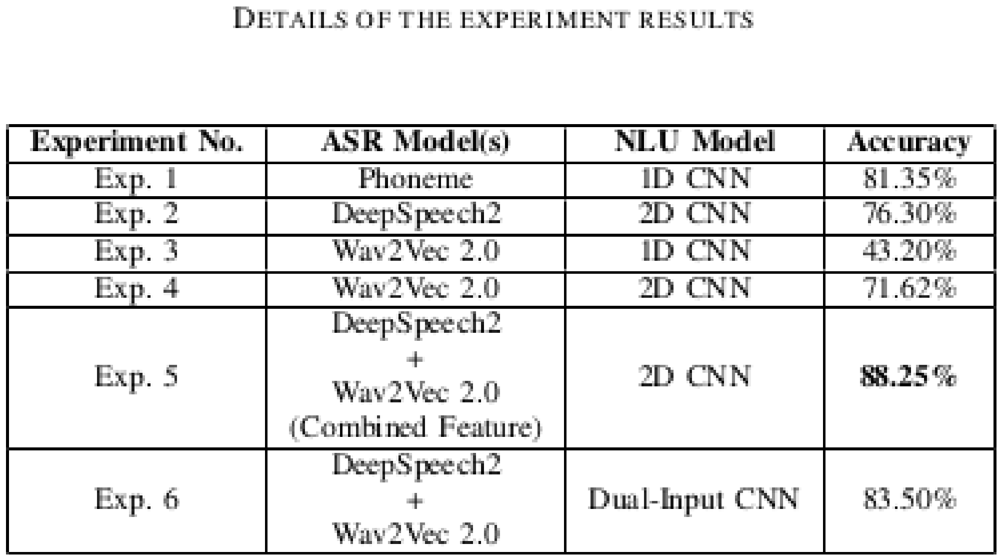

Conventional SLU
Speech
Text
Intent
ASR
TURN THE KITCHEN LIGHTS ON
NLU
{ action: "activate", object: "lights", location: "kitchen" }
Speech
End-to-end SLU
Intent
SLU
{ action: "activate", object: "lights", location: "kitchen" }

Figure 2.7: Conventional ASR-NLU system vs End-to-End SLU

Low resource Multi-ASR Speech Command Recognition

In [37], a pipeline approach was used. In this paper, Mohamed and Thayasivam proposes a multi-ASR set-up to improve the accuracy of low-resource speech-command transfer-learning tasks. The authors combine the outputs of pre-trained ASR models, DeepSpeech2 and Wav2Vec 2.0, using a CNN-based setup. The experiment shows an 8% increase in accuracy over the current State Of The Art (SOTA) low-resource speech-command phoneme-based speech intent classification methodology.
The paper addresses an important issue in Spoken Language Understanding (SLU) studies, particularly for low-resource languages where training an ASR model requires a significant amount of audio input. This approach can help improve the accuracy of NLU models, despite the errors produced by the ASR models.
The experimental results in figure 2.8 presented in the paper demonstrate the effectiveness of the proposed multi-ASR setup in improving the accuracy of low-resource speech-command recognition and topic identification tasks. Figure 2.8 reports an impressive accuracy of 88.25% with less than 0.5 hours of Tamil[5] audio clips using a combined ASR approach. The authors recommend the application of this setup to different domains and low-resource languages plus the expansion of the multi-ASR model to include more than two ASR models and phoneme-based ASR models.
This paper presents a novel approach to addressing the challenges of low-resource SLU tasks. The results suggest that combining the outputs of multiple ASR models can significantly improve the accuracy of downstream NLU models. The proposed approach has important implications for improving speech recognition and natural language understanding in low-resource settings.

[5]Tamil is a Dravidian language spoken mainly in the Indian state of Tamil Nadu and the Union Territory of Puducherry, as well as in some other parts of India and Sri Lanka.



DETAILS OF THE EXPERIMENT RESULTS

Experiment No.	ASR Model(s)	NLU Model	Accuracy
Exp. 1	Phoneme	1D CNN	81.35%
Exp. 2	DeepSpeech2	2D CNN	76.30%
Exp. 3	Wav2Vec 2.0	1D CNN	43.20%
Exp. 4	Wav2Vec 2.0	2D CNN	71.62%
Exp. 5	DeepSpeech2 + Wav2Vec 2.0 (Combined Feature)	2D CNN	**88.25%**
Exp. 6	DeepSpeech2 + Wav2Vec 2.0	Dual-Input CNN	83.50%

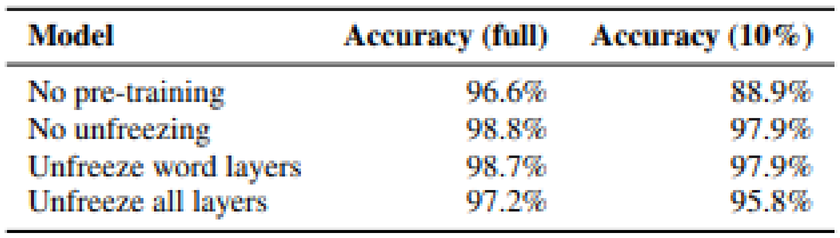
Figure 2.8: Details of the different Experiments and Results

Speech model pre-training for end-to-end Spoken Language Understanding

This paper demonstrates the use of the end-to-end approach of the Speech-to-text-to-intent technique. In [38], Lugosch et al. presents a pre-training methodology for end-to-end spoken language understanding (SLU) models that map speech directly to intent through a single trainable model. The proposed methodology involves pre-training the model to predict words and phonemes, which learns good features for SLU. A new SLU dataset, Fluent Speech Commands[32] was introduced in this paper which is used to demonstrate the efficacy of their pre-training techniques.
The end-to-end SLU model consists of three modules: phoneme level, word level, and intent level. The phoneme module is implemented using a SincNet layer, which processes the raw input waveform, followed by multiple convolutional layers and recurrent layers with pooling (to reduce the sequence length) and dropout. The phoneme module is pre-trained to predict phoneme targets using a linear classifier.
The output of the phoneme module serves as the input for the word module, which is made up of recurrent layers with dropout and pooling. The word module is pre-trained to predict whole-word targets using another linear classifier. The third module, which is not pre-trained, maps word-level representations to the predicted intent.

The paper evaluates the proposed methodology using various experiments. Lugosch et al. use the Montreal Forced Aligner to obtain word and phoneme-level alignments for LibriSpeech[6] and pre-train the model on the entire 960 hours of training data using these alignments. They also compare the accuracy of different models when given the full training dataset or a 10% subset of the training data.
The experiments show that the pre-trained models outperform the randomly initialized model in all cases. The best results are obtained when only the word layers of the pre-

[6]https://www.tensorflow.org/datasets/catalog/librispeech



Model	Accuracy (full)	Accuracy (10%)
No pre-training	96.6%	88.9%
No unfreezing	98.8%	97.9%
Unfreeze word layers	98.7%	97.9%
Unfreeze all layers	97.2%	95.8%

Figure 2.9: Accuracy on the test set for different models, given the full training dataset or a 10% subset of the training data

trained model are unfrozen. The authors suggest that this may be because the fully unfrozen model begins to forget the more general phonetic knowledge acquired during pre-training.
However, the paper also discusses the limitations of the end-to-end SLU model, such as its inability to handle new phrases and synonyms not observed in the SLU dataset. The authors suggest that it may be possible to teach the pre-trained part of the model to output ”embedding-like” word representations so that the intent module can recognize the meaning of phrases with synonyms.

2.6.3 Related work: Speech Intent Recognition for IoT applications, A Case study for Smart Homes

In [8],M.K.M et al. presents a prototype for a smart home automation system that uses natural language processing and machine learning to enable voice-controlled operation of devices in the home. This system features an interactive and humanoid central processing unit that eliminates the need for physical devices. Residents can use voice commands to control devices such as lights, doors, and fans. Additionally, the system is capable of searching the internet for information that it does not understand. The system includes a camera that continuously monitors the home and uses image processing techniques to identify people attempting to access the house. An automated mailing system is also included that sends photos captured by the camera based on whether users are valid or invalid. The system is designed to be easy to use, cost-effective, and non-physical. However, the authors note that voice recognition accuracy could be improved. Overall, the proposed system represents a significant advancement in home automation technology, offering users a more intuitive and efficient way to control their homes.

In [15], Kumar et al. presents a Smart Home Automation system that uses Natural Language Processing (NLP) and IoT cloud solutions to remotely control smart homes in a secure and customized manner. GoogleAPI is used to integrate devices and precisely monitor home devices. The proposed NLP-based IoTCloud infrastructure involves transmitting the sound or voice of the user to a microphone, smartphone, or any other device. Appliances like AC, refrigerators, fan, or bulb receive the sound command and use communication protocols such as ZigBee, WiFi, Bluetooth, internet, etc. to send the sound command to the cloud for processing and storing commands. The sound is enhanced using various techniques and processed to identify the user's voice. If the voice is matched with the pre-trained dataset, the command is executed, and the necessary actuators work.
The paper provides a prototype of several devices and sensors that are voice-controlled. Speech intent recognition is performed on the cloud using sound processing. Overall, the proposed system offers an efficient and secure way to control smart homes using NLP and IoT cloud solutions.

In [39], Reshamwala et al. presents an idea for creating an IoT-based Home Automation system that uses hardware devices such as Arduino Uno, ESP8266 NodeMCU, and sensors, along with software applications like Google Firebase, a real-time database, to create a fully functional smart home. The proposed system allows home appliances to be controlled through a user's mobile application via the internet from any location worldwide, while minor devices and their tasks are fully automated by the system. Additionally, voice commands can be integrated into the system using Google's Voice Assistant system. However, the paper focuses only on two actions for each device in the smart home. Overall, the proposed system offers a convenient and efficient way to control a smart home through IoT-based technology, with the potential for expansion to include more complex functionalities in the future.

In [40], Rani et al. presents a project aimed at developing a voice-based home automa-

tion system that uses the Internet of Things (IoT), Artificial Intelligence, and Natural Language Processing (NLP) to enable cost-effective and efficient control of home appliances using voice commands.
The NLP component of the system provides a personal touch, and the user is authenticated by providing a username and password on their mobile device. They can then issue voice commands, which are interpreted by the device and sent to the relevant appliance. The mobile device uses NLP to interpret the user's voice command.
The appliances are connected to an Arduino Board, which is used to demonstrate the IoT concept, and programmed to respond to mobile inputs. The project aims to automate the operation of every appliance in the home, leading to a significant reduction in power consumption due to the avoidance of excess or wasteful use of appliance services.
Overall, this project presents a promising approach to home automation, leveraging advanced technologies such as NLP and IoT to create a more intuitive and efficient user experience.

2.6.4 Discussion

The current state of Luganda intent classification for IoT device control reveals significant deficiencies, particularly the absence of a Luganda voice command dataset and a preliminary classification model. Existing research solutions in this domain fail to adequately address the gap in speech intent recognition for Luganda.
Our research project aims to rectify these shortcomings by developing a comprehensive Internet of Things (IoT) application that accurately recognizes speech commands in the Luganda language. This entails addressing the specific challenges of Luganda speech intent recognition for IoT applications and emphasizing the importance of deployment considerations.
In contrast to the reviewed papers, which predominantly focus on English or other languages for device control, our research specifically targets Luganda as the primary language for controlling devices in a smart home environment. This research endeavour is particularly significant for regions where Luganda is widely spoken.
Moreover, while most of the reviewed papers concentrate on cloud-based IoT solutions for device control, our proposed research takes a more practical and localized approach. By deploying the Natural Language Processing (NLP) model on the Raspberry Pi and establishing communication with ESP32 nodes, we enable localized and efficient smart home automation, which is especially advantageous in areas with limited internet connectivity.
Furthermore, none of the papers discusses the complete IoT implementation, from model design to deployment and integration with IoT devices. Our research project addresses these gaps by providing a holistic approach that encompasses the entire process, from developing the models to deploying and integrating them within an IoT framework. Additionally, it is important to highlight that the papers focused solely on model development did not consider the aspects of deployment and optimization for real-world applications. In contrast, the papers that did discuss deployment in a smart home setting lacked comprehensive discussions on the various techniques employed in building the models, such as quantization and model architecture.

Our research project recognizes the criticality of training data in a specific language for accurate speech recognition and translation into device commands. Consequently, we have curated a newly constructed dataset of Luganda voice commands, specifically tailored for training our NLP model. This dataset covers a comprehensive range of 20 distinct intents, enabling the control of up to 8 devices, surpassing the limited device and command range discussed in prior literature.
The implementation of our system on a Raspberry Pi, communicating with ESP32 nodes, distinguishes our research project. This integration effectively combines the advantages of NLP and IoT technologies, offering users an intuitive and efficient means of controlling their devices using natural language commands in their local language.
In terms of model development, we considered the use of Mel-frequency cepstral coefficients (MFCCs) as input, drawing inspiration from the approaches discussed in the research papers by Yang et al. [33] and Buddhika et al. [36]. By incorporating MFCCs, we capture relevant acoustic features that contribute to the speech-to-intent model's accuracy. Convolutional Neural Networks (CNNs) were employed to process the MFCC features and extract meaningful representations, enhancing the model's performance.
Considering the deployment scenario and the need for fast inference in a smart home environment, we made the decision to focus on the speech-to-intent approach, rather than the speech-to-text approach. This choice was influenced by the real-time requirements and the intention to deploy the models on edge devices. By leveraging MFCCs and CNNs, we achieve efficient and accurate speech classification capabilities, aligning with the deployment constraints and objectives of our project.
To enhance the robustness of the models, we explored augmentation techniques. These techniques aimed to increase the diversity and variability of the training data, thereby improving the models' ability to generalize and adapt to different speech patterns and environmental conditions.
To ensure successful deployment on edge devices, we paid particular attention to model quantization techniques, reducing the model size to enable faster inference and meet the memory requirements. Optimization and quantization were crucial factors in our research project.

The upcoming chapter will provide further insights into the methodology employed in the project, detailing the specific techniques and approaches used to develop and optimize the Luganda speech-to-intent models for deployment on edge devices.

Chapter 3

Methodology

Introduction

The methodology chapter presents a detailed description of the research approach and procedures employed in this study. It outlines the systematic framework adopted to achieve the research objectives and provides insights into the various methods, tools, and techniques utilized for data collection, analysis, model development, deployment, IoT integration and lastly system evaluation. The procedures taken are summarised in figure 3.1.

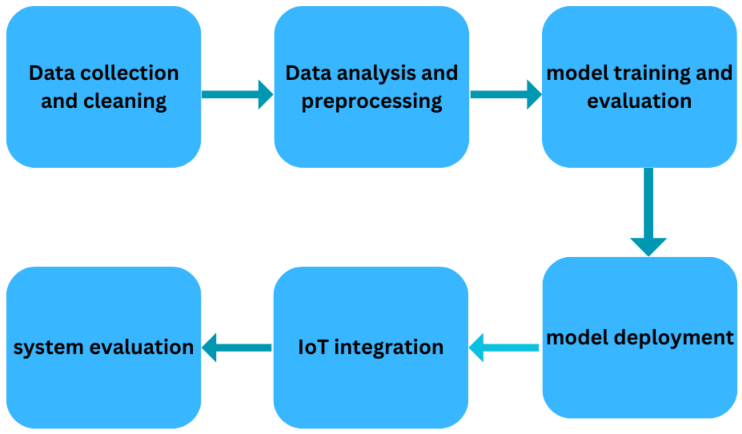

Data collection and cleaning
Data analysis and preprocessing
model training and evaluation
model deployment
IoT integration
system evaluation

Figure 3.1: Project Methodology

3.1 Data Collection and Cleaning

Data collection for this research project followed the following procedure:
The dataset was collected based on a list of transcriptions for each intent. The dataset was collected through crowd-sourcing, which involved recruiting multiple individuals to record the utterances. Participants were provided with a set of predefined intents and were asked to speak corresponding commands corresponding to various smart home devices and actions in Luganda. The intents were identified on 2 slots: *action, device*. The smart home devices and their associated intents that were considered during the collection of the dataset are in the table 3.1.

It is important to note that each intent could have various transcriptions, capturing

Table 3.1: Intents and Actions for Home Automation Devices

Device	**Intents and Actions**
Door	- Open - Close
Lights	- On - Off
Fan	- On - Off - Increase speed - Decrease speed
Alarm	- On - Off
TV,	- On - Off
Fridge	- On - Off
Camera	- On - Off
Speaker	- On - Off - Increase volume - Decrease volume

the natural language variations and expressions used by different individuals. For example, for the "lights on" intent in Luganda, transcriptions such as "saako ettaala," "ntelaako ettaala," "ettaala gyitekeeko," and others could be present. The different transcriptions on how users could interact with those devices using voice formed the basis of the dataset.
The data collection process was carried out using smartphones, capturing audio record-

ings in various formats such as .wav, .aac, and .m4a. To maintain consistency, all audio files were converted to the .wav format for further processing. This conversion was done using ffmpeg[1], a command-line tool for audio processing. Each utterance is recorded as a 16 kHz single-channel .wav file.
A total of 10,200 audio files were collected from 81 participants, comprising 43 males and 38 females. The participants' age range was between 20 to 25 years. The distribution of the data classes in the dataset is shown in Figure 3.2 below.

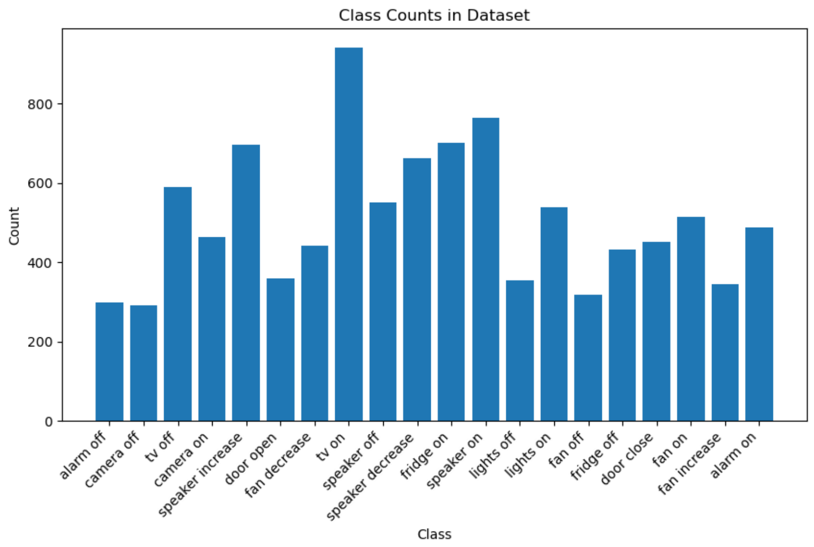


Figure 3.2: Distribution of classes in the dataset

The collected data was split into training, test, and validation sets, as shown in Table 1. During the data labelling process, special attention was given to the test set to ensure that the audio files were from different speakers whose audios were not exposed to the model during training.
The reason for having this test set with audio files from different speakers is to simulate real-world scenarios where the model may encounter unseen speakers during deployment. This helps evaluate the model's performance on speaker-independent speech intent classification, ensuring its robustness and ability to generalize to new speakers.

Data Split	**Number of Audio Files**
Training	8,127
Test	1,068
Validation	1,005
Total	**10,200**

Table 3.2: Data Splitting

The collected data, with various intents related to different smart home devices, ensures

[1]https://ffmpeg.org/

a comprehensive representation of speech commands typically used in a smart home environment. This allows the trained model to effectively classify the intended actions based on the input speech commands.
The subsequent steps of data cleaning, data preprocessing, model development and evaluation, model deployment, IoT integration, and system evaluation were performed to develop and assess the Luganda speech intent classification system.

3.2 Data Preprocessing

Data preprocessing plays a crucial role in preparing the collected audio data for further analysis and model training. In this section, the methodology employed for data preprocessing in our research project is presented, with a specific focus on the extraction of audio features with a goal for deployment on both the Wio Terminal and Raspberry Pi platforms. The extracted features serve as essential inputs for training and evaluating our speech intent classification models.
The data preprocessing stage encompasses various tasks, including audio signal processing, feature extraction, and data augmentation. Specifically, two sets of experiments were conducted: one for the Wio Terminal and another for the Raspberry Pi. These experiments involved different feature extraction parameters and techniques to capture the distinct characteristics of the audio signals from each platform. Mel Frequency Cepstral Coefficients were used as features in this step. The transition of raw audio to MFCCs is illustrated in figure 3.3.

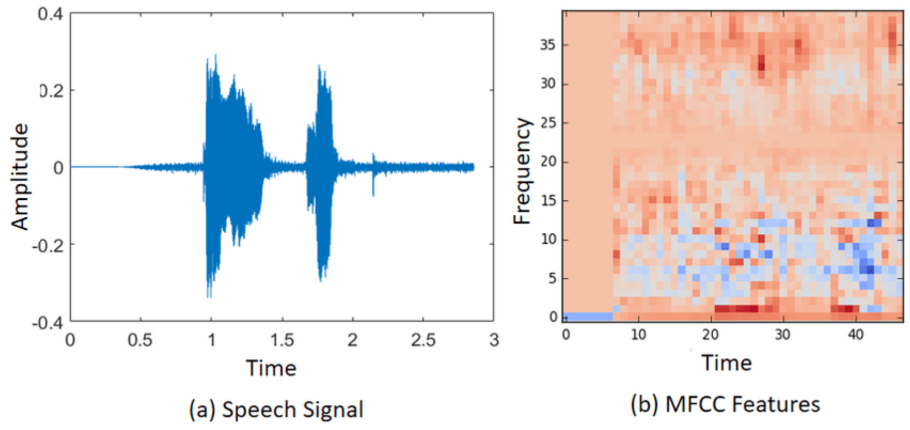


Figure 3.3: The transition from raw speech to MFCC

For the Wio Terminal, experiments were conducted to extract features using both augmented and unaugmented audio data. Furthermore, the impact of feature dimensionality was considered by extracting 10 features from the audio signals. On the other hand, for the Raspberry Pi, two different feature sets were extracted: 10 MFCCs and 13 MFCCs.
Additionally, experiments were conducted with and without data augmentation to in-

vestigate the influence of augmentation techniques on the performance of the models. The feature extraction was done in Jupyter Notebooks[2].

3.2.1 Data Augmentation

In the context of this project, data augmentation techniques were employed to enhance the training data for audio-based machine learning models. Specifically, two augmentation techniques, namely adding white noise and pitch scaling, were implemented using the Librosa library. These techniques aimed to introduce variability and diversity into the training data, improving the model's ability to generalize and handle different acoustic conditions.
The first augmentation technique involved adding white noise to the input audio signals. Random noise with an intensity of 0.2 was generated and added to the original signals. This process simulated various background noise scenarios that the model could encounter during inference.
The second augmentation technique applied pitch scaling to the audio signals. A pitch scale factor of 2 was utilized to modify the pitch or frequency content of the signals. This adjustment allowed the model to handle variations in pitch, which naturally occur in speech or audio recordings. Figure 3.4 shows the variations of one of the signals extracted from the dataset where augmentation was applied in code.

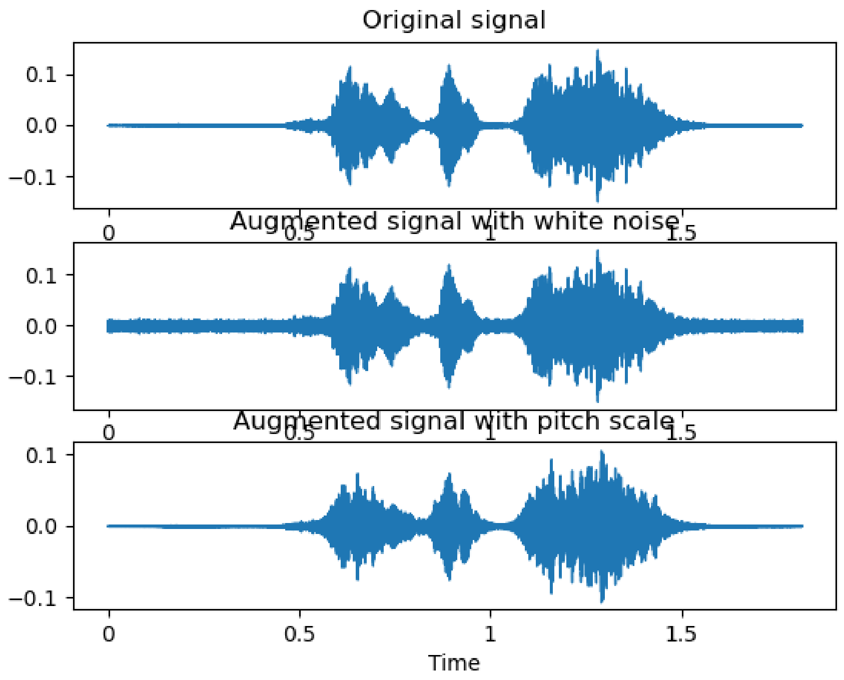


Figure 3.4: The changes in the signal during augmentation

The data augmentation process was implemented using the Librosa library, which pro-

[2]https://jupyter.org/

vided a range of audio processing functions and utilities. The augmentation function took the original signal as input and returned the original signal along with two augmented versions: one with added white noise and another with pitch scaling applied. These augmented signals expanded the training dataset, providing additional variations and realistic scenarios for the model to learn from.

3.2.2 Feature Extraction: Wio Terminal

These were the steps taken to extract features from the audio for the model developed with the goal of deployment on a Wio Terminal.

1. **Loading Audio:** The audio files were loaded using the librosa[3] library with a sample rate of 16,000 Hz, ensuring consistency across all audio samples.
2. **Resizing Audio:** To standardize the length of the audio samples, they were resized to a fixed duration of 3 seconds.
3. **MFCC Feature Extraction:** Mel Frequency Cepstral Coefficients (MFCCs) were extracted as the acoustic features from the preprocessed audio signals. The TensorFlow Python Operations library was chosen for MFCC calculation due to its compatibility with the MFCC Opcode from Arm's repository for Keyword Spotting on ARM Microcontrollers. This ensured consistency in the feature extraction process.

   The following parameters were used during the MFCC feature extraction:

   - **Minimum Frequency:** 100 Hz
   - **Maximum Frequency:** 8000 Hz
   - **Window Size:** 20 milliseconds (ms)
   - **Window Increase:** 20 ms
   - **Number of Cepstral Coefficients:** 10

3.2.3 Feature Extraction: Raspberry Pi

These were the steps taken to extract features from the audio for the model developed with the goal of its deployment on a Raspberry Pi.

1. **Loading Audio:** The audio files were loaded using the librosa[4] library with a sample rate of 16,000 Hz, ensuring consistency across all audio samples.
2. **Resizing Audio:** To standardize the length of the audio samples, they were resized to a fixed duration of 4 seconds.
3. **MFCC Feature Extraction:** Mel Frequency Cepstral Coefficients (MFCCs) were extracted as the acoustic features from the preprocessed audio signals. Two sets of features were generated based on both 10 and 13 mfccs. The following parameters were used during the MFCC feature extraction:

   - Number of MFCCs: 10 and 13

[3]https://librosa.org/doc/latest/index.html

[4]https://librosa.org/doc/latest/index.html

- FFT window size: 2048
- Minimum frequency: 100 Hz
- Hop length: 512
- Window type: Hanning window

3.2.4 Label Encoding

The speech intent label associated with each audio sample was encoded using LabelEncoder from the scikit-learn[5] library. This encoding step ensured that the labels were represented numerically, facilitating audio classification tasks.

By applying these data preprocessing steps, we obtained the MFCC features that capture the relevant acoustic characteristics of the audio signals, which serve as the input for the subsequent audio classification model development and evaluation stages. The encoded labels were also extracted which served as the labels for the classification tasks.

3.3 Model Development and Evaluation

In this research project, Convolutional Neural Networks (CNNs) were employed for the development of the Luganda speech intent classification model. The MFCC features extracted during the data preprocessing phase served as the input to the CNN model. Model development was done in Google Colaboratory[6] utilising the free access to GPUs. The CNN model was carefully designed and optimized to achieve the best performance on the validation set. The model architecture, hyperparameters, and training process were fine-tuned to maximize the validation accuracy. This was done using a callback known as EarlyCallback with a patience of 20. The model architecture was as shown in figure 3.5.
2 CNN model architectures were used in the study. The first architecture, as shown in figure 3.5(b), consisted of multiple convolutional layers followed by batch normalization and max pooling layers. It further included a global max pooling layer, dropout layers, and dense layers. At the classification layer, a softmax activation function was applied, which transformed the output of the previous dense layer into a probability distribution over the different classes. The softmax function ensured that the predicted class probabilities sum up to 1, allowing the model to make predictions based on the highest probability class. This architecture, with the original Global Max Pooling layer, was used to process audio features extracted from either 10 or 13 Mel Frequency Cepstral Coefficients (MFCCs) both with and without augmentation.
In a modified architecture shown in Figure 3.5(a), the Global Max Pooling layer was replaced by a flattening layer. This layer reshaped the output from the previous convolutional layers into a 1-dimensional vector, which was then fed into the subsequent dropout and dense layers. The softmax activation function was applied at the classi-

[5]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
[6]https://colab.research.google.com/

fication layer, enabling the model to generate class probabilities based on the learned features. Both architectures were trained and evaluated using the respective sets of audio features, aiming to classify the audio samples into different classes.
The parameters used in the training were as shown in Table 3.3 below.

Hyperparameter	**value**
Optimizer	Adam
Learning rate	0.001
Batch size	32

Table 3.3: Hyperparameter Selection

To assess the effectiveness of the models, separate features from test datasets were prepared specifically for evaluating their performance. These test datasets were distinct from the data used for training and validation purposes. For instance, the models trained on the set of 13 MFCCs were evaluated using the corresponding set of MFCCs that had been reserved solely for testing. This approach ensured an unbiased evaluation of the models' capabilities and provided insights into their generalization and predictive accuracy.
Finally, to enable easy deployment and integration into the smart home system, the best model from the assessment of results was saved in the .tflite format. This compact file format allows for efficient storage and execution of the model on edge devices. The model was further quantized to enable faster inference in real-world scenarios. Different quantization techniques were explored.
The model choice for the Wio Terminal and Raspberry Pi was based on a number of considerations based on the experiments done on the sets of features extracted for the 2 modes of deployment on the Wio terminal and Raspberry Pi. The various experiments are presented in the next subsection 3.3.

Experiments

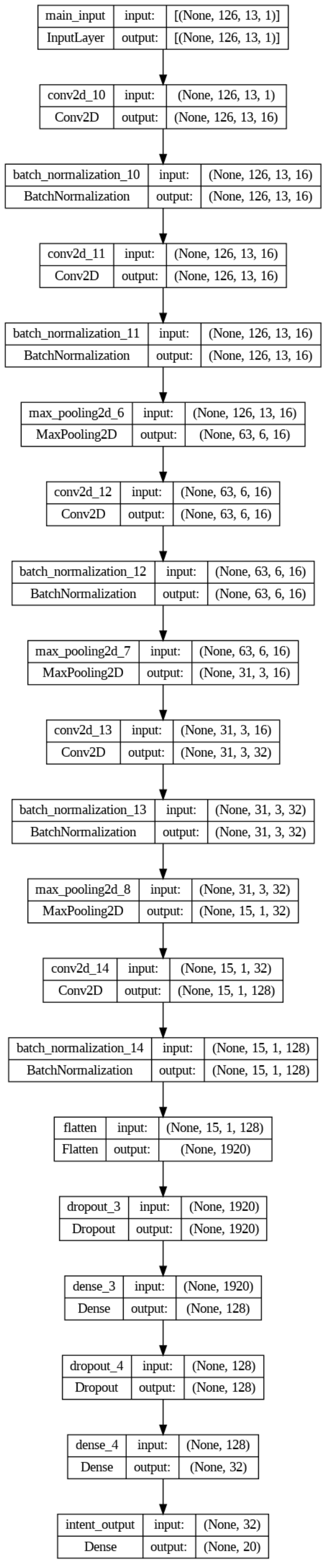
Experiments A to D were done to find the best model for deployment on the Wio Terminal. Experiment A involved training the model using a Global Max Pooling layer on the training set with 10 MFCCs extracted without augmentation. The purpose of this experiment was to assess the performance of the model without incorporating data augmentation techniques.
Experiment B followed a similar approach as Experiment A but included data augmentation. The model was trained using 10 MFCCs extracted from the training set with the addition of augmentation techniques. This experiment aimed to evaluate whether data augmentation could enhance the model's performance.
Moving on to Experiment C, the focus shifted to using a Flattening layer instead of Global Max Pooling. The model was trained on the training set with 10 MFCCs without data augmentation. The objective was to compare the performance of the model when a different architecture was employed. Lastly, Experiment D replicated Experiment C but included data augmentation. The model was trained using 10 MFCCs extracted from the training set with the incorporation of augmentation techniques. The purpose

was to evaluate the combined impact of data augmentation and the Flattening layer on the model's performance.
These experiments were conducted to explore the effects of different architectural choices and data augmentation techniques on the accuracy and performance of the Wio Terminal models. The results obtained from these experiments were documented and are summarized in Table 4.1.

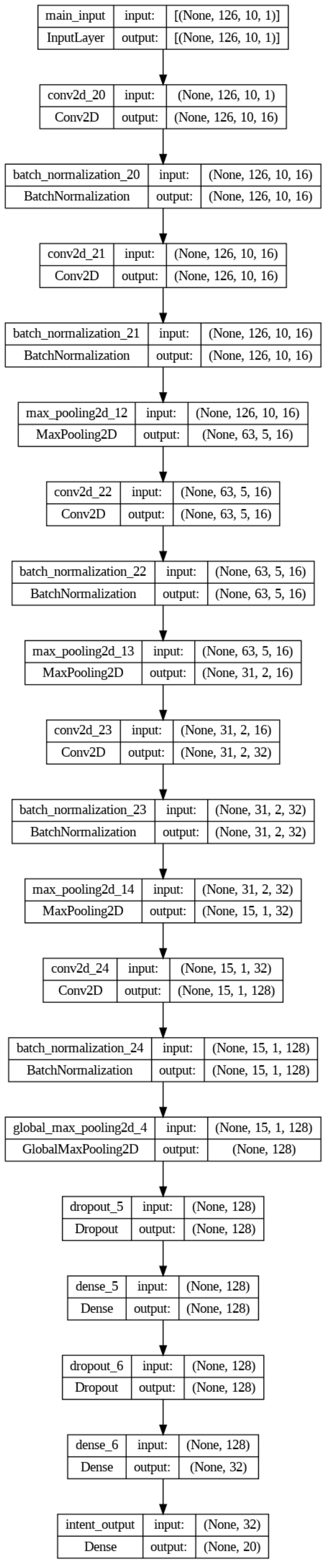
Experiments 1 to 8 were done to assess the best model for deployment on the Raspberry Pi.
Experiment 1 involved training the model for deployment on the Raspberry Pi using a Global Max Pooling layer. The model was trained on the training set with 10 MFCCs extracted without data augmentation. The objective of this experiment was to assess the performance of the model without incorporating data augmentation techniques.
Experiment 2 followed a similar approach as Experiment 1 but included data augmentation. The model was trained using 10 MFCCs extracted from the training set with the addition of augmentation techniques. This experiment aimed to evaluate whether data augmentation could improve the model's performance when deployed on the Raspberry Pi.
Moving on to Experiment 3, the focus shifted to using a Flattening layer instead of Global Max Pooling. The model was trained on the training set with 10 MFCCs without data augmentation. The purpose was to compare the performance of the model when a different architectural choice was made for deployment on the Raspberry Pi.
Experiment 4 replicated Experiment 3 but included data augmentation. The model was trained using 10 MFCCs extracted from the training set with the incorporation of augmentation techniques. The goal was to assess the combined impact of data augmentation and the Flattening layer on the model's performance when deployed on the Raspberry Pi.
Next, Experiment 5 explored the use of 13 MFCCs without data augmentation and a Global Max Pooling layer. The model was trained on the training set with the aim of evaluating the performance of the model with an increased number of MFCCs.
Experiment 6 mirrored Experiment 5 but included data augmentation. The model was trained using 13 MFCCs extracted from the training set with the addition of augmentation techniques. This experiment sought to assess the impact of data augmentation on the performance of the model with an increased number of MFCCs.
Experiment 7 examined the use of a Flattening layer with 13 MFCCs without data augmentation. The model was trained on the training set to evaluate the performance of this architectural choice when combined with an increased number of MFCCs.
Lastly, Experiment 8 replicated Experiment 7 with the inclusion of data augmentation. The model was trained using 13 MFCCs extracted from the training set with the incorporation of augmentation techniques. The purpose was to assess the combined impact of data augmentation and the Flattening layer on the model's performance with an increased number of MFCCs.
These experiments aimed to explore the effects of different model architectures and data augmentation techniques on the accuracy and performance of the models for deployment on the Raspberry Pi. The results obtained from these experiments were

documented and are summarized in Table 4.3.
For each of the modes of deployment, the best model was chosen and taken for quantization. Results for the model quantization for the Wio Terminal and Raspberry Pi are discussed in Tables 4.2 and 4.5 respectively.

Slot Filling: In our research project, we also employed the technique of Slot Filling to investigate how our model generalizes to different outputs based on specific slots. The model was designed with two slots, namely "action" and "object," which constituted the outputs of the model. The outcomes and findings of this technique will be further discussed in the upcoming chapter. The model architecture utilized for Slot Filling is visually depicted in Figure 3.6. This architecture provides a clear representation of the two output slots and their relationship within the model. The performance of this architecture and its implications will be elaborated upon in subsequent sections of this report.


main_input | input: | [(None, 126, 13, 1)]
InputLayer | output: | [(None, 126, 13, 1)]
conv2d_10 | input: | (None, 126, 13, 1)
Conv2D | output: | (None, 126, 13, 16)
batch_normalization_10 | input: | (None, 126, 13, 16)
BatchNormalization | output: | (None, 126, 13, 16)
conv2d_11 | input: | (None, 126, 13, 16)
Conv2D | output: | (None, 126, 13, 16)
batch_normalization_11 | input: | (None, 126, 13, 16)
BatchNormalization | output: | (None, 126, 13, 16)
max_pooling2d_6 | input: | (None, 126, 13, 16)
MaxPooling2D | output: | (None, 63, 6, 16)
conv2d_12 | input: | (None, 63, 6, 16)
Conv2D | output: | (None, 63, 6, 16)
batch_normalization_12 | input: | (None, 63, 6, 16)
BatchNormalization | output: | (None, 63, 6, 16)
max_pooling2d_7 | input: | (None, 63, 6, 16)
MaxPooling2D | output: | (None, 31, 3, 16)
conv2d_13 | input: | (None, 31, 3, 16)
Conv2D | output: | (None, 31, 3, 32)
batch_normalization_13 | input: | (None, 31, 3, 32)
BatchNormalization | output: | (None, 31, 3, 32)
max_pooling2d_8 | input: | (None, 31, 3, 32)
MaxPooling2D | output: | (None, 15, 1, 32)
conv2d_14 | input: | (None, 15, 1, 32)
Conv2D | output: | (None, 15, 1, 128)
batch_normalization_14 | input: | (None, 15, 1, 128)
BatchNormalization | output: | (None, 15, 1, 128)
flatten | input: | (None, 15, 1, 128)
Flatten | output: | (None, 1920)
dropout_3 | input: | (None, 1920)
Dropout | output: | (None, 1920)
dense_3 | input: | (None, 1920)
Dense | output: | (None, 128)
dropout_4 | input: | (None, 128)
Dropout | output: | (None, 128)
dense_4 | input: | (None, 128)
Dense | output: | (None, 32)
intent_output | input: | (None, 32)
Dense | output: | (None, 20)

(a) With Flattening layer


main_input | input: | [(None, 126, 10, 1)]
InputLayer | output: | [(None, 126, 10, 1)]
conv2d_20 | input: | (None, 126, 10, 1)
Conv2D | output: | (None, 126, 10, 16)
batch_normalization_20 | input: | (None, 126, 10, 16)
BatchNormalization | output: | (None, 126, 10, 16)
conv2d_21 | input: | (None, 126, 10, 16)
Conv2D | output: | (None, 126, 10, 16)
batch_normalization_21 | input: | (None, 126, 10, 16)
BatchNormalization | output: | (None, 126, 10, 16)
max_pooling2d_12 | input: | (None, 126, 10, 16)
MaxPooling2D | output: | (None, 63, 5, 16)
conv2d_22 | input: | (None, 63, 5, 16)
Conv2D | output: | (None, 63, 5, 16)
batch_normalization_22 | input: | (None, 63, 5, 16)
BatchNormalization | output: | (None, 63, 5, 16)
max_pooling2d_13 | input: | (None, 63, 5, 16)
MaxPooling2D | output: | (None, 31, 2, 16)
conv2d_23 | input: | (None, 31, 2, 16)
Conv2D | output: | (None, 31, 2, 32)
batch_normalization_23 | input: | (None, 31, 2, 32)
BatchNormalization | output: | (None, 31, 2, 32)
max_pooling2d_14 | input: | (None, 31, 2, 32)
MaxPooling2D | output: | (None, 15, 1, 32)
conv2d_24 | input: | (None, 15, 1, 32)
Conv2D | output: | (None, 15, 1, 128)
batch_normalization_24 | input: | (None, 15, 1, 128)
BatchNormalization | output: | (None, 15, 1, 128)
global_max_pooling2d_4 | input: | (None, 15, 1, 128)
GlobalMaxPooling2D | output: | (None, 128)
dropout_5 | input: | (None, 128)
Dropout | output: | (None, 128)
dense_5 | input: | (None, 128)
Dense | output: | (None, 128)
dropout_6 | input: | (None, 128)
Dropout | output: | (None, 128)
dense_6 | input: | (None, 128)
Dense | output: | (None, 32)
intent_output | input: | (None, 32)
Dense | output: | (None, 20)

(b) With Global Max Pooling

Figure 3.5: Comparison of Model Architectures

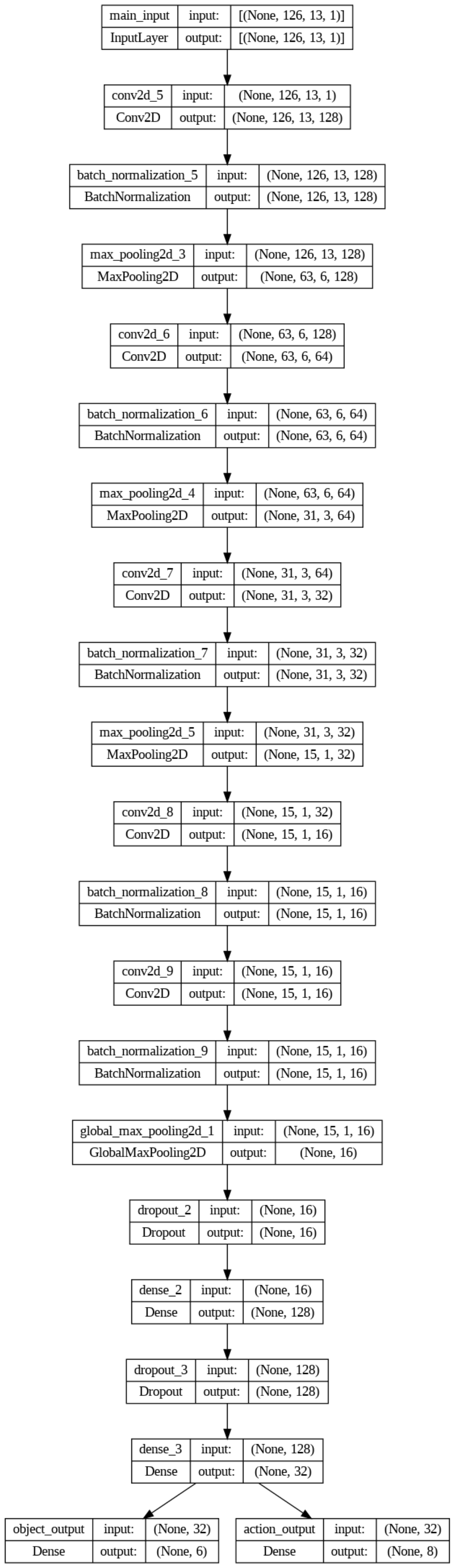

main_input | input: | [(None, 126, 13, 1)]
InputLayer | output: | [(None, 126, 13, 1)]
conv2d_5 | input: | (None, 126, 13, 1)
Conv2D | output: | (None, 126, 13, 128)
batch_normalization_5 | input: | (None, 126, 13, 128)
BatchNormalization | output: | (None, 126, 13, 128)
max_pooling2d_3 | input: | (None, 126, 13, 128)
MaxPooling2D | output: | (None, 63, 6, 128)
conv2d_6 | input: | (None, 63, 6, 128)
Conv2D | output: | (None, 63, 6, 64)
batch_normalization_6 | input: | (None, 63, 6, 64)
BatchNormalization | output: | (None, 63, 6, 64)
max_pooling2d_4 | input: | (None, 63, 6, 64)
MaxPooling2D | output: | (None, 31, 3, 64)
conv2d_7 | input: | (None, 31, 3, 64)
Conv2D | output: | (None, 31, 3, 32)
batch_normalization_7 | input: | (None, 31, 3, 32)
BatchNormalization | output: | (None, 31, 3, 32)
max_pooling2d_5 | input: | (None, 31, 3, 32)
MaxPooling2D | output: | (None, 15, 1, 32)
conv2d_8 | input: | (None, 15, 1, 32)
Conv2D | output: | (None, 15, 1, 16)
batch_normalization_8 | input: | (None, 15, 1, 16)
BatchNormalization | output: | (None, 15, 1, 16)
conv2d_9 | input: | (None, 15, 1, 16)
Conv2D | output: | (None, 15, 1, 16)
batch_normalization_9 | input: | (None, 15, 1, 16)
BatchNormalization | output: | (None, 15, 1, 16)
global_max_pooling2d_1 | input: | (None, 15, 1, 16)
GlobalMaxPooling2D | output: | (None, 16)
dropout_2 | input: | (None, 16)
Dropout | output: | (None, 16)
dense_2 | input: | (None, 16)
Dense | output: | (None, 128)
dropout_3 | input: | (None, 128)
Dropout | output: | (None, 128)
dense_3 | input: | (None, 128)
Dense | output: | (None, 32)
object_output | input: | (None, 32)
Dense | output: | (None, 6)
action_output | input: | (None, 32)
Dense | output: | (None, 8)

Figure 3.6: Architecture for Slot Filling showing the 2 output slots

3.4 Model Deployment

The best models from the quantisation process for deployment on the Wio Terminal and Raspberry Pi were deployed on the corresponding devices.

3.4.1 Deployment on the Wio Terminal

The deployment of the model on the Wio Terminal involved several steps, including model conversion, code generation, and integration with the Arduino IDE. The process can be divided into three main parts: audio acquisition, MFCC calculation, and inference on MFCC features. This subsection provides a concise overview of the deployment process.

Model Conversion and Code Generation

To deploy the model on the Wio Terminal, it was first converted to the TensorFlow Lite format with int8 quantization. The converted model was then further transformed into a .h file suitable for Arduino sketches. This step ensured compatibility and efficient execution on the Wio Terminal.

Arduino IDE Setup and Sketch Creation

The deployment required the installation of the TensorFlow Lite library in the Arduino IDE. Most of the new library versions didn't work as expected and The TensorFlow Lite Library was downloaded from the Arduino TensorFlowLite github repository [41] from a commit labelled *219ac1dfed8a8ba0edfdbaae51aed5dc9b208c0c* from 2021 as directed by the Seeed Studio support page for TensorFlow Lite [7]. Once the library was installed, a new Arduino IDE sketch was created to facilitate the deployment process. It is crucial to ensure that the Seeed SAMD board definitions are at least version 1.8.2 to avoid any compilation errors with the TensorFlow Lite library. The Tensorflow Interpreter was instantiated on startup and all necessary operations were included within the sketch.

Audio Acquisition and MFCC Calculation

The capture of audio was started by the press of a button and it lasted 3 seconds. The WIO-KEY-A on the Wio Terminal was utilised. In order to process sound captured by the Wio Terminal's built-in microphone, the DMA ADC (Direct Memory Access Analog-to-Digital Converter) function of the Cortex M4F MCU was utilized. The DMA functionality allowed for efficient data transfer from the microphone to the memory without significant involvement from the MCU. The captured audio was then subjected to MFCC (Mel Frequency Cepstral Coefficients) calculation, which involved extracting the relevant acoustic features from the audio data.
The calculation of MFCC features on the Wio Terminal's Cortex M4F MCU was borrowed from Arm's repository for Keyword Spotting on ARM Microcontrollers[8] which

[7]https://wiki.seeedstudio.com/Wio-Terminal-TinyML-TFLM-1/
[8]https://github.com/ARM-software/ML-KWS-for-MCU

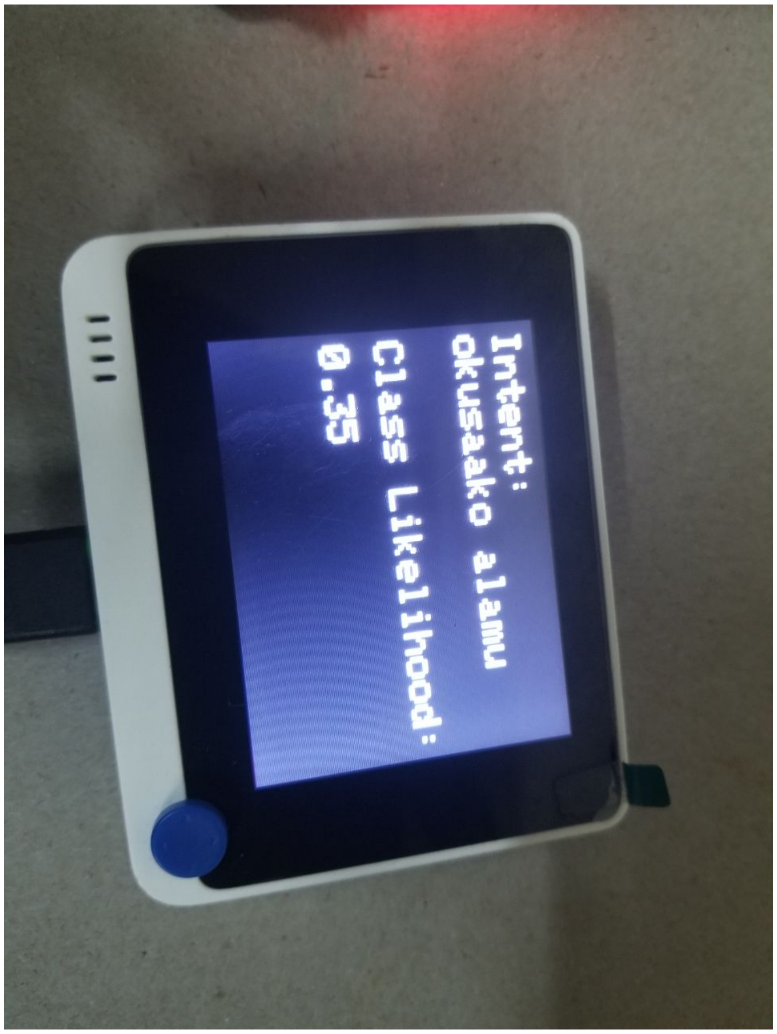


Figure 3.7: Wio terminal displaying an intent

provided the necessary code implementation in C++ to match the TensorFlow MFCC Op code that was used at the feature extraction phase in Python and ensured accurate feature extraction on the Wio Terminal.

Inference on MFCC Features

Once the audio within one sample (e.g., 3 seconds) was processed and converted into MFCC features, the MFCC feature array was converted from FLOAT32 to INT8 values. These INT8 values were then fed into the model interpreter for inference using the TensorFlow Lite for Microcontrollers library.

Display of Results

The results of the inference process were displayed on the Wio Terminal's screen, providing real-time feedback on the classification outcomes.

Overall, the deployment process involved converting the model, setting up the Arduino IDE, acquiring audio data, calculating MFCC features using code borrowed from Arm's repository, performing inference, and displaying the results. These steps ensured the successful integration and execution of the model on the Wio Terminal, enabling it to classify speech intents in real time.
However, during the course of the project, we encountered a challenge when working with the Wio Terminal board. It was discovered that the board had limitations in simultaneously supporting both the model and WiFi capabilities. Although the individual components could function independently when executed in separate sketches, integrating them into a single program resulted in program hang-ups. Additionally, uploading new code to the board necessitated a forced reboot, further hindering the coexistence of the model and WiFi functionalities. Consequently, we were unable to achieve our third objective of IoT integration using the Wio Terminal. With this limitation, we explored alternative options and ultimately turned to the Raspberry Pi for its ability to accommodate both the model and WiFi functionalities simultaneously, enabling us to proceed with our intended IoT integration goals. The deployment of the model on the Raspberry Pi is discussed in the next subsection 3.4.2.

3.4.2 Deployment on Raspberry Pi

The trained Luganda speech intent classification model was successfully deployed on a Raspberry Pi for real-time inference and integration into the smart home system. The operation of the model on the Raspberry Pi can be put in the following steps:

Model Processing on Raspberry Pi

The saved .tflite model was loaded onto the Raspberry Pi, which served as the edge device for executing the inference process locally.

Installation of libraries

The necessary libraries for audio capture and model processing and MQTT transmission were installed onto the Raspberry Pi. The specific library versions utilized in the project are listed in Table 3.4. The tflite-runtime library played a crucial role in executing the .tflite model, while Numpy was employed for resizing loaded audio arrays and expanding array dimensions. The audio files were loaded using the librosa library, and the paho-mqtt library facilitated the publication of MQTT messages on a broadcast topic.

Audio Capture on Microphone:

In the deployment on the Raspberry Pi, the audio capture was facilitated by a microphone connected to the input port of the device. The reSpeaker 2-Mics Pi HAT, equipped with two analogue microphones and the WM8960 Audio Codec, played a crucial role in achieving high-definition voice capture for our application.
To initiate the recording process, the reSpeaker 2-Mics Pi HAT's inbuilt button was utilized. In our project, the recording process was triggered by pressing this button.

Library	Version
librosa	0.9.1
numpy	22.04
scipy	1.6.0
tf-lite-runtime	2.9.1
paho-mqtt	1.6.1

Table 3.4: Library versions for the Raspberry Pi

Once the button was pressed, the audio capture commenced, and the recording process began.
In addition, the reSpeaker 2-Mics Pi HAT featured programmable APA102 RGB LEDs, which were effectively used to create an engaging interface display. These LEDs served the purpose of visually indicating when the recording had started, providing users with clear feedback on the status of the audio capture.
To capture the audio snippets, the Raspberry Pi employed the *arecord* command-line utility, which was executed using the subprocess Python library. The audio snippets were recorded for a duration of 3 seconds at a sampling rate of 32000 Hz. The recorded audio was saved as "test.wav" and utilized for further preprocessing and analysis.
The utilization of the reSpeaker 2-Mics Pi HAT, along with its inbuilt recording button and the APA102 RGB LEDs, offered a comprehensive and interactive audio capture setup in our project. This setup enhanced the user experience by providing a convenient button for recording initiation and clear visual indications of the recording process. Figure 3.8 shows the lighting of the APA102 RBG LEDs.

Feature Extraction based on MFCCs:

The audio file *test.wav* was loaded using librosa and decimated to 16000 Hz using the Scipy library by using a decimation factor of 2. The captured audio was then preprocessed to extract MFCC features using the same parameters as during the training phase. These features served as the input to the deployed model.

Model Inference:

The MFCC features were fed into the deployed model on the Raspberry Pi, which performed the speech intent classification. The model made predictions based on the trained weights and architecture, determining the intended actions from the input speech commands.

Broadcast to IoT Devices:

Once the speech intent was classified by the deployed model, the Raspberry Pi sent commands to the corresponding IoT devices in the smart home system. For example, if the intent was to turn on the lights, the Raspberry Pi would send the appropriate command to the lights device, triggering the desired action. This is further discussed

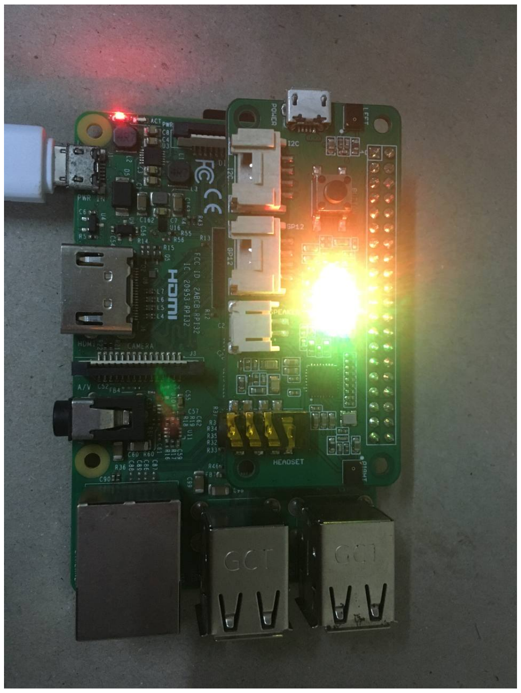


Figure 3.8: reSpeaker LEDS lighting during recording

in the next subsection 3.4.3.

Through the successful deployment of the trained model on the Raspberry Pi, the smart home system became capable of accepting voice commands in Luganda, processing them locally, and controlling the IoT devices accordingly.

3.4.3 IoT Integration

The Luganda speech intent classification system was seamlessly integrated with the Internet of Things (IoT) devices in the smart home environment. The integration process involved connecting multiple devices, including a Raspberry Pi, a Wio Terminal, and two ESP boards. The Raspberry Pi, therefore, acted as the central hub for processing the inferred commands and distributing them to the respective IoT devices.
The MQTT protocol was utilized for efficient and reliable message transmission. This required a WiFi connection and this was provided by a smartphone's hotspot to which all the devices were connected. The Raspberry Pi served as the MQTT server, facilitating communication between the different microcontrollers. This was done by successfully installing the Mosquitto broker on the Raspberry Pi. The IP address of the Raspberry Pi when connected to this hotspot was issued as the MQTT server when setting up MQTT on the ESP32 boards and Wio Terminal. The integration workflow consisted of the following steps:

Connection Setup:

The Raspberry Pi was connected to the Wio Terminal and the two ESP32 boards on the same WiFi network provided by a smartphone. This established the wireless connection between the central device and the IoT devices.

Command Transmission via MQTT:

Upon inferring the speech intent, the Raspberry Pi sent the corresponding command to the IoT devices using the MQTT protocol. It published the command to the topic called *rpi/broadcast*, to which the ESP32 boards and the Wio Terminal subscribed. The Raspberry Pi also inferred the level of confidence with which the prediction was made. This was done to monitor the correct execution of only commnads with a reasonable accuracy. If the confidence level is less than 75%, a report of low confidence is issued to the broadcast and no commnad is sent or executed.

Wio Terminal Display:

The Wio Terminal served as a display device, providing a user-friendly interface for visualizing the executed commands. It received the command information via MQTT subscription to the *rpi/broadcast* and displayed it on its screen, providing real-time feedback to the user.

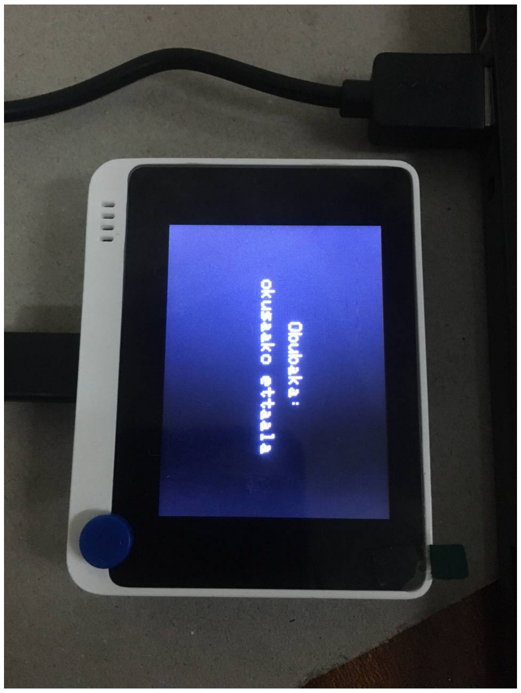


Figure 3.9: Wio Terminal display of inference

ESP32 Control of Devices:

To ensure seamless communication and control between the Raspberry Pi and the IoT devices, MQTT callbacks were employed. Each ESP32 board, acting as a microcontroller, utilized a unique client-id for identification on the network and subscribed to the *rpi/broadcast* topic. Through the implementation of callbacks, the boards were able to receive MQTT messages from the designated topic *rpi/broadcast*.
Upon receiving a command, the respective ESP32 board interpreted the message and executed the corresponding action on the associated IoT device. Each ESP32 board was assigned to control a set of devices to illustrate the concept of IoT in a home with different rooms. For instance, if the command was to turn on the lights, the ESP32 board responsible for controlling the lights device would execute the necessary action. This approach allowed for real-time responsiveness, ensuring that commands were promptly executed whenever they were received, enabling efficient and reliable control over the IoT devices.
To fully demonstrate the items in the smart home, a few simple demonstration devices were used for the proof of concept. These devices are shown in table 3.5 below. The

Device	**Demonstration device**
alarm	buzzer and red LED
door	servo motor
fan	dc motor
lights	solar bulb
camera	blue LED
fridge	yellow LED
television	red LED

Table 3.5: Demonstration devices for each smart home device

full system setup is shown in figure 3.10.
This integration of the Luganda speech intent classification system with the IoT devices made the smart home environment fully voice-controlled. The seamless coordination between the Raspberry Pi, Wio Terminal, and ESP32 boards allowed for the efficient and reliable execution of commands based on the interpreted speech intents.

3.4.4 System Evaluation

The Luganda speech intent classification system underwent a thorough evaluation to assess its performance and effectiveness. The evaluation process focused on various aspects of the system's functionality and behaviour. The following observations were made during the evaluation:

Accuracy and Confidence in Low-Noise Environments:

The model exhibited more accurate predictions and higher confidence levels when operating in low-noise environments. The absence of background noise facilitated clearer speech recognition and improved the overall performance of the system.

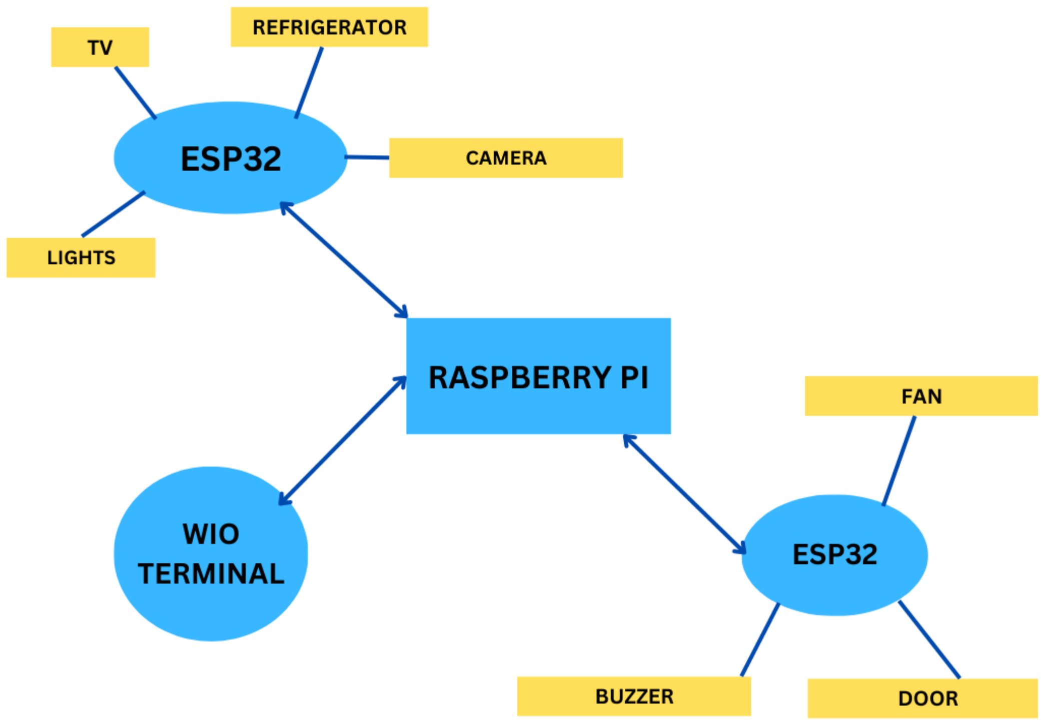


Figure 3.10: System Setup

Time Lag on First Prediction:

A slight time lag was observed in the first prediction made by the system. This can be attributed to the initialization and loading of the model upon system startup and usage of libraries for the first time. Subsequent predictions were executed smoothly without any noticeable delays. To solve this, a random recording was done on startup from which features were extracted and a prediction made which was not sent to the IoT connection.

Inference Performance on Audible Speakers:

The system demonstrated better inference and higher confidence levels when processing speech commands from audible speakers. Clear and audible speech inputs resulted in more accurate interpretations and reliable classification of speech intents.

Gender Fairness:

The system exhibited fairness in accuracy across different speakers' genders. It achieved comparable accuracy rates for both male and female speakers, ensuring gender neutrality in the classification of speech intents.

Low Confidence on Non-Fluent Speakers:

The system provided lower confidence levels and less accurate predictions when processing speech commands from non-fluent speakers. The variations in pronunciation, accent, or speech patterns among non-fluent speakers posed challenges for accurate classification. The model was however able to classify the intents of these speakers with a good confidence score.

The system evaluation provided valuable insights into the performance characteristics and limitations of the Luganda speech intent classification system. These findings can guide further improvements and refinements to enhance the system's accuracy, robustness, and adaptability in various real-world scenarios.

This chapter has presented a systematic approach to address the research objectives of this project. The chapter discussed the various stages involved in the development of the Luganda speech intent classification system, including data collection, preprocessing, feature extraction, model development, IoT integration and evaluation. In the upcoming chapter, we will analyze and interpret the experimental findings obtained from the data extraction and model development experiments. The results will be presented in a structured manner, providing insights into the performance of the system under different conditions and configurations.

Chapter 4

Results and Discussion

Introduction

This chapter presents the results and outcomes of the experimental investigations conducted in this project as discussed in the previous chapter. It offers a comprehensive overview of the findings obtained from the data extraction and model development experiments, focusing on the performance of the Luganda speech intent classification system. The section aims to provide a structured and organized presentation of the key results, highlighting the influence of various factors on the system's performance.

4.1 Luganda Voice Commands dataset

The figures 4.1 and 4.2show the distribution of intent classes in the part of the dataset separated for training and testing respectively.

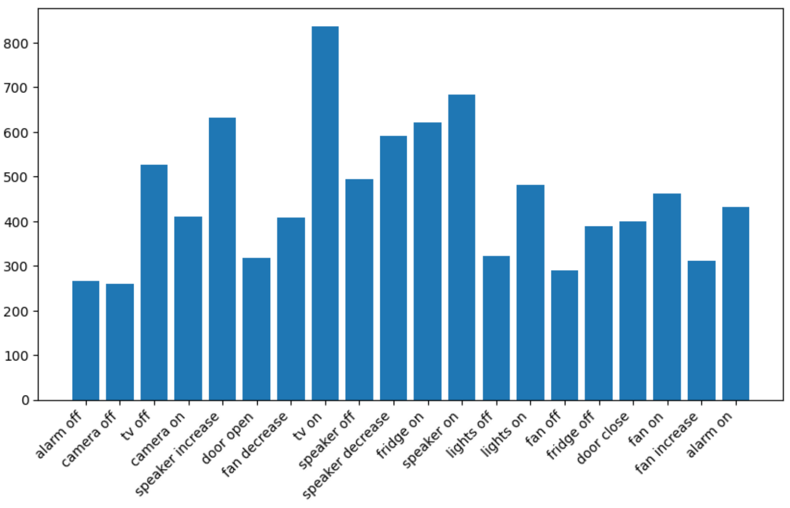


Figure 4.1: Distribution of classes in the Train dataset

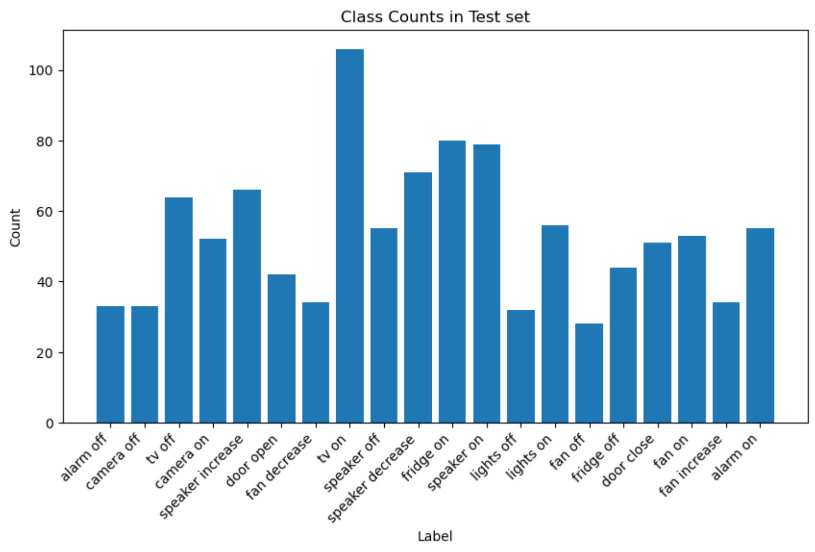


Figure 4.2: Distribution of classes in the Test dataset

4.2 Wio Terminal

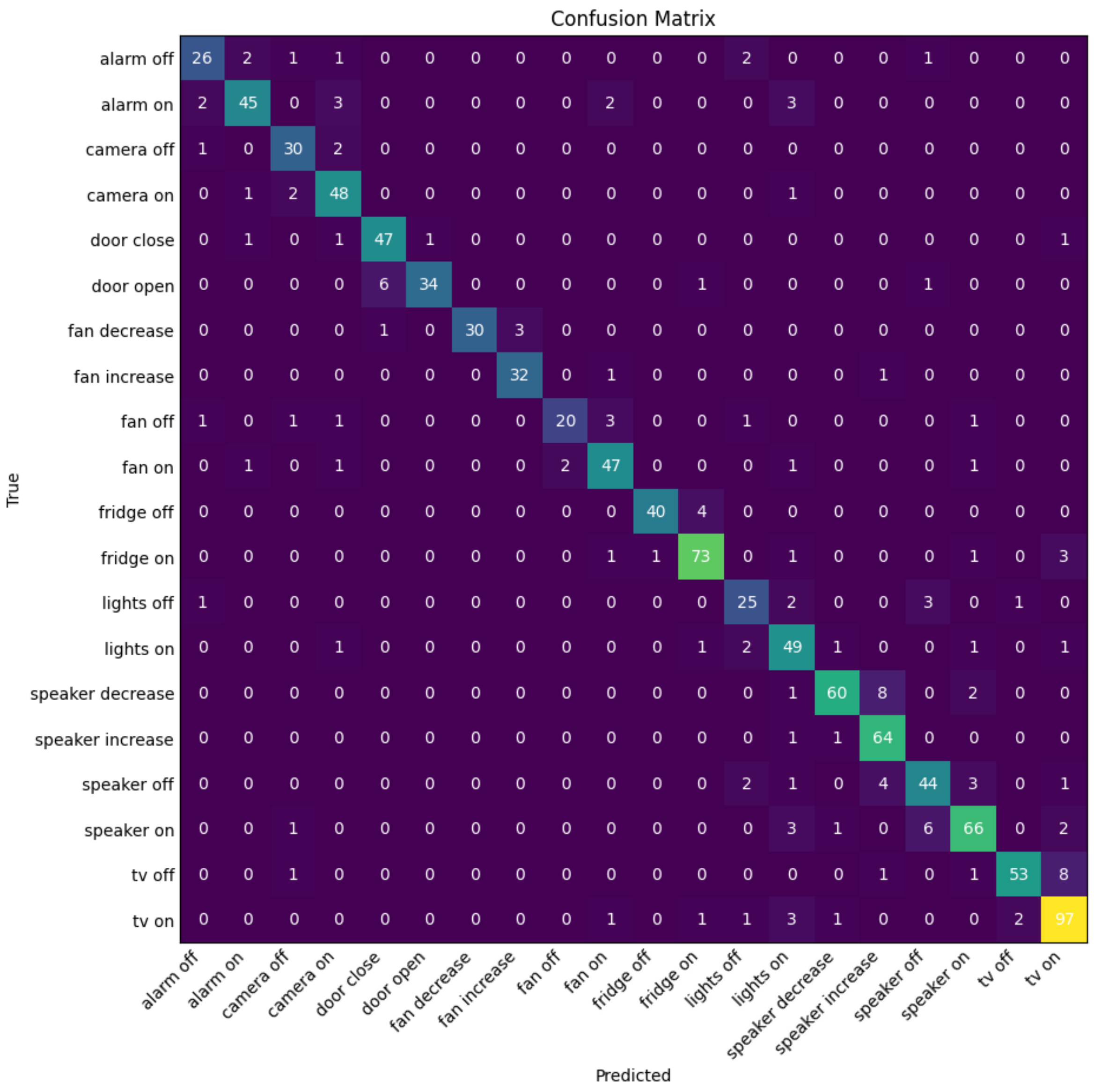
The methodology involved training the model with the intention of deploying it on the Wio Terminal involved a number of experiments as presented in section 3.3. The results of this training phase are presented in Table 4.1, which showcases the outcomes of Experiments A to D. To analyze the performance of the model, training and loss curves are displayed in Figures 4.4, 4.5, 4.6, and 4.7 for both the Global Max Pooling and Flattening layer configurations both with and without augmentation. Additionally, Table 4.2 provides insights into the results obtained after quantizing the model for deployment on the Wio Terminal. These findings contribute to a comprehensive evaluation of the model's suitability for the Wio Terminal deployment scenario discussed in the section 4.4.

Table 4.1: Results of Data Extraction and Model Development Experiments for the Wio Terminal

Exp	MFCCs	Augmentation	Architecture	Accuracy(%)	Loss	Parameters
A	10	Without	Global Max Pooling	84.64	0.6105	45,524
B	10	With	Global Max Pooling	87.08	0.5252	45,524
C	10	Without	Flattening	72.10	1.2226	324,052
D	10	With	Flattening	82.02	0.8250	324,052

4.2.1 Model Quantisation

The best model selected from Experiments A to D was quantized, and the table 4.5 below demonstrates the impact of different quantization techniques on the model size and accuracy of the model on the same set of test features.



True \ Predicted	alarm off	alarm on	camera off	camera on	door close	door open	fan decrease	fan increase	fan off	fan on	fridge off	fridge on	lights off	lights on	speaker decrease	speaker increase	speaker off	speaker on	tv off	tv on
alarm off	26	2	1	1	0	0	0	0	0	0	0	0	2	0	0	0	1	0	0	0
alarm on	2	45	0	3	0	0	0	0	0	2	0	0	0	3	0	0	0	0	0	0
camera off	1	0	30	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
camera on	0	1	2	48	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
door close	0	1	0	1	47	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
door open	0	0	0	0	6	34	0	0	0	0	0	1	0	0	0	0	1	0	0	0
fan decrease	0	0	0	0	1	0	30	3	0	0	0	0	0	0	0	0	0	0	0	0
fan increase	0	0	0	0	0	0	0	32	0	1	0	0	0	0	0	1	0	0	0	0
fan off	1	0	1	1	0	0	0	0	20	3	0	0	1	0	0	0	0	1	0	0
fan on	0	1	0	1	0	0	0	0	2	47	0	0	0	1	0	0	0	1	0	0
fridge off	0	0	0	0	0	0	0	0	0	0	40	4	0	0	0	0	0	0	0	0
fridge on	0	0	0	0	0	0	0	0	0	1	1	73	0	1	0	0	0	1	0	3
lights off	1	0	0	0	0	0	0	0	0	0	0	0	25	2	0	0	3	0	1	0
lights on	0	0	0	1	0	0	0	0	0	0	0	1	2	49	1	0	0	1	0	1
speaker decrease	0	0	0	0	0	0	0	0	0	0	0	0	0	1	60	8	0	2	0	0
speaker increase	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	64	0	0	0	0
speaker off	0	0	0	0	0	0	0	0	0	0	0	0	2	1	0	4	44	3	0	1
speaker on	0	0	1	0	0	0	0	0	0	0	0	0	0	3	1	0	6	66	0	2
tv off	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	53	8
tv on	0	0	0	0	0	0	0	0	0	1	0	1	1	3	1	0	0	0	2	97

Figure 4.3: Confusion Matrix for Experiment B

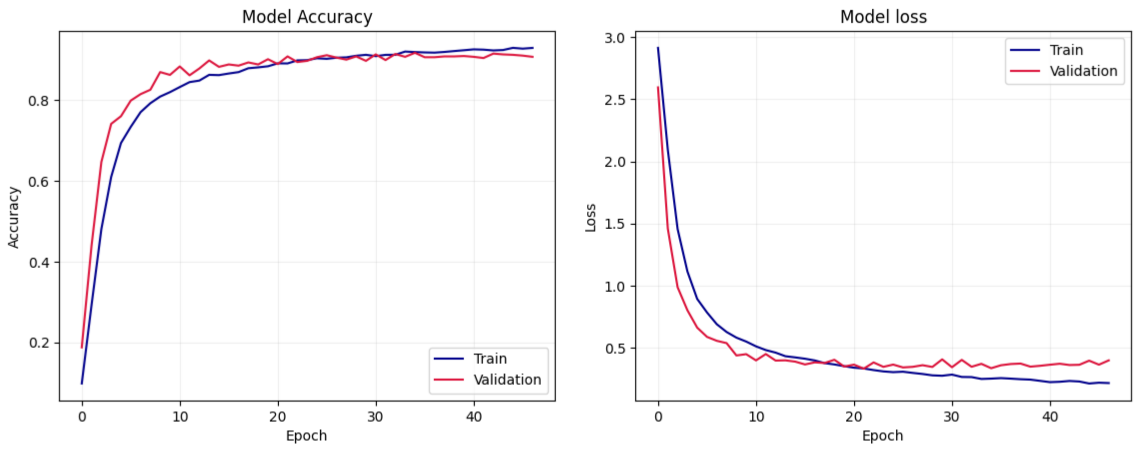


Figure 4.4: Training and loss Curves for Experiment A

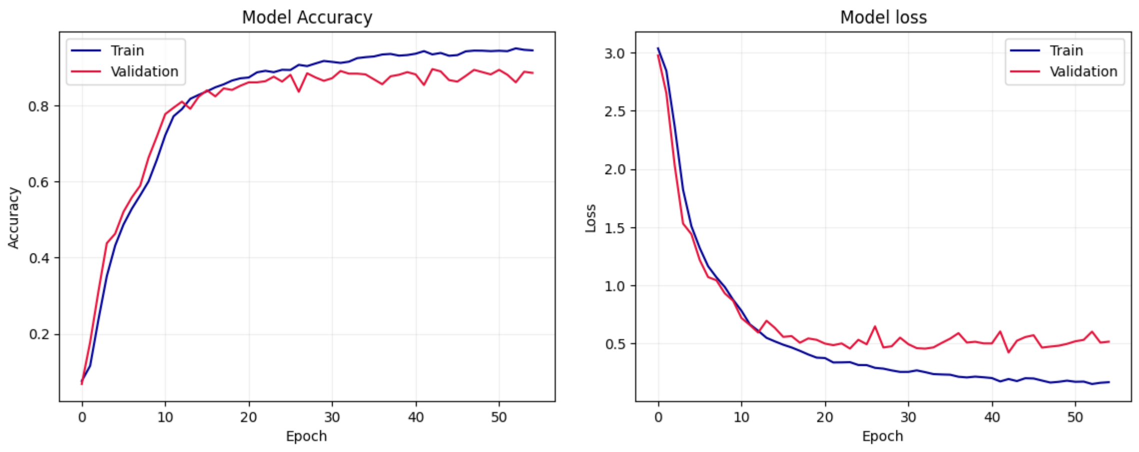


Figure 4.5: Training and loss Curves for Experiment B

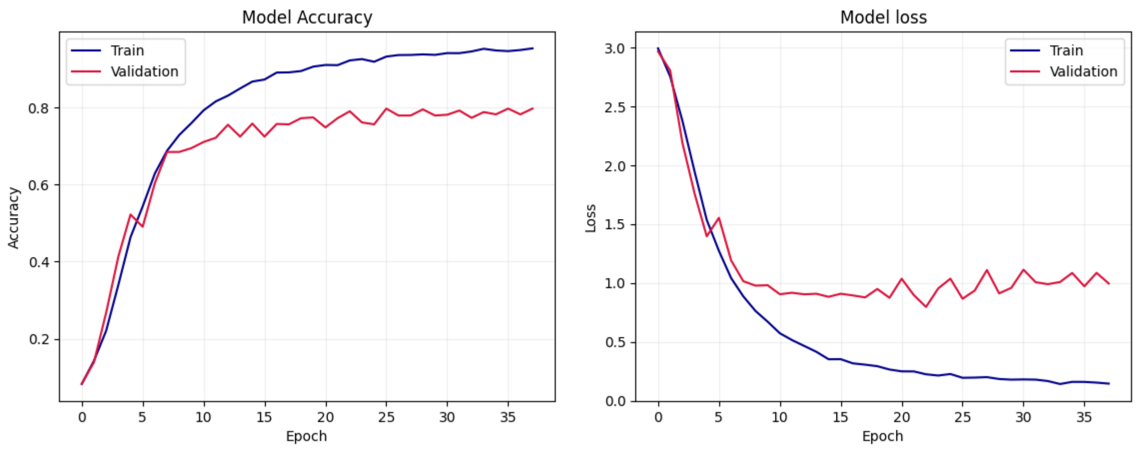


Figure 4.6: Training and loss Curves for Experiment C

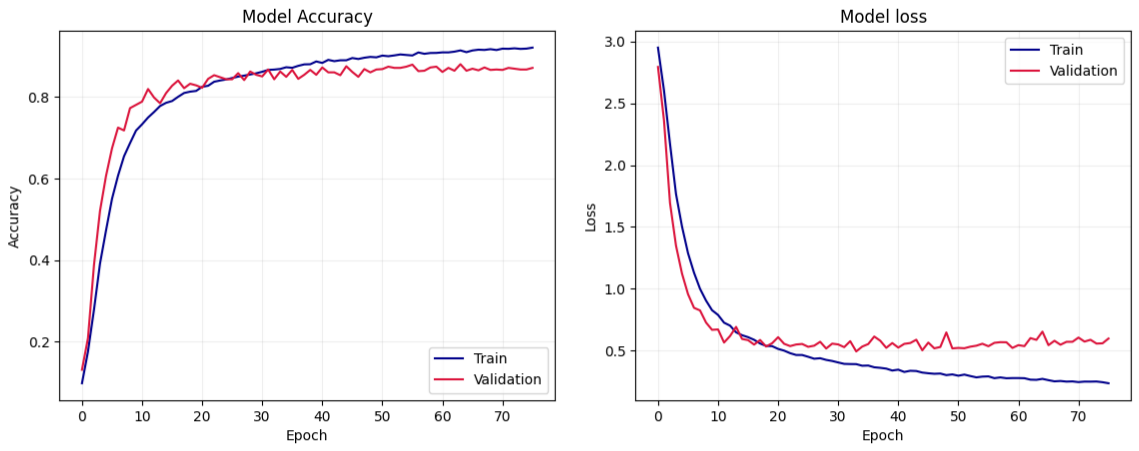


Figure 4.7: Training and loss Curves for Experiment D

Table 4.2: Results of Model Quantisation: Wio Terminal

Experiment	**size(KB)**	**Model Accuracy**	**Accuarcy Tradeoff**	**Reduction%**
.h5 model	654.75	87.08	0	0
Default	183.762	87.08	0	71.94
Float 16	99.051	87.08	0	84.88
Int 8	59.242	84.55	- 2.53	90.954
Size	58.703	87.36	+0.28	91.035

4.3 Raspberry Pi

The results from Model training for the Raspberry Pi were as shown below. The table 4.3 summarises Experiments 1 to 8. The results are further discussed in section 4.5. The training and loss curves for the model experiments with both the Global Max Pooling layer and Flattening layer both with and without data augmentation were as shown;

Exp.	MFCCs	Augmentation	Architecture	Accuracy(%)	Loss	Parameters
1	10	Without	Global Max Pooling	84.64	0.7150	45,524
2	10	With	Global Max Pooling	87.08	0.4709	45,524
3	10	Without	Flattening	80.15	0.7582	274,900
4	10	With	Flattening	80.43	0.9648	274,900
5	13	Without	Global Max Pooling	83.80	0.7144	45,524
6	13	With	Global Max Pooling	87.36	0.6486	45,524
7	13	Without	Flattening	83.05	0.8888	274,900
8	13	With	Flattening	80.99	1.1151	274,900

Table 4.3: Results of Data Extraction and Model Development Experiments: Raspberry Pi

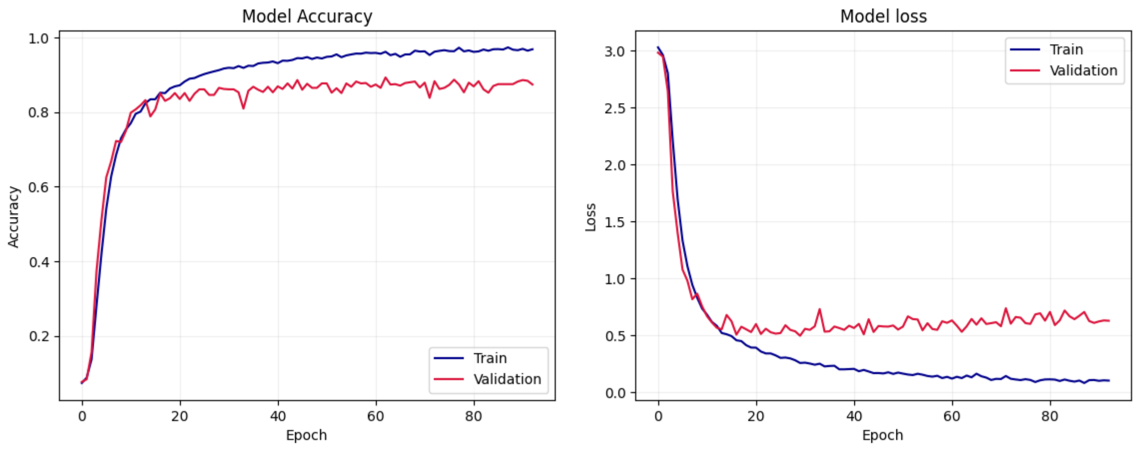


Figure 4.8: Training and loss curves for Experiment 1

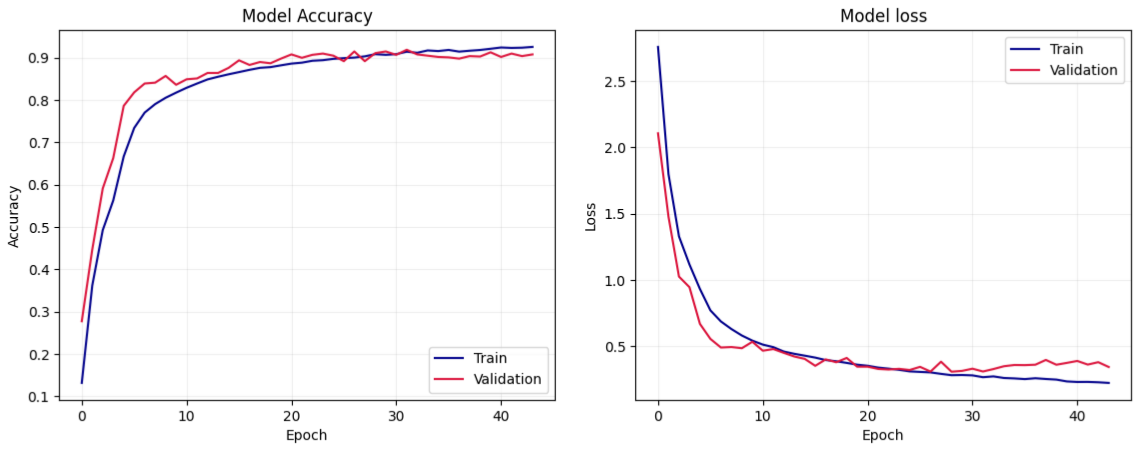


Figure 4.9: Training and loss curves for Experiment 2

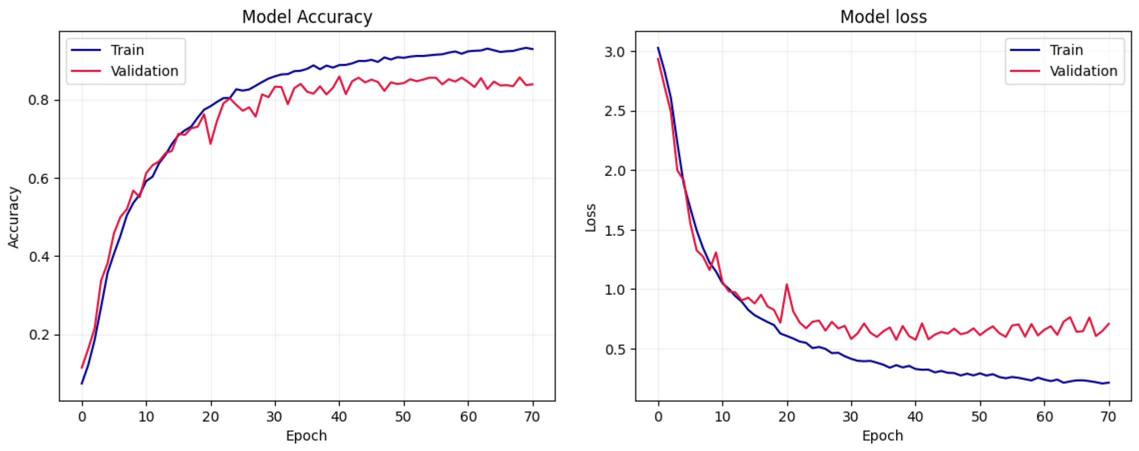


Figure 4.10: Training and loss curves for Experiment 3

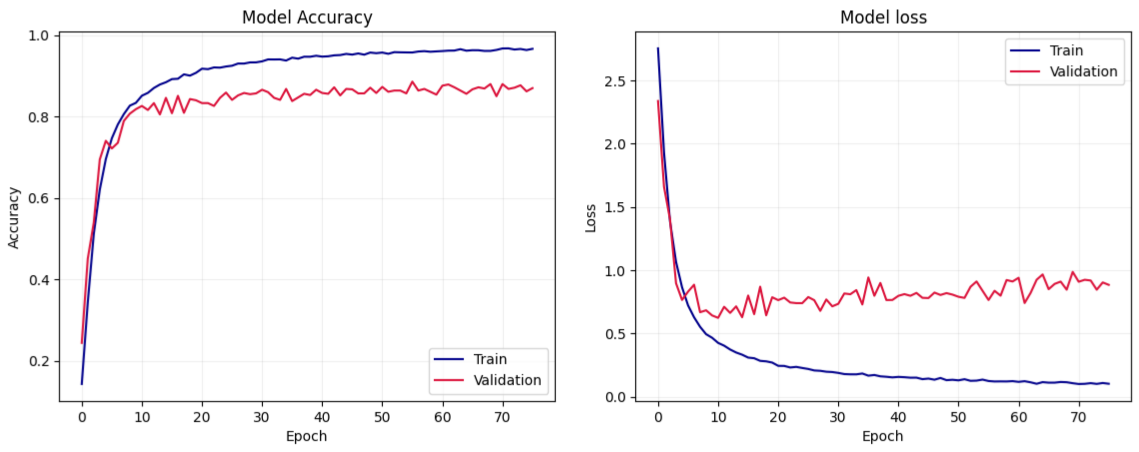


Figure 4.11: Training and loss curves for Experiment 4

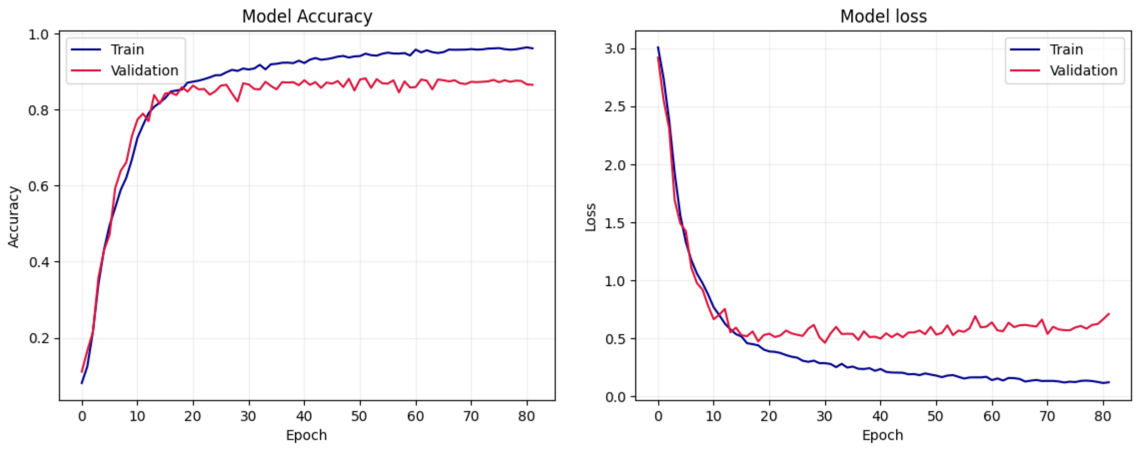


Figure 4.12: Training and loss curves for Experiment 5

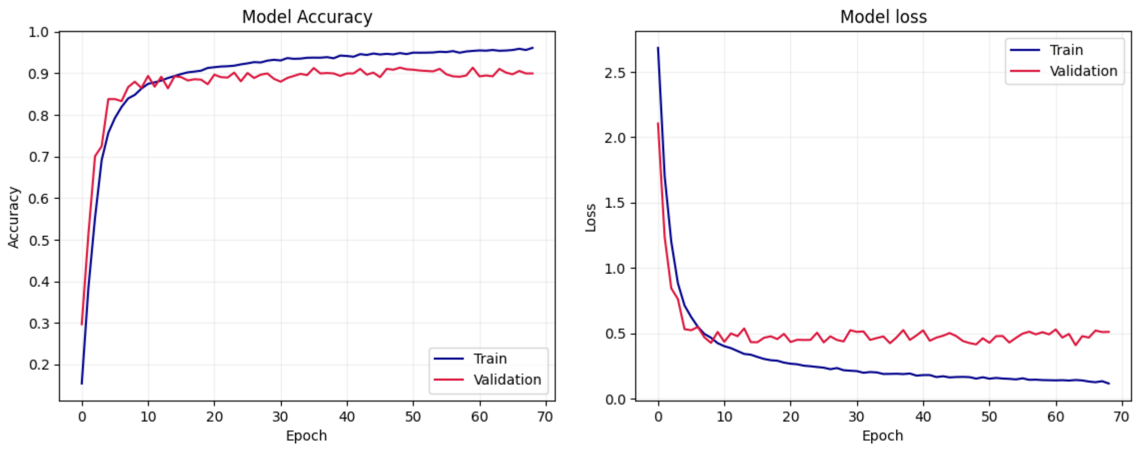


Figure 4.13: Training and loss curves for Experiment 6

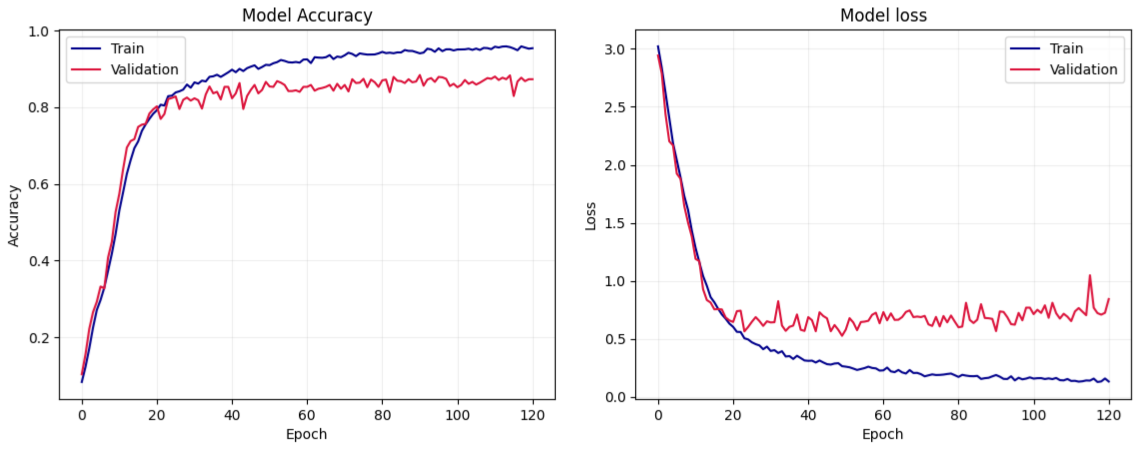


Figure 4.14: Training and loss curves for Experiment 7

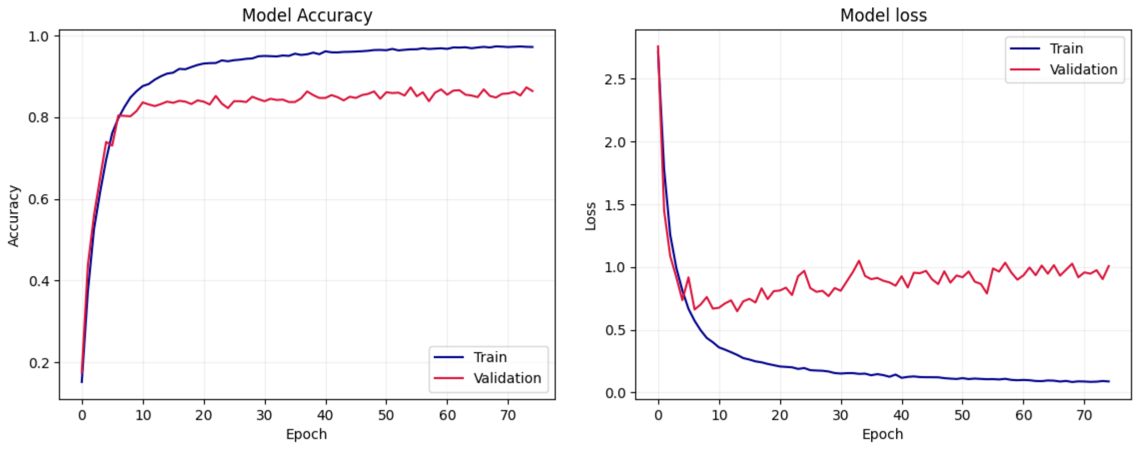
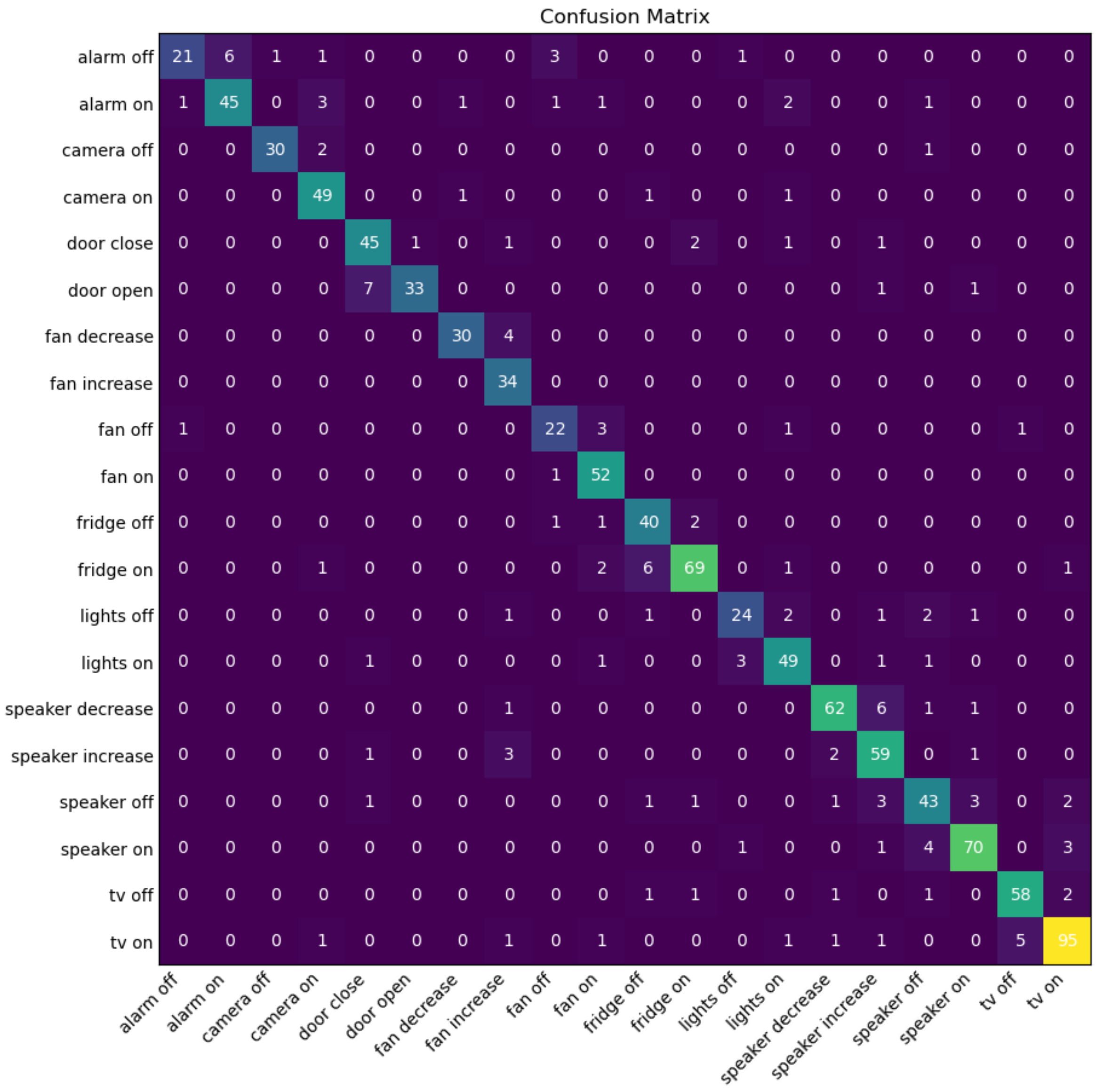


Figure 4.15: Training and loss curves for Experiment 8

Confusion Matrices for the best experiments on the Test sets



Confusion Matrix

	alarm off	alarm on	camera off	camera on	door close	door open	fan decrease	fan increase	fan off	fan on	fridge off	fridge on	lights off	lights on	speaker decrease	speaker increase	speaker off	speaker on	tv off	tv on
alarm off	21	6	1	1	0	0	0	0	3	0	0	0	1	0	0	0	0	0	0	0
alarm on	1	45	0	3	0	0	1	0	1	1	0	0	0	2	0	0	1	0	0	0
camera off	0	0	30	2	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
camera on	0	0	0	49	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0
door close	0	0	0	0	45	1	0	1	0	0	0	2	0	1	0	1	0	0	0	0
door open	0	0	0	0	7	33	0	0	0	0	0	0	0	0	0	1	0	1	0	0
fan decrease	0	0	0	0	0	0	30	4	0	0	0	0	0	0	0	0	0	0	0	0
fan increase	0	0	0	0	0	0	0	34	0	0	0	0	0	0	0	0	0	0	0	0
fan off	1	0	0	0	0	0	0	0	22	3	0	0	0	1	0	0	0	0	1	0
fan on	0	0	0	0	0	0	0	0	1	52	0	0	0	0	0	0	0	0	0	0
fridge off	0	0	0	0	0	0	0	0	1	1	40	2	0	0	0	0	0	0	0	0
fridge on	0	0	0	1	0	0	0	0	0	2	6	69	0	1	0	0	0	0	0	1
lights off	0	0	0	0	0	0	0	1	0	0	1	0	24	2	0	1	2	1	0	0
lights on	0	0	0	0	1	0	0	0	0	1	0	0	3	49	0	1	1	0	0	0
speaker decrease	0	0	0	0	0	0	0	1	0	0	0	0	0	0	62	6	1	1	0	0
speaker increase	0	0	0	0	1	0	0	3	0	0	0	0	0	0	2	59	0	1	0	0
speaker off	0	0	0	0	1	0	0	0	0	0	1	1	0	0	1	3	43	3	0	2
speaker on	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	4	70	0	3
tv off	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	58	2
tv on	0	0	0	1	0	0	0	1	0	1	0	0	0	1	1	1	0	0	5	95

Figure 4.16: Confusion matrix for Experiment 2

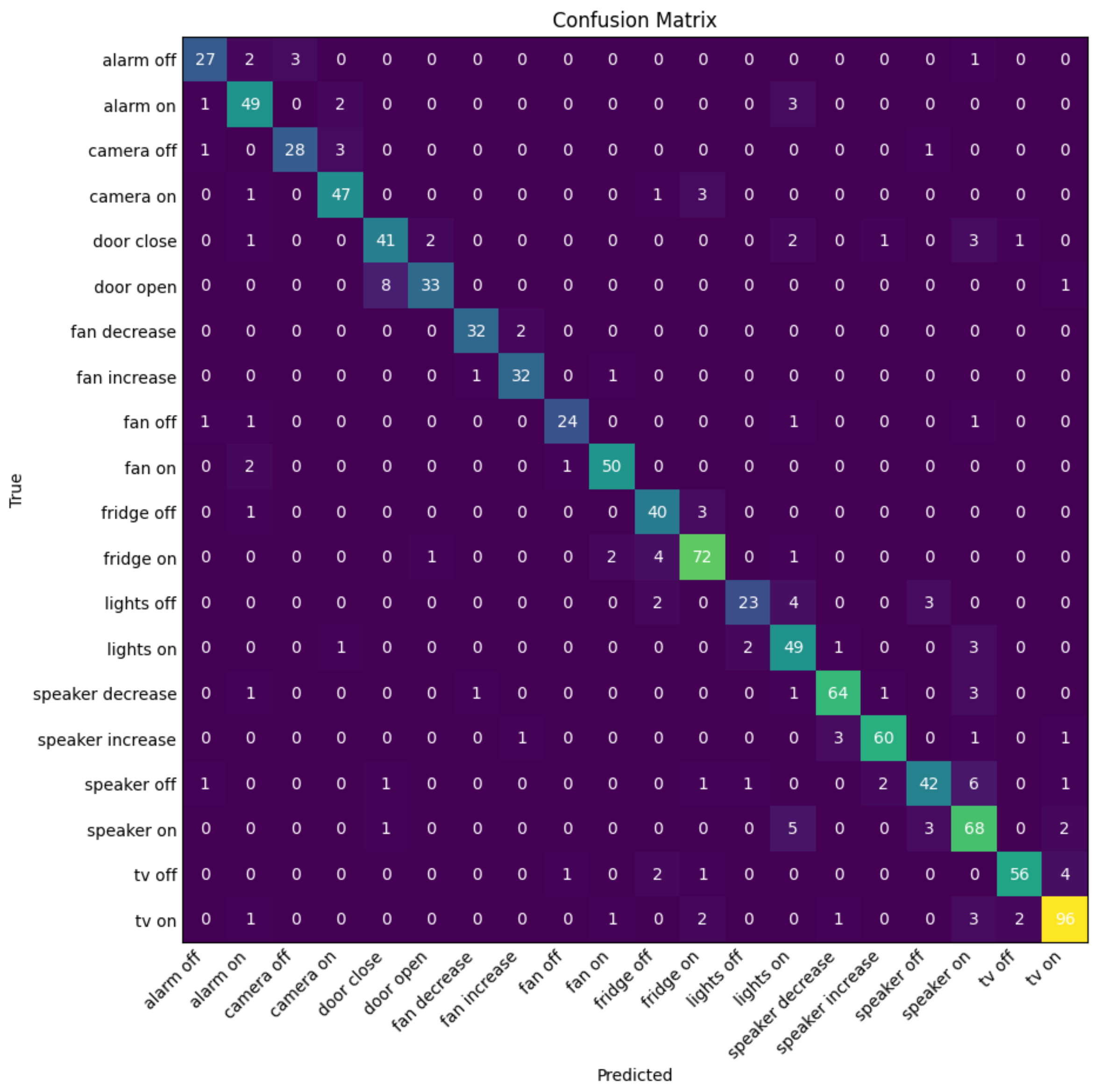


Figure 4.17: Confusion matrix for Experiment 6

4.3.1 Slot filling

The performance of the model for multiple slot outputs is presented in the following section. Table 4.4 provides the Classification Accuracy and Loss values for the Action and Object slots of the model. The training and loss curves based on the Object and Action Slots are depicted in Figure 4.18 and 4.19, respectively. Furthermore, the confusion matrices and Classification Reports of the slots can be found in Figure 4.20 and 4.21. The results highlight the accuracy and loss metrics achieved for each slot, shedding light on the effectiveness of the model's performance in slot filling.

Slot	Accuracy	Loss
Action	94.29	0.2828
Object	91.85	0.3658

Table 4.4: Performance of Slot Filling

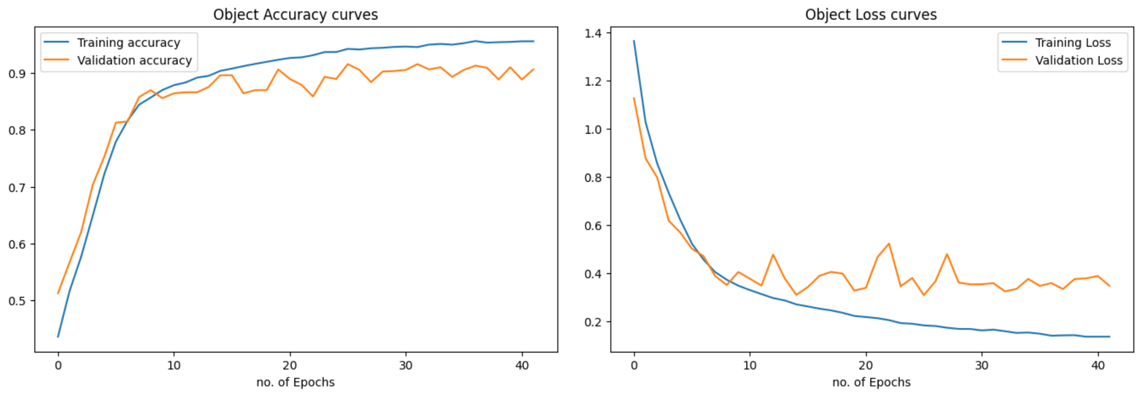


Figure 4.18: Training and loss curves for Object output

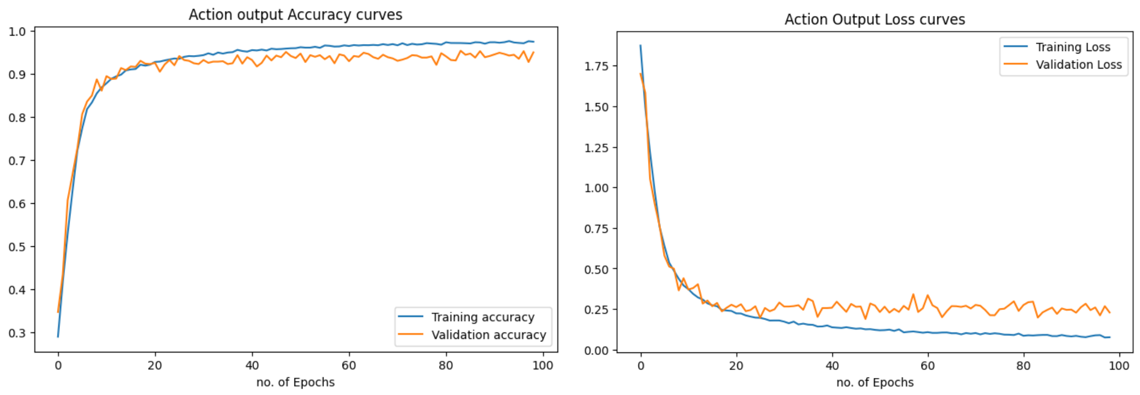


Figure 4.19: Training and loss curves for the Action output slot

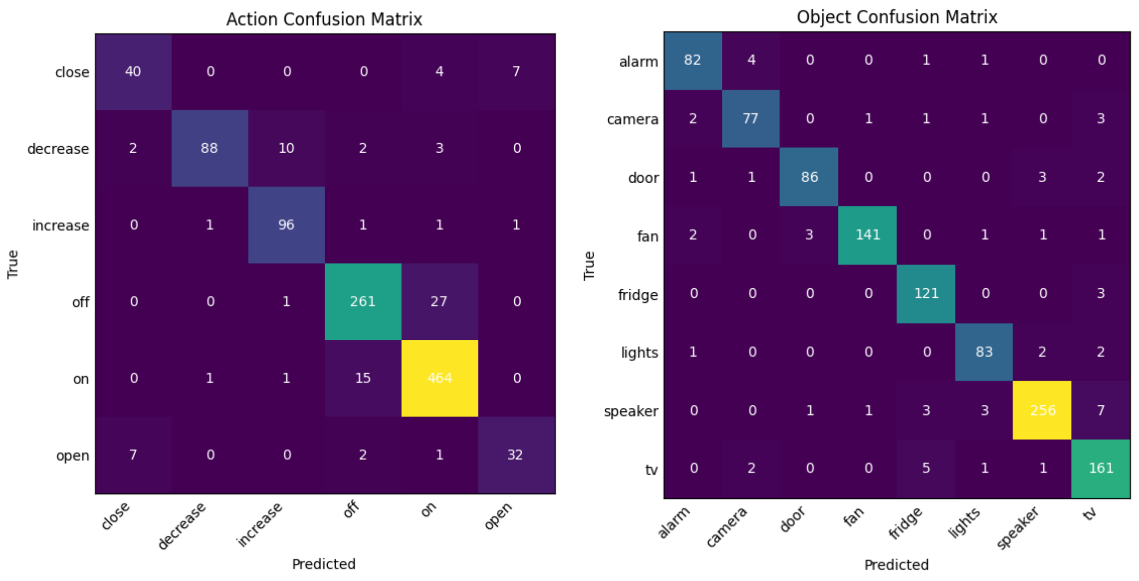
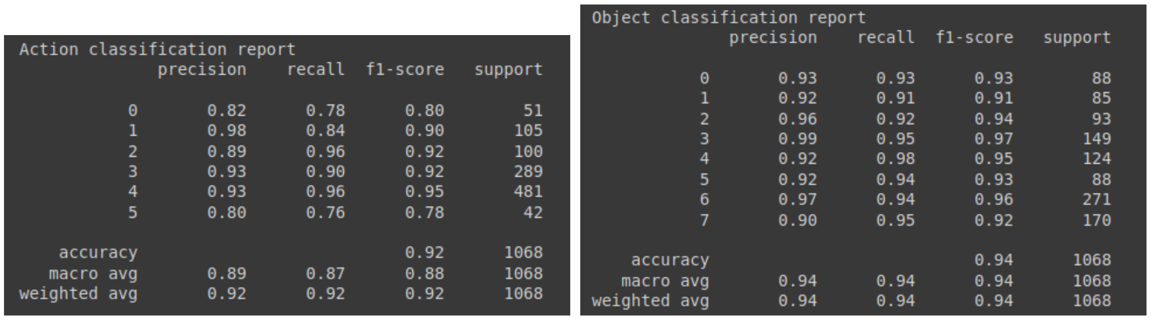


Figure 4.20: Action and Object Classification Matrices



```
Action classification report
              precision    recall  f1-score   support

           0       0.82      0.78      0.80        51
           1       0.98      0.84      0.90       105
           2       0.89      0.96      0.92       100
           3       0.93      0.90      0.92       289
           4       0.93      0.96      0.95       481
           5       0.80      0.76      0.78        42

    accuracy                           0.92      1068
   macro avg       0.89      0.87      0.88      1068
weighted avg       0.92      0.92      0.92      1068
```

```
Object classification report
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        88
           1       0.92      0.91      0.91        85
           2       0.96      0.92      0.94        93
           3       0.99      0.95      0.97       149
           4       0.92      0.98      0.95       124
           5       0.92      0.94      0.93        88
           6       0.97      0.94      0.96       271
           7       0.90      0.95      0.92       170

    accuracy                           0.94      1068
   macro avg       0.94      0.94      0.94      1068
weighted avg       0.94      0.94      0.94      1068
```

Figure 4.21: Action and Object Classification Reports

4.3.2 Model Quantisation

The best model as discussed in section 4.5.1 was chosen for quantization, and the table 4.5 below demonstrates the impact of different quantization techniques on size and accuracy on the same set of test features.

Table 4.5: Results of Model Quantisation for Raspberry Pi

Experiment	Model size(KB)	Accuracy	Accuracy change	Reduction%
.h5 model	647.5	87.36	0	0
Default	183.859	87.36	0	71.60
Float 16	99.211	87.36	0	84.67
Size	58.805	87.73	+0.37	90.91

4.3.3 Final Prototype

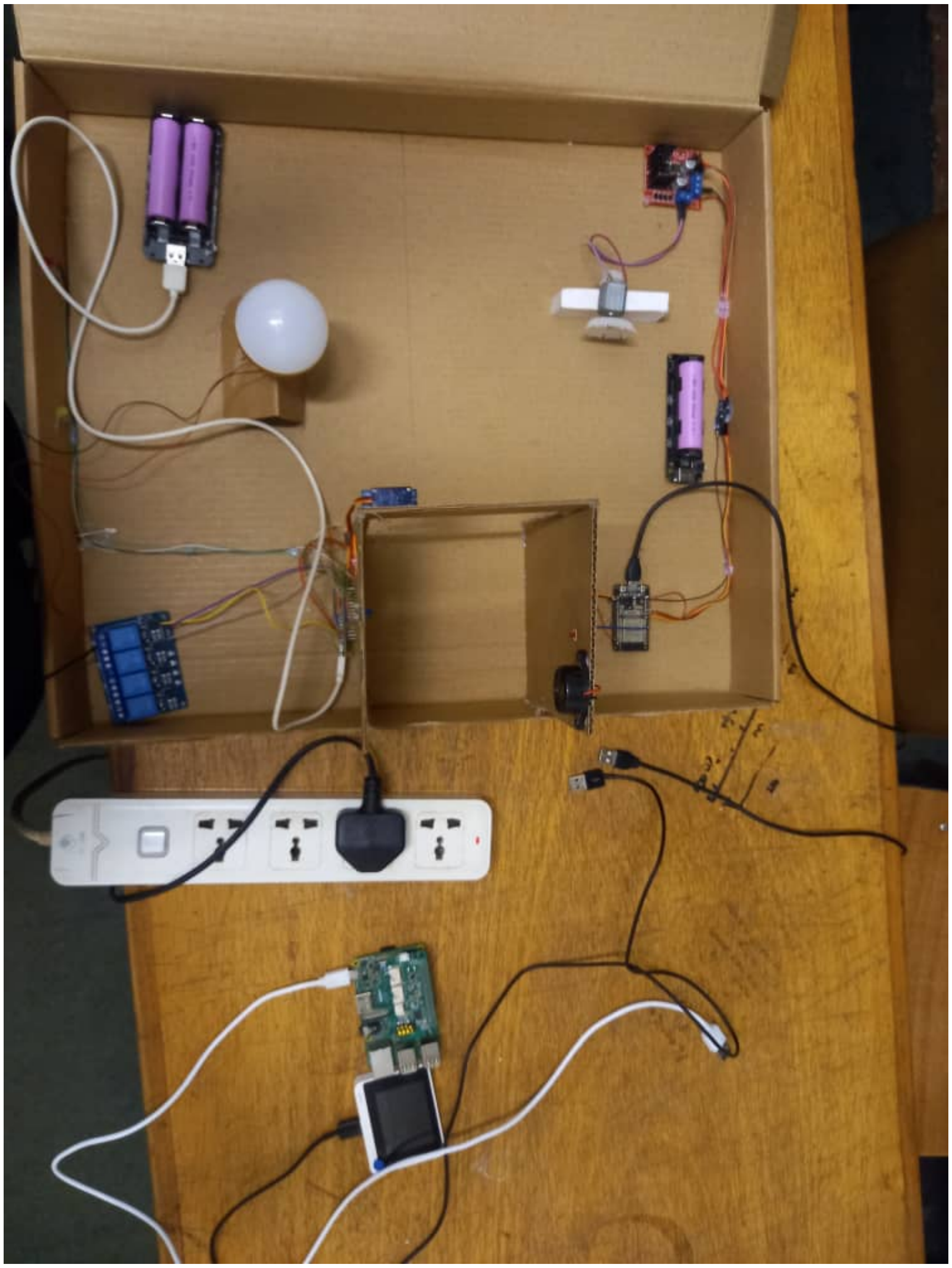


Figure 4.22: Final System Prototype

Discussion

In this section, we delve into the decisions made during the project based on the presented results in the preceding section. These decisions were carefully considered to optimize the performance of the model and improve its suitability for deployment on edge devices, specifically the Raspberry Pi and Wio terminal.

4.4 Wio Terminal

Based on the results presented in Table 4.1, Experiment B, which involved the use of augmentation with 10 MFCCs and the Global Max Pooling architecture, achieved the highest accuracy of 87.08% compared to the other experiments. Additionally, Experiment B had a relatively low loss value of 0.5252, indicating better model performance and convergence.
The decision to take the model from Experiment B ahead for quantisation was based on several factors. Firstly, the higher accuracy achieved in Experiment B suggests that the model trained with augmentation and the Global Max Pooling architecture was able to effectively capture and generalize the patterns in the data, resulting in more accurate predictions. This indicates that the model has the potential to perform well in real-world scenarios.
Secondly, the relatively low loss value in Experiment B further supports the effectiveness of the model. A lower loss indicates that the model has minimized the discrepancy between the predicted and actual outputs, indicating a better fit to the training data. Considering these factors, the model from Experiment B was chosen for quantisation as it demonstrated superior performance in terms of accuracy and loss compared to the other experiments. By selecting this model, it is expected that the quantised version will preserve the high accuracy achieved during training while reducing the model size and computational requirements, enabling its deployment on resource-constrained devices or systems.
It was also observed that Experiment C and Experiment D, which involved the Flattening architecture, had lower accuracies compared to Experiment B. This suggests that the Global Max Pooling architecture was more suitable for the given task and dataset, leading to better classification performance. Therefore, Experiment B was considered the most promising approach to further optimize and deploy the model for quantisation.

4.4.1 Model choice for deployment on the Wio Terminal

The results of the model quantization experiments conducted on the Wio Terminal, as presented in Table 4.2, provide valuable insights into the tradeoff between mainly model size and accuracy. On the Wio Terminal with limited requirements, Size is very important but good accuracy is also needed.
The baseline model, saved in the .h5 format, has a size of 654.75 KB and an accuracy of 87.08%. This serves as a reference point for evaluating the subsequent quantization experiments.
Applying the default quantization method resulted in a model size reduction of 71.94%, with a model size of 183.762 KB, while maintaining the same accuracy of 87.08%. This indicates that significant compression can be achieved without sacrificing model performance.
The float16 quantization method further reduced the model size to 99.051 KB, achieving a model reduction of 84.88%. Remarkably, the model accuracy remained consistent at 87.08%.
Introducing int8 quantization resulted in a model size of 59.242 KB, a reduction of 90.954% compared to the baseline model. However, there was a slight decrease in accuracy to 84.55%, reflecting a tradeoff of -2.53%. Nonetheless, this reduction in model size can be advantageous for resource-constrained devices.
The last experiment focused on optimizing the model for size. With a model size of 58.703 KB, there was a marginal improvement in accuracy to 87.36% (+0.28%) compared to the baseline model. The resulting model reduction achieved in this experiment was 91.035%.
These results demonstrate the efficacy of model quantization techniques in significantly reducing the model size while maintaining acceptable levels of accuracy.
The int8 quantization method proved to be the most effective in terms of compression and complexity, making it a suitable choice for deploying the model on the Wio Terminal. This approach significantly reduced the model size, optimizing the utilization of the limited resources available on the device.
Furthermore, the int8 quantization method allowed for the conversion of model inputs, such as the MFCC features, to the int8 data type making it a preferred choice over the model optimized for size. This ensured efficient resource management, enabling the Wio Terminal to handle the computational requirements while minimizing memory usage.

Therefore the int8 model was deployed on the Wio Terminal.

4.5 Raspberry Pi

In the conducted experiments, two different architectural choices were explored: models with Global Max Pooling and models with Flattening layers. The Global Max Pooling layer condenses the spatial dimensions of the input feature maps by selecting the maximum value within each region, reducing the overall parameter count. On the other hand, the Flattening layer reshapes the input into a one-dimensional vector, retaining

all the spatial information.
Comparing the results, it can be observed that models with Global Max Pooling consistently achieved higher accuracy compared to models with Flattening layers. For instance, Experiment 2 with Global Max Pooling achieved an accuracy of 87.08% while Experiment 4 with Flattening achieved an accuracy of 80.43%. This suggests that the Global Max Pooling layer effectively captured the salient features for classification, resulting in improved performance.
Regarding the effects of augmentation, it can be noted that the inclusion of data augmentation techniques generally led to improved accuracy across the experiments. For instance, Experiment 2 with augmentation achieved an accuracy of 87.08%, whereas Experiment 1 without augmentation achieved an accuracy of 84.64% on the same set of features. Similarly, Experiment 6 with augmentation achieved an accuracy of 87.36%, while Experiment 5 without augmentation achieved an accuracy of 83.80%. This indicates that data augmentation helped in enhancing the model's ability to generalize and classify intent more accurately.
Additionally, the number of MFCCs extracted also had an impact on the classification performance. Comparing experiments with 10 MFCCs (Experiments 1-4) and experiments with 13 MFCCs (Experiments 5-8), it can be observed that models utilizing 13 MFCCs achieved slightly higher accuracies in most cases. For example, Experiment 6 with 13 MFCCs and augmentation achieved an accuracy of 87.36%, whereas Experiment 2 with 10 MFCCs and augmentation achieved an accuracy of 87.08%. This suggests that extracting a larger number of MFCCs provided more discriminative information for the intent classification task, contributing to improved accuracy.
Based on the comparison between architectural choices and the observation on augmentation, it can be inferred that models utilizing Global Max Pooling and data augmentation exhibited superior accuracy. Additionally, the number of MFCCs extracted also played a significant role in the classification performance. These findings highlight the effectiveness of Global Max Pooling, data augmentation, and the appropriate selection of the number of MFCCs in improving the classification capabilities of the models.
This knowledge was borrowed for slot filling and the Global Max Pooling layer and 13 MFCCs with data augmentation were utilized to capture the slot information. The results obtained from the slot-filling experiments are analyzed and discussed in this section. Table 4.4 presents the performance metrics, including accuracy and loss, for the Action and Object slots.
For the Action slot, the model achieved an accuracy of 94.29% with a corresponding loss of 0.2828. This indicates that the model successfully classified and predicted the actions with a high degree of accuracy.
Similarly, for the Object slot, the model achieved an accuracy of 91.85% with a loss value of 0.3658. This demonstrates the model's ability to accurately identify and classify the objects within the input data.

This model from Experiment 6 and that based on slot-filling were selected for more analysis.

4.5.1 Model choice for quantization

From the results presented in Table 4.3, it can be observed that Experiment 6, which involved using 13 MFCCs with data augmentation and global max pooling architecture, achieved a relatively high accuracy of 87.36% and a low loss value of 0.6486. This combination of configuration parameters yielded promising performance in terms of speech recognition accuracy.
Slot filling achieved good classification accuracy on the action and output slots with significantly low loss values as discussed in Section 4.3.1. The results are displayed in Table 4.4 and Figures 4.20 and 4.21. It can therefore be inferred that these 2 models outperformed the rest of the models from the experiments in model development for deployment on the Raspberry Pi and a detailed analysis of the 2 experiments was done to choose the best model for quantisation.
In real-time, slot filling can however introduce new combinations of intents when using multiple slot outputs. For example, *action* and *object* slot outputs could be inferred as 'lights' and *'decrease'* respectively creating a new intent *"lights decrease"* that never existed in the original dataset and is not catered for in hardware integration. This error can be reduced by introducing confidence levels. However, there are also new complications in setting confidence levels for each slot, especially in real-time scenarios where inconsistencies between slot confidence levels may arise. A scenario where model confidence of one slot may pass the threshold while the other fails could repeatedly happen between the 2 slots leading to user frustration.
Experiment 6, on the other hand, employed a single output approach, focusing on the overall classification of the intent. It notably achieved a relatively high accuracy of 87.36% and a low loss value of 0.6486 as discussed earlier. Although Experiment 6 exhibited a lower overall accuracy in classifying the intent compared to slot filling, it offered advantages in terms of monitoring and managing confidence levels. Treating the intent as a unified entity allows the confidence levels to be effectively monitored, ensuring greater consistency and reliability in real-time applications. There was also no chance of creating new intents at deployment since there was a clear and well-defined set of intents that could be obtained from the model output.
Considering these factors, Experiment 6 emerged as a suitable candidate for further analysis in the context of model quantization. Its strong performance in terms of accuracy and loss made it a representative case to explore the impact of reducing model precision on the overall system. By quantizing Experiment 6, we aimed to evaluate the impact of reducing the model's precision on its performance. This experiment allowed us to assess the trade-off between model size and accuracy, as well as to investigate the feasibility of deploying a more resource-efficient model without significant degradation in recognition performance. The results for quantization are discussed in the next subsection 4.5.2.

4.5.2 Model Choice for Deployment

After evaluating the results of model quantization on the Raspberry Pi platform, we can make an informed decision regarding the best model for deployment. Table 4.5

provides an overview of the quantization experiments performed on the model from Experiment 6, including the model size in kilobytes (KB), accuracy, accuracy change compared to the original saved .h5 model, and the percentage reduction in size.
The quantization experiments aimed to improve the speed of inference on the Raspberry Pi by reducing the model size and leveraging the platform's computational resources efficiently. While size reduction was a factor in the quantization process, the primary focus was on achieving faster inference without sacrificing accuracy.
The default quantization technique successfully reduced the model size to 183.859 KB, maintaining the original accuracy of 87.36%. This indicates that the default quantization method effectively compressed the model while preserving its performance.
Additionally, the Float16 quantization technique further reduced the model size to 99.211 KB, with no loss in accuracy. By utilizing 16-bit floating-point representation, this technique aimed to strike a balance between model size reduction and precision.
Notably, the size optimization technique resulted in the smallest model size of 58.805 KB, representing a 90.91% reduction compared to the original model. More importantly, this technique improved the accuracy slightly to 87.73%, demonstrating a positive impact on model performance.
Considering the objectives of deploying the model on the Raspberry Pi platform, where computational resources and memory constraints were present in making predictions in real-time, the quantized model obtained through the size optimization technique is recommended since a balance was achieved between model size reduction and accuracy improvement which makes it the most suitable choice for deployment.
This model was deployed on the Raspberry Pi for real-time inference. This ensured efficient and reliable inference while maintaining a reasonable level of accuracy.

Summary

In summary, the following decisions were noted. The utilization of Global Max Pooling yielded superior results for model development on the Raspberry Pi and Wio terminal. This pooling technique helped in reducing the number of parameters while maintaining good accuracy. It outperformed the Flattening layer, which suggests that the Global Max Pooling layer is more suitable for the given context.
Another conclusion drawn was that utilizing 13 MFCCs led to better accuracy as compared to using 10 MFCCs. This implies that a higher number of MFCCs captures more relevant information from the audio signals, resulting in improved model performance.
Another important finding was the effectiveness of quantization in reducing the model size and enabling faster inference on edge devices.This reduction in size is particularly crucial for edge devices with limited computational resources and storage capacity, such as the Raspberry Pi and Wio terminal.
By incorporating Global Max Pooling, optimizing MFCCs, and leveraging quantization, researchers and developers can achieve more accurate and efficient speech intent classification on edge devices, unlocking new possibilities for IoT applications in the Luganda language.

In conclusion, this chapter provides a comprehensive overview of the key findings and discussions for the choices taken throughout the project. The chapter highlights the performance of various models and techniques employed in the development of the Luganda speech intent recognition system for IoT applications. In the next chapter, we will discuss the limitations of the project and the recommendations for improvement to the research.

Chapter 5

Limitations, Recommendations and Conclusion

5.1 Limitations

While this project has achieved significant progress in Luganda speech intent recognition for IoT applications, it is important to acknowledge certain limitations that were encountered during the course of the research. These limitations include:

Limited Dataset Size:

The availability of a limited dataset for Luganda speech commands posed a challenge in training and evaluating the models. The small dataset size may have an impact on the generalization and robustness of the models.

Limited Hyperparameter Tuning:

Due to time and resource constraints, the hyperparameter tuning process for the models was limited. Optimal hyperparameter selection plays a crucial role in model performance, and further exploration in this area could potentially yield improved results.

Absence of Testing on Children and other age groups:

The evaluation of the system did not include testing on children. As speech patterns and language usage may differ among different age groups, the performance of the system specifically for children and other age groups outside 20-25 years remains unexplored.

Absence of a Hot Wake Word:

The current implementation did not include a hot wake word functionality, which allows continuous listening for voice commands without the need for a button. Integrating a hot wake word feature could enhance the user experience by providing a more seamless and hands-free interaction.

Hardware Constraints:

The project was developed and tested on limited hardware resources, specifically the Raspberry Pi and Wio Terminal. While efforts were made to optimize the models and ensure efficient performance, the hardware limitations imposed certain restrictions on the system's capabilities.

Despite these limitations, this project serves as a valuable foundation for further advancements in Luganda speech intent recognition and demonstrates the potential for leveraging natural language processing techniques in IoT applications.

5.2 Recommendations

Future research should focus on expanding the dataset for Luganda speech intent classification by collecting a larger and more diverse set of speech samples. This would help improve the generalizability of the models and enhance the reliability of the findings.
Researchers should also explore alternative feature representations beyond MFCCs to capture both acoustic and linguistic aspects of Luganda speech. This could involve investigating spectrograms, word embeddings, linguistic features, or even adopting end-to-end deep learning approaches. A detailed comparison and combination of these different feature representations may lead to improved performance.
An in-depth hyperparameter tuning should also be performed to optimize the architecture and hyperparameters of the classification models. This involves exploring different network architectures, activation functions, regularization techniques, and optimization algorithms.
It is also recommended to conduct further experiments to fine-tune the preprocessing parameters such as window size, hop length, and frequency range. This analysis would ensure that the extracted features preserve important information and minimize the introduction of noise.
Future researchers should also consider exploring the efficacy of the speech-to-text-to-intent technique for classifying Luganda speech intents. Additionally, the integration of a high-quality text-to-speech model into the system can enhance the user experience by providing audio feedback.
Furthermore, it is recommended that the inclusion of a wake word functionality be considered in future enhancements. The integration of a wake word feature can provide a more seamless and user-friendly experience by allowing users to initiate voice commands by simply speaking the designated wake word. This addition can further streamline the interaction between users and the system, enhancing overall usability.
Collectively, these improvements can significantly enhance the Luganda Speech Intent Classification for IoT applications.

5.3 Conclusion

In this project, we have undertaken a comprehensive exploration of various aspects of speech intent recognition for IoT applications in the Luganda language. We have made significant progress in dataset creation, feature extraction, model development, quantization, deployment, IoT integration, and system evaluation .
First, we curated a Luganda voice command dataset, enabling us to train and evaluate our speech-to-intent models effectively. The dataset played a crucial role in capturing the linguistic characteristics and context-specific commands necessary for accurate intent recognition.
Next, we employed Mel-Frequency Cepstral Coefficients (MFCCs) as input features, capturing essential acoustic information for our models. This feature extraction technique proved to be effective in representing speech signals and contributed to the overall performance of our models.
Through rigorous model development, we explored various architectures, including Convolutional Neural Networks (CNNs), to achieve high accuracy in speech intent recognition. We also implemented data augmentation techniques to improve the robustness and generalization capabilities of our models.
To optimize our models for deployment on resource-constrained devices, we employed model quantization techniques, reducing their size while maintaining reasonable accuracy. This allowed for faster inference and efficient utilization of computational resources on devices such as the Raspberry Pi and Wio Terminal.
The successful deployment of our models on edge devices laid the foundation for IoT integration. The final system utilized the Raspberry Pi as the central device, the reSpeaker 2-Mics Pi HAT for audio capture, Wio Terminal for display and ESP32 boards to control multiple IoT devices based on user voice commands. This integration showcased the practicality and effectiveness of our speech intent recognition system in a smart home environment.
To evaluate the performance of our system, we conducted extensive testing and analysis. The results demonstrated the high accuracy achieved in recognizing speech intents and effectively controlling connected devices. The system exhibited reliability, responsiveness, and seamless integration with IoT devices.
In conclusion, our project has made significant contributions to the field of speech intent recognition for IoT applications in the Luganda language. Through dataset creation, feature extraction, model development, quantization, deployment, IoT integration, and system evaluation, we have developed a robust and efficient speech-to-intent system. This system has the potential to enhance the user experience and accessibility of smart home environments in Luganda-speaking regions.
By bridging the gap between local languages and IoT technologies, this project opens opportunities for enhanced user experiences, greater accessibility, and improved integration of IoT devices into everyday life. Future work could explore further improvements in real-time interaction by use of hot wake word detection, multi-language support, and the integration of advanced machine learning techniques for even more accurate and intelligent intent recognition.

References

[1] T. Dinushika, L. Kavmini, P. Abeyawardhana, U. Thayasivam, and S. Jayasena. Speech command classification system for sinhala language based on automatic speech recognition. In *2019 International Conference on Asian Language Processing (IALP)*, pages 205–210, 2019. doi: 10.1109/IALP48816.2019.9037648.

[2] Michael McTear, Zoraida Callejas, and David Griol. *The conversational interface: Talking to smart devices.* Springer International Publishing, January 2016. ISBN 9783319329659. doi: 10.1007/978-3-319-32967-3.

[3] Claire Babirye, Joyce Nakatumba-Nabende, Andrew Katumba, Ronald Ogwang, Jeremy Tusubira Francis, Jonathan Mukiibi, Medadi Ssentanda, Lilian D Wanzare, and Davis David. Building text and speech datasets for low resourced languages: A case of languages in east africa. In *3rd Workshop on African Natural Language Processing*, 2022. URL `https://openreview.net/forum?id=SO-U99z4U-q`.

[4] Language hackathon - deep learning indaba x uganda chapter. FruitPunch AI. URL `https://app.fruitpunch.ai/activity/38fa571b-3970-4c0f-89d3-f1926df70774`. Accessed July 1, 2023.

[5] New Vision Official. 20 million people can speak luganda - linguists - new vision official, Nov 2022. URL `https://www.newvision.co.ug/category/entertainment/20-million-people-can-speak-luganda---linguis-90236`. Accessed Nov. 30, 2022.

[6] Artificial intelligence: What it is and how it is used. `https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp`. Accessed on Dec. 15, 2022.

[7] Expert.ai. What is the definition of machine learning? `https://www.expert.ai/blog/machine-learning-definition/`. Accessed on Dec. 16, 2022.

[8] M.K.M, K.B. Mukesh Kumar, L. Sharma, M.Z. Sayeed Pasha, and K.H.D. An interactive voice controlled humanoid smart home prototype using concepts of natural language processing and machine learning. In *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, pages 1537–1546, 2018. doi: 10.1109/RTEICT42901.2018.9012359.

[9] I. Mykhailichenko, H. Ivashchenko, O. Barkovska, and O. Liashenko. Application of deep neural network for real-time voice command recognition. In *2022 IEEE 3rd KhPI Week on Advanced Technology (KhPIWeek)*, pages 1–4, 2022. doi: 10.1109/KhPIWeek57572.2022.9916473.

[10] P. Wang. Research and design of smart home speech recognition system based on

deep learning. In *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, pages 218–221, 2020. doi: 10.1109/CVIDL51233.2020.00-98.

[11] Simplilearn. What is deep learning and how does it work. `https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-deep-learning`. Accessed on Dec. 16, 2022.

[12] T.N. Sainath and C. Parada. Convolutional neural networks for small-footprint keyword spotting. In *Interspeech 2015*, pages 1478–1482, 2015. doi: 10.21437/Interspeech.2015-352.

[13] IBM. What are neural networks? `https://www.ibm.com/cloud/learn/neural-networks`. Accessed on Dec. 16, 2022.

[14] Rajeev Kumar and Vinay Sahula. Intelligent approaches for natural language processing for indic languages. In *2021 IEEE International Symposium on Smart Electronic Systems (iSES)*, pages 331–334, 2021. doi: 10.1109/iSES52644.2021.00084.

[15] S. Kumar, S. Benedict, and S. Ajith. Application of natural language processing and iotcloud in smart homes. In *2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT)*, pages 20–25, 2019. doi: 10.1109/ICCT46177.2019.8969066.

[16] T. Desot, F. Portet, and M. Vacher. Towards end-to-end spoken intent recognition in smart home. In *2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, pages 1–8, 2019. doi: 10.1109/SPED.2019.8906584.

[17] Gokhan Tur and Renato De Mori. *Spoken Language Understanding*. John Wiley & Sons, 2011.

[18] Thierry Desot, François Portet, and Michel Vacher. Slu for voice command in smart home: Comparison of pipeline and end-to-end approaches. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 822–829, 2019. doi: 10.1109/ASRU46091.2019.9003891.

[19] T. Desot, F. Portet, and M. Vacher. End-to-end spoken language understanding: Performance analyses of a voice command task in a low resource setting. *Computer Speech & Language*, 75:101369, Sep 2022. doi: 10.1016/j.csl.2022.101369.

[20] D. Buddhika, R. Liyadipita, S. Nadeeshan, H. Witharana, S. Javasena, and U. Thayasivam. Domain specific intent classification of sinhala speech data. In *2018 International Conference on Asian Language Processing (IALP)*, pages 197–202, Nov. 2018. doi: 10.1109/IALP.2018.8629103.

[21] M. Mehrabani, S. Bangalore, and B. Stern. Personalized speech recognition for internet of things. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 369–374, 2015. doi: 10.1109/WF-IoT.2015.7389082.

[22] I. B. C. Irugalbandara, A. S. M. Naseem, M. S. H. Perera, and V. Logeeshan. HomeIO: Offline smart home automation system with automatic speech recogni-

tion and household power usage tracking. In *2022 IEEE World AI IoT Congress (AIIoT)*, pages 571–577, 2022. doi: 10.1109/AIIoT54504.2022.9817282.

[23] Seeed Studio. Wio terminal: Atsamd51 core with realtek rtl8720dn ble 5.0 & wi-fi 2.4g/5g dev board with free course. `https://www.seeedstudio.com/Wio-Terminal-p-4509.html`, February 11 2023. Accessed on February 15, 2023.

[24] Espressif Systems. ESP32 Wi-Fi & Bluetooth MCU I Espressif Systems. `https://www.espressif.com/en/products/socs/esp32`, n.d. Accessed on February 11, 2023.

[25] MQTT. The standard for iot messaging. `https://mqtt.org/`. Accessed on June 1, 2023.

[26] TechTarget. What is mqtt and how does it work? *IoT Agenda*, January 27 2021. URL `https://www.techtarget.com/iotagenda/definition/MQTT-MQ-Telemetry-Transport`. Accessed on April 27, 2023.

[27] Eclipse Mosquitto. Eclipse mosquitto. Website, January 08 2018. URL `https://mosquitto.org/`. Accessed on April 15, 2023.

[28] NEUROTECH AFRICA. Low resource languages vs conversational artificial intelligence. `https://blog.neurotech.africa/low-resource-languages-vs-conversational-artificial-intelligence/`, September 26 2022. Accessed on February 11, 2023.

[29] Language Connections. Luganda language: The tongue of buganda, Nov 2022. URL `https://www.languageconnections.com/blog/luganda-the-language-of-buganda/`. Accessed Nov. 30, 2022.

[30] Prime Uganda Safaris & Tours. Culture in uganda — culture of uganda, Oct 2020. URL `https://www.primeugandasafaris.com/culture-in-uganda`. Accessed Dec. 01, 2022.

[31] How to make conversational ai work for low-resource languages, 2022. URL `https://keyreply.com/blog/conversational-ai-low-resource-languages/`. accessed Nov. 30, 2022.

[32] Fluent.ai. Fluent speech commands: A dataset for spoken language understanding research. `https://fluent.ai/fluent-speech-commands-a-dataset-for-spoken-language-understanding-research/`, April 19 2021. Accessed on December 2, 2022.

[33] Xuebin Yang, Hongzhi Yu, and Lei Jia. Speech recognition of command words based on convolutional neural network. In *2020 International Conference on Computer Information and Big Data Applications (CIBDA)*, pages 465–469, 2020. doi: 10.1109/CIBDA50819.2020.00110.

[34] Google AI Blog. Launching the speech commands dataset. `https://ai.googleblog.com/2017/08/launching-speech-commands-dataset.html`, August 24 2017. Accessed on December 2, 2022.

[35] Dmitriy Serdyuk, Yongqiang Wang, Christian Fuegen, Anuj Kumar, Baiyang Liu, and Yoshua Bengio. Towards end-to-end spoken language understanding. *CoRR*, abs/1802.08395, 2018. URL `http://arxiv.org/abs/1802.08395`.

[36] D. Buddhika, R. Liyadipita, S. Nadeeshan, H. Witharana, S. Javasena, and U. Thayasivam. Domain specific intent classification of sinhala speech data. In *2018 International Conference on Asian Language Processing (IALP)*, pages 197–202, Nov. 2018. doi: 10.1109/IALP.2018.8629103.

[37] Isham Mohamed and Uthayasanker Thayasivam. Low resource multi-asr speech command recognition. In *2022 Moratuwa Engineering Research Conference (MERCon)*, pages 1–6, 2022. doi: 10.1109/MERCon55799.2022.9906230.

[38] Loren Lugosch, Mirco Ravanelli, Patrick Ignoto, Vikrant Singh Tomar, and Yoshua Bengio. Speech model pre-training for end-to-end spoken language understanding, 2019. URL `https://arxiv.org/abs/1904.03670`.

[39] T. Reshamwala, C. Shah, and S. Naik. Computerization in home: Change in way of life. In *2021 Second International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)*, pages 1–6, 2021. doi: 10.1109/ICSTCEE54422.2021.9708568.

[40] P. J. Rani, J. Bakthakumar, B. P. Kumaar, U. P. Kumaar, and S. Kumar. Voice controlled home automation system using natural language processing (nlp) and internet of things (iot). In *2017 Third International Conference on Science Technology Engineering & Management (ICONSTEM)*, pages 368–373, Mar. 2017. doi: 10.1109/ICONSTEM.2017.8261311.

[41] Automated sync from github.com/tensorflow/tflite-micro · tensorflow/tflite-micro-arduino-examples@219ac1d. GitHub. URL `https://github.com/tensorflow/tflite-micro-arduino-examples/commit/219ac1dfed8a8ba0edfdbaae51aed5dc9b208c0c`. Accessed on July 3, 2023.

MAKERERE UNIVERSITY

COLLEGE OF ENGINEERING, DESIGN, ART AND TECHNOLOGY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Luganda Speech Intent Classification For IoT Applications

Submitted by:
JOHN TREVOR KASULE 19/U/14669/PS BSCE
ELVIS MUGUME 19/U/0676 BELE

Main supervisor: **Dr. Andrew Katumba**
Co-supervisor: **Dr. Ronald Kizito**

A Final Year Full Project Proposal Submitted to the Department of Electrical and Computer Engineering, Makerere University in partial fulfilment of the requirements for the award of Degree of Bachelor of Science in Electrical Engineering.

24 February 2023

Abstract

The ability to control Internet of Things (IoT) devices using voice commands has gained popularity in recent years. However, most existing systems are designed for high-resource languages, and not suitable for low-resource languages like Luganda. To address this issue, we propose a project to develop a Luganda speech intent recognition system for controlling IoT devices. This project involves collecting a dataset of Luganda voice commands, building and training natural language processing (NLP) models based on both speech-based and automatic speech recognition (ASR) approach, and implementing the best-performing model on a Wio Terminal. The Wio Terminal will communicate with ESP32 microcontroller nodes using MQTT protocol to control 8 devices with 20 intents. Our proposed system aims to allow Luganda-speaking individuals to interact with smart home devices using voice commands in their native language, improving their experience and enabling more efficient control of their devices. This project will contribute to the fields of natural language processing and IoT and promote the inclusion of low-resource language speakers in the design of smart home systems. The developed system can be extended to other languages with limited resources and can serve as a starting point for future research.

List of Figures

Contents

Chapter 1

Introduction

1.1 Project Background

Speech command classification, also known as intent classification, is a key research area in the field of conversational AI[1]. Recent advances in conversational Artificial Intelligence (AI) have resulted in conversation-based applications with a wide range of supported platforms. Speech recognition is the ability of machines or programs to identify or acknowledge words and phrases from spoken language and convert them to a machine-readable format[2]. This technology has supported the growth of voice-based interactions.[3] These systems respond to voice commands, whether they are for playing music, turning on the lights, or answering questions. In order to be effective, these systems must be able to extract the intent of the user from the voice command. This is known as intent recognition. The intent is something the user wants to do. Intent classification is a sub-task of Spoken Language Understanding and it aims to extract the aim of the user in a given voice command. Voice-based intent recognition is typically performed through automatic speech recognition (ASR) and intent classification from the transcriptions in a pipeline[4].

With the rapid increase of smart devices, there has been a growing interest in the concept of the Internet of Things (IoT). The internet has changed the way people communicate with each other by creating a virtual world for both professional and social lives during the past decades[1]. The Internet of Things (IoT) can be defined as a worldwide network of interconnected physical things. IoT can be considered an expansion of the internet that will impact our everyday lives and the way we interact with things. The foundational elements for IoT are smart objects, which are everyday items with embedded intelligence and internet connectivity. A smart object could be a smart bulb that lights when you get home[5].

Creating a network of interconnected objects is the foundation for IoT. While this is the case, the interaction between people and the internet of things is another crucial component of building a connected life. Users can interact directly with a variety of things by creating embedded interactions within IoT objects. Enabling a Speech-based interaction with these devices improves user experience[6].

Voice commands are increasingly common in consumer electronics. Home assistants like Google Home, Siri, or Alexa[7] offer a natural voice interface to digital services and home automation with an almost non-existing entry barrier or learning curve for the user[2]. Cars offer voice interfaces to control air conditioning or media playback. This is all achievable by speech intent classification.

Work has been done in relation to speech intent classification using IoT devices. However, most of the existing work in speech intent classification and speech recognition at large is only available in major languages such as English, French and German and there has been less work done with low-resource languages. Low-resource languages are those that have relatively fewer data for training conversational AI systems[8]. Most of the native languages in Uganda and East Africa fall under this category.

This work focuses on Luganda, the official spoken language of Buganda referring to the central region of Uganda where the largest number of native speakers are found. After English which is the official language of Uganda, Luganda is the most spoken language in the country (even more so than the second official language, Kiswahili)[9]. Luganda is spoken by about 20 million people in the country and beyond[10].

1.2 Problem Statement

There is increasing demand for voice command systems to control devices in homes and industries. However, most of the work that has been done and developed is in foreign languages, creating a gap between high-level resource languages and low-resource languages. Despite a large number of speakers of native languages like Luganda, which is the most spoken language in Uganda with at least 20 million speakers, there has been limited work in developing voice command systems for these languages. This project aims to address this gap by implementing a speech intent classification for IoT applications using Luganda voice commands.

1.3 Justification

Uganda is a culturally diverse and multilingual country with about 41 native languages[11]. All the native languages in Uganda are classified as low-resource languages in the conversational AI field[11]. This is because there are not enough resources available to work with the languages in AI tasks, especially Natural Language processing[8]. There is a steady advance in speech recognition technology throughout the world and Africa and the technology have not benefited any of these languages classified as low-resource for ASR applications. It is important to note that the absence of adequate resources and data for low-resource languages is a significant obstacle in the development of NLP tools. As a result, there is a need for concerted efforts toward creating solutions for low-resource languages to promote linguistic diversity and inclusivity in the field of AI. The lack of resources or research for the development of speech intent recognition systems in these low-resource languages means that speakers of these languages cannot benefit from speech intent recognition systems and control devices in their local languages. Luganda is the most spoken native language in Uganda and is used for communication in a large part of Central Uganda where technology is mostly used[11]. There is a need to include Luganda in speech recognition and voice command systems due to its great coverage in Uganda. There is no dataset for Luganda voice commands that can be used to develop a model to classify intents in an IoT setup. There is also no preliminary model for Luganda intent classification that can be used to control devices. And more so, the existing solutions for Luganda are not implemented to solve this gap in Speech intent recognition. The project will build a foundation for speech recognition tasks in low-resource languages in East Africa. The project will also provide a basis to create smart homes and smart devices that use Luganda to be controlled. The project addresses the gap by providing a dataset of Luganda voice commands that can be used for Luganda speech intent recognition. The project also presents research in the Conversational AI field that can be used as a basis for more research on other native languages in the region/country.

The rest of the document is organized as follows. Section 2 provides a literature review of related work in the field of speech intent classification and IoT applications. Section 3 describes the Project Plan which includes the proposed system's design and architecture, including the NLP model, the dataset of Luganda voice commands, and the hardware components used.

Chapter 2

Literature Review

In this chapter, we delve into the existing literature on Luganda speech intent classification for IoT applications, exploring the latest advances and identifying the gaps that our proposed project aims to fill.

2.1 Brief Theory

2.1.1 Machine Learning

Machine Learning is a subset of Artificial intelligence (AI) that gives computers the ability to learn without being programmed[12]. Machine learning uses technologies where the rules or algorithms are created from the input data, that is; the data on which the system is trained. In machine learning, the computer is allowed to come up with its solution based on the data it is given instead of telling it what to do.
Machine learning involves giving machines to learn from raw data so that the computer program can change when exposed to new data (learning from experience).[13], [14] There are many different types of machine learning which include; unsupervised learning, supervised learning and reinforcement learning. In supervised learning, a system is trained using labelled data and the correct output is provided for each input. In unsupervised learning, the system is trained using unlabelled data and it is tasked to learn to identify patterns and relationships in the data on its own. In reinforcement learning, the system is trained by learning from its actions and receiving feedback in the form of rewards or penalties[15].

2.1.2 Deep Learning

Deep learning is a type of machine learning that involves using neural networks with multiple layers to learn complex patterns and relationships in data. It involves a hierarchy of nonlinear transformation of input that can be used to generate a statistical model as output.[15–17]

Neural Networks

Artificial Neural Networks are normally called Neural Networks (NN). A neural network is a type of machine-learning model that is inspired by the structure and function of the brain. It is composed of a large number of interconnected artificial neurons, which are computational units that are designed to mimic the behaviour of biological neurons.
In a neural network, the neurons are organized into layers, with each layer receiving input from the previous layer and passing its output to the next layer. This allows the network to learn and represent increasingly complex and abstract patterns in the data. A large amount of labelled data is provided for training a neural network and the network uses this data to adjust the strengths of the connections between the neurons. A trained neural network can be used to make predictions on new data. This involves providing the input data to the network and running the computational graph to generate the

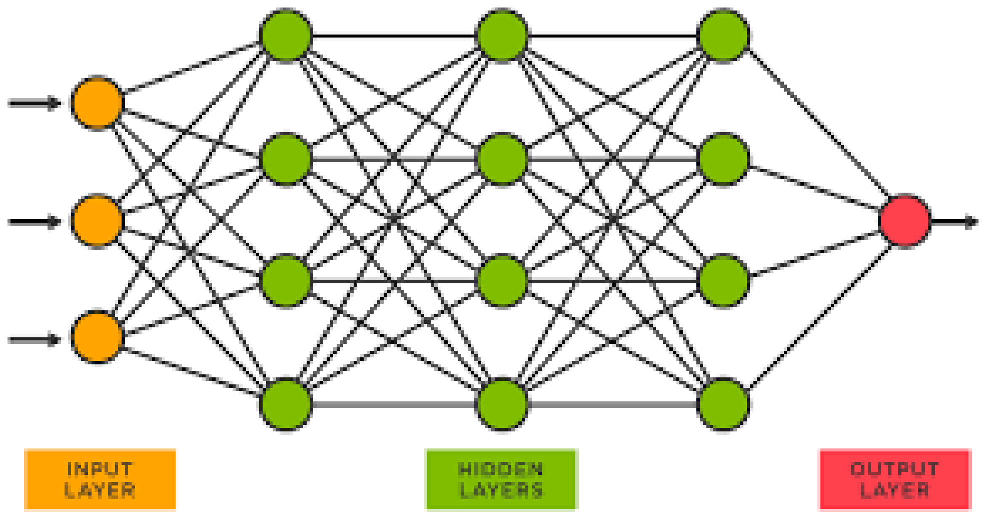


Figure 2.1: Neural Network architecture

output.[15, 18, 19] From the image above, it can be seen that a neural network is made of interconnected neurons. Each of them is characterised by its weight, bias, and activation function. Here are other elements of this network[9].

- Input Layer: The input layer takes raw input from the domain. No computation is performed at this layer. Nodes here just pass on the information (features) to the hidden layer.
- Hidden Layer: As the name suggests, the nodes of this layer are not exposed. They provide an abstraction to the neural network. The hidden layer performs all kinds of computation on the features entered through the input layer and transfers the result to the output layer.
- Output Layer It's the final layer of the network that brings the information learned through the hidden layer and delivers the final value as a result.

All hidden layers usually use the same activation function. However, the output layer will typically use a different activation function from the hidden layers. The choice depends on the goal or type of prediction made by the model.

2.1.3 Natural Language Processing

Natural Language Processing is a term used in the field of computer science to describe the study of human natural languages. Natural Language Processing (NLP) is a subfield of semantics, software engineering, and artificial intelligence dealing with the coordination between computers and human language, specifically how to program computers to be able to process and investigate a lot of natural language information. The objective is to program a computer to understand the written texts or spoken utterances, including the context of the language inside them. To understand the structure of the language knowledge of various units of language is required. The major units of language are phonemes, morphemes, lexemes, syntax and context. When syntax, Semantics, and Pragmatics are conceptualised along with these units, they help to obtain an understanding of the language and communicate among the people. The smallest unit of sound is the phoneme. It brings changes in the meaning of a language but it does not hold any meaning when used alone. A morpheme is the smallest unit of words. A lexeme is the arrangement of all the inflected forms of a single word. The syntax is the arrangement of rules by which an individual builds full sentences while the context conveys a particular meaning of a sentence based on the interaction of all the units of language.[20] Most of the challenges in NLP are listed as follows: human language understanding, enabling computers to derive meaning from the voice commands given to them through human (natural) language, and others involve natural language interaction between computers and humans.[21]

Automatic Speech Recognition

Speech recognition also called, Automatic Speech Recognition or speech-to-text is the ability of a machine or program to identify words spoken aloud and convert them into readable text.[20] [21]

2.1.4 Spoken language Understanding

Spoken Language Understanding (SLU) is a subfield of Natural Language Processing (NLP) that focuses on developing algorithms and models to enable machines to comprehend and interpret human language in spoken form.
Spoken Language Understanding (SLU) refers to the ability of a machine to extract semantic information from speech signals in a form that is amenable to further processing including dialogue, voice commands, information retrieval, etc.[22]
SLU involves the analysis of the acoustic signal of speech to identify phonemes and words and then to extract meaning from the identified words. This includes tasks such as automatic speech recognition, intent detection, slot filling, and dialogue management[23].
Automatic Speech Recognition (ASR) is a critical component of SLU, as it converts spoken words into text that can be analyzed by NLP algorithms. Intent detection identifies the user's intention behind the spoken words, while slot filling extracts relevant information from the user's utterances. Dialogue management involves managing the flow of conversation, generating appropriate responses, and handling errors and uncertainties in the user's input.[23]
SLU is used in a variety of applications, including virtual assistants, chatbots, and voice-enabled devices. By enabling machines to understand and respond to spoken language, SLU improves the user experience, making it more intuitive and natural.

Slot Filling

Slot filling is a task in Natural Language Processing (NLP) that involves extracting specific pieces of information, known as slots, from natural language input(speech or text)[24]. The slots are predefined categories of information that the system is looking for in the user's utterance, such as date, time, location, or any other relevant information that is required to complete a task.
For example, if a user says ”Schedule a meeting with John at 3 PM on Friday,” the system can extract the following slots: ”John” (the person's name), ”3 PM” (the time), and ”Friday” (the date). These slots can then be used to complete the task of scheduling a meeting with John on Friday at 3 PM.
Slot filling is commonly used in conversational systems, such as chatbots or virtual assistants, to extract information from the user's input and complete a specific task. It is usually performed in conjunction with intent detection, which identifies the user's intention in the input. Once the intent is detected, slot filling can be used to extract the relevant information needed to fulfil the intent[23].

Speech Intent Classification

Speech intent classification/identification/recognition is a branch of Spoken Language Understanding that enables systems to extract the intent of the user from the speech in a particular domain. The intent is something the user wants to do. Speech intent classification can be done either by converting speech to text and doing a text classification or directly using a classification algorithm for extracted features of speech.
The classical approach to Speech Intent Classification and Spoken Language Understanding has been a pipeline of Automatic Speech Recognition (ASR) to transcribe speech signals into a textual representation that feeds a Natural Language Understanding (NLU) module[25]. This method is widely used in many generic ASR systems, whereas the second method is more applicable to specific domains. For a specific domain, the speech vocabulary is limited and well-defined[26]. Speech command classification is a part of speech intent classification that covers research on the use of voice commands.

Speech Intent classification works along with slot filling which helps to extract specific parts of information from a speech command or utterance.
For example, if a user provides an utterance/command saying ”turn on the lights in the kitchen,” the system can extract the following slots: ”activate” (action), ”lights” (the object), and ”kitchen” (the location). These slots can then be used to complete the intent of activating the lights in the kitchen.
Speech intent classification has led to progress in the development of the use of voice commands to control devices. The recognition of the intent from a voice command is what is known as voice command recognition. Voice command recognition has been implemented in home automation using the help of IoT[21].

2.1.5 IoT

The Internet of Things (IoT) refers to the connection of devices such as appliances, vehicles, and other items embedded with electronics, software, sensors, and network connectivity which enable these objects to collect and exchange data. This technology allows for the automation of various processes and the ability to monitor and control devices remotely. The IoT has the potential to greatly improve efficiency and convenience in many areas of our lives, such as healthcare, transportation, and energy management.
The IoT market is growing at a rapidly growing technology market and has many different concepts. IoT technology is most often associated with ”smart home” related products, including gadgets and appliances. The devices include; lighting fixtures, thermostats, home security systems, cameras, and other home appliances that support one or more common ecosystems, and can be controlled via devices associated with that ecosystem, such as smartphones and speakers[21]
The relation within the IoT network is usually people-to-people, people-to-things, and things-to-things. This concept has greatly changed the way people interact with these devices. This is because people are able to control a number of interconnected devices at the same time. The connection can be set up using modules such as Bluetooth, Wi-Fi, and Zigbee among others.[14] The interconnection set up using these modules is what is known as the Internet.[17] The traditional approach to controlling devices has always been by the use of remotes but recently, the use of voice has sparked up and improved the use of IoT.[21], [27], [28]
The use of voice commands to control devices is efficient as it enables users to control devices without raising a hand or leaving a position. With IoT, it is achievable to manage a number of devices using voice with the help of speech intent recognition which helps to determine the intent of the voice command utterance. When the intent is obtained, a particular action occurs on a device(s). The use of such commands can vary from “turn off the lights in the kitchen” to “close all doors”.

WIO Terminal

The WIO Terminal, a device built with a powerful ATSAMD51 microcontroller and wireless connectivity via Realtek RTL8720DN, has been chosen to establish an IoT configuration along with three ESP32 nodes in the proposed project. This configuration is intended to control eight devices set up at different ESP32 nodes, with the WIO Terminal serving as the central hub for communication and control[29].

ESP32

The ESP32, a low-cost and power-efficient system-on-a-chip, was selected as the primary node for this project due to its reliable wireless communication capabilities, including Wi-Fi and Bluetooth, which are essential for IoT applications. Additionally, the ESP32’s dual-core processor, with a clock speed of up to 240MHz, enables it to perform complex tasks efficiently[30].

MQTT

The MQTT protocol was chosen as the primary communication protocol for this project. MQTT, or Message Queuing Telemetry Transport, is a lightweight messaging protocol that is well-suited for IoT applications due to its low power and bandwidth requirements. It allows for reliable communication between devices, even in low-bandwidth or unreliable network conditions.

2.1.6 Low-resource languages

Low-resource languages are those which have limited amounts of data available for building Artificial Intelligence (AI) systems such as Conversational AI and machine translation. These languages are primarily found in Africa and Asia, where data storage is limited. As a result, it becomes challenging to develop natural language processing (NLP) solutions like conversational AI systems. Oral languages with few written resources are particularly challenging to work with[31].

In order to develop language-based solutions, access to a significant amount of raw text data from various sources such as books, emails, social media content, and scientific papers is crucial. Task-specific resources like parallel corpora for machine translations, various kinds of annotated text to be used in part-of-speech tags, and dictionaries for developing named entity recognition systems are also necessary. The lack of working solutions and available data makes it difficult to fine-tune models for downstream tasks, limiting the range of possible tasks that can be solved with low-resource NLP tools[32].

Auxiliary data such as labelled data in different languages, as well as lexical, syntactic, and semantic resources like dictionaries, dependency tree corpora, and semantic databases like WordNet, are also essential for developing AI solutions for low-resource languages. It is important to note that the absence of adequate resources and data for low-resource languages is a significant obstacle in the development of NLP tools. As a result, there is a need for concerted efforts toward creating solutions for low-resource languages to promote linguistic diversity and inclusivity in the field of AI.

Luganda

Luganda is the official spoken language of Buganda referring to the central region of Uganda where the largest number of native speakers are found in the country[9]. After English which is the official language of Uganda, Luganda is the most spoken language in the country (even more so than the second official language, Kiswahili)[11]. Luganda is spoken by about 20 million people in the country and beyond[10]. Luganda and all the native languages in Uganda are classified as low-resource languages in the conversational AI field. This is because there are not enough resources available to work with the languages in AI tasks, especially Natural Language processing [8] [32].

2.2 Related work

This section discusses work in relation to speech intent classification for IOT applications.

2.2.1 Voice commands datasets

Fluent Speech Commands dataset

The Fluent Speech Commands dataset is a collection of 30,043 spoken commands that are designed to train and test speech recognition systems for smart home scenarios. The dataset was collected through crowd-sourcing, which involved recruiting multiple individuals to record the utterances. This approach has several benefits, including the ability to include a diverse range of voices and speech patterns in the dataset, which makes it more representative of real-world usage scenarios[33].

The dataset includes 97 speakers, and each utterance is recorded as a 16 kHz single-channel .wav file. The dataset is labelled with three slots - action, object, and location - and each slot can take on multiple values. The combination of slot values represents the intent of the spoken command. For example, a spoken command might have the intent of ”turn on the lights in the kitchen,” where ”turn on” is the action, ”lights” is the object, and ”kitchen” is the location.

There are multiple possible wordings for each intent, which makes the dataset more challenging and realistic for speech recognition systems to recognize. In total, the dataset has 248 phrasing mappings to 31 unique intents. The demographic information about the speakers, including age range, gender, and speaking ability, is included in the dataset.

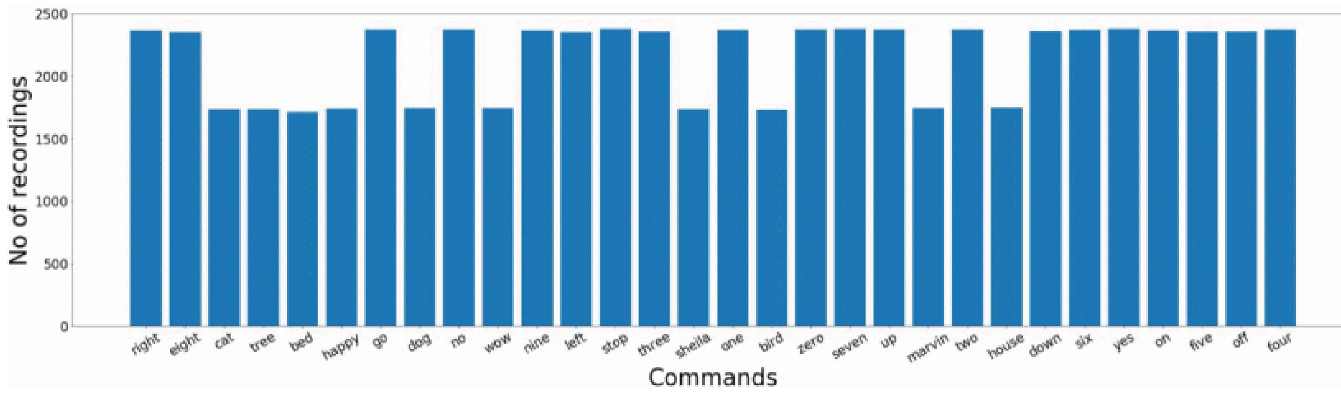
The dataset is randomly divided into the train, valid, and test splits, with no speaker appearing in more than one split. Each split includes all possible wordings for each intent, although there is an option to include data for only certain wordings for different sets to test the model's ability to recognize wordings not seen during training.

The Fluent Speech Commands dataset is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International license. This license allows the dataset to be shared, copied, and redistributed as long as it is not used for commercial purposes, and any adaptations or modifications of the dataset are not allowed. This license ensures that the dataset is freely available to the research community for non-commercial research purposes while also protecting the rights of the dataset creators.

The Fluent Speech Commands dataset is therefore a valuable resource for researchers and developers working on speech recognition systems for smart home scenarios. The fact that it was collected through crowd-sourcing makes it a more representative and diverse dataset, which can improve the performance of speech recognition systems in real-world scenarios[33, 34].

Google Speech Commands Dataset

The Google Speech Commands dataset[35] is a valuable resource for researchers and practitioners interested in developing and testing speech recognition models. It was created by the TensorFlow and AIY teams to showcase speech recognition examples using the TensorFlow API and contains 65,000 clips of one-second-long duration. Each clip contains one of 30 different words spoken by thousands of different subjects. The clips were recorded in realistic environments with phones and laptops. The number of clips for each command in the dataset is shown in figure 2.2 below.



ss

Figure 2.2: Commands in Google Speech Commands Dataset

One notable feature of the Google Speech Commands dataset is that it includes noise words, which add complexity to the task of recognizing specific words in noisy or cluttered environments. This is particularly relevant for speech recognition in industry settings, where background noise or interference may be common. By including noise words, the dataset provides a more realistic and challenging task for speech recognition models to solve.

The availability and accessibility of the Google Speech Commands dataset through the TensorFlow API and other platforms also make it a valuable resource for further research and experimentation in speech recognition. Its use can help researchers and practitioners develop and test new speech recognition models, and can also facilitate the comparison and evaluation of different approaches[35].

2.2.2 State-of-the-art: Speech Intent Recognition

There are **two** primary approaches to speech intent Recognition: speech-to-intent directly and speech-to-text-to-intent.

1. The speech-to-intent directly approach aims to map spoken language directly to its intent without first transcribing the speech into text. These features can include MFCCs, spectrograms, or other time-frequency representations that capture relevant information about the audio signal. Machine learning models or deep learning models are then trained on these features to classify the user's intent. This approach has the advantage of not relying on intermediate text transcription, which can introduce errors and reduce the accuracy of the final prediction[26, 36].

2. The speech-to-text-to-intent approach, on the other hand, involves first transcribing the spoken language into text using automatic speech recognition (ASR) and then extracting intent from the text using natural language understanding (NLU) techniques. There are two primary methods within this approach: the pipeline approach and the end-to-end approach[24]. The approaches are shown in Figure 2.3.

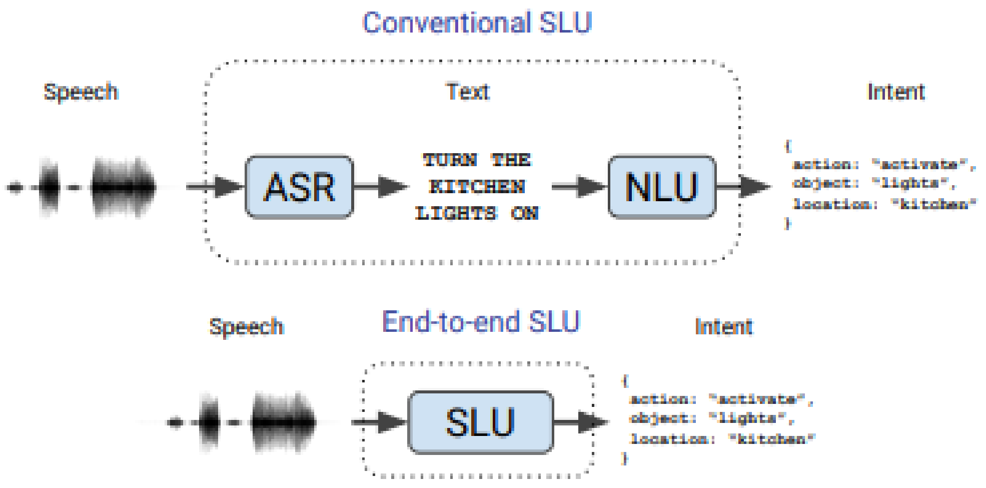

Conventional SLU
Speech
Text
Intent
ASR
TURN THE KITCHEN LIGHTS ON
NLU
{ action: "activate", object: "lights", location: "kitchen" }
End-to-end SLU
Speech
Intent
SLU
{ action: "activate", object: "lights", location: "kitchen" }

Figure 2.3: Conventional ASR-NLU system vs End-to-End SLU

The pipeline approach to speech-to-text-to-intent involves separate modules for ASR and NLU. The ASR module converts spoken input into text, and the NLU module extracts intent and slots from the transcribed text. Although this approach has been widely adopted and is generally effective, it has some limitations. For example, errors introduced by the ASR module can propagate through the pipeline and degrade the performance of the SLU system. Additionally, the separate modules may be optimized under different criteria, leading to suboptimal performance.

In contrast, the end-to-end approach to speech-to-text-to-intent combines the ASR and NLU tasks into a single model. The model is trained directly on spoken input and can learn to extract high-level concepts related to the task, rather than relying on word-by-word recognition. This approach has the potential to improve performance by avoiding errors introduced by the separate ASR and NLU modules. However, it requires large amounts of domain-specific data for training, which can be difficult to acquire for new domains.

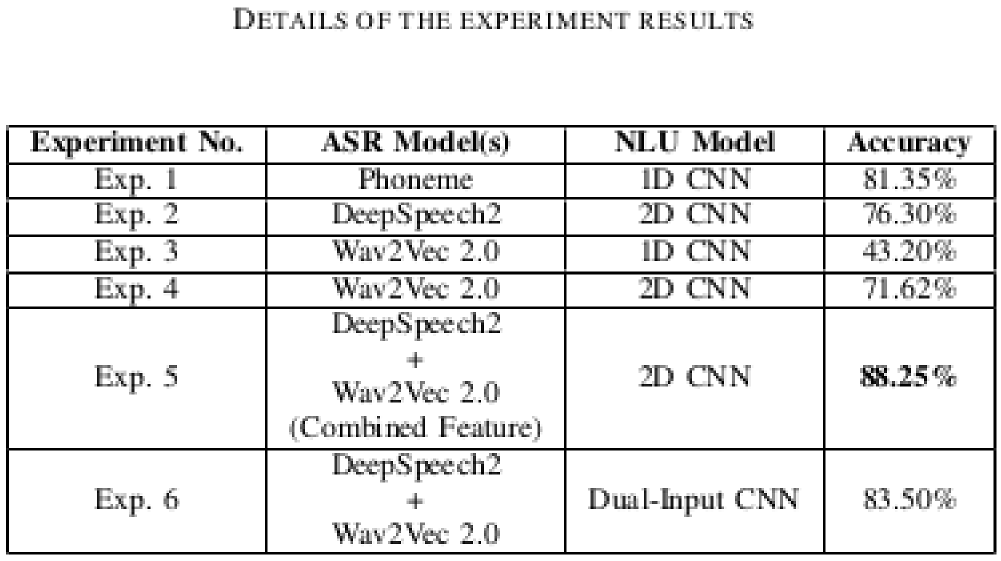
From the explanation above, it can be inferred that a variety of model units, including the ASR unit, NLU unit, and the direct speech-to-intent classifier, are involved in speech intent recognition. A variety of model strategies for these units can be used to identify speech intents.
ASR can be done using acoustic models, ranging from traditional HMM models to those based on

Artificial Neural networks. Natural Language Understanding (NLU) tasks simply involve classifying text and can be performed using standard machine learning classification algorithms like Support Vector Machines (SVMs) and decision trees. Additionally, Neural Networks can also be utilized for this purpose. Speech intent recognition has also been done using a hybrid methodology. In this method, neural networks and HMMs are combined, or neural networks are used for acoustics and machine learning techniques for classification[26].

1. In [37], Mohamed and Thayasivam proposes a multi-ASR set-up to improve the accuracy of low-resource speech-command transfer-learning tasks. The authors combine the outputs of pre-trained ASR models, DeepSpeech2 and Wav2Vec 2.0, using a CNN-based setup. The experiment shows an 8% increase in accuracy over the current state-of-the-art low-resource speech-command phoneme-based speech intent classification methodology.

   The paper addresses an important issue in SLU studies, particularly for low-resource languages where training an ASR model requires a significant amount of audio input. The proposed approach of combining outputs from multiple ASR models can help improve the accuracy of downstream NLU models, despite the errors produced by the ASR models.

   The experimental results in figure 2.4 presented in the paper demonstrate the effectiveness of the proposed multi-ASR setup in improving the accuracy of low-resource speech-command recognition and topic identification tasks. The reported accuracy of 88.25% with less than 0.5 hours of Tamil[1] audio clips is particularly impressive.



DETAILS OF THE EXPERIMENT RESULTS

Experiment No.	ASR Model(s)	NLU Model	Accuracy
Exp. 1	Phoneme	1D CNN	81.35%
Exp. 2	DeepSpeech2	2D CNN	76.30%
Exp. 3	Wav2Vec 2.0	1D CNN	43.20%
Exp. 4	Wav2Vec 2.0	2D CNN	71.62%
Exp. 5	DeepSpeech2 + Wav2Vec 2.0 (Combined Feature)	2D CNN	**88.25%**
Exp. 6	DeepSpeech2 + Wav2Vec 2.0	Dual-Input CNN	83.50%

Figure 2.4: Details of the different Experiments and Results

   The authors mention future work in applying this setup to different domains and low-resource languages. They also plan to expand the multi-ASR model to include more than two ASR models and phoneme-based ASR models.

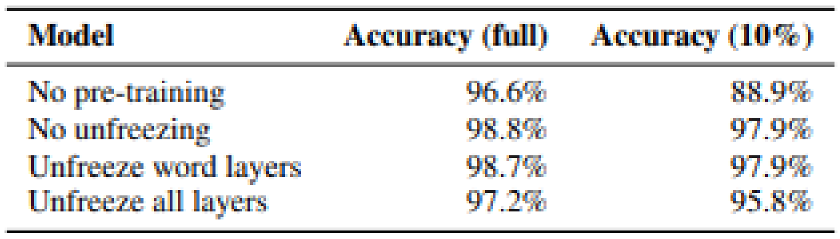
   Overall, this paper presents a novel approach to addressing the challenges of low-resource SLU tasks. The results suggest that combining the outputs of multiple ASR models can significantly improve the accuracy of downstream NLU models. The proposed approach has important implications for improving speech recognition and natural language understanding in low-resource settings.

[1]Tamil is a Dravidian language spoken mainly in the Indian state of Tamil Nadu and the Union Territory of Puducherry, as well as in some other parts of India and Sri Lanka.

2. In [38], Lugosch et al. presents a pre-training methodology for end-to-end spoken language understanding (SLU) models that map speech directly to intent through a single trainable model. The proposed methodology involves pre-training the model to predict words and phonemes, which learns good features for SLU. The authors also introduce a new SLU dataset, Fluent Speech Commands[33], to demonstrate the efficacy of their pre-training techniques.

   The end-to-end SLU model consists of three modules: phoneme level, word level, and intent level. The phoneme module is implemented using a SincNet layer, which processes the raw input waveform, followed by multiple convolutional layers and recurrent layers with pooling (to reduce the sequence length) and dropout. The phoneme module is pre-trained to predict phoneme targets using a linear classifier.

   The output of the phoneme module serves as the input for the word module, which is made up of recurrent layers with dropout and pooling. The word module is pre-trained to predict whole-word targets using another linear classifier. The third module, which is not pre-trained, maps word-level representations to the predicted intent.



Model	Accuracy (full)	Accuracy (10%)
No pre-training	96.6%	88.9%
No unfreezing	98.8%	97.9%
Unfreeze word layers	98.7%	97.9%
Unfreeze all layers	97.2%	95.8%

Figure 2.5: Accuracy on the test set for different models, given the full training dataset or a 10% subset of the training data

   The paper evaluates the proposed methodology using various experiments. Lugosch et al. use the Montreal Forced Aligner to obtain word and phoneme-level alignments for LibriSpeech, and pre-train the model on the entire 960 hours of training data using these alignments. They also compare the accuracy of different models when given the full training dataset or a 10% subset of the training data.

   The experiments show that the pre-trained models outperform the randomly initialized model in all cases. The best results are obtained when only the word layers of the pre-trained model are unfrozen. The authors suggest that this may be because the fully unfrozen model begins to forget the more general phonetic knowledge acquired during pre-training.

   However, the paper also discusses the limitations of the end-to-end SLU model, such as its inability to handle new phrases and synonyms not observed in the SLU dataset. The authors suggest that it may be possible to teach the pre-trained part of the model to output ”embedding-like” word representations so that the intent module can recognize the meaning of phrases with synonyms.

3. In [39], Yi et al. performed a recognition task using a pre-trained wav2vec model to conduct experiments on CALLHOME corpus, including six languages: Mandarin, English, Japanese, Arabic, German, and Spanish.

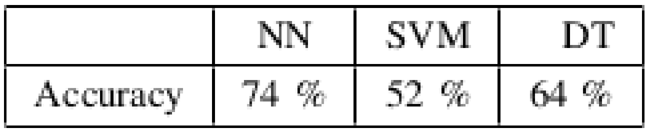
Furthermore, features that are taken from speech can be input into neural networks or machine learning algorithms for direct intent categorization.

1. In [40], Buddhika et al. proposes a domain-specific intent classification system for the Sinhala[2] language using a feed-forward neural network with backpropagation. Buddhika et al. extracted

[2]Sinhala is a low-resource language primarily spoken in Sri Lanka, a country located in South Asia.

Mel Frequency Cepstral Coefficients (MFCC) from a Sinhala speech corpus of 10 hours to train the neural network. The performance of the system was evaluated using the recognition accuracy of the speech queries.

This excerpt from the paper indicates that the authors have conducted a comparison of the intent classification accuracy achieved by three different approaches: Neural Network (NN), Support Vector Machine (SVM), and Decision Tree (DT). The performance of these approaches was evaluated and compared in figure 2.6 below, which shows the accuracy achieved by each approach.



	NN	SVM	DT
Accuracy	74 %	52 %	64 %

Figure 2.6: Comparison of NN , SVM and DT

According to the figure, the Neural Network approach outperformed both SVM and DT approaches in the domain-specific intent classification tasks. This suggests that the proposed approach using a feed-forward neural network with backpropagation is a promising method for intent classification.

One of the interesting findings in this paper is that the accuracy of the model did not exceed 66% when the neural network was trained with raw recordings in the corpus. However, after preprocessing the corpus and removing over-recorded samples, recordings that did not contain the full inflection, and speech samples with heavy noise, the model accuracy improved to 73%. This demonstrates the importance of having a preprocessed corpus for the proposed system. mfcc features were extracted from the preprocessed audio and used to train a deep neural network intent classifier for which results are shown.

However, the work was done on only 6 intents which represents a small scope.

2. In [34], Yang et al. presents a method of speech recognition using convolutional neural networks (CNNs) for recognizing command words in industry settings. The authors compare the proposed CNN-based method with traditional deep neural network (DNN) and recurrent neural network (RNN) training methods and report that the CNN-based method significantly improves the accuracy of recognizing command words. The results are shown in the table 2.1 below;

Approach	Accuracy
DNN	82.46%
RNN	89.45%
CNN	92.88%

Table 2.1: Comparison of accuracies for model approaches

The experimental results show that the CNN-based method outperforms DNN and RNN models on the Google Speech Commands dataset, which suggests that CNNs could be a promising solution for speech recognition tasks in industry settings. The features extracted from the speech are mfccs which are fed to the different model architectures. This finding is significant because speech recognition technology has not developed as rapidly as other areas of artificial intelligence, such as natural language processing and image recognition.

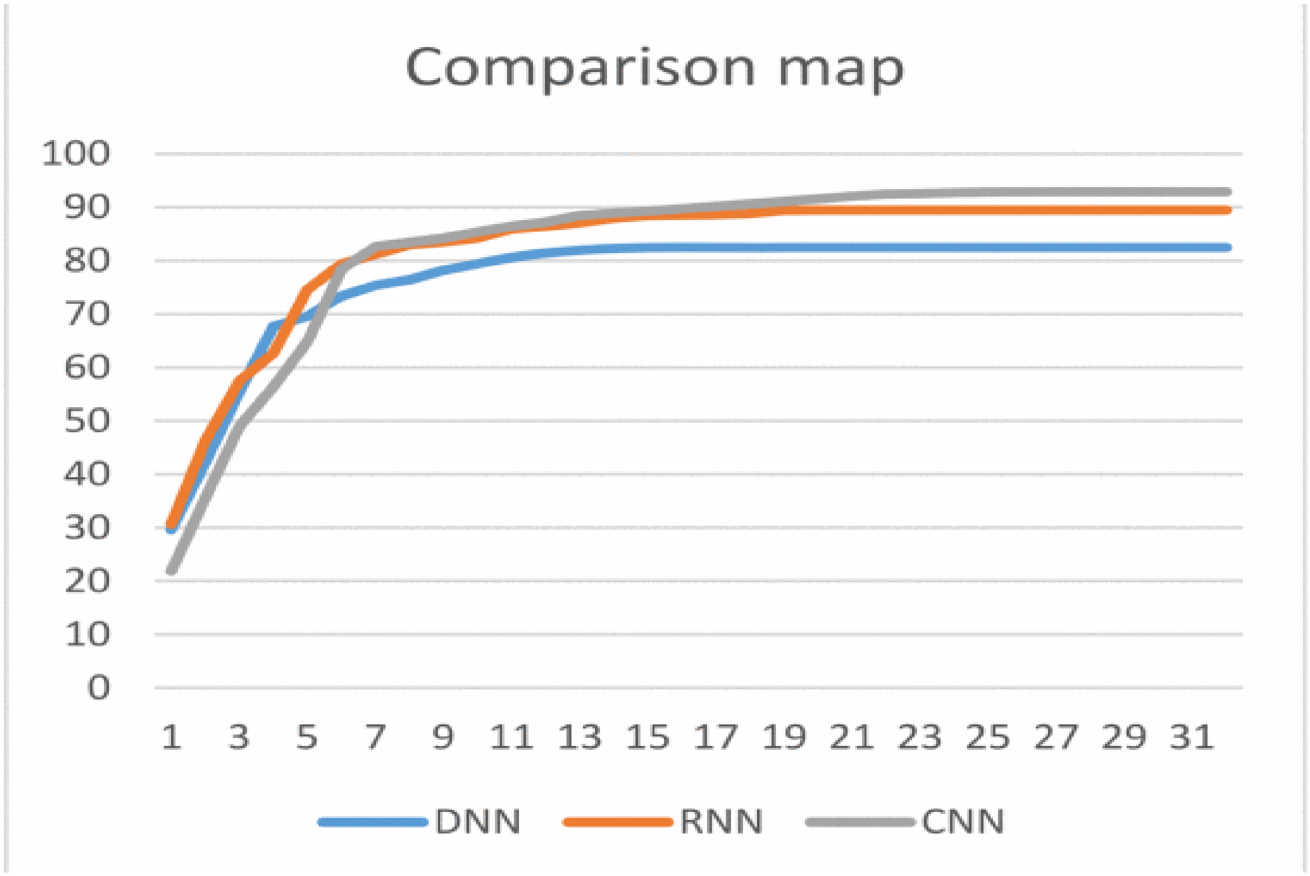


Figure 2.7: Comparison map of accuracies for different model approaches

The proposed approach is well-supported by the literature, which suggests that CNNs are adequate for speech recognition tasks. The authors also provide some insights into the architecture and hyperparameters used in the CNN model, which can be helpful for researchers and practitioners who want to apply this method to other speech recognition tasks.

However, the paper could benefit from providing a more detailed discussion of the limitations of the proposed approach and potential future directions for improving speech recognition technology. For example, it would be interesting to see how the proposed CNN-based method performs on other datasets with larger vocabularies, or how it compares to other state-of-the-art speech recognition methods such as transformer models.

In summary, the paper provides a useful contribution to the development of speech recognition technology, and its potential applications in industry settings can be substantial. Nonetheless, further research is needed to fully understand the capabilities and limitations of the proposed approach and to explore potential ways of improving speech recognition technology.

2.2.3 Related work: Speech Intent Recognition for IoT applications

In [14],M.K.M et al. presents a prototype for a smart home automation system that uses natural language processing and machine learning to enable voice-controlled operation of devices in the home. This system features an interactive and humanoid central processing unit that eliminates the need for physical devices. Residents can use voice commands to control devices such as lights, doors, and fans. Additionally, the system is capable of searching the internet for information that it does not understand. The system includes a camera that continuously monitors the home and uses image processing techniques to identify people attempting to access the house. An automated mailing system is also included that sends photos captured by the camera based on whether users are valid or invalid. The system is designed to be easy to use, cost-effective, and non-physical. However, the authors note that voice recognition accuracy could be improved. Overall, the proposed system represents a significant advancement in home automation technology, offering users a more intuitive and efficient way to control their homes.

In [21], Kumar et al. presents a Smart Home Automation system that uses Natural Language Processing (NLP) and IoT cloud solutions to remotely control smart homes in a secure and customized manner. GoogleAPI is used to integrate devices and precisely monitor home devices. The proposed NLP-based IoTCloud infrastructure involves transmitting the sound or voice of the user to a microphone, smartphone, or any other device. Appliances like AC, refrigerator, fan, or bulb receive the sound command and use communication protocols such as ZigBee, WiFi, Bluetooth, internet, etc. to send the sound command to the cloud for processing and storing commands. The sound is enhanced using various techniques and processed to identify the user's voice. If the voice is matched with the pre-trained dataset, the command is executed, and the necessary actuators work.

The paper provides a prototype of several devices and sensors that are voice-controlled. Speech intent recognition is performed on the cloud using sound processing. Overall, the proposed system offers an efficient and secure way to control smart homes using NLP and IoT cloud solutions.

In [41], Reshamwala et al. presents an idea for creating an IoT-based Home Automation system that uses hardware devices such as Arduino Uno, ESP8266 NodeMCU, and sensors, along with software applications like Google Firebase, a real-time database, to create a fully functional smart home. The proposed system allows home appliances to be controlled through a user's mobile application via the internet from any location worldwide, while minor devices and their tasks are fully automated by the system. Additionally, voice commands can be integrated into the system using Google's Voice Assistant system. However, the paper focuses only on two actions for each device in the smart home. Overall, the proposed system offers a convenient and efficient way to control a smart home through IoT-based technology, with the potential for expansion to include more complex functionalities in the future.

In [42], Rani et al. presents a project aimed at developing a voice-based home automation system that uses the Internet of Things (IoT), Artificial Intelligence, and Natural Language Processing (NLP) to enable cost-effective and efficient control of home appliances using voice commands. The NLP component of the system provides a personal touch, and the user is authenticated by providing a username and password on their mobile device. They can then issue voice commands, which are interpreted by the device and sent to the relevant appliance. The mobile device uses NLP to interpret the user's voice command. The appliances are connected to an Arduino Board, which is used to demonstrate the IoT concept, and programmed to respond to mobile inputs. The project aims to automate the operation of every appliance in the home, leading to a significant reduction in power consumption due to the avoidance of excess or wasteful use of appliance services. Overall, this project presents a promising approach to home automation, leveraging advanced technologies such as NLP and IoT to create a more intuitive and efficient user experience.

2.2.4 Discussion

In relation to the proposed research topic of Luganda Speech intent recognition for IoT applications, the reviewed papers have some gaps that are worth discussing.

Firstly, none of the papers explicitly discusses using Luganda as the language for controlling devices in a smart home. While the reviewed papers present various natural language processing techniques and models for controlling devices, they use English or other languages as the primary language of communication. This highlights the need for research to explore speech intent recognition for non-English languages such as Luganda, especially for applications in regions where Luganda is commonly spoken.

Secondly, most of the reviewed papers focus on the use of cloud-based IoT solutions for controlling devices, which may not be suitable for all environments, especially in regions with limited internet connectivity. The proposed research, which involves placing the NLP model on a Wio Terminal and communicating with ESP32 nodes to manage devices, provides a more practical and localized approach to smart home automation.

It is notable that the previous papers do not explicitly address the language of the commands used for NLP models in their respective systems. This is a significant gap as natural language processing models require a large amount of training data in a specific to accurately recognize speech and translate it into device commands. Additionally, the previous papers discuss work limited to controlling only a few devices, which limits the scope of their systems.

The current state of Luganda intent classification for IoT device control is inadequate due to the absence of a Luganda voice command dataset and a preliminary classification model. Existing solutions do not sufficiently address this gap in speech intent recognition for Luganda.

In contrast, the proposed research focuses on developing a more comprehensive Internet of Things application that is capable of recognizing speech commands in the Luganda language. This research addresses the aforementioned gaps by using a newly constructed dataset of Luganda voice commands for training the NLP model. The proposed system is also designed to control up to 8 devices based on 20 different intents, which is a more extensive range of devices and commands than those discussed in the previous papers.

Moreover, the proposed system is unique in that it is implemented on a Wio Terminal communicating with different ESP32 nodes, which manage the connected devices. This approach combines the benefits of both NLP and IoT technologies, offering users an intuitive and efficient way to control their devices using natural language commands in a local language.

In the development of a speech-to-intent model, we propose to use both the speech-to-intent and speech-to-text approach. To develop the speech-to-intent model, we propose to use mfccs as input to a Convolutional neural network as discussed by Yang et al. in [34]. Our work also proposes the use of spectrograms to make a comparison with mfccs for performance on low-resorce languages unlike in [40] and [34]. In the speech-to-text approach, we propose to use the pipeline approach and compare with the speech-to-intent model. Since there is no pretrained wav2vec model for Luganda, we propose to use a Coqui Speech to text model to fill the gap. The Natural Language Understanding model will be developed based on LSTMs to track patterns in the data.

The upcoming chapter will detail the work plan, methodology, project scope, limitations, budget, and timeline that will be utilized to achieve the project objectives.

Chapter 3

Project Work Plan

This chapter contains the objectives of the project, and how they will be obtained. the scope and limitations of the project are also discussed.

3.1 Objectives

3.1.1 Main Objective

The main objective of this project is to design, develop, and evaluate a Luganda speech intent recognition system for IoT applications.

3.1.2 Specific Objectives

The specific objectives of this project are as follows:

1. To collect and prepare a large dataset of Luganda speech commands for training a machine-learning model.
2. To train and optimize an NLP model for recognizing the intent behind different Luganda speech commands.
3. To implement the trained model into an IoT system for voice-controlled devices using the Luganda language.
4. To evaluate the system.

3.2 Methodology

A detailed explanation of how each of the specific objectives will be achieved is summarized below;

3.2.1 Objective 1: Data Collection and Preparation

To collect and prepare a large dataset of Luganda speech commands for training a machine-learning model.

Deliverable

A clean and well-labelled dataset of utterances of Luganda voice commands in .wav format labelled with their intents.

Tools

Smartphones, research papers, ffmpeg, Audacity

Approach

Luganda voice commands will be collected using smartphones. The recordings will be in the .wav format and will be collected from Luganda speakers aged between 20 to 30 years. The audio files will be recorded based on a list of transcriptions for 20 intents for 8 devices. The audio will be recorded in both noisy and clean environments to cater for the aimed working environment. The goal is to collect 9000 audio files of Luganda voice commands which will be identified on 2 slots: *action, intent*

The collected data set will be preprocessed to enhance its quality. The preprocessing techniques will include normalization, trimming, and removal of any noise. The dataset will be split into training, validation, and testing sets with a ratio of 80:10:10, respectively.

Noise will be removed with tools such as ffmpeg[1] and librosa[2] Data augmentation techniques such as the addition of white noise, pitch scaling, signal inversion, and time stretching will be applied to increase the size of the dataset.

3.2.2 Objective 2: Model Development and Optimization

To train and optimize an NLP model for recognizing the intent behind different Luganda speech commands.

Deliverable

Trained NLP model architecture, training, and validation report

Tools

Python, TensorFlow framework, Librosa, Google Colab

Approach

The prepared dataset will be used to train an NLP model based on neural networks. Mel-Frequency cepstral coefficients (MFCCs) will be the input to a neural network architecture based on Convolutional Neural Networks (CNN). Other model techniques will also be considered, including Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) as in [34]. The performance of the different model architectures will be compared.

The hyperparameters of the model will be tuned using to achieve optimal performance. The performance of the model will be evaluated using different evaluation metrics such as accuracy, F1-score, and confusion matrix.

An additional approach will be implemented, which involves using a pre-trained Luganda Automatic Speech Recognition (ASR) model to transcribe the speech commands into text. The ASR model is a Coqui Speech-to-text model[3]. The transcribed text will be used to train a text intent classification model to recognize the intent behind different commands. The classifier model will be based on LSTMs architecture. The performance of the model pipeline will be tested and evaluated.

The performance of the ASR-based approach and speech-based approach will be compared. The models will be developed in Google Colab, and the final model will be saved and converted to TensorFlow Lite (.tflite) format for deployment.

[1]https://ffmpeg.org/
[2]https://librosa.org/doc/latest/index.html
[3]https://github.com/coqui-ai/stt

3.2.3 Objective 3: System Implementation

To implement the trained model into an IoT system for voice-controlled devices using the Luganda language.

Deliverable

IoT prototype

Tools

Wio Terminal, ESP32 nodes, servo motor, dc motor, speaker, connecting wires, buzzer etc

Approach

A Luganda speech intent recognition model(saved (.tflite) model) will be placed on the Wio Terminal. When a voice command is spoken, the Wio Terminal will use its built-in microphone to capture the audio signal. The Luganda speech intent recognition model will process the audio signal and identify the specific intent of the command.

Based on the identified intent, the Wio Terminal will then send a command message using MQTT protocol to the ESP32 nodes. The ESP32 nodes, each controlling a specific device(s), will receive the command message, process it, and perform the corresponding action. The system will have 8 devices and 20 intents. These devices include a door, lights, fan, alarm, camera, TV, fridge, and speaker. Each device will have specific intents that correspond to its control options. For example, the door will have two intents, "open" and "close," while the lights will have "on" and "off." The fan will have four intents, "on," "off," "increase speed," and "decrease speed." The alarm, camera, TV, fridge, and speaker will all have two intents, "on" and "off," but the speaker will also have two additional intents, "increase volume" and "decrease volume."

To demonstrate the functionality of the system, the various devices will be connected to the ESP32 modules. A servo motor will be used to demonstrate the opening and closing of the door. A light bulb will be used to demonstrate the on-and-off functionality of the lights. A DC motor driver will be used to demonstrate the fan's on-and-off functionality, as well as its ability to increase and decrease speed. A buzzer and a small camera with an LED indicator will be used to demonstrate the alarm and camera's on-and-off functionality. LEDs will be used to demonstrate the TV and refrigerator's on-and-off functionality. Finally, a speaker, which can connect wirelessly to the Wio Terminal, will be used to demonstrate its on-and-off functionality, as well as its ability to increase and decrease volume. The system design is shown in figure 3.1

3.2.4 Objective 4: System Evaluation

To evaluate the system

Deliverables

Usability and effectiveness evaluation report recommendations

Tools

Interviews and questionnaires, smartphones

Approach

To evaluate the developed system, a user study will be conducted to gather feedback on its usability and effectiveness. The study will involve both male and female participants who are good at Luganda and those who are not so fluent to ensure a diverse set of feedback. The evaluation approach will involve both qualitative and quantitative methods to gather feedback.

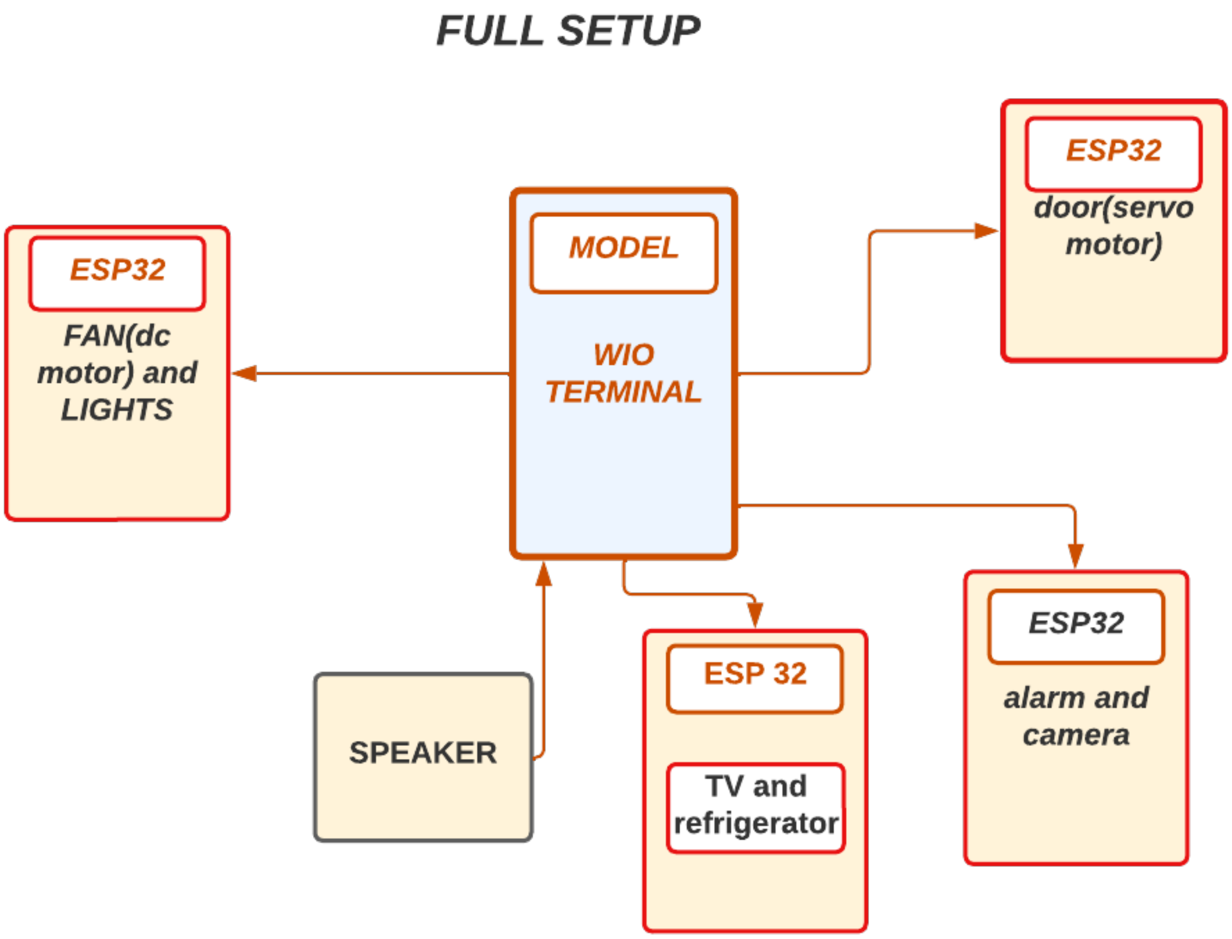


Figure 3.1: IoT Design

Questionnaires will be distributed to the participants to gather their opinions and satisfaction levels with the system's performance, ease of use, and overall experience.

After gathering feedback and analyzing the results, recommendations will be made to refine the developed system. The recommendations will be aimed at improving the system's usability, effectiveness, and overall user experience. The feedback and recommendations will be used to guide future development and adjustments to the system.

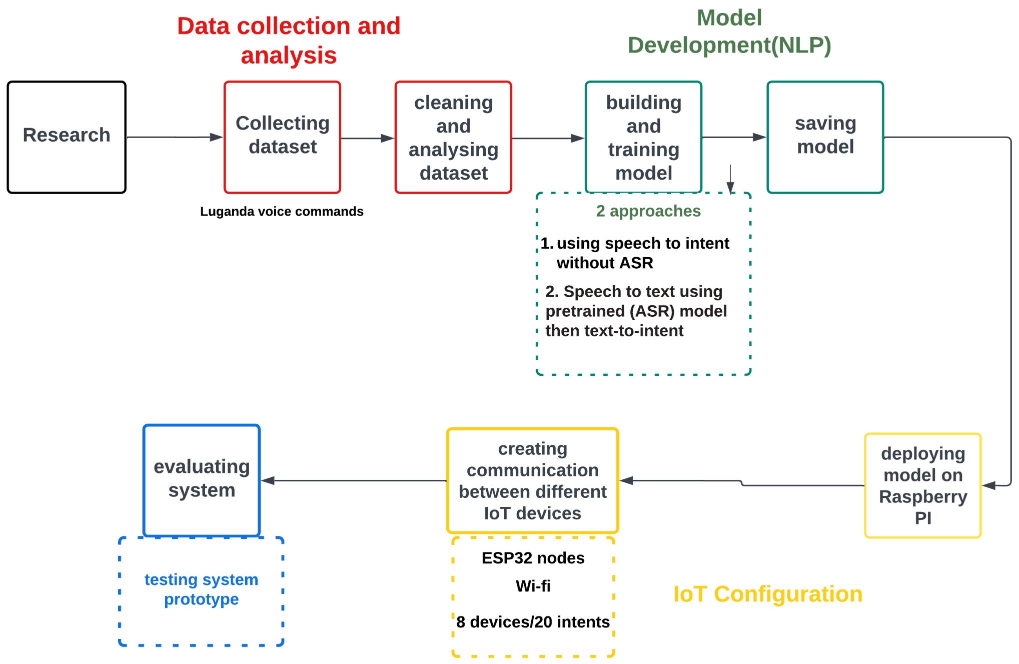

Data collection and analysis
Model Development(NLP)
Research
Collecting dataset
cleaning and analysing dataset
building and training model
saving model
Luganda voice commands
2 approaches
1. using speech to intent without ASR
2. Speech to text using pretrained (ASR) model then text-to-intent
evaluating system
testing system prototype
creating communication between different IoT devices
ESP32 nodes
Wi-fi
8 devices/20 intents
deploying model on Raspberry PI
IoT Configuration

Figure 3.2: Project Methodology

3.3 Project Scope

3.3.1 Scope

This project aims to develop a Luganda speech intent recognition system for IoT applications using machine learning techniques.

The scope of the project includes collecting a large dataset of Luganda speech commands for training and testing different machine learning models, including an ASR-based approach, and a speech-based approach. The developed system will be integrated into a Wio Terminal-based system for controlling IoT devices through voice commands in Luganda. The performance of the developed models will be evaluated using different evaluation metrics, and the system will be tested with real users to gather feedback on its usability and effectiveness.

3.3.2 Limitations

The limitations of the project include the availability of a pre-trained Luganda ASR model, which may affect the performance of the ASR-based approach. Additionally, the limited availability of Luganda speech data may affect the generalization capability of the developed models. The project assumes that the Wio Terminal and ESP32 nodes can communicate seamlessly and that the IoT devices connected to the ESP32 nodes can be controlled through voice commands. The project also assumes that there are no privacy or security concerns with the use of voice commands for IoT applications.

The constraints faced during the project include the limited availability of Luganda speech data, which may require the use of data augmentation techniques to increase the dataset size. The limited computational resources such as access to GPU may also affect the choice of neural network models and the training process. The time frame of the project may also limit the number of machine learning models that can be trained and evaluated.

3.4 Expected results

Upon completion of the project, we anticipate having a functional prototype of an IoT device network that can effectively communicate with one another and react to a particular intent recognized from audio inputs spoken in Luganda by a user through a microphone.
At the end of the project, we also anticipate obtaining a Luganda voice commands dataset, which could be valuable for future research endeavours in the field, as well as for students and researchers.

3.5 Project Budget

Table 3.1 shows the budget for the project.

Item	**Cost**
4 ESP 32 boards	280000
Speaker	50000
buzzer	5000
LEDs	5000
dc motor	10000
Servo motor	50000
bulb	5000
connecting wires	5000
Other expenses	50000
Wio Terminal	350000
total	**910,000**

Table 3.1: Project Budget

3.6 Project Timeline

The project timeline is shown in figure 3.3

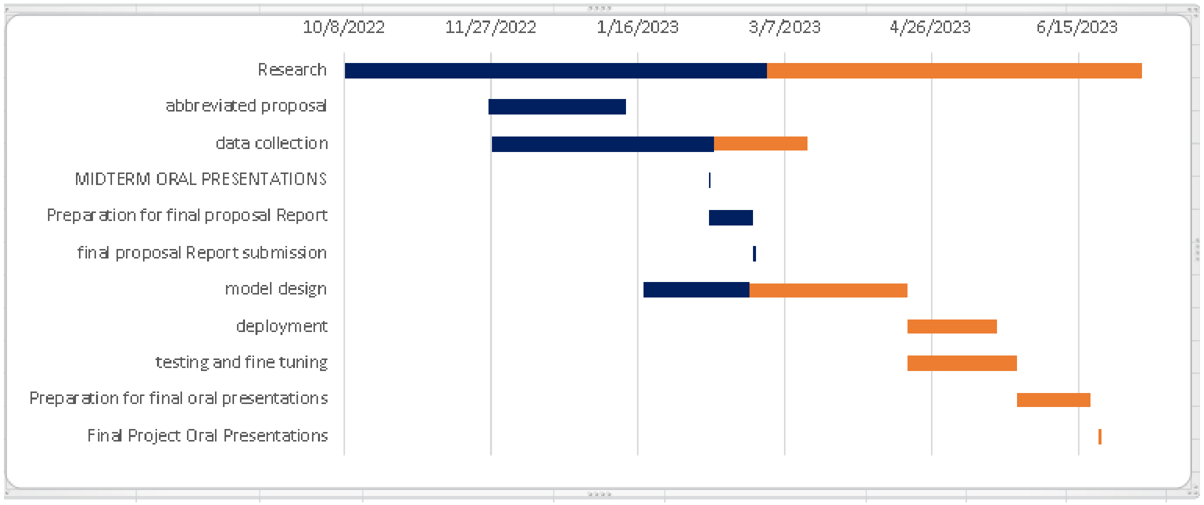


Figure 3.3: Project Timeline.

Bibliography

[1] T. Dinushika, L. Kavmini, P. Abeyawardhana, U. Thayasivam, and S. Jayasena. Speech command classification system for sinhala language based on automatic speech recognition. In *2019 International Conference on Asian Language Processing (IALP)*, pages 205–210, 2019. doi: 10.1109/IALP48816.2019.9037648.

[2] K.N. Kavmini, R. V., S. S. S., and D. R. Intelligent personal assistant - implementing voice commands enabling speech recognition. In *2020 International Conference on System, Computation, Automation and Networking (ICSCAN)*, pages 1–5, 2020. doi: 10.1109/ICSCAN49426.2020.9262279.

[3] Michael McTear, Zoraida Callejas, and David Griol. *The conversational interface: Talking to smart devices*. Springer International Publishing, January 2016. ISBN 9783319329659. doi: 10.1007/978-3-319-32967-3.

[4] T. Desot, F. Portet, and M. Vacher. Towards end-to-end spoken intent recognition in smart home. In *2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, pages 1–8, 2019. doi: 10.1109/SPED.2019.8906584.

[5] M. Mehrabani, S. Bangalore, and B. Stern. Personalized speech recognition for internet of things. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 369–374, 2015. doi: 10.1109/WF-IoT.2015.7389082.

[6] P. Wang. Research and design of smart home speech recognition system based on deep learning. In *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, pages 218–221, 2020. doi: 10.1109/CVIDL51233.2020.00-98.

[7] Amazon (Alexa). Amazon alexa voice ai — alexa developer official site, 2022. URL `https://developer.amazon.com/en-US/alexa.html`. accessed Nov. 30, 2022.

[8] How to make conversational ai work for low-resource languages, 2022. URL `https://keyreply.com/blog/conversational-ai-low-resource-languages/`. accessed Nov. 30, 2022.

[9] Language Connections. Luganda language: The tongue of buganda, Nov 2022. URL `https://www.languageconnections.com/blog/luganda-the-language-of-buganda/`. Accessed Nov. 30, 2022.

[10] New Vision Official. 20 million people can speak luganda - linguists - new vision official, Nov 2022. URL `https://www.newvision.co.ug/category/entertainment/20-million-people-can-speak-luganda---lingu` Accessed Nov. 30, 2022.

[11] Prime Uganda Safaris & Tours. Culture in uganda — culture of uganda, Oct 2020. URL `https://www.primeugandasafaris.com/culture-in-uganda`. Accessed Dec. 01, 2022.

[12] Artificial intelligence: What it is and how it is used. `https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp`. Accessed on Dec. 15, 2022.

[13] Expert.ai. What is the definition of machine learning? https://www.expert.ai/blog/machine-learning-definition/. Accessed on Dec. 16, 2022.

[14] M.K.M, K.B. Mukesh Kumar, L. Sharma, M.Z. Sayeed Pasha, and K.H.D. An interactive voice controlled humanoid smart home prototype using concepts of natural language processing and machine learning. In *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, pages 1537–1546, 2018. doi: 10.1109/RTEICT42901.2018.9012359.

[15] I. Mykhailichenko, H. Ivashchenko, O. Barkovska, and O. Liashenko. Application of deep neural network for real-time voice command recognition. In *2022 IEEE 3rd KhPI Week on Advanced Technology (KhPIWeek)*, pages 1–4, 2022. doi: 10.1109/KhPIWeek57572.2022.9916473.

[16] P. Wang. Research and design of smart home speech recognition system based on deep learning. In *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, pages 218–221, 2020. doi: 10.1109/CVIDL51233.2020.00-98.

[17] Simplilearn. What is deep learning and how does it work. https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-deep-learning. Accessed on Dec. 16, 2022.

[18] T.N. Sainath and C. Parada. Convolutional neural networks for small-footprint keyword spotting. In *Interspeech 2015*, pages 1478–1482, 2015. doi: 10.21437/Interspeech.2015-352.

[19] IBM. What are neural networks? https://www.ibm.com/cloud/learn/neural-networks. Accessed on Dec. 16, 2022.

[20] Rajeev Kumar and Vinay Sahula. Intelligent approaches for natural language processing for indic languages. In *2021 IEEE International Symposium on Smart Electronic Systems (iSES)*, pages 331–334, 2021. doi: 10.1109/iSES52644.2021.00084.

[21] S. Kumar, S. Benedict, and S. Ajith. Application of natural language processing and iotcloud in smart homes. In *2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT)*, pages 20–25, 2019. doi: 10.1109/ICCT46177.2019.8969066.

[22] T. Desot, F. Portet, and M. Vacher. Towards end-to-end spoken intent recognition in smart home. In *2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, pages 1–8, 2019. doi: 10.1109/SPED.2019.8906584.

[23] Gokhan Tur and Renato De Mori. *Spoken Language Understanding*. John Wiley & Sons, 2011.

[24] Thierry Desot, François Portet, and Michel Vacher. Slu for voice command in smart home: Comparison of pipeline and end-to-end approaches. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 822–829, 2019. doi: 10.1109/ASRU46091.2019.9003891.

[25] T. Desot, F. Portet, and M. Vacher. End-to-end spoken language understanding: Performance analyses of a voice command task in a low resource setting. *Computer Speech & Language*, 75: 101369, Sep 2022. doi: 10.1016/j.csl.2022.101369.

[26] D. Buddhika, R. Liyadipita, S. Nadeeshan, H. Witharana, S. Javasena, and U. Thayasivam. Domain specific intent classification of sinhala speech data. In *2018 International Conference on Asian Language Processing (IALP)*, pages 197–202, Nov. 2018. doi: 10.1109/IALP.2018.8629103.

[27] M. Mehrabani, S. Bangalore, and B. Stern. Personalized speech recognition for internet of things. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 369–374, 2015. doi: 10.1109/WF-IoT.2015.7389082.

[28] I. B. C. Irugalbandara, A. S. M. Naseem, M. S. H. Perera, and V. Logeeshan. HomeIO: Offline smart home automation system with automatic speech recognition and household power usage tracking. In *2022 IEEE World AI IoT Congress (AIIoT)*, pages 571–577, 2022. doi: 10.1109/AIIoT54504.2022.9817282.

[29] Seeed Studio. Wio terminal: Atsamd51 core with realtek rtl8720dn ble 5.0 & wi-fi 2.4g/5g dev board with free course. `https://www.seeedstudio.com/Wio-Terminal-p-4509.html`, February 11 2023. Accessed on February 15, 2023.

[30] Espressif Systems. ESP32 Wi-Fi & Bluetooth MCU I Espressif Systems. `https://www.espressif.com/en/products/socs/esp32`, n.d. Accessed on February 11, 2023.

[31] NEUROTECH AFRICA. Low resource languages vs conversational artificial intelligence. `https://blog.neurotech.africa/low-resource-languages-vs-conversational-artificial-intelligence` September 26 2022. Accessed on February 11, 2023.

[32] Claire Babirye, Joyce Nakatumba-Nabende, Andrew Katumba, Ronald Ogwang, Jeremy Tusubira Francis, Jonathan Mukiibi, Medadi Ssentanda, Lilian D Wanzare, and Davis David. Building text and speech datasets for low resourced languages: A case of languages in east africa. In *3rd Workshop on African Natural Language Processing*, 2022. URL `https://openreview.net/forum?id=SO-U99z4U-q`.

[33] Fluent.ai. Fluent speech commands: A dataset for spoken language understanding research. `https://fluent.ai/fluent-speech-commands-a-dataset-for-spoken-language-understanding-research/` April 19 2021. Accessed on December 2, 2022.

[34] Xuebin Yang, Hongzhi Yu, and Lei Jia. Speech recognition of command words based on convolutional neural network. In *2020 International Conference on Computer Information and Big Data Applications (CIBDA)*, pages 465–469, 2020. doi: 10.1109/CIBDA50819.2020.00110.

[35] Google AI Blog. Launching the speech commands dataset. `https://ai.googleblog.com/2017/08/launching-speech-commands-dataset.html`, August 24 2017. Accessed on December 2, 2022.

[36] Dmitriy Serdyuk, Yongqiang Wang, Christian Fuegen, Anuj Kumar, Baiyang Liu, and Yoshua Bengio. Towards end-to-end spoken language understanding. *CoRR*, abs/1802.08395, 2018. URL `http://arxiv.org/abs/1802.08395`.

[37] Isham Mohamed and Uthayasanker Thayasivam. Low resource multi-asr speech command recognition. In *2022 Moratuwa Engineering Research Conference (MERCon)*, pages 1–6, 2022. doi: 10.1109/MERCon55799.2022.9906230.

[38] Loren Lugosch, Mirco Ravanelli, Patrick Ignoto, Vikrant Singh Tomar, and Yoshua Bengio. Speech model pre-training for end-to-end spoken language understanding, 2019. URL `https://arxiv.org/abs/1904.03670`.

[39] Cheng Yi, Jianzhong Wang, Ning Cheng, Shiyu Zhou, and Bo Xu. Applying wav2vec2.0 to speech recognition in various low-resource languages. *CoRR*, abs/2012.12121, 2020. URL `https://arxiv.org/abs/2012.12121`.

[40] D. Buddhika, R. Liyadipita, S. Nadeeshan, H. Witharana, S. Javasena, and U. Thayasivam. Domain specific intent classification of sinhala speech data. In *2018 International Conference on Asian Language Processing (IALP)*, pages 197–202, Nov. 2018. doi: 10.1109/IALP.2018.8629103.

[41] T. Reshamwala, C. Shah, and S. Naik. Computerization in home: Change in way of life. In *2021 Second International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)*, pages 1–6, 2021. doi: 10.1109/ICSTCEE54422.2021.9708568.

[42] P. J. Rani, J. Bakthakumar, B. P. Kumaar, U. P. Kumaar, and S. Kumar. Voice controlled home automation system using natural language processing (nlp) and internet of things (iot). In *2017 Third International Conference on Science Technology Engineering & Management (ICONSTEM)*, pages 368–373, Mar. 2017. doi: 10.1109/ICONSTEM.2017.8261311.