Intelligent Sound Localization and Recognition with Tactile Feedback System to Assist Hearing-Impaired Individuals

by

Radia Tasnim Arony
21201209
Naomi Afrin Jalil
21201325
Maisha Raidah Chowdhury
21201560
Umma Hafsa Tahsin
21201523
Mohammad Shabab Bin Hasan
21201285

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
October 2025

© 2025. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Radia Tasnim Arony
21201209

Naomi Afrin Jalil
21201325

Maisha Raidah Chowdhury
21201560

Umma Hafsa Tahsin
21201523

Mohammad Shabab Bin Hasan
21201285

Approval

The thesis/project titled "Intelligent Sound Localization and Recognition with Tactile Feedback System to Assist Hearing-Impaired Individuals" submitted by

1. Radia Tasnim Arony(21201209)
2. Naomi Afrin Jalil(21201325)
3. Maisha Raidah Chowdhury(21201560)
4. Umma Hafsa Tahsin(21201523)
5. Mohammad Shabab Bin Hassn(21201285)

Of Summer, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on October 10, 2025.

Examining Committee:

Supervisor:
(Member)

Dr. Farig Yousuf Sadeque
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

A significant portion of the global population suffers to some extent from hearing loss and requires assistive devices. The everyday lives of hearing-impaired individuals are hampered by their inability to recognize important environmental sounds in their surroundings. This thesis proposes an innovative system that can both recognize and localize critical environmental sounds while providing tactile feedback to enhance user awareness in real time. Machine learning models such as Convolutional Neural Networks (CNNs) are utilized to classify specific sounds such as car horn, dog bark, scream, and calling bell. Furthermore, localization algorithms are integrated into a wearable device to determine the direction of these sounds. Once these sounds are detected and localized, the user is notified through vibration feedback from their smartphone or smartwatch, while the sound type, confidence level, and direction are simultaneously displayed. The proposed system is designed to be a low-cost, compact, and accessible solution for recognizing important environmental sounds and represents a meaningful advancement in the field of assistive devices by significantly enhancing the safety and quality of life of individuals with hearing impairments.

Keywords: Wearable Technology, Hearing Impairment, Audio Classification, Real-time Feedback, Sound Localization, Tactile Feedback, Machine Learning, Convolutional Neural Network (CNN), TensorFlow Lite, Mobile Applications, Wear OS, Android Development, Vibration Alerts, Accessibility Technology, Human-Computer Interaction, Envirnomental Sound Detection, On-device Inference. Smartwatch Applications

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Hearing is a very important sense for a human being to feel secure and safe in the world. A hearing-impaired individual suffers everyday greatly due to not being able to recognize important enviromental sounds such as car horn, dog bark, scream or calling bell, which often indicate danger or communication intent. While surveying for the research in "Dhaka Bodir High School" the students were asked if they ever faced security issues or problems because they could not hear a sound; in answer, all the participating students replied yes. Usual hearing aids that are available now only amplify the environment's sound but do not differentiate or locate them, which can be hard for people with hearing impairments when they are out in the streets.

Recent progress in artificial intelligence and machine learning has enabled intelligent sound recognition that can detect and classify specific sounds. Such a model is CNN, which is great for extracting temporal and spatial features from an audio spectrogram and is also lightweight, but the existing model does not target a small number of classes with high precision for each. Moreover, there are many sound recognition models like YAMNet, VGGish, and PANNs which are trained on large datasets like AudioSet but they are large and computationally heavy to work in real-time. There is also no study on the integrated approach of sound recognition and localization for real-time approach which will be easy and lightweight enough to use through android or wear os.

This research integrates machine learning based real-time sound detection with sound localization to create an affordable alternative to the usual hearing aids for the hearing impaired people. Moreover, these will be notified to the users through tactile feedback like vibration, in their Android phone or Wear OS devices, making them alert in time to avoid potential hazards or unsafe situations.

1.2 Rationale of the Study or Motivation

In recent years, real-time sound localization has been able to acheive significant attection because of its applications in assistive technologies and intelligent auditory systems. However, most commercially available hearing aids remain prohibitively expensive for individuals with limited financial resources. Typically, such devices

range in price from approximately Tk. 15,000 for entry-level models to over Tk. 500,000 for advanced systems, representing a substantial financial burden for many families.

A survey conducted at "Dhaka Bodir High School" revealed that 47.4% of respondents currently use an application or device to assist in word recognition, 42.1% do not use any assistive technology, and 10.2% expressed interest in adopting such solutions in the future. Importantly, all respondents (100%) indicated a strong desire to perceive the direction of surrounding sounds, emphasizing that unawareness of environmental audio cues poses a significant safety risk.

In addition, existing solutions for single-sided deafness (SSD) people face an additional challenge in perceiving the direction of sound sources, as audio input is received primarily through one functioning for them CROS, BiCROS, and cochlear implant systems remain cost-prohibitive. Cochlear implants start at approximately Tk. 200,000, while CROS/BiCROS devices typically begin at around Tk. 48,000, further highlighting the need for affordable alternatives.

This study is therefore motivated by the necessity to design a low-cost assistive system capable of enhancing spatial and environmental awareness for hearing-impaired individuals.

1.3 Problem Statement

Despite significant advancements in machine learning and audio signal processing, real-time sound localization systems remain computationally demanding and unsuitable for deployment on resource-constrained embedded platforms. Existing approaches primarily focus on high-performance applications rather than practical, low-cost assistive solutions that enable environmental and spatial awareness from everyday sounds for general-purpose applications. To tackle these limitations, this study offers an design and implementation of a low-power, real-time sound localization and classification system that can effectively operate on embedded hardware while providing meaningful spatial awareness cues to users with hearing impairments.

1.4 Objectives

1. Collect enough data from validated sources and the environment, process and label them for training to achieve appropriate detection

2. Study existing literature on sound detection models, such as CNN (Convolutional Neural Network), and build a new custom model for accurate sound detection.

3. Explore sound localization hardware tools that accurately localize detected sounds.

4. Make the localization and detection real-time by using an appropriate pipeline.

5. Develop app for Android and Wear OS tailored for the detection and localization of the selected sounds for hearing-impaired individuals, using Kotlin.

6. Develop a user-friendly and affordable assistive device accessible to a broad spectrum of hearing-impaired individuals.

1.5 Methodology in Brief

Apart from the background study and stakeholder survey, the thesis has four major active components: creating and cleaning the dataset, developing a model for sound recognition, developing the hardware and algorithm for sound localization, and designing a usable application combining sound recognition and localization. For the dataset, various audio files from Kaggle, Freesound, Huggingface, and audio collected by us were used. These audio files were preprocessed as needed and carefully labeled for the model training. For sound recognition, developing a compact custom Convolutional Neural Network was chosen based on the requirements of the study. The model was optimized for accuracy through many trials. For hardware setup, four microphones were used for four directions: front, back, left, and right. Input from the microphones is processed using an ESP32 microcontroller. The microcontroller computes the direction of the sound for sound localization and sends the information to the connected smart devices using the microcontroller's own wifi network. The application that combines sound recognition and localization and displays the results was developed using Android Studio to ensure optimal performance on Android smartphones and Wear OS smartwatches. The trained model was converted to TensorFlow Lite (TFLite) to make it lightweight and suitable for application. Additionally, the user interface was designed to be simple and intuitive.

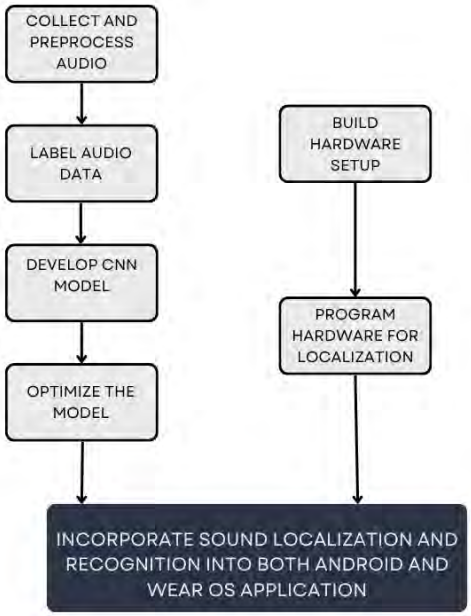


Figure 1.1: Brief Methodology Flowchart

1.6 Scopes an Challenges

The scope of this study is limited to the design and development of a low-cost, real-time sound classification and localization framework intended for hearing-impaired users. The system focuses on recognizing and localizing everyday alert-worthy sounds from indoor and noisy environments identified through a user survey and delivering directional and alert-based feedback suitable for wearable applications.

The proposed design integrates a lightweight convolutional neural network (CNN) for sound classification with analog signal processing for localization during indoor and noisy environments, intended for operation on power-limited embedded devices.

A major challenge during capturing of sounds for classification was in the microphone of the wearable device, which has been set up with a 0.5-second delay when capturing sounds immediately following a vibration feedback. This is because these vibrations are captured by the microphone of the device itself as input. This constraint introduces some loss of audio information from the environment needed during real-time scenarios.

Another significant challenge observed during the sound localization process is the effective range of the electret microphones, which are able to capture sound within a short distance and then alert the user to the direction of the sound.

Furthermore, the system is currently capable of detecting sound from only four primary directions: front, back, left, and right. Sounds originating from intermediate angles, such as between the front and right, cannot be distinctly localized or represented by the system.

Chapter 2

Related Work

2.1 Survey Methodology

Our collection of papers started with the identification of keywords that accurately addressed the topic of our research. First, we obtained a general collection of articles with the help of academic databases such as Semantic Scholar and Google Scholar. The papers were then assessed on the eligibility criteria of the articles based on their relevance to our study, publication date, publication venue, and the number of citations. We tried to collecting papers from recent years that are relevant to our research. Based on this first filtering, we have analyzed the technologies and methodologies used in the papers further to determine whether they can be applied in our research or not. This helped us to select papers that are relevant to our research. We prioritized the papers, which included electronic mapping, voice localization, or convolutional neural networks (CNNs). Also, papers which involved audio-to-tactile sensory conversion, key words extraction in speech, multi-speaker speech recognition,and recognition in spatially separated noise. Lastly, we tabulated and summarized the results in a chronological order giving a logical structure that fits our research process.

2.2 Related Work

In this research, the type of localization studied by Saxena and Ng (2009) [1] was sound localization without more than one microphone, along with an artificial pinna (outer ear). Although sound localization with the help of more than one microphones has been investigated in a thorough manner, multiple microphones tend to make the system larger and difficult to use. On the other hand, a single-microphone system is smaller, less energy-consuming, and less expensive. These authors were motivated by biological monoaural systems (one ear). They have employed a Hidden Markov Model (HMM) and publicly available sets of sounds, like the speech of humans, barks of dogs, and waves of oceans. The designs of four artificial pinna were tested, and two of them were significantly better than the rest. The most functioning design was the design with smooth grooves, and the second-best one was the design that had two sharp corners and a number of notches. This system performed optimally with wideband noises (e.g., radio static) and natural continuous noises (e.g., ocean waves); however, it had trouble with pure tones, which can also be seen with the human ear. The localization error of the wideband noise of the pinna with the best

performance was just 4.3°, although the other designs had 8.8°, 22.3°, and 42.6°.In the case of overlapping speech sounds, the best pinna, and the second best were 7.7 degrees and 19.3 degrees. In brief, with natural sounds, such as dog barks, the performance was marginally poorer with errors of 14.2 degrees and 18.3 degrees, respectively. Altogether, the artificial pinna which is the highest performing had an average localization error of 13.5, indicating that their small single-microphone model could have helped to accurately localize the sound source.

Lim et al.[2] introduced a new technique, Steered Response Voice Power (SRVP), for speaker localization in environments that are noisy. With the help of this method, human speech can be located despite background noise by devices such as TVs and humanoid robots. Conventional algorithms, such as SRP-PHAT, usually fail in these settings due to the fact that such algorithms tend to detect the sound source that is the loudest instead of the human sound source. This implies that in a noisy environment, the system may be confused by a non-human noise as human speech. This problem is addressed by The SRVP approach by integrating SRP-PHAT with Voice Activity Detection, which enables it to differentiate between human voices and other noises. The SRVP algorithm initially applies SRP-PHAT to find potential sound sources and the best ones are choosen. It then identifies the source that best fits the features of the human speech with the aid of VAD. This method will enable human speakers to be localized more precisely in noisy localities. There were two experiments performed by the researchers, the first made with simulated data, and the second with recordings of the real-world. In both instances, an array of eight microphones was circular. In the case of the simulated test, the array was positioned inside a virtual room with simulated noise sources, whereas in the case of the real-world test, it was positioned in a real-life room. There was only one loudspeaker that served as the source of voice, and a number of noise sources were positioned in different locations. It was declared that the localization was successful when the algorithm was able to identify the position of the speaker within a 5-degree range of the actual position. The tests were conducted with varying Signal-to-Noise Ratios (SNRs) between -5 dB and 15 dB, with the higher SNR having the effect of giving a clearer voice signal over the noise in the background. The findings indicated that SRVP was significantly better than the traditional SRP-PHAT algorithm in every noise level. Based on this example, at 0 dB SNR (when voice and noise are equal), SRP-PHAT did not reach a high accuracy of 18.8% whereas SRVP had 30.6% accuracy. SRP-PHAT also had 23.7% accuracy on real-world tests at 0 dB SNR compared with 42.9% on SRVP. These findings indicate that the SRVP tool is much stronger and efficient in localizing speakers in noisy conditions, in contrast to the traditional tools, such as SRP-PHAT.

Archana et al. [3] created a miniature assistant device, which assists visually impaired people to perceive the surrounding objects and get notified about nearby obstacles in the form of vibration, beep, and voice messages. The device uses an ultrasonic sensor (HC-SR04) that sends sound waves of high frequency and obtains the time to bounce back of the sound waves to determine the presence and the distance of the obstacles. An AT89C51 microcontroller receives the signals of the sensor, and in response to the direction of an obstacle detected, the corresponding alerts are activated. Two kinds of buzzers, which are continuous and intermittent

types that were applied to either show the presence or absence of obstacles on the right or left. However, the obstacles that were in the direct path activated voice notifications. The entire system was configured and tested in an environment that simulated the conditions of the real world, and the device was proven to be able to detect the presence of an obstacle at a distance of 0.3 to 4 meters and in a field of view of 60 degrees. The findings showed that the device is useful in improving spatial perception and mobility of the users since it precisely detects the direction and the proximity of the objects around them.

Audhkhasi et al. (2017) [4] created a new end-to-end (E2E) keyword search (KWS) model that is capable of finding keywords directly on a speech without utilizing automatic speech recognition (ASR). This method is much faster than the traditional ASR-based KWS systems; no training data input in the target language is needed, and it can be trained on multilingual datasets to enhance its performance on multiple languages. The model can identify the in-vocabulary (IV) and out-of-vocabulary (OOV) words. It is composed of three primary parts: (1) a recurring neural network (RNN) that ciphers the acoustic input into a vector, (2) a mixture of RNN and CNN that encodes the text query and (3) a comparison module that identifies the instances of the key word by matching the encoded audio with the text query. It is important to note that the RNN does not need to be trained on transcribed speech or target language data. The researchers tried a number of techniques to reduce the MSE of the acoustic autoencoder and discovered that a batch size of 80 had the least MSE. Also, the frame stacking and the decimation strategies were used to decrease the computation and training time. The ASR-free KWS model was superior because it detected more OOV words with 70.77 percent accuracy, as opposed to 50 percent accuracy of the ASR-based systems, which, at times, were unable to detect the OOV words at all. Despite the fact that the ASR-free model was slightly inferior for the IV words, it was a better improvement compared to the previous methods. The paper has also investigated how the model can determine the time that a keyword appeared, which shows that the model can resolve time-localized keyword searches.

Detection and localization of speech in confined areas are suggested by Vecchiotti et al. [5] as a fully data-driven method with the help of a joint neural network called the Joint VAD-SLOC Model. This system implements both Neural Voice Activity Detection (VAD) and Neural Speaker Localization (SLOC) into one system. The cascade architecture is used with a Neural SLOC that has been trained with an Oracle VAD, which only chooses the speech components of an audio signal to assist in the assessment of localization performance. The model uses a Convolutional Neural Network (CNN), which receives two different input features -Log-Mel and GCC-PHAT processed in two stacks of convolutional layers and then dense layers producing three features: a voice-activity prediction and two speaker-position coordinates predictions. Neural VAD takes binary labels to recognize non-speech and speech signals, whereas the Neural SLOC provides coordinate values (between 0 and 1). This model applies ReLU and Hard Tanh activation functions, is implemented in Python with Keras, and trained with backpropagation and Adam optimizer, early stopping, and varying learning rates. The researchers used the DIRHA dataset for evaluation, which was recorded with 40 microphones spaced 50 cm between and dis-

tributed in five rooms. All microphones were used to extract log-Mel features used in VAD, and pairs of neighboring microphones were used to extract GCC-PHAT in SLOC. The training was based on a 10-fold cross-validation structure where 8 folds were taken as training, 1 fold as validation, and 1 fold as testing. The metrics that were used to measure the performance were deletion rate (Del), false alarm rate (FA), root mean squared error (RMSE), speech activity detection (SAD) in VAD, and and P(cor) in localization accuracy. It was found that the proposed joint model resulted in a 33% reduction of the average SAD error over existing methods, and even though the RMSE did not change by much, only 2.7% with respect to the standalone Neural SLOC, the system as a whole produced more accurate and integrated speech detection and localization.

Mahamud and Zishan [6] came up with a wearable device that can assist hearing-impaired people in communicating more conveniently since spoken language is converted into readable text in real time. The system operates on a mobile application that captures the voice and translates it into text via Google's speech recognition technology and transmits the text to a wearable device with an LCD screen via Bluetooth technology. The text converted is then displayed on the screen so that the user can read the text being said in real time. The hardware that will be used in the system consists of an Arduino Uno microcontroller, a 16x2 LCD screen, and a Bluetooth module (HC-05). The Arduino Uno is the processing unit that receives and decodes the text information and transmits that to the display. The wearable device is user-friendly, small, lightweight, and uses a 5V power supply, and is resistant to external interference. The corresponding mobile device, called ANKON, allows connecting with the wrist device without any issues and transfers data easily. It has a small yet crisp display, which can display 16 characters in a line and can scroll longer messages, thus it can be used in long conversations. Long sentence such can be displayed on the screen to allow the users to read it in a comfortable manner. The system was accurate during simulations and real testing, where speech was translated into text, and it was also shown in real time. The developers are also aiming to extend the system to accommodate several languages in the future, such that users can use their first language. In general, this device provides a convenient, cheap, and effective solution to enhance the everyday communication of hearing-impaired persons.

In a study conducted by A. H. Michaely et al. [7], the focus was on a two-tiered Keyword Spotting (KWS) system that would enhance precision by ensuring that the number of false accepts is reduced to a minimal degree, and the number of false rejects is also minimized. KWS systems are popular in hands-free voice activation, and they recognize certain trigger phrases, like "Hey Google" or "Alexa". In this method, the core KWS works in the local area on a device that has limited computing capacity to identify possible trigger phrases. Whenever this kind of phrase is detected, the device transfers an entire audio segment, such as the trigger and the query that follows the trigger, to a secondary KWS system on the server. On-server KWS platforms rely on bigger Automatic Speech Recognition (ASR) models and sophisticated algorithms to examine the audio on a finer scale. Then it decodes the query as well as the trigger with the help of a context-based recognizer, and in the case that the trigger phrase is inaccurate, it blocks any query and eliminates false

accepts. The process enhances the system to accommodate the speech variations like the coarticulation effect and background noise. Other functions are the detection and elimination of the trigger phrase during transcription, thereby guaranteeing better processing of queries, and the second pass re-analyses on the server to further narrow the results. This approach was found in field tests to cut false accepts by 89 and the false rejects only increased by 0.2 percent, a very negligible trade-off to the accuracy gain. Besides, it enhanced WER in ASR by 10-50, and therefore, it became more effective in various languages and settings. In general, the suggested system has a solid and scalable enhancement of the conventional KWS systems.

Yağanooglu and Kose [8] thought of a wearable device that can be used to detect important environmental sounds in real time to help the hearing-impaired individuals identify which sounds can influence their safety or their daily life. The idea behind the device is aimed at improving the daily lives of such people by notifying them of significant sounds like the **doorbells, alarms, telephone ringing, and car honks using a specific pattern of vibration. Its parts consist of four primary elements, namely a Grove shield, a microphone, a Raspberry Pi, and a vibration motor. There is a microphone that records the sounds around, and the Raspberry Pi processes it through the application of audio fingerprinting to determine the type of sound. After the recognition of a sound, the device will vibrate a particular pattern in order to alert the user about it. Training and testing of this system was based on a dataset of 10,000 sound samples in eight categories using techniques like K-Nearest Neighbors, Mel-Frequency Cepstral Coefficients, Line Spectral Frequencies, Zero Crossing Rate, Log Area Ratio, Audio Fingerprinting, and Linear Prediction Coefficients. The audio fingerprinting technique operates in that a sound gets a unique fingerprint based on its frequency peaks, so that new sounds can be compared to the existing fingerprints in order to correctly classify the sound. The device was high-performing with a score of 99% accuracy on the simple sounds (such as phone rings and alarms) and 91-93% accuracy on complex sounds (such as car horns) and gave an alert to the user within the timeframe of 1.9 seconds. Even though there was a minimal reduction in the accuracy in noisy conditions, it was still better than other sound recognition systems. Indoor and outdoor testing confirmed the reliability of the device, and user comments added that the device is capable of significantly advancing the safety and awareness of hearing-impaired people.

Seki et al. (2018) [9] have suggested an end-to-end sequence-to-sequence model that can segment and identify multiple speakers in a single input channel without requiring separate stages of speech recognition and source separation. In comparison to the previous models that used isolated source data to make references, their model does not need these data but instead uses a characteristic objective function that enhances the difference in hidden vectors of various speakers and thus performance. Conventionally, speech recognition involving multiple speakers was done in two different steps; the first step involved the separation of the mixed audio, followed by the recognition of the speech of each speaker. Other past end-to-end models integrated the two tasks, but nonetheless needed source references that were isolated. Conversely, the framework by Seki et al. combines both tasks with an architecture that is an **attention-based encoder-decoder** and straightforwardly maps label sequences to mixed speech inputs. The encoder transforms the input

into hidden representations, and an attention mechanism summarizes into a context vector, highlighting the important timeframes in which to predict labels. In order to deal with several speakers, the technique of Permutation Invariant Training (PIT) is used to make sure that every prediction sequence of labels is correctly associated with a corresponding speaker, even when the order of speakers changes. The decoder is an independent decoder of the hidden representations of one speaker, and an extra task based on Kullback-Leibler divergence is used to increase the contrast between the hidden vectors of different speakers. The two tested architectures were the splitting of hidden vectors by two architectures: VGG and Bi-directional Long Short-Term Memory. The mixed-speech data based on the **WSJ (English) and CSJ (Japanese) corpora were experimented with in terms of the combination of utterances of two speakers with varying signal-to-noise ratios. In single-speaker (unmixed) data, the model obtained a Character Error Rate (CER) of 2.6% in English and 7.8% in Japanese, which is the lower bound of the performance. The baseline (no hidden vector split) gave an 83 percent CER on English in mixed-speech conditions, whereas the VGG split and the BLSTM split decreased it to 16.5 percent and 14 respectively. The highest score of 13.7-14.9 percent CER on English and Japanese, respectively, was attained by the **BLSTM split with the loss of KL divergence. Generally speaking, the proposed model worked better than the traditional systems based on separate separation (DPLC) and recognition (ASR) modules, providing a rate of Word Error Reduction (WER) by almost 2.6 percent, and shows that the end-to-end approach is effective in dealing with multi-speaker recognition without special source separation.

Fletcher et al. [10] conducted a study on electrohaptic stimulation for Cochlear Implant (CI) users to demonstrate that providing haptic feedback through the skin can improve speech intelligibility in noisy environments, a situation where CI users typically face significant challenges. In the experiment, ten participants participated in the experiment where they encountered speech perception tests during the pre-training sessions and post-training sessions. The testing was done with two different speech corpora: the BKB Institute of Hearing Research corpus, and the training was carried out using the RealSpeech (UK) content library. The RealSpeech content was a combination of various general-interest stories and non-portable multi-speaker sounds, which were recorded by the National Acoustics Laboratories. These speech and noise signals were converted into tactile sensations using an elaborate signal processing pipeline. The frequencies of the tactile signals were 50230 Hz, which are appropriate for low-power haptic devices. The audio was reduced to 16 Hz and run through a 4-channel Finite Impulse Response filter bank, further subdivided equally on the Equivalent Rectangular Bandwidth scale. Afterwards, a Hilbert envelope was computed and low-pass filtered channel-wise. These envelopes varied the strength of four tonal carriers that were fixed, and then subjected to a function known as an expander to reduce noise and improve other speech-related temporal modulations - in essence, speech cues become more distinguishable by feel. The experimental apparatus was set up in a sound-insulated booth. The generation and control of stimuli were controlled through MATLAB scripts that were run on a laptop in a separate room. The soundcard was an RME Babyface Pro and the system was Genelec 8020 PM Bi-amplified Monitor System with the calibration being done by Brueel and Kjaer Type 2250 sound level meter and Type 4231 Type 4 sound cali-

brator. The vibrotactile feedback was realized through two HVLab vibrometers on the basis of the tactile modification. The results were recorded over four sessions (two of these were testing and the remaining two training), a mean of 8.3 percentage points of speech intelligibility was observed with haptic feedback. Though this was a little less than the rest of the studies (with reported rates of about 10.8%), the difference was thought to be caused by the position of the vibrometers, length of the test, and demographics. The minimum and maximum percentage points gained by individuals were 0.5 percentage points and 21.8 percentage points, respectively. Interestingly, the participant who improved the most had a prior history of tactile training through playing the flute, implying that prior knowledge of fine tactile stimuli may have an effect on being more adaptable to electrohaptic feedback. On the whole, the results affirmed that haptic stimulation has a beneficial effect on CI users to improve their speech perception in noisy conditions and, consequently, their communication and quality of life.

Chang et al. (2020) [11] performed the study with the help of a neural sequence-to-sequence (seq2seq) structure that identifies and isolates speech within a multi-channel, multi-speaker input system, producing multiple output text sequences of various speakers. The experiment was based on the reworked version of the WSJ1 dataset, which is used to train speech multi-speaker and includes overlapping speech and background noise of different speakers. The suggested model, which is implemented in PyTorch and is based on the ESPnet framework, is made up of three big layers to separate and identify speech and microphone array inputs. To begin with, in the multichannel input, each speaker and channel is modeled independent of the others by estimating a time-frequency mask with Short-Term Fourier Transformation (STFT) to form a mask each of the speakers and an extra mask of background noise. They are calculated using the masks to compute beamformer filter coefficients at each frequency in every speaker using the PSDs of these speakers and the noise as second noise, respectively. Applied to the inputs of the multichannel, multi-speaker filter banks are these neural beamforming filters which give the outputs of the system one noise-free channel per speaker. Lastly, the beamformed outputs are first normalized and then sent to a sub module of end-to-end speech recognition which uses Permutation Invariant Training (PIT) to assign correct input-reference sequences. The model was further expanded by taking into consideration curriculum learning as a way of improving performance in training. The system was also trained using multichannel, multi-speaker data, as well as on single-speaker, single-channel data, to be robust at both configurations. Experiments, especially on the case of two speakers and two microphones, showed that the conventional beamforming algorithms failed when speech overlaps, and that the proposed MIMO speech model worked better than the baselines with Character Error Rate (CER) of 3.75 percent and Word Error Rate (WER) of 7.55 percent on multi-speaker recognition in anechoic environments. When no curriculum learning was implemented, the performance went down to approximately 12% CER, which indicated its significance in training convergence. Nonetheless, curriculum learning did not produce a significant effect on speech separation quality in addition to the fact that it did improve recognition accuracy. Assessment scales like Perceptual Evaluation of Speech Quality (PESQ) and Signal-to-Distortion Ratio (SI-SDR) were used to show that the neural beamforming module was indeed able to distinguish between speakers, thus making

the MIMO speech model an extremely useful tool in the multi-channel multi-speaker speech separation and recognition.

Shen et al. [12] have suggested a method of localizing human or objects in an environment based on algorithms which uses the estimation of the Angle of Arrival (AoA) using a single microphone array. The main problems that were solved included determination of AoA without knowledge of the original signal and properly tracking these AoAs to find the location of the user. Their system was mainly concentrated on two things, namely, the accurate AOA detection and joint wall geometry estimation. In order to have high accuracy in AoA detection, they proposed a new algorithm, Iterative Align-and-Cancel (IAC) algorithm, which executes both the alignment and cancellation (as opposed to the old methodology which only does the former). The IAC algorithm is useful in eliminating repeated correct AoAs, and delayed AoAs due to echoes or reflections, which means only direct paths are taken into consideration. The system architecture is based on two AoAs (AoA1 and AoA2) derived based on the raw audio input and three geometric parameters: wall orientation, wall distance, and user height. A basic step that is taken with these parameters is to run them through a fusion model to generate the correct location estimations. To apply the solution to practice, the researchers employed a Seeed Studio 6-microphone array that records at 16 kHz which was connected to a raspberry Pi and a laptop to process the data. Every one of four indoor settings- a studio apartment, kitchen, student office, and large conference room had experiments under controlled conditions where the stimulus remained at locations not exceeding a range of 4-5 meters and did not have any significant barriers. The data was comprised of 2,350 voice commands and non voice sounds. Comparison of performance indicated that the generating algorithm was considerably better than the traditional algorithms including Delay-and-Sum, MUSIC, and GCC-PHAT algorithms in the AoA detection accuracy. On the whole, the suggested system was able to reduce median localization error by 52% relative to GCC-PHAT with an overall median error of 0.44 meters. There were results that showed that the localization errors became greater in the bigger space such as the conference room and that the object localization was simpler in comparison to human localization. Moreover, the system did not react to the changes in the voices and speaking patterns of the users, but was sensitive to the clutter density and background noise- denser environment and continuous noise caused larger median errors. The suggested IAC algorithm is therefore a major breakthrough in the estimation of AoA and sound source localization in indoors with single-microphone.

Haoheng Song et al. [13] proposed a new system named as electro-haptic stimulation (EHS) system which offers a combination of electrical auditory stimulation provided by cochlear implants (CIs) with haptic feedback to the wrists so that hearing-impaired individuals will have better speech perception. Despite the fact that cochlear implants go a long way in restoring hearing, CI users are still experiencing severe difficulties in noisy/ multi- talker situations where the speech and background noise are hard to differentiate. To cope with it, the researchers developed a system that imparts the missing speech and localization hearing data with the help of the extra haptic clues, which enriches the acoustic output of the CI. The suggested approach presents a wearable neuroprosthetic device that is non-invasive

and inexpensive and can be easily combined with the current CI technology. This haptic feedback provides tactile signals synchronized with,audio feedback that improves the ability of users to discriminate and isolate speech among an array of other competing sources of noise. In experimental assessments, there were significant improvements in speech understanding by CI users after 30 minutes of training and especially when the sound and speech had different spatial origins. The research has noted an average improvement of 2.6 dB when noise and speech came from opposite sides and 2.8 dB in speech reception threshold (SRT) when noise and speech came from the same side. The gains were also repeatedly reported in the participants presented with three dissimilar CI systems suggesting that the EHS approach is also flexible and is compatible with other cochlear implant technologies. Altogether, this new approach proves to be one of the major improvements of CI performance as it suggests that a combination of auditory and haptic stimulation can considerably increase the speech intelligibility in noisy real-life scenarios, eventually leading to improved communication skills and the quality of life of hearing-impaired.

The paper by Kim et al. [14] has covered in detail the various components, algorithms, and optimization of end-to-end (E2E) speech recognition systems especially on-equipment implementation. The traditional speech recognition systems usually need various independent modules, including the language model, acoustic model, pronunciation model, and a Weighted Finite State Transducer (WFST) decoder. Conversely, systems based on end-to-end neural speech recognition are more effective in large datasets of data than WFST based decoders or n-gram language models and are additionally better in performance based on the subword units in Byte Pair Encoding (BPE) subwords. Gated Recurrent Neural Networks (RNNs) have a smaller memory footprint than other E2E models, and support streaming when used in the unidirectional mode, whereas models based on Convolutional Neural Networks (CNNs) use nonlinear rectifiers and groups of filters but do not need to compute the prior time steps to compute the current one, so can be more efficiently realized with Single Instruction Multiple Data (SIMD) support. Connectionist Temporal Classification (CTC) with stacked layers of neural networks, CTC loss, and parameters, and Recurrent Neural Network Transducer (RNN-T) which introduces explicit feedback path on the output label to its prediction network to overcome the constraints of CTC, attention-based models, and Monotonic Chunkwise Attention (MoChA)-based models (use a combination of hard monotonic attention and soft chunk-wise attention) are described as E2E architectures. The easiest model compression method is TensorFlow Lite 8-bit quantization, another method is Low-Rank Approximation (LRA), and pruning is the most successful when hardware is available; in other cases, knowledge distillation and factorization should be used depending on the size of the model. In order to enhance the performance, Kim et al. proposed non-streaming mode combination, shallow fusion of language models (LM) and improving Named Entity Recognition (NER) performances using spell corrector.

Korayem et al. [15] suggested a low-cost voice recognizing system based on Hidden Markov Models (HMM), the aim of this technology was to take individual word rates and apply them to the robotic systems, which had the capability of demonstrating facial gestures and movement. The research was meant to realize functionalities of complex systems at lower computational cost. The system was put into practice

after identifying the voice commands on two robotic platforms a robotic face that would show the expressions and a scout robot that would physically move towards the sound source thereby introducing a crude voice localization. They experimented with a small non-native English speech dataset that was developed by the Persian speakers, and included 10 English command words that they used: ready, forward, back, left, right, stop, move, faster, slower, go. Each speaker was recorded three times saying the words, making 18 speakers, each recording 540 total files of 16,000 Hz bearable audio labeled as .wav. The data was split into training and test groups where the individuals in both groups were dissimilar. The HMM-based recognition system consisted of three major steps, which include; feature extraction, acoustic modeling, and decoding. As features, Mel-Frequency Cepstral Coefficients (MFCC) were obtained, and in the case of acoustic modeling, each voice command was divided into 25 ms frames, and one HMM per word was trained (10 HMMs, total) (10 HMMs control words, and a single HMM silence). Word arrangements were defined by the use of dictionary, and the decoding was undertaken by the use of Viterbi algorithm. In voice localization, HARK (a voice localization package) was used and voice synthesis utilized ROS middleware to enhance human-robot interaction. The system proved to be quite impressive as it was able to recognize trained commands in 0.9 seconds with a word error rate of 0% among English speakers. But in case of non-native (Persian) speakers the accuracy dropped to 85.16, and this demonstrates that the system is sensitive to the pronunciation differences.

Fletcher et al. [16] investigates whether haptic stimulation of three body parts, including lower triceps, palm and dorsal wrist, can offer tactile intensity differences (TIDS) to supplement cochlear implant (CI) electric stimulation in individuals with hearing disabilitiest. Although prior papers primarily targeted fingers tips/palm areas, this study investigated the wrist and triceps since they were found to be more convenient and less prone to interference with devices and movement distortions. The triceps lower part was considered as one of the potential options due to discrete positioning and ease of self-fitting. A total of twelve were screened against HVLab Vibrotactile Perception Meter that was equipped with a 6 mm contactor which was calibrated using BK exciter. The testing phase of the experiment involved using custom MATLAB scripts to control a Max 8 Patch to measure the threshold with stimuli presented using an RME Babyface Pro soundcard and two HVLab tactile vibrometers with built-in accelerometers. A set of amplitude-modulated and unmodulated stimuli were used, based on a dataset of the University of Southampton Research Data Management Repository, and by a similar protocol as earlier studies of Fletcher et al. Signal detection involved a three-alternative forced-choice (3AFC) paradigm and TID discrimination involved a two-alternative forced-choice (2AFC) paradigm, using a two-down one-up adaptive strategy. The statistical analysis of the detection and discrimination thresholds was performed using repeated-measures ANOVA (RM-ANOVA). The results were 55.6 dB average dynamic range of the dorsal wrist, 57.6 dB of the palmar and 54.3 dB of the lower triceps. The TID detection threshold of the dorsal wrist, palmar wrist, and lower triceps were 0.0114 and 0.0091 m/s 2, and 0.0116 and 0.0137 m/s 2 respectively. Generally, the discrimination threshold of TID was about 1 dB at all sites, which compares to binaural auditory discrimination in normal-hearing subjects, and the palmar wrist was a little more sensitive (by about 0.3 dB).

Hanan Aldarmaki et al. [17] suggested a speech recognition (ASR) system that is unsupervised and can transcribe speech to text without using large labeled datasets or annotations made by humans. The purpose of this approach is to aid low-resource languages or languages that do not have a written form by learning directly on speech signals in raw form. The processes used by the researchers to deal with the problem of low-resource ASR and human language acquisition included speech segmentation, feature extraction and embedding, acoustic unit discovery, cross-modal grounding, unsupervised sub-word and word modeling, iterative refinement, and pseudo-supervised learning. The combination of these elements in their system substitutes known supervised training with non-supervised ones to create effective ASR models to languages with limited resources. The phonetic segmentation model performed very well with a top boundary F-score of more than 80 percent which was a significant advancement in unsupervised speech processing. Nevertheless, the segmentation of words is more complicated as there is still research on how to use collocations, pauses, and syllabic segmentation to further establish boundaries of words. Cross-modal mapping models such as Generative Adversarial Networks (GANs), were also used to make promising progress in the context of focusing on alignment of speech and text segments without any explicit supervision, and achieved a word error rate of as little as 11% in certain models. These results show that there is much to be done in closing the disparity between unsupervised and supervised ASR systems. A future research roadmap that highlights the combination of bottom-top phonetic modeling with top-top semantic learning, better word segmentation, and better speaker-independent embeddings is a conclusion of the study on robust unsupervised ASR performance.

Chen et al. [18] proposed a system named Voicemap which is developed to autonomously complete voice localization by having a collaborative relationship between a sweeping robot and voice equipment. The sweeping robot which acted as an appliance system in a smart home searched through its surroundings to map it and identify the positioning details of the voice devices present in the environment. In this way, the system would be able to locate the voice sources on the map. Localization of the robot itself was done using the microphone array, which formed a positional relationship between the robot and the voice devices. The principal issues were the correct positioning of the constantly moving robot, the isolation of clean voice samples with the noise produced during the operation of the robot, and the correspondence of the coordinate system of the robot with the microphone array. To resolve these problems, the researchers adopted an inertial-based super-resolution technique to compute the Angle of Arrival (AoA), which incorporated the data of object motion with AoA estimation to improve its accuracy by the use of time-frequency analysis. The filtering process was created to locate the intersections of two or more peaks taking into account the trajectory of the robot. Once AoA estimation was completed, the team aligned the map that the SLAM generated with the AoA data to bring the coordinate system of the moving robot and the microphone array into correspondence. After the location of the device was obtained in the SLAM map, precise voice localization was realized. A COTS SLAM robot, Seeedstudio ReSpeaker 4-microphone linear array, and 6-microphone circular array with a Raspberry Pi 4 Model B were used to assemble the prototype, and the sound sam-

pling rate was 48 kHz. The AoA estimation of the system reached the localization error of 0.12 m, which was by the margin 43.9% lower than the state-of-the-art, and the mapping performance was 0.12 m and 0.13 m in two test conditions. In general, the proposed model had greater accuracy in localizing human voice compared to the Symphony and the MAVL models with a median localization error of 0.22 m.

Fletcher et al. [19] examined the possibility of using multiple frequency maps to improve tactile stimulation to phoneme discrimination using spectral information, the frequency content of sound, transferred by the use of tactile movement on the wrist. An audio-to-tactile vocoder signal-processing technique similar to that of Fletcher et al. [3] was used in the study with a frequency range of 50-7000Hz; this was the range of frequencies of the tactile system and was extended to achieve a better sensitivity range. The tactile messages were sent to one of the dorsal-wrist locations to ensure a maximum number of differences in voicing and place of articulation. The study implemented a custom-built haptic stimulation rig in an isolated room with twenty-six participants that were conducted by the EHS Research Group. The system consisted of a Ling Dynamic Systems V101 shaker, fitted with a 3D-printed circular probe with a diameter of 10 mm, driven by a MOTU UltraLite-mk5 sound card, RME QuadMic II preamplifier and HVLab Tactile Vibrometer power amplifier. The measurements of the vibration levels were taken with a BK 4533-B-001 accelerometer and a BK Type 4294 calibration exciter. The participants were given a three interval three alternative forced choice phoneme discrimination task in which pairs of phonemes were given to them in the form of tactile stimuli of a set of 53 phoneme pairs. The experiment had five conditions namely, (i) one frequency band and one vibrotactile tone (1FB1T), (ii) one frequency band and four vibrotactile tones (1FB4T), (iii) four frequency bands and four vibrotactile tones (4FB4T), (iv) one frequency band and eight vibrotactile tones (1FB8T), and (v) eight frequency bands and eight vibrotactile tones (8FB8T). It was found that performance with four out of ten bands was better by an average of five and nine percent better than that with a single band. Consonants did better than vowels, where the means of the four and eight band consonants and vowels were improved by 8.6 and 14.8, respectively. In general, the scores of participants varied with the gender of the speaker: participants of the male gender had 43.4% and 44.6% to distinguish tactile phoneme with consonants and vowels, respectively, whereas female participants had 51.7% and 45.7% with the same, which showed that multiple frequency channels could significantly contribute to the phoneme discrimination in the tactile channel, especially in consonant sounds.

Mark D. Fletcher et al. [20] introduced a more sophisticated noise reduction and speech enhancement dual-path recurrent neural network, DPRNN, that is used to create noise-reducing and speech-enhancing systems in hearing-assistive devices, such as those used by cochlear implant users. They had a behavioral experiment of 16 adults aged between 18 to 37, with non-stationary noise, i.e. party noise with an ILTASS spectrum and stationary noise which had been filtered to correlate with ILTASS. The DPRNN divided speech into blocks that are subjected to the bidirectional or unidirectional LSTM layers and used mask to remove noise to get clean speech. To process online, a quasi-causal DPRNN was used with a chunk size of 20 resulting in a capability to look 5.75 milliseconds in the future, which is appropriate to real time hearing-aid tasks. The Hilbert transform was used to extract speech am-

plitude envelopes which were filtered with a zero-phase 6 th order Butterworth filter with a 23 Hz cut-off to eliminate silences and stabilize signal-to-noise ratio between successive sentences. The vibrotactile stimulation was administered with the use of a Ling Dynamic Systems V101 shaker placed on the back of the wrist and the tactile tones were presented in the same frequency, and varied in detectability depending on the individual thresholds. Quasi-causal, non-causal, and causal variants of the DPRNNs were trained with big data to guarantee that they would be stable in the presence of different noise levels and SNRs. Although the classic log-MMSE techniques are effective in stationary noise, they are inadequate in non- stationary, real world situation such as by multi-talker conditions, but the DPRNN especially the quasi-causal and non- causal type obtained better noise reduction. This, together with an optimized system of tactile vocoder, improved the noise management and tactile speech perception of hearing-impaired users.

Chapter 3

Impacts

3.1 Final Requirements and Specifications

From surveying 19 students from a government high school (a school for hearing-impaired students), it was clear that making a system that works on high-powered devices would not be useful for underprivileged people with hearing disabilities. The survey showed that 15 out of the 19 students used smartphones, and none of them used Computers, laptops, tablets, etc. In practical use, smart watches are very useful for getting tactile feedback for important notifications, as it is always attached to the user's body. Multiple microphones are necessary for sound localization. Commonly used smartphones do not have a multiple-microphone setup that is needed for sound localization. Hence, the study needed to include a hardware setup for sound localization.

Functional requirements for the study are that it should be accurate in sound localization and recognition. It should also be smartphone and smartwatch-compatible. Non-functional requirements are that the solution should be low cost, it should process and give results in real time, it should be lightweight, and it should be easily usable for hearing-impaired people.

Based on the stakeholder survey and requirements, many specifications were found. The four classes for sound recognition were finalized using the survey results. The system was incorporated into an Android application, as Android devices are more widely available at a comparatively cheaper price. ESP32 microcontroller was used for sound localization because of its cooperatively low price. The CNN model was chosen to make the solution lightweight. Preprocessing audio was put inside the model to cut down latency for better real-time performance.

3.1.1 Application Configuration and Technical Specifications

The app for both Android and Wear OS was created with Android Studio Ladybug (2024.2.1) and was developed with the Kotlin programming language. The build configuration for both application was based on Gradle Kotlin DSL, SDK version was set to 34, and the minimum SDK version was 28, and the compile SDK version was 35. The application was developed to work with the devices that support Java 11 and Kotlin JVM target 11. To implement machine learning, TensorFlow Lite

(version 2.17.0) was applied, and the TensorFlow Lite Support and Task Audio libraries (version 0.4.4) were used to execute the trained model EnviroSoundNet.tflite on an on-device. The TensorFlow Lite Converter was used to convert the model with support for float32 inputs to enable efficient operations on mobile devices. To design a modern and accessible interface, Jetpack Compose and Material 3 were used to create the application interface. To communicate with the ESP32 microcontroller, the app relied on Java-WebSocket (version 1.5.4) through a local WebSocket connection (ws://192.168.4.1: 81). The app needs to access the microphone to record real-time audio to identify sounds and works with Android phones and devices with the Wear OS platform based on the ARM or x86 architecture.

Functional requirements include real-time audio recording, on-device sound classification using the TFLite model, direction detection using ESP32 communication, vibration feedback on sound detection, and a user interface display that shows the detected sound, direction, and confidence level.

The non-functional requirements are low-latency in sound class detection, which is approximately 0.05 seconds, a lightweight (3.279 MB) model, and good memory use to run smoothly with compatibility with various Android version devices, and secure and reliable communication between the app and ESP32 through local WebSocket.

3.2 Social Impact

The social impact of the suggested sound recognition and localization system is substantial, especially for people who are deaf or partially deaf. By recognizing and localizing important environmental sounds, such as car horns, screams, dog barks and calling bells and alerting them with visual and haptic cues, it improves their situational awareness. This increases their safety, self-reliance, and quality of life in both indoors and outdoors. For example, they can comfortably stay alone in a house, knowing that if someone rings the doorbell, they will not miss that. They can move around outside comfortably, knowing if a vehicle is approaching them, and it uses the horn, they will instantly know about the horn and which direction the vehicle is coming from and move towards safety. Their friends and family can be reassured that if they yell for them loudly, even when they are not in visual proximity, they will know they have been called and which way the call came from. So, if needed, one will be able to call for their hearing-impaired friends or family members and they will come towards the caller. If an aggressive barking dog comes barking from behind towards a hearing-impaired person, they will be aware and be safe from an accident. By enabling hearing-impaired users to participate more confidently in everyday activities without constant dependence on others, the system also fosters social inclusion.

3.3 Environmental Impact

The thesis has minimal environmental impact. It was developed and trained using a cloud-based GPU of Google Colab, which did not have substantial power consumption. After the model was built, the trained model was used in smartphones and

smartwatches, which did not consume much power. A rechargeable power bank was used for the hardware; hence, no batteries ended up in the landfill.

3.4 Ethical Issues

All the data used for training and testing the model were taken from publicly available sources or collected by us. The survey was done with the permission of the Principal and Vice Principal of the school, as well as the consent of the students. Participants' anonymity was also maintained. The mobile and watch applications do not store any audio of the user, preserving their privacy and security. All the work has been done maintaining high ethical standards and professionalism.

3.5 Project Management Plan

At the beginning, three months were spent doing literature review. Next, 2 months were spent collecting, preprocessing, and labeling data. Three weeks were spent on model development and optimization. A month was spent building the application. A month was spent developing the hardware setup and doing sound localization. Once the system was built, we were able to go back to improve the dataset and the model. The last two weeks were spent testing in the real world and improving the system. The work was distributed among group members to increase efficiency and cover more area.

A total of 30$ was spent on Google Colab Pro for model training. The audio was saved as .npy on the drive using free CPU. A100 GPU and T4 GPU were used for model training. The purchased computing units are used carefully to avoid wastage. Best performing models were saved for easy re-use. Once the model is trained, no more budget needs to go towards computing units. Around 6000tk was spent on hardware components. Different components were tested, and some duplicates were bought as needed. 2000tk was spent designing and creating the 3D printed model to keep the hardware components. The 3D printed model made the system wearable for the user.

3.6 Economic Analysis

For a very long time, hearing aids have been the only substitute for hearing-impaired people. By considering the market price and its availability, it is certain that it is not a product that can be afforded by all. Besides, if the severity of hearing problems increases, the price and unavailability of specialized hearing aid also increase. From that point of view, this solution device can be a good alternative. It can be a good option either for completely deaf people or a person with a few difficulties. The setup that has been implemented till now is cheaper than the conventional hearing aid.Followings are the components cost of the setup:

Table 3.1: Components Cost

Components Name	**Quantity**	**Price (TK)**
ESP32 DevKit V1	1	520
MAX9814 Microphone Modile	4	800
Power Source	1	400
3D Printed Headset	1	2000
		Total : 3720 /-

Even though when a product goes to production and gets real-life implementation, the price gets higher due to accuracy confirmation in each part of the device, at least, it is safe to say a new alternative to hearing aid can be added to the marketplace.

3.7 Risk Management

Data privacy may be an issue for our system. Our design, however, minimizes this risk. No personal or sensitive information is contained in the training data, hence there is no threat of data breaches. The application logs brief 3-second sound bites merely to identify sounds. The audio is automatically deleted from the device right after processing. This guarantees complete privacy of the users, and at the same time makes the system secure and ethical.

Chapter 4

Methodology

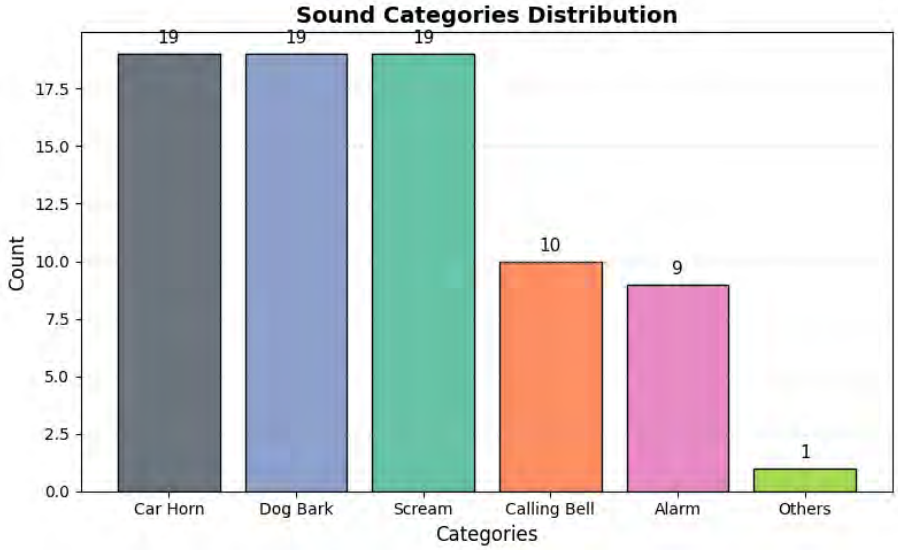
4.1 Dataset

4.1.1 Analysis of Dataset

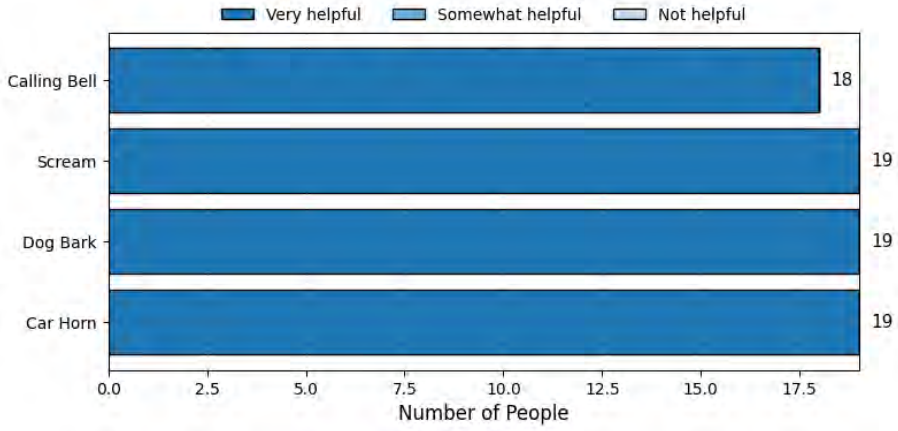
For the research, work was done on audio datasets for recognizing car horns, screams, dog barks, calling bells, and others. These audio data are collected from both external sources and manually. There are a total of 10528 audio datasets from which are divided into 10528 rows and 5 columns. Columns contain filename, duration, class, and target. There are a total of 5 classes, which are ”car_horn”, ”scream”, ”dog_bark”, ”calling_bells”, and ”other”. Any kind of noise or environmental sound that is not a car horn or scream or dog bark or calling bell falls under the ”other” class. The target column contains the numeric value against each class. The ”car_horn” class represented 1, ”scream” represented 2, ”dog_bark” represented 3, ”calling_bells” represented 4, and ”other” represented 0. The length of each audio file is set to 3 seconds to ensure a fixed input dimension so that the model can process all the data simultaneously.

Why This Specific Dataset

To validate the research, a survey was done among 19 hearing-impaired teenage students of “Dhaka Bodir High School”. Here, they were asked what type of sound they wished to hear for their safety, and their answer showed the following results-



And when asked if they wanted to know where the sound was coming from and all of them agreed to that.



The following result shows their response to the question about how helpful they think it will be to get notified about these certain sounds and their location for their safety. These results were utilized in finalizing the dataset classes.

Class Based Analysis

For the portion of the car horn dataset, audio files were collected from three different sources — two different Kaggle datasets and one free audio website platform named "Freesound.org". From "Freesound", 998 audio files were extracted where 838 contain car horns and 160 contain environmental noise. From the first Kaggle dataset, 1,080 audio files were extracted that contained car horns. Lastly, another Kaggle dataset consisted of 1,920 files filled with different types of noise. Some car horn audio data was also collected by recording the traffic environment of Dhaka. Thus, total audio files for car horn detection reached 3,998. From there, 412 clean car horns and 1,630 noisy car horns were labeled, ending up with 2,042 car horn audio data for detection. All the general noise and environmental noise fell under the "other" class.

For the detection of screams, a dataset from Kaggle was used that provided a total of 2,029 audio files after preprocessing. The scream audios were also manually labeled, which consisted of 1,065 clean screams and 964 noisy screams. Both clean screams and noise-mixed screams were included in these audio files.

For the `dog_bark` class, another 2,000 audio files were processed from three different sources — two datasets from Kaggle and one dataset from the Hugging Face website. Here, 47.35% of audio files were noise-mixed dog barks, and the rest 52.65% audio files were clean dog barks. 1,057 files were clean dog barks, and 947 files were mixed with noises.
Finally, for the calling bell class, various types of calling bells were collected through families and friends. The "Freesound.org" website was also used to collect calling bell audio files. Combining these audio files, a total of 2,094 audio files were prepared for calling bell detection, containing 1,486 clean calling bells and 608 noisy calling bells.

Class Distribution

The class distribution represents the number of samples in each class. This class distribution was visualized for the five different classes of the dataset. The following is the figure that represents class distribution:

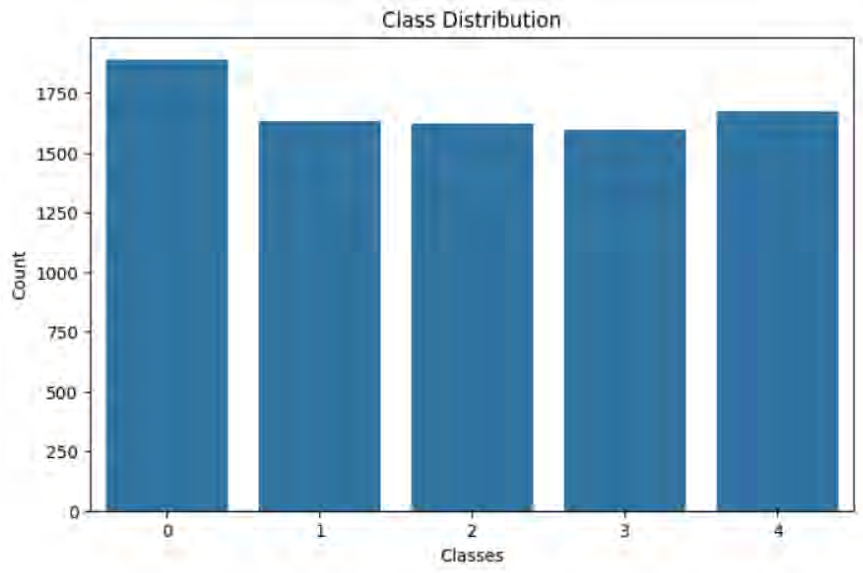


Figure 4.1: Class Distribution of five classes

Here, the x-axis represents the target value of each class, and the y-axis represents the sample present in the train dataset, which is not the literal value but rather converted to a range from 0 to more than 1750. 0 represents the "other class, 1 represents the "car_horn" class, 2 represents the "scream" class, 3 represents the "dog_bark" class, and 4 represents the "calling_bells" class. As almost the same number of sample data was supplied for each detection, the bar chart represents a uniform distribution among them. Noise data("other" class) was slightly higher

than the other three classes, as we supplied 2363 audios for this section. This distribution ensures that there is a perfect balance among classes and there is no underrepresented class in the train dataset

Mean & Variance Intensity

Mean & variance intensity analysis helps in understanding the energy distribution within the dataset. The mean intensity represents the average loudness or average energy of an audio file, while the variance intensity represents the fluctuation of energy or loudness over time. The mean and variance intensity for the five different classes in the dataset were plotted as follows:

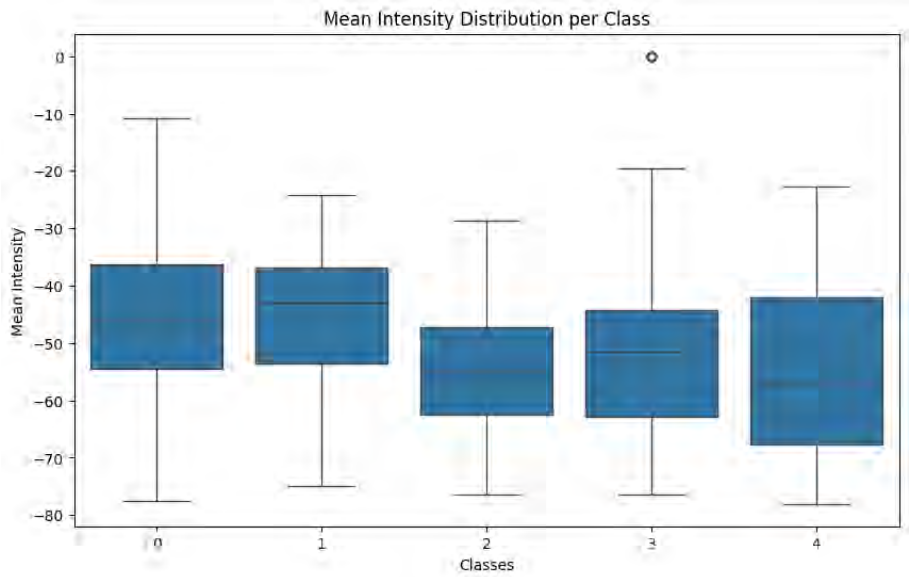


Figure 4.2: Mean Intensity of Five Classes

From the figure 4.2, it can be seen that the ”other”(0) class holds the highest range of mean intensity, which is approximately -55 dB to -34 dB. The 2nd highest intensity range is occupied by the ”car_horn”(1) class, which is approximately from -54 dB to -33 dB. Thirdly comes the ”calling_bells”(4) class, which has a range from -69 dB to -42 dB approximately. Then comes the ”dog_bark”(3) dataset, which holds the range from -62 dB to -43 dB approximately. And lastly, comes the ”scream”(2) dataset, holding the range from -61 dB to -58 dB approximately. Higher mean intensity indicates audio files with a higher energy level, and lower mean intensity indicates a subdued audio signal. So it can be easily interpreted that the ”other” class has higher or louder audio signals. It happened because this class holds noise of various kinds, which may contain very high pitch and bandwidth. ”Other” class was not bound to any certain noise level. On the other hand, collected scream dataset was from a single source, which contained the same kind of internal quality-based dataset, which may lead to a lower mean intensity.

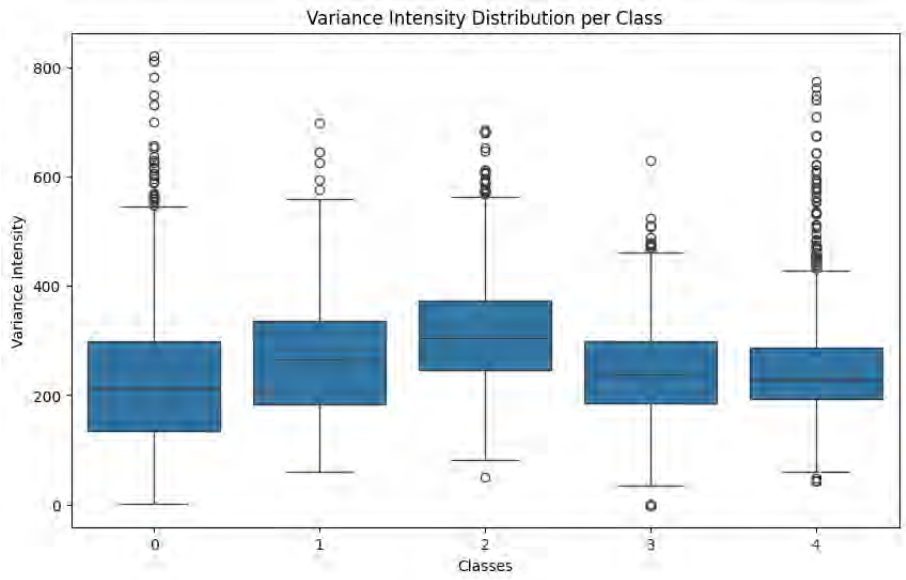


Figure 4.3: Variance Intensity of Five Classes

From the figure 4.3, the variance intensity of each class was found. Higher variance intensity indicates a highly fluctuating audio dataset presence, and lower variance intensity upholds the opposite. In the dataset 'scream' class has the highest variance intensity, which is logical as scream has the tendency of sudden outbursts after a quiet moment. It upholds the range from 250 dB to 350 dB approximately. On the contrary, the 'other' class has the lowest intensity (from 140 dB to 290 dB approximately), which is also reasonable because usually noise holds a continuous form of sound from any environment. The other two classes have medium-level variance intensity as expected.

Spectrogram Analysis

A spectrogram helps to represent the distinct characteristics of different kinds of audio datasets. As the dataset has five different classes, each of them plotted a unique spectrogram, which is a representation of the energy of audio signals across frequencies over time. The plotted spectrograms are as follows:

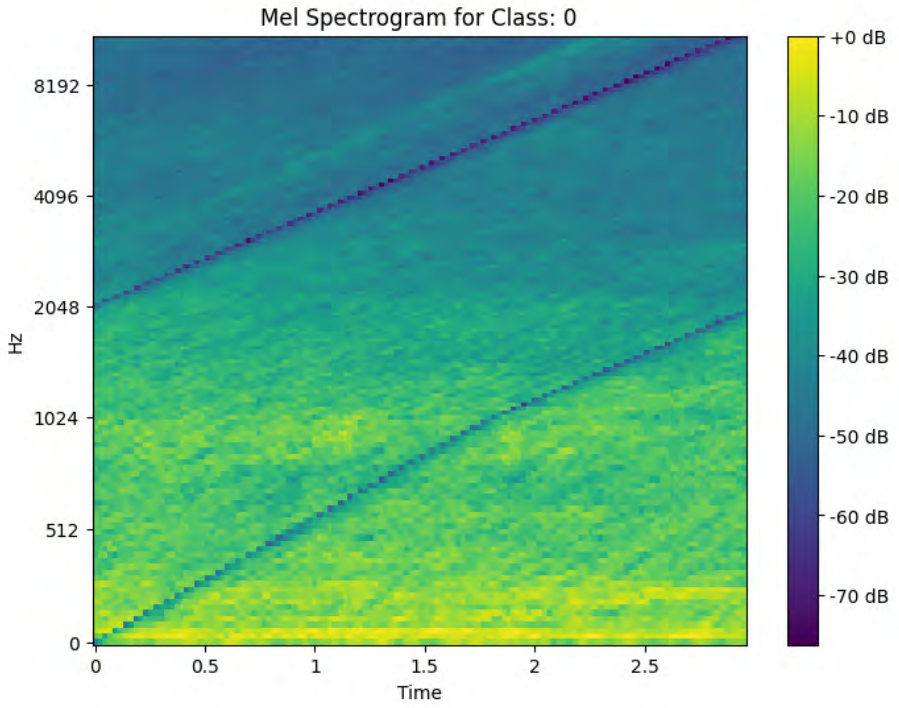


Figure 4.4: Spectrogram of Class 0: Other

Figure 4.4 holds the 'other' class, which shows unstructured characteristics because it contains so many kinds of noises that do not maintain any harmonic pattern. It is mainly a fuzzy and diffuse distribution that is broadly spread across frequencies. Randomness of noise makes it unpredictable and shows no constant patterns. There are no sharp picks or clear lines throughout the spectrogram.

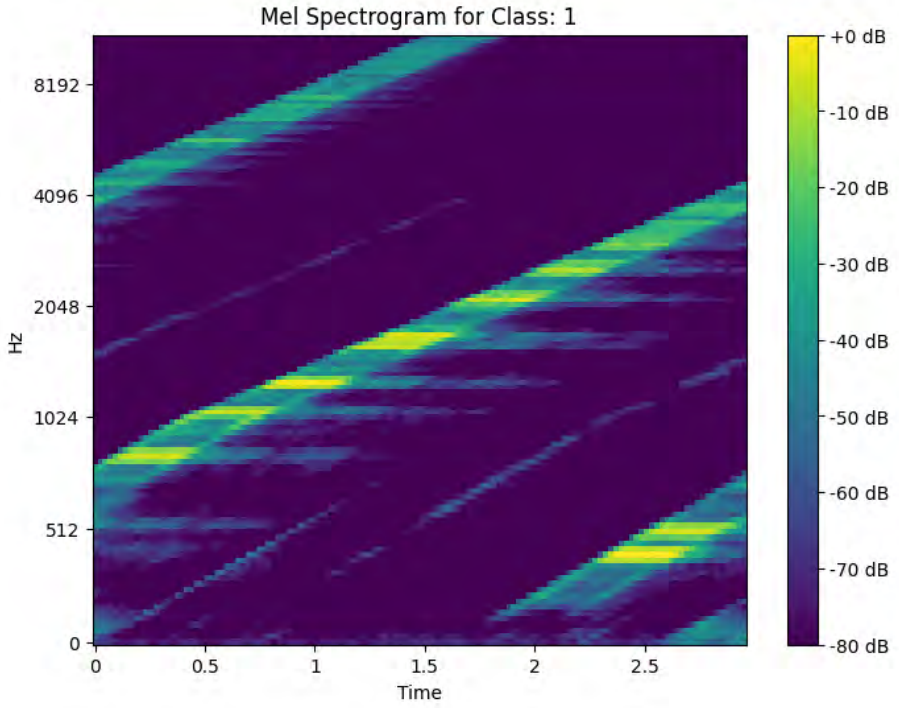


Figure 4.5: Spectrogram of Class 1: Car Horn

In the figure 4.5, the 'car_horn' class was plotted. The brighter part of the spectrogram indicates higher energy across a certain frequency. Unlike noise, it shows a harmonic or periodical pattern throughout the gram. Distinct bands of signals are concentrated in specific frequency bands. No sudden burst of energy is shown, which means fewer fluctuations.

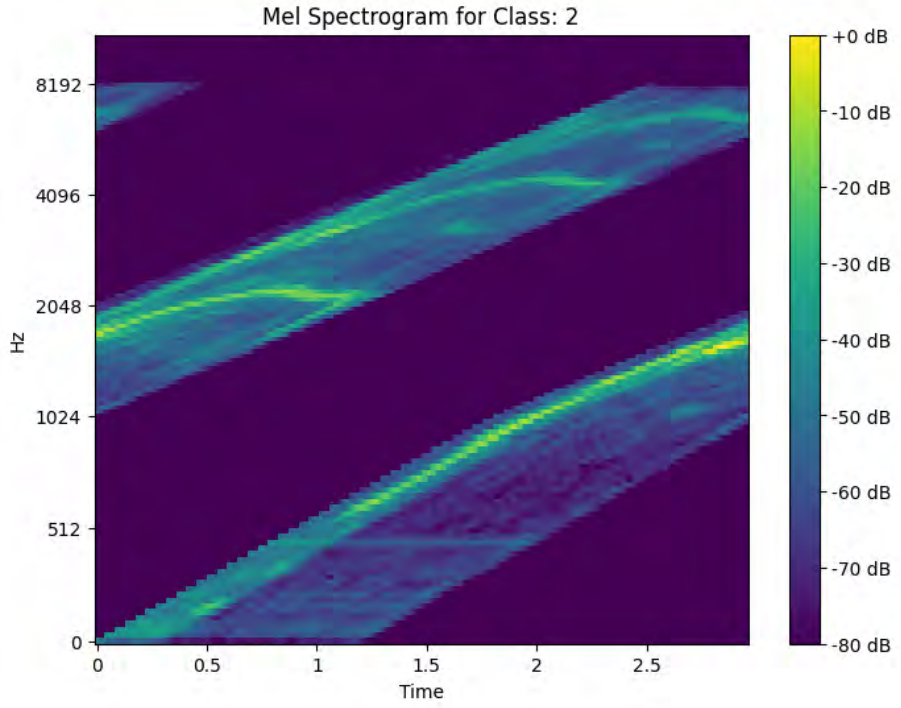


Figure 4.6: Spectrogram of Class 2: Scream

Figure 4.6 shows the 'scream' dataset, which holds its common nature with its higher spread of energy in a border range of frequency. Variability of the signal shows a high pitch, which is likable for scream audios. Both high and low frequency shows a significant amount of energy. Unlike car horns, it shows irregular patterns in its band.

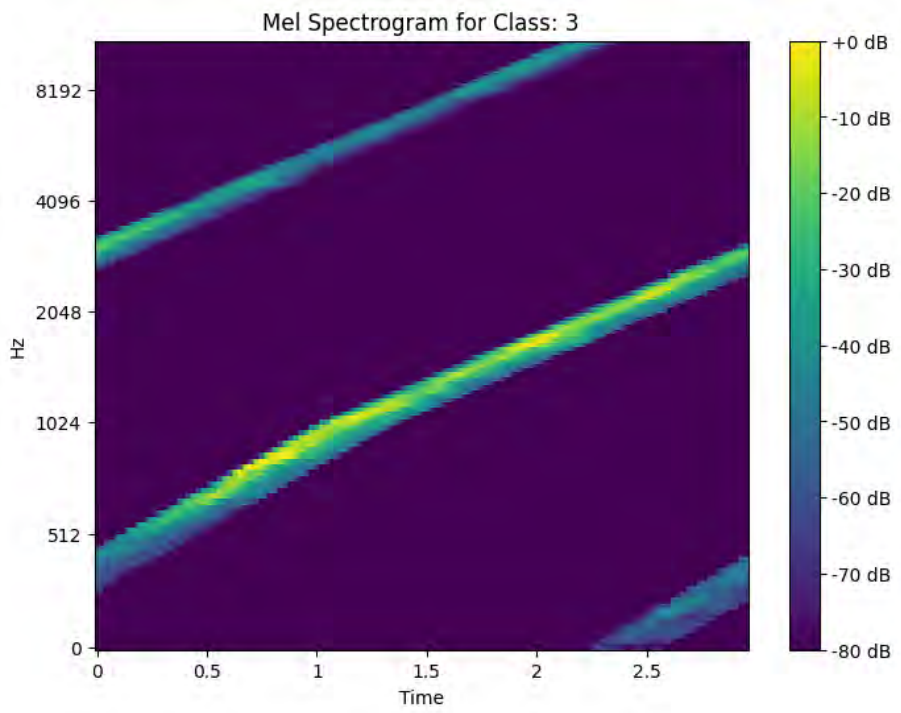


Figure 4.7: Spectrogram of Class 3: Dog Bark

'Dog_bark' dataset is represented in figure 4.7. It is likely with car horn data, but the band of signals is more specified with clearer lines. It happened due to the preprocessing of the dog bark dataset, where almost 50% audio files provided clean dog barks that have no background noise. This portion of the dataset holds the highest amount of clean data, so the bands are more specific and periodic.

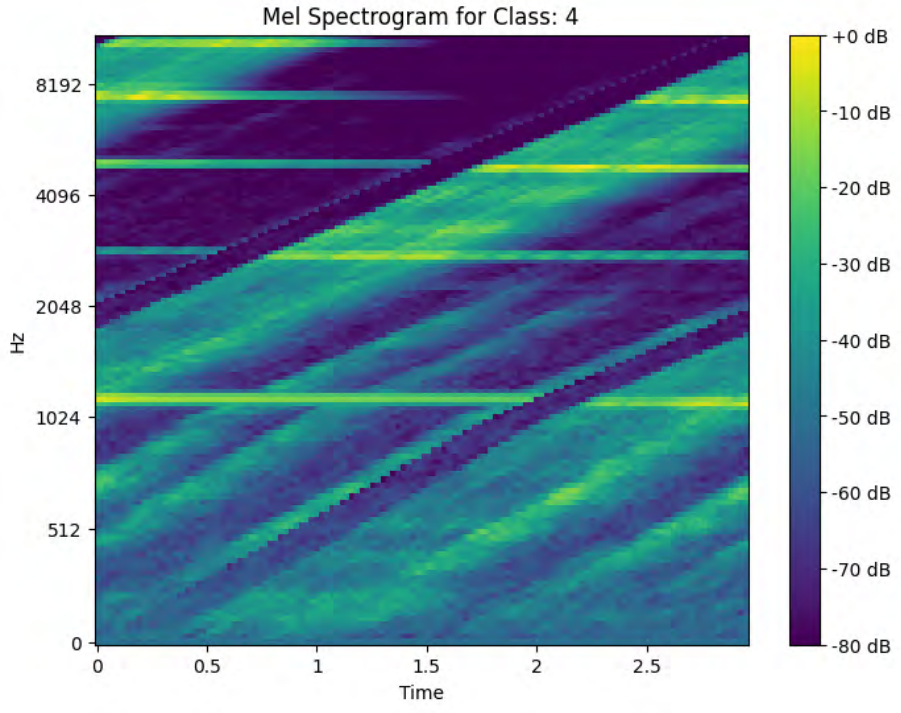


Figure 4.8: Spectrogram of Class 4: Calling Bell

Figure 4.8 shows the 'calling_bells' dataset, which is very unique and different from the other datasets so far. It has bright horizontal lines indicating strong energy at specific frequencies because bell sounds generally produce strong and harmonic overtones. On the other hand, both high and low frequencies in the wide frequency coverage show a significant amount of energy, which is due to the bells' metallic and sharp tone. As we know, calling bells fade gradually in the end, the intensity of the mel spectrogram also fades gradually.

Mean vs Variance Intensity overlaps among Classes

The mean intensity and variance intensity for each class are plotted here as a mean vs variance graph, which helps to analyze different kinds of features of the dataset. The following figure shows it:

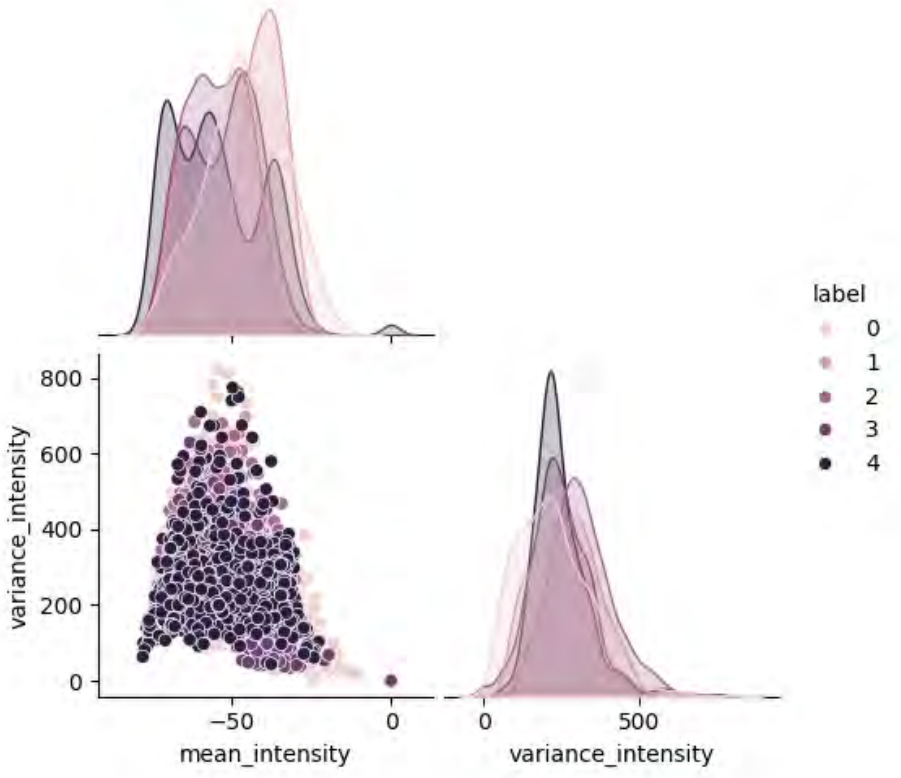


Figure 4.9: Visualization of overlapping features

The top-left plotting represents the mean intensity overlaps among classes, and the bottom-right plotting represents the variance intensity overlaps among classes. From the comparison, it can be easily said that variance will be a better feature to distinguish among classes than mean intensity, as mean intensity upholds higher overlapping than variance intensity. After doing the scatter plot, which is situated at the bottom-left part of the figure, we can declare that maximum overlapping happens between classes 1 and 2, which will misguide the separating process. There are comparatively higher chances of getting lower accuracy during predicting classes 1 and 2.

4.1.2 Dataset Preprocessing

As five different classes were used in the dataset, for each class data from different sources were collected and preprocessed. After doing individual preprocessing, they were combined into a single dataset. After combining, the whole dataset was split into training and testing parts, which are 80% and 20% respectively. The testing dataset has 2109 audio files, and the training dataset has 8419 audio files.

Car Horn and Noise Dataset Preprocessing

For the portion of car horn data, four different sources were used, which include two different Kaggle datasets and one free audio website platform named "Freesound.org". From the website platform, 270 audio files were collected that contained car horns and environmental noise. Then these audio files were trimmed into 3 seconds each, creating 918 files. These audio files were then labeled into three categories: car horn, not car horn, and clean car horn. A clean car horn was a part of a car horn that provided minimal background noise.

Augmentation on the not car horn part was done, as there were environmental noises that were repetitive, making the audio files similar. Thus, 838 files of car horns & 160 files of environmental noise were created. Between the two datasets from Kaggle, the first one contained 540 audio files, which were then converted to 1,080 audio files containing clear car horns. Each file was originally 1 second long; for half of them, silent padding was added at both the start and end of the audio to make it 3 seconds. For the other half, looping was done to make them 3 seconds long. Background noise was also added with a portion of a clear car horn for better model learning.

Another Kaggle dataset contained 2,000 audio files, from which 1,920 files filled with different types of noises were collected. An additional 80 environmental noise audio files were added from the previous 160 files of environmental noise collected from the "Freesound" website. Thus, a total of 2,000 files of noise were prepared from this portion.

All the audio files were then combined, and the total audio files for car horn detection reached 3,998. The dataset was split into training and testing sets, where the training set included 80% and testing included 20% of the dataset. For both parts, an equal portion of car horns and noises was ensured.

Real-life car horns were also collected for better performance in real environments. These were divided into 3-second portions, and the audios were manually reviewed to extract clean and noisy car horn samples. About 30% to 40% clean car horn data were added in each car horn portion at the beginning. After all preprocessing, 412 clean and 1,630 noisy car horn audio files for the "car_horn" dataset were prepared.

For the 'other' class, daily life environmental noises were used as well, like- cafeteria and class recording as audio files for better performance. That gave a total of 2363 noisy audio files for the 'other' dataset.

Finally, these two dataset labeled 'car_horn' and 'other' were prepared with their target values 1 and 0, respectively..

Scream Dataset Preprocessing

One source was mainly used for the scream dataset, which came from a Kaggle dataset. Primarily, it had 1583 scream audio files and 1583 non-scream or noise audio files. Each of them was 5 seconds long. Firstly, the 1583 scream audios were trimmed to make them 3 seconds long. Then, to balance the scream dataset with the car horn dataset, 417 audios were randomly picked from the same scream audios and then noise was added. These noises were taken from the 1583 non-scream audios. Thus, collectively, 2029 audio files of the 'scream' dataset were prepared, where 1065 were clean and 964 were noisy scream audio files, and its target value was set to 2 under the 'scream' class.

Dog Bark Dataset Preprocessing

Firstly, 1053 clean dog bark sounds were collected from two different sources: one Kaggle dataset and one dataset from the Hugging Face website platform. These audios had different lengths. They were trimmed to 3 seconds. Then, with random selection, 947 audio files were selected from these 1053 audio files and added noise from the third dataset, which was collected from Kaggle. This dataset was full of different kinds of noise. Thus, a total of 2000 audio files were prepared, where 1057 were clean and 943 were noisy dog bark, and their target value was set to 3 under the 'dog_bark' class.

Calling Bell Dataset Preprocessing

For calling bell detection, the data was primarily collected from friends and family through forms and divided into 3-second audios. From there approximately 800 calling bell sounds were selected. Then, to balance with the other datasets more audio files were collected from the 'Freesound.org' website, that have a 'Creative Commons License. And after reviewing all the audio files got 1486 clean and 608 noisy calling bell sounds for detection. Thus, a total of 2094 audio files of the calling bell dataset were prepared, and its target value was set to 4 under the 'calling_bells' class.

4.2 Model

4.2.1 Overview

A custom Convolutional Neural Network(CNN) was developed for sound recognition. The model needed to be able to identify five sound classes (Car Horn, Scream, Dog bark, Calling bell, and Others), be compact in size, and be capable of functioning within an app for practical use.

In the preprocessing stage, three-second raw sound waves are converted to 1D float arrays, and then they are mapped to Mel spectrograms as a 2D representation of the sounds. The CNN model was trained and tested using labeled sounds. The

CNN model is comprised of a dense layer and multiple convolutional and maxpool layers. Many techniques, such as regularization, batch normalization, dropouts, etc, are used to optimize the model.

During practical application, three-second sound clips are continuously taken using the user's mobile or smart watch mic. The sound is converted to a Mel spectrogram, which is fed into the pre-trained CNN model, and the model gives a prediction of the category of the sound. The whole pipeline is compact and performs efficiently in real time.

4.2.2 Model Justification

CNN is proven to work well for two-dimensional data, as it can capture local and spatial patterns. Alternative approaches, such as Recurrent Neural Network(RNN) can capture temporal patterns but lack in special pattern recognition. Transformers perform well on special patterns too, but they require substantially more data and computing power, which makes them less practical for this use. CNN balanced performance and computational efficiency for the sound recognition task.

Mel scaled short-term Fourier transform (Mel spectrogram) performs very well in CNN models for audio classification tasks [21]. Hence, at first, the raw sound waves are converted to Mel spectrograms, which are a 2D image representation of a sound. Next, the CNN model is trained on these images to classify which sound is from which category by seeing the local and spatial patterns.

4.2.3 Model Architecture

The model takes a float array representation of a three-second raw sound as input and gives a category name as an output. At first, the sound is converted to a Mel spectrogram. There are five convolution blocks consisting of convolution layers followed by a pooling layer; these work on the 2D representation of the sound. After these five blocks, the data is flattened, and it is followed by a fully connected dense layer. Softmax is used to get output from the fully connected layer.

Table 4.1: Model Architecture Details

Block	**Number of Filters**	**Kernel Size**	**Dropout Rate**	**Regularization**
1	16	(3×3)	0.3	L2 (1e-3)
2	32	(3×3)	0.4	L2 (1e-3)
3	64	(3×3)	0.45	L2 (1e-3)
4	128	(3×3)	0.5	L2 (1e-3)
5	256	(3×3)	0.5	L2 (1e-3)

Batch normalization is used in every block, ReLU is used as the activation function for each convolution layer and Maxpool is used for every pooling layer. The output from the last block is flattened and it goes to the fully connected layer, which also uses ReLU activation, L2 Regularization, batch normalization and a dropout rate of 0.5. The output from the dense layer uses the softmax activation function to give

an output. The output layer gives a category number (0=Others, 1 = Car Horn, 2 = Scream, 3 = Dog Bark, 4 = Calling Bell) which indicates the category of the input sound. The model uses the ADAM optimizer when compiling and uses the accuracy metric for evaluation.

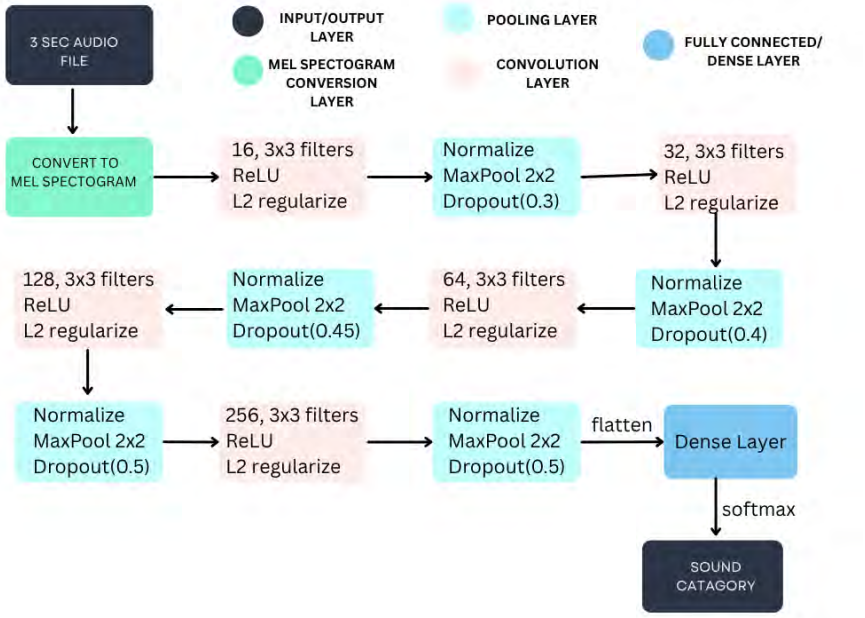


Figure 4.10: CNN Model Flowchart

4.2.4 Model Optimization

In order to improve the performance and generalizability of the model, many hyperparameter tuning and architectural experiments were conducted in the model development process. For example, different numbers or layers, number of filters, activation functions, pooling layers, dropout rates, etc, were tried. Batch normalization and L2 regularization and high dropout rates were used to stop the model from overfitting the training data.

Models using three to seven convolution blocks were tested and it was found that 5 convolution blocks performed the best. Models with fewer than five blocks were too simple and did not have high accuracy; more than five blocks did not perform better than five blocks while being more complex. Five convolution blocks performed the best while remaining less complex. Different numbers of filters on each layer were also tested. Starting with fewer than 16 filters did not improve the model, and starting with 32 filters or more took significantly more time and computational power, while they did not improve the performance by much. Five convolutional blocks, each with one convolution layer with a number of filters 16,32,64,128, and 256, overall performed the best.

Different activation functions, including ReLU, Leaky ReLU, GeLU, and tanh, etc, were tried in order to find the optimal non-linearity for feature learning. ReLU was selected finally because it provided the best performance and quicker convergence.

Different dropout rates on different layers were tested. The final rates gave high accuracy while preventing overfitting. For the pooling layer, Maxpool, Averagepool and a combination of both were tested, but using Maxpool on all convolution blocks performed the best.

4.3 Android and Wear OS Application Development

4.3.1 Purpose and Scope of the Applications

Two applications were developed to support accessibility within the proposed system: one for Android smartphones and another for Wear OS devices. Both applications can be used to access the system's core functionality of detecting and localizing important sounds, supplemented with a tactile feedback mechanism. Only one application is needed for the system, so these two applications are completely independent and compatible with the headset. The purpose behind developing two independent applications was to provide users with the flexibility to choose the application based on their access to an Android phone or a Wear OS smartwatch.

4.3.2 User Interface Implementation

Android Application

The user interface of the Android application was implemented using Jetpack Compose. The whole application was written in Kotlin. For a clean, consistent, and accessible interface, Material Design 3 components were employed.
The main elements of the interface include:

- **Status Text:** Displays the current state of the system. For example, *Waiting to Start*, *Recording*, or *Recording Stopped*. During the detection cycle, it shows the real-time detected sound class, prediction confidence, and direction of the sound source.
- **Progress Bar:** A linear progress indicator representing the 3-second detection window. This bar slowly fills up during the window and resets at the end to mark the start of a new 3-second window. With this element, the user can understand the progress of the detection cycle.
- **Start/Stop Recording Button:** Allows the user to begin or end an audio recording session for detection and localization.
- **Vibration Feedback:** Triggered whenever an important sound class such as *car horn*, *calling bell*, *scream*, or *dog bark* is detected. This provides tactile feedback to notify the user.

Figure 4.11 represents the user interface with the detected sound category, corresponding confidence level, and direction of the sound source. It also includes the progress indication for the next recording session.

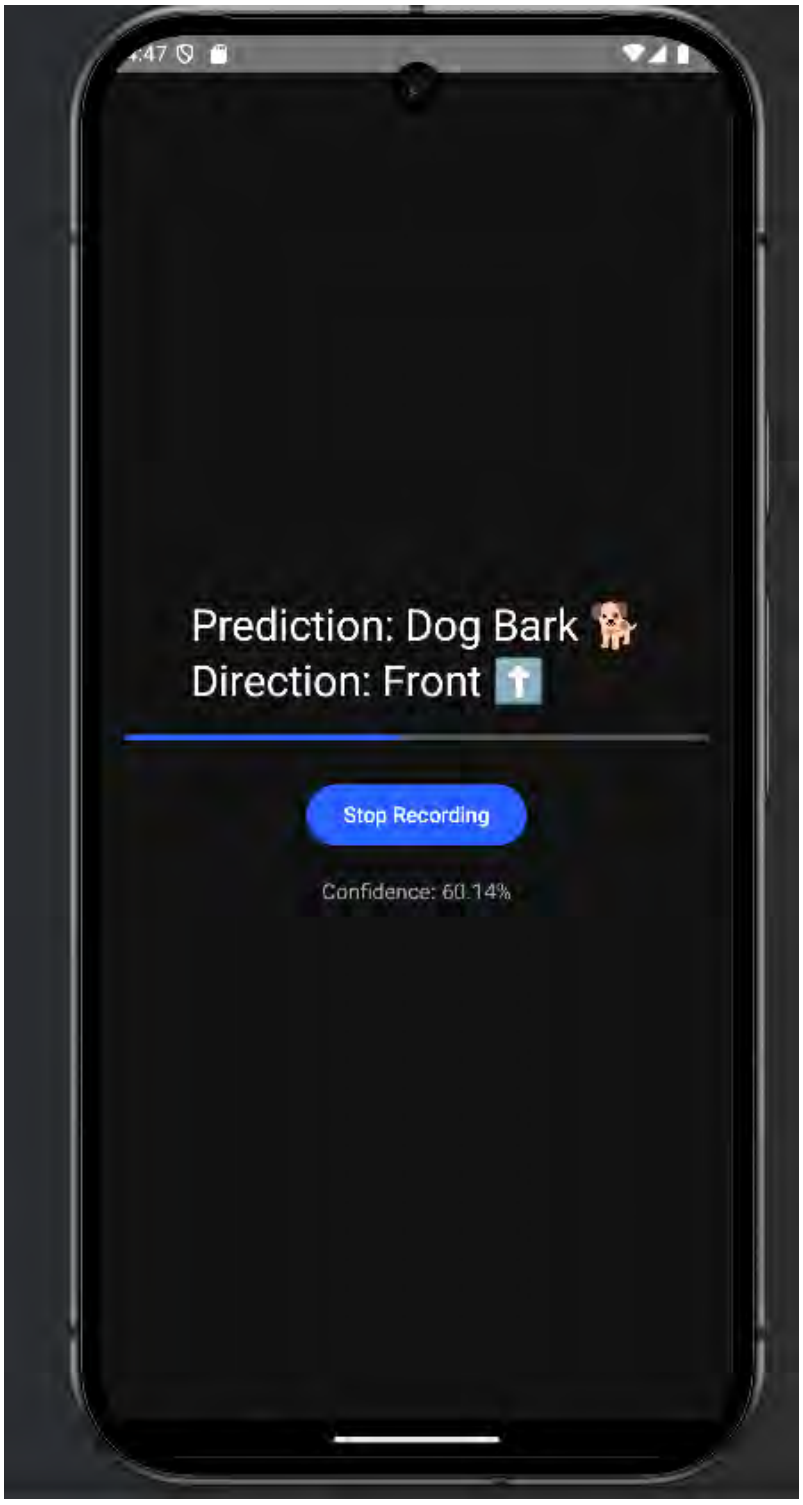


Figure 4.11: User interface with the result of the Android Phone Application

Wear OS Application

The user interface of the standalone Wear OS application was also implemented using Jetpack Compose and written in Kotlin. Just like the Android application, Material Design 3 components were also employed.
The main elements of the interface include:

- **Status Text:** Displays the current state of the system, such as *Recording* or

Recording Stopped. During the detection cycle, it shows the real-time detected sound class, prediction confidence, and direction of the sound source.

- **Start/Stop Recording Button:** Allows the user to begin or end an audio recording session for detection and localization.
- **Vibration Feedback:** Triggered whenever an important sound class such as *car horn*, *calling bell*, *scream*, or *dog bark* is detected, notifying the user through tactile feedback.

Figure 4.12 represents the user interface after pressing the "Record" button. It includes the detected sound category, direction of the sound source, and corresponding confidence level.

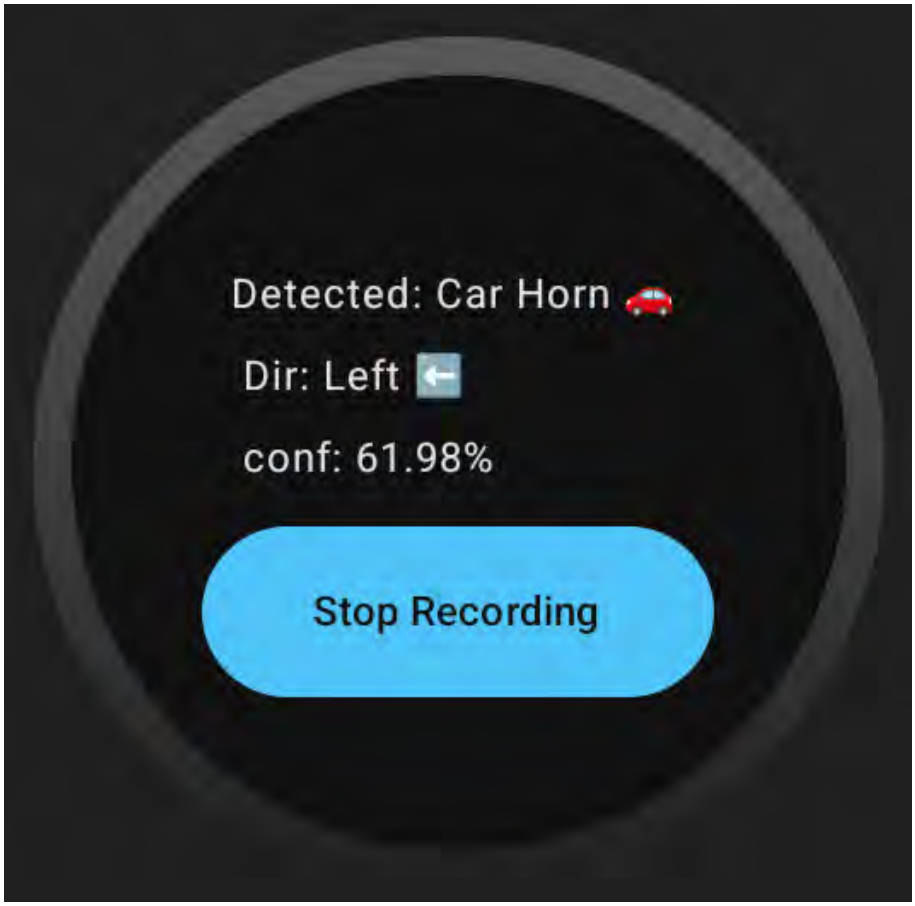


Figure 4.12: User interface with the result of the Wear OS Application

4.3.3 Backend Implementation Logic of Android and Wear OS

The system back-end was written in Kotlin and includes sound classification, feedback control, and real-time audio processing. The application captures 3-second audio samples with the Android `AudioRecord` API, normalizes the data, and sends it through a TensorFlow Lite model to get an inference. This model makes predictions of the type of sound — which can be either a car horn, a scream, a dog bark, or other — and its respective confidence score.

To perform localization, the application connects to the ESP32 headset over a WebSocket connection to receive an approximation of the direction of the detected sound, which can be either *front*, *back*, *left*, or *right*. If the classification outcome is not "Other," the application provides vibration feedback using the built-in vibrator of the device to notify the user.

The backend implementation logic is identical in both Android and Wear OS, except for vibration feedback handling. In Android, recording does not stop during vibration, so every 3 seconds, the gap between recordings is approximately 0.02–0.05 seconds. However, as the built-in vibrator tends to be very close to the microphones of Wear OS devices, audio recording is paused while giving vibration feedback. Therefore, in this case, every 3 seconds, the gap between recordings is about 0.56 seconds.

The complete backend is asynchronous to maintain smooth real-time execution and constantly update the user interface with the latest detection results, confidence, and direction.

Figure 4.13 represents the flowchart of backend processing logic showing asynchronous audio classification and headset communication. Both threads operate in parallel to provide real-time detection, direction retrieval, and vibration feedback, with minor variations between Android and Wear OS implementations.

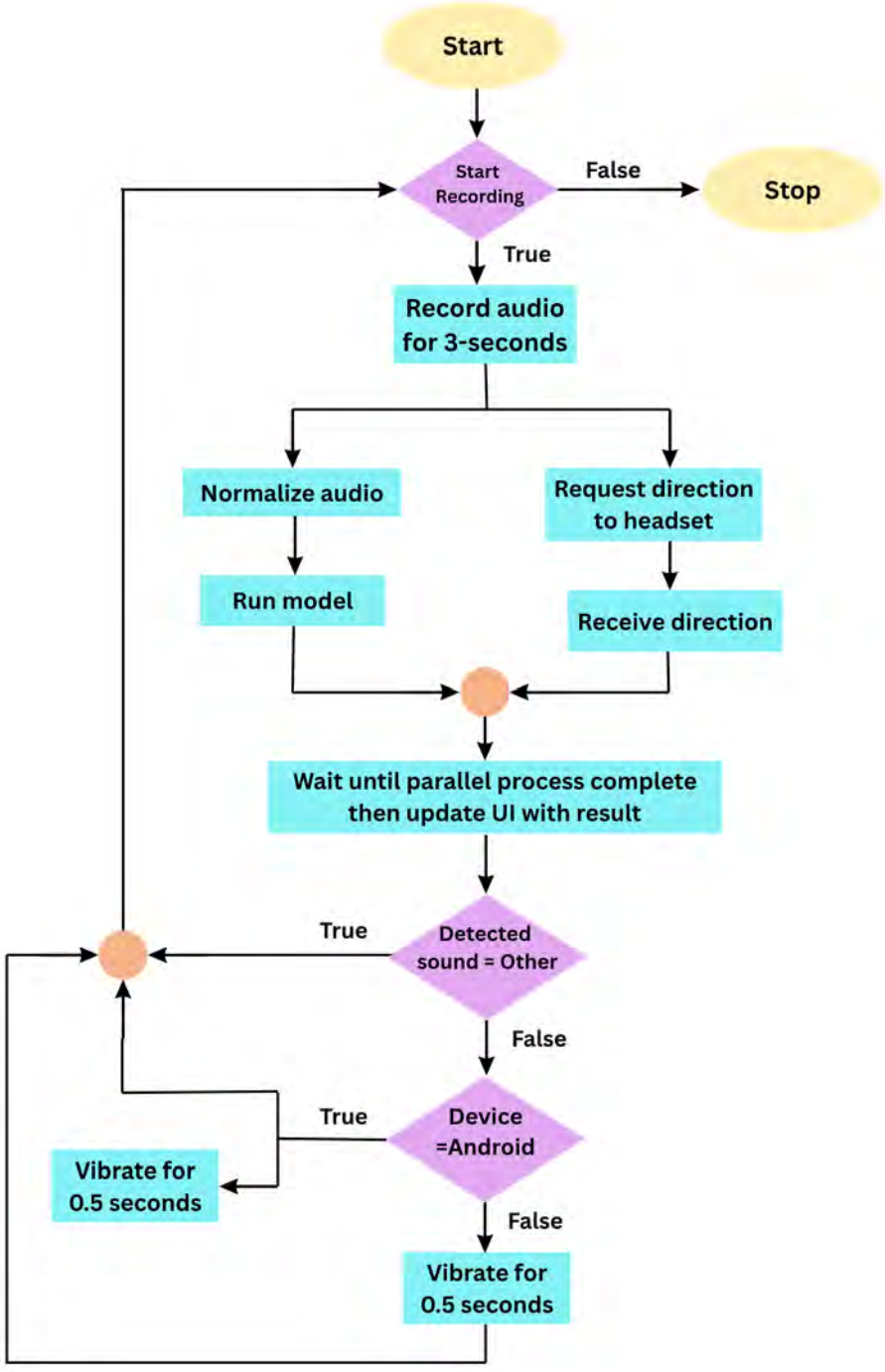


Figure 4.13: Flowchart of backend processing logic

4.4 Hardware Architechture

4.4.1 Overview

Any kind of android smartphones and smartwatches are acting as the solution device. After detecting the sound and localizing its position, the result is shown in the monitor of that smart device and vibrates as tactile feedback. The two core functionalities of the system, sound detection and localization use two different architectures.

For sound detection, the signal is received by any embedded microphone of an android smartphone or smartwatch. For localization, at least four microphones are needed to capture the sound signal from different positions . As no android smartphone or smartwatch offers this setup, a separate solution made with four mics and a microcontroller is made and connected with our solution device via Wi-Fi. The full localization process is done by the microcontroller and this setup is attached to the human body with a 3D printed headset . So the full solution is totally portable and user friendly.

4.4.2 Hardware Used for Localization

To set up the prototype of localization, four MAX9814 Microphone Amplifier modules are used to capture the sound wave from the environment. They need 2.2V to 5.5V for operating which is perfectly compatible for common microcontrollers. One Esp32 Development Board DEVKIT V1 is acting as the microcontroller and helping to calculate the direction from the received signals coming through four different mics. One mini breadboard and few wires are used to connect the Esp32 with four mics. Four mics are used to identify four different locations: front , back, left and right primarily. These four mics are connected with four ADC pins of ESP32: GPIO pin 35, GPIO pin 33, GPIO pin 34, GPIO pin 32 with front mic, back mic, right mic and left mic respectively. Each mic takes its power from the 3.3V pin of ESP32 and uses a ground pin to complete the circuit. The Esp32 board gets power from two 1.5V batteries which combinedly supplies 3V and regulates the localization process.

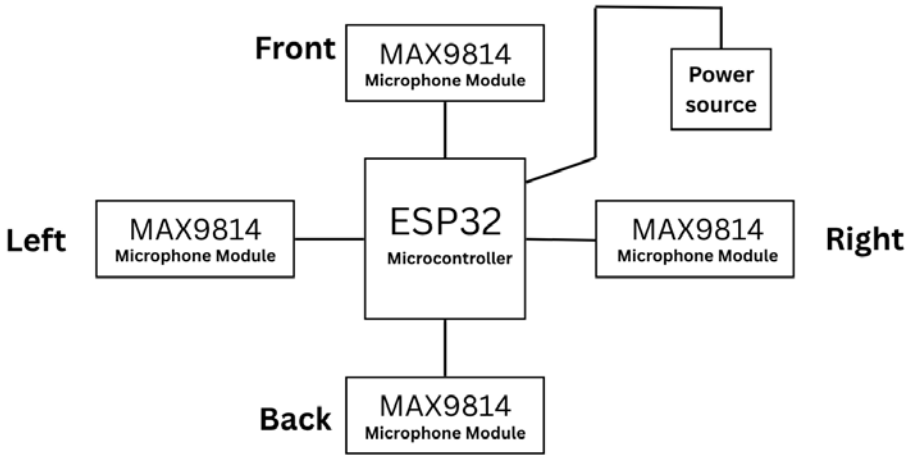


Figure 4.14: Block Diagram of Components Used for Localization

4.4.3 Logic of Localization

For building up the localization logic, a simple yet effective method is used. From 4 different pins, signals are read using analogRead() function and stored in 4 different variables: frontVal, backVal, rightVal and leftVal. Two thresholds are declared for measuring the silence of the environment as sound is received in the form of waves. Upper threshold is "ut" and lower threshold is "lt" and their values are 2500 and 300 respectively. The signal of sound received and stored in each variable is continuously

measured with the both upper and lower threshold. If the signal read is higher than the upper threshold or lower than the lower threshold, each variable increases its count variable by one. This value of the count variable updates continuously till it gets a request from the application to send the direction. If it gets the request, it compares the values of these four count variables and selects which one is the highest for this certain period. Then send that mic name as the direction. If no request is received, it continues to do the receiving signal and increasing count variable for each mic till it gets a request.

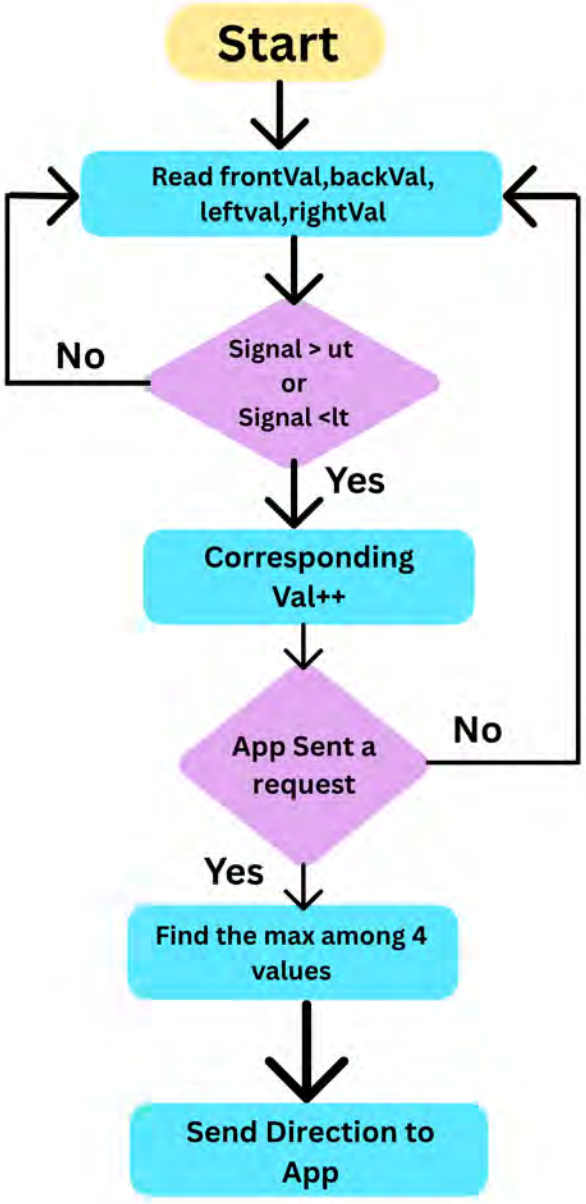


Figure 4.15: Flowchart of the process of Localization

4.4.4 Connection with Solution Device

To make this system portable, the Wi-Fi feature of ESP32 is used to send this direction to the application. ESP32 offers a great feature where it works in the Access point mode and acts as a router to create its own Wi-Fi network. This way, after providing the name and password of the Wi-Fi, it creates its own network using WiFi.softAP() function and assigns an IP address by using softAPIP() function. Any amount of clients can be added to this network and receive direction. WiFi.softAPgetStationNum() helps to know the total connected clients at the starting of the process. To communicate with clients, WebSocketServer() function is used to create a "webSocket" named object which listens on port 81 and handles client connection, incoming or outgoing messages. A callback function named onWebSocketEvent() is called using the webSocket event library. When the application sends the message "GETDIRECTION", this function receives it and calls the calculateDirection() function which runs the full localization logic mentioned previously. Thus the webSocket receives direction and sends it as an outgoing message to single or multiple clients using webSocket.sentTXT(). This process of sending and receiving messages continuously works with websocket.loop() function providing a real time feedback until it is forcefully stopped by the application.

4.4.5 3D Printed Headset

The 3D printed headset is a secured place for the localization setup and helps to get the perfect coordinate to the 4 mics as it is situated on the head of the user. It has a cylindrical shape with a height of 10.5 cm and a radius of 8.6 cm which is a standard to set up on a human head. There are two rectangular shapes on both the right and left side of the cylinder situated 2 cm above from the bottom to attach elastic which can be tied up with the chin to make the object more stable. A plate situated 6.5 cm above from the bottom helps to keep the microcontroller, mini breadboard and power source. Four different circle shapes with 0.7 cm radius are positioned 2.8 cm above from the plate to attach four mics. They are situated in four different directions of the cylinder: front, left, back and right.

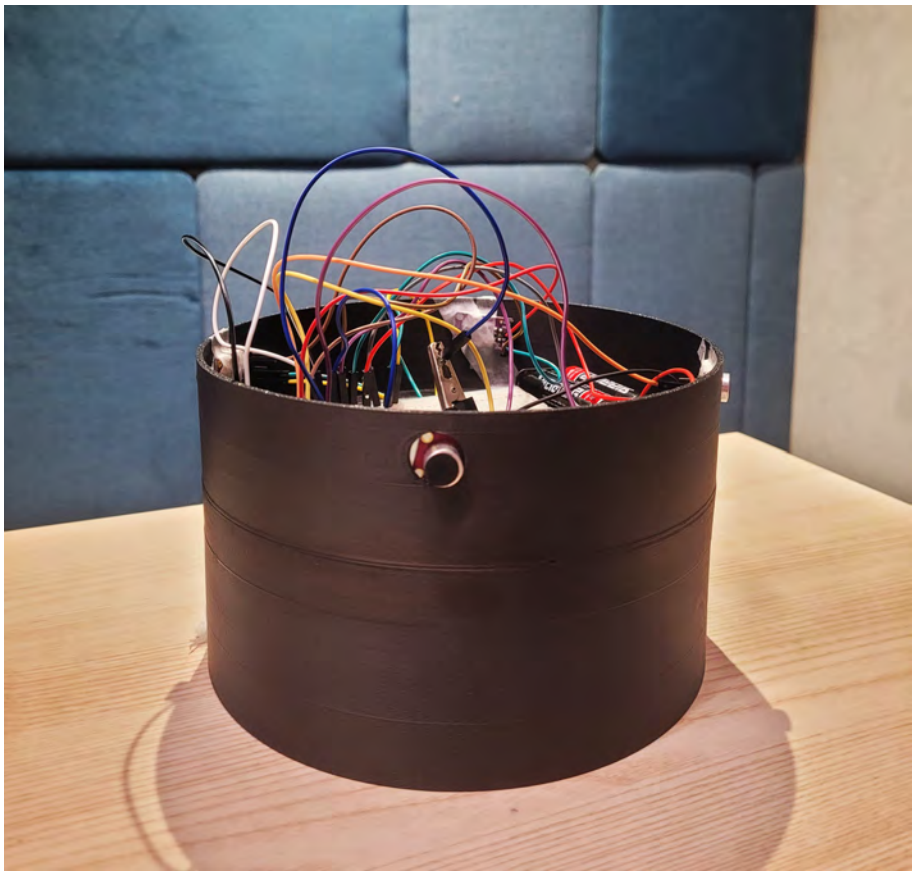


Figure 4.16: 3D Printed Headset

Figure 4.17: A Human Wearing The Headset

4.4.6 Setup for Users

To use this solution, the user needs to wear the headband and download the application. After downloading the application on a smartphone or smartwatch, the user needs to connect the smartphone or smartwatch with ESP32 Wi-Fi network. At first, the ESP32 board needed to be powered up. Then, from the available Wi-Fi networks, the user needs to connect with "ESP32" named Wi-Fi. After that, the user can open the application and see the result and get the feedback by clicking the "Start Recording" button.

Chapter 5

Results

5.1 Performance Evaluation

5.1.1 Sound Recognition Performance Evaluation

Sound recognition was done using a Custom CNN model. The model's performance was evaluated using standard evaluation metrics, including accuracy, precision, recall, and the F1-score. These together provide a comprehensive assessment of the model's performance. Confusion matrix was used to visualize the results through a heatmap and statistically analyze the model's performance on the different classes. The maximum number of epochs was set to 60, but the training stopped at 50 epochs because of the early stopping mechanism. The CNN model achieved 92.64% accuracy on training data and 88.34% on validation data. The precision, recall and f1 scores are given in the table below

Table 5.1: Classification Performance for Each Class

Class	**Class Name**	**Precision**	**Recall**	**F1-Score**
0	Others	0.77	0.88	0.82
1	Car Horn	0.94	0.78	0.85
2	Scream	0.96	0.93	0.94
3	Dog Bark	0.89	0.90	0.89
4	Calling Bell	0.91	0.93	0.92

Over all class 2(scream) is the best performing class. Class 0 (others) is the worst-performing class. Others is a vast category containing the greatest variety of sounds. Any sound that is not car horn, scream, dog bark or calling bell is supposed to go to this class. In the overall dataset, "others" sounds are higher in number, just as in real life, which affected the model's result.

Precision:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

The precision of a class measures how many of the predictions of that class were correct. High precision means low false positives.

Recall:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

The recall value indicates how many times the actual sound of a category was correctly identified. High recall indicates low false negatives.

F1 Score:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1 score is the harmonic mean of Precision and Recall.

Precision of class 0 is comparatively low, which means sounds of different categories are sometimes falsely predicted to be in "others". Recall of class 1 is the lowest, indicating some car horns are often predicted to be on some other category. Others classes' f1 score is lowest as it has comparatively low precision and recall values.

Precision is the highest for class 2, which means the model rarely falsely classifies a sound as a scream. Scream and calling bell have the highest recall value, which means if someone screams or rings a calling bell, the model will rarely predict it as any other category. Scream has best precision and recall hense it also has the highest f1 score.

5.1.2 Sound Localization Performance Evaluation

To evaluate the localization process, real time testing was conducted.After building the set up, different sound sources were provided from different angles. Within the range, each mic shows 100% accuracy. When the source of sound goes slowly out of range, the accuracy starts to drop. Here, each mic was assigned ten tests. So, a total of 40 sets of actual vs predicted data were collected. At the point of going out of range, the accuracy was 80.01%.
The following table shows the precision, recall and f1-score of the colleceted dataset:

Table 5.2: Localization Performance for Each Mic

Mic	**Precision**	**Recall**	**F1-Score**
Front	0.69	0.90	0.78
Back	0.80	0.80	0.80
Left	1.00	0.70	0.82
Right	0.80	0.80	0.80

5.2 Analysis of Design Solutions

5.2.1 Analysis of the Sound Recognition

The proposed model is definitely viable because it has been deployed in smartphones and in smartwatches. It is compact in size and gives usable predictions in real-world

applications. The model was trained using a comparatively small dataset. The model's performance can be improved by using a larger dataset.

5.2.2 Analysis of the Sound Detection Application

The sound detection application was meant to offer real-time audio classification and localization for the hearing-impaired users. The operational efficiency of the solution is one of its major strengths. The app can do inference instantly through the use of the lightweight TensorFlow Lite model, which is directly integrated into Android and Wear OS devices without depending on cloud services. This is to guarantee quick predictions with low latency, which is suitable for real-time situations. Also, the app is accurate in detecting target sound categories.

In terms of usability, the app offers a simple and intuitive user experience. The use of everyday devices like smartphones and smartwatches will result in very little extra cost to end users, as they will not require any special hardware other than the inexpensive headset. The user interface is designed using Jetpack Compose and Material 3 elements and has a simple and intuitive look for ease of use. Vibration feedback also helps, as it gives immediate feedback to the user based on the detected sounds.

Despite these strengths, there are some limitations. The functionality of the app is determined by the capabilities of the devices, like the quality of the microphones and the processing power, which differ device to device. Although the model is lightweight, extremely low-end devices could have some slight delays or higher battery usage in case of prolonged use. The Wear OS recording gap is higher compared to Android app. Consequently, audio detection is quicker than Wear OS. Furthermore, the detection would have been faster if it recognized one-second audio as opposed to three-second audio.

Scalability is also a factor since more sound categories or the capability to support multiple simultaneous sound sources might need retraining or a redesign of the model, and this might also raise computational requirements.

The application is accessible and affordable to the target users in terms of cost. As the standard Android development tools and common hardware are all that is needed to develop and deploy it. As a result, the design offers a reasonable and efficient assistive device for hearing-impaired individuals while remaining mindful of potential limitations and areas for future improvement.

5.2.3 Analysis of the Sound Localization

As a separate solution connected via wifi with the solution device, it is very light weight, user friendly and easy to carry. Any person regarding age, gender can wear it without facing any difficulties. It has a simple logic to calculate its direction, so the computational power does not increase due to the algorithm. If the logic implementation gets heavier due to future modification, it can easily access the processor of the smartphones as it is already connected to them. Processors of Android

smartphones can offer much better processing units than common microcontrollers.

The main limitation of the hardware is the range it is covering. Further improvement in building mic setup is necessary to overcome this limitation. Introducing microphone array and beamforming technology may increase the accuracy of the localization process.

5.3 Statistical Analysis

5.3.1 Sound Recognition Confusion Matrix

Mapping the confusion matrix as a heatmap helped to easily visualize the performance of each class. It is clear from the confusion matrix that most of the predictions made by the model were correct. It can also be seen that high number of false positives are in class 0(others), which reaffirms that this class has the lowest precision.

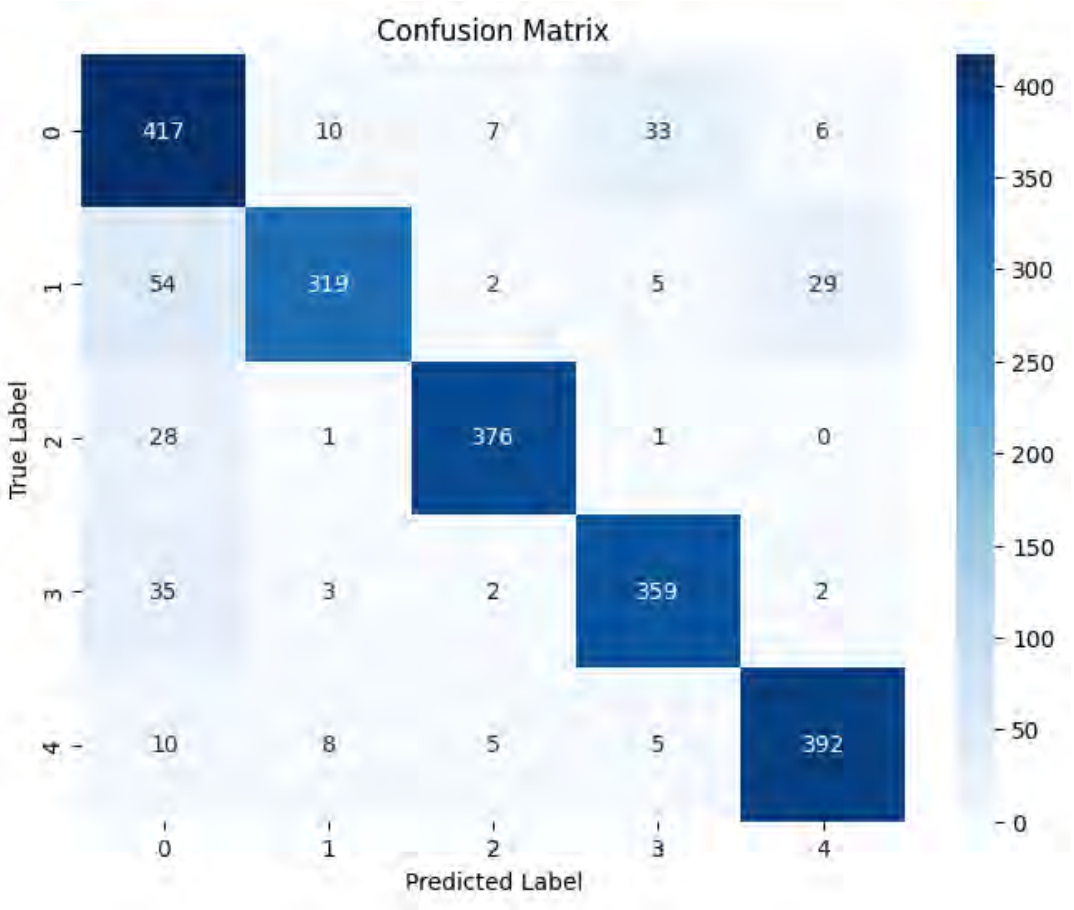


Figure 5.1: Confusion Matrix Sound Recognition

From the precision score and confusion matrix, it is evident that the model sometimes predicts “Others” even when the true label is something else. From using the model on smart mobiles and smart watches, it was found that the model predicts correctly when the sound is near or loud. However, if the sound is far away and faint it is sometimes predicted as others

5.3.2 Sound Localization Confusion Matrix

The heatmap of the collected dataset created from the confusion matrix is pictured below. The performance is recorded while the sound source is near the edge of localization range. Here the result shows, for the front mic, 9 out of 10 results were correct. For both the back & right mic, 8 out of 10 results were correct. And for the left mic, 7 out of 10 assumptions were correct.

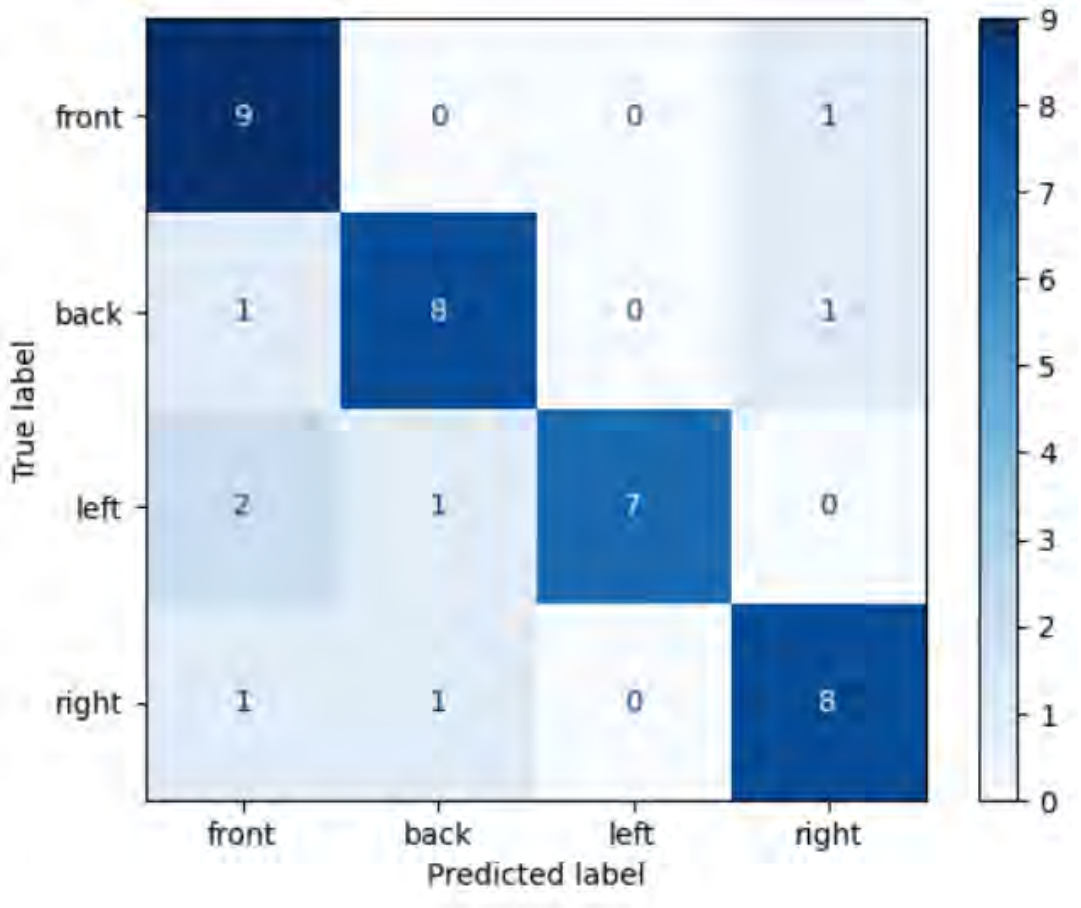


Figure 5.2: Confusion Matrix of Localization

5.4 Comparisons and Relationships

For smartphone deployment, our study required a model that could be converted to TFLite format for mobile deployment. YAMNet is a popular sound classification model available in TFLite format.

Table 5.3: Comparison of Sound Recognition Models

Model Name	**Classes**	**Parameters**
Custom CNN model	5	723,269
YAMNet	521	3.7 million

YAMNet is much larger in size than our custom model, as it has 5 times more parameters. YAMNet includes the five classes of the proposed model and 516 more.

It can identify guitar sounds, wind, ocean, clap, etc categories, which is not possible for our custom model. However, these are not necessary for a person with hearing impairment. These will not help their safety in regular life. Pre-trained models can identify more sounds at the cost of being bulky in size. Our proposed model identifies five specifically chosen classes at a very small compact size.

5.5 Discussion

The performance evaluation was done to evaluate the accuracy and reliability of the System. The Custom CNN model achieved an overall accuracy of 88.34%, with the highest F1-score observed for the 'scream' class (0.92). The four distinct classes have specific sounds; the "others" class is not distinctive at all, and sounds with many different characteristics belong here. That could be a possible explanation for its low precision. The training accuracy is 4.3% higher than the validation accuracy. Which means the model is still slightly overfitting. Making the dataset larger and more robust can help prevent it. This model is compact in size, it was converted to TFLite for deployment, and gives reliably accurate predictions in real-time, especially with loud and clear sounds. Therefore, the model meets the requirements for the study.

5.5.1 Performance Evaluation of Android and Wear OS Applications

The two applications that have been developed for Android and Wear OS illustrate an efficient real-time sound classification and direction recognition system. An Android application was used as the first test platform to provide a strong implementation of audio recording, mel spectrogram, and TFLite-based classification of sounds that included car horns, dog barks, and screams. After being confirmed, these essential functions were adjusted to the Wear OS application and allowed compact, on-wrist sound awareness with vibration and visual displays.

The apps demonstrated that real-time sound classification and localization are feasible at high efficiency on both mobile and wearable devices. Android applications had more processing capabilities and display, and Wear OS was more focused on responsiveness and accessibility. To resolve the problem of interference, a small break of the microphone during the vibration feedback was made in the Wear OS, which ensured a more accurate detection.

According to the results, there is a high possibility of real-life implementation, especially for users with hearing impairments. Nonetheless, drawbacks in the form of interference with background noise, small training data, and short battery life on the smartwatch are still challenges. Future developments may involve multi-microphone recording for better accuracy.

Overall, both Android and Wear OS apps will play an essential role in creating affordable and accessible AI-based sound recognition and localization, as well as wearable technology, to support safety, awareness, and inclusivity in daily life.

Chapter 6

Conclusion

6.1 Summary of Findings

This study's goal of developing a cost-effective sound recognition and localization system that notifies hearing-impaired individuals of specific sounds and their location was met. We found that up to 88% accuracy is achievable in sound recognition using a small Convolutional Neural Network. The model is deployable on smartphones, which are easily accessible and affordable for the general public. Also, a well-designed end-to-end model enables real-time processing, even on small devices such as smartwatches. Furthermore, giving vibration feedback is the most optimal way to notify a user who is hearing impaired when a sound class is recognized. We also found that it is possible to accurately localize four directions, which are front, back, left, and right, using four microphones which are positioned around a person's head in the headset. We were able to make it totally wearable, portable, and an offline device. People can use it whether in their home or office, or any other place. People will have the opportunity to consider it as a substitute for a hearing aid.

6.2 Contribution to the Field

Traditional hearing aids that are expensive, our aim was to create a cost-effective alternative to help users get alerted through the integration of a downloadable application software for phones or for smart watches in combination with a wearable headset. The approach delivered satisfactory results in indoor and even in noisy environments, which has always been a challenge traditionally in terms of sound classification and localization, especially during real-world scenarios.

6.3 Future Works

- Refinement will focus on lowering the audio capture window from 3 seconds down to 1 second. For real time scenarios it is optimal for a short burst of input to be processed swiftly to improve responsiveness.
- Rather than sequentially processing 3-seconds audio segments we plan to adopt a sliding window approach in which overlapping audio segments will be analyzed. This method will help the model to capture finer variations and tran-

sient sounds that might occur between the fixed intervals thereby enhancing the detection accuracy.

- At present the system captures audio at two sources namely electret microphones of the headset and built-in device microphones. Further improvement goals aim to merge the input source to a singular point on top of the headset to feed into the CNN.
- For improved directional accuracy and multi-angle sound localization, the system will incorporate a microphone array consisting of more than four microphones to capture sounds from angles beyond just the four primary directions.
- Further work will also aim to enhance the localization distance of the electret microphones to capture and process sounds from an increased effective range.
- As well as an addition of a severity scale for equal intensity noise coming from multiple directions, to process which direction and sound should be given priority

Bibliography

[1] A. Saxena and A. Y. Ng, "Learning sound location from a single microphone," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 1737–1742. DOI: 10.1109/ROBOT.2009.5152861

[2] H. Lim, I.-C. Yoo, Y. Cho, and D. Yook, "Speaker localization in noisy environments using steered response voice power," *IEEE Transactions on Consumer Electronics*, vol. 61, no. 1, pp. 112–118, 2015. DOI: 10.1109/TCE.2015.7064118

[3] H. Archana, C. Chellaram, and M. Rajalakshmi, "Obstacle detection for visually impaired patients," in *2014 International Conference on Science Engineering and Management Research (ICSEMR)*, 2014, pp. 1–3. DOI: 10.1109/ICSEMR.2014.7043565

[4] K. Audhkhasi, A. Rosenberg, A. Sethy, B. Ramabhadran, and B. Kingsbury, "End-to-end asr-free keyword search from speech," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1351–1359, 2017. DOI: 10.1109/JSTSP.2017.2759726

[5] P. Vecchiotti, E. Principi, S. Squartini, and F. Piazza, "Deep neural networks for joint voice activity detection and speaker localization," in *2018 26th European Signal Processing Conference (EUSIPCO)*, 2018, pp. 1567–1571. DOI: 10.23919/EUSIPCO.2018.8553461

[6] M. S. Mahamud and M. S. R. Zishan, "Watch it: An assistive device for deaf and hearing impaired," in *2017 4th International Conference on Advances in Electrical Engineering (ICAEE)*, 2017, pp. 556–560. DOI: 10.1109/ICAEE.2017.8255418

[7] A. H. Michaely, X. Zhang, G. Simko, C. Parada, and P. Aleksic, "Keyword spotting for google assistant using contextual speech recognition," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2017, pp. 272–278. DOI: 10.1109/ASRU.2017.8268946

[8] M. Yağanoğlu and C. Köse, "Real-time detection of important sounds with a wearable vibration based device for hearing-impaired people," *Electronics*, vol. 7, no. 4, 2018, ISSN: 2079-9292. DOI: 10.3390/electronics7040050 [Online]. Available: https://www.mdpi.com/2079-9292/7/4/50

[9] H. Seki, T. Hori, S. Watanabe, J. Le Roux, and J. R. Hershey, "A purely end-to-end system for multi-speaker speech recognition," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, I. Gurevych and Y. Miyao, Eds., Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 2620–2630. DOI: 10.18653/v1/P18-1244 [Online]. Available: https://aclanthology.org/P18-1244

[10] M. D. Fletcher, A. Hadeedi, T. Goehring, et al., "Electro-haptic enhancement of speech-in-noise performance in cochlear implant users," *Scientific Reports*, vol. 9, p. 11 428, Aug. 2019. DOI: 10.1038/s41598-019-47718-z [Online]. Available: https://doi.org/10.1038/s41598-019-47718-z

[11] X. Chang, W. Zhang, Y. Qian, J. L. Roux, and S. Watanabe, "Mimo-speech: End-to-end multi-channel multi-speaker speech recognition," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019, pp. 237–244. DOI: 10.1109/ASRU46091.2019.9003986

[12] S. Shen, D. Chen, Y.-L. Wei, Z. Yang, and R. R. Choudhury, "Voice localization using nearby wall reflections," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '20, London, United Kingdom: Association for Computing Machinery, 2020, ISBN: 9781450370851. DOI: 10.1145/3372224.3380884 [Online]. Available: https://doi.org/10.1145/3372224.3380884

[13] M. D. Fletcher, H. Song, and S. W. Perry, "Electro-haptic stimulation enhances speech recognition in spatially separated noise for cochlear implant users," *Scientific Reports*, vol. 10, p. 12 723, Jul. 2020. DOI: 10.1038/s41598-020-69697-2 [Online]. Available: https://doi.org/10.1038/s41598-020-69697-2

[14] C. Kim et al., "A review of on-device fully neural end-to-end automatic speech recognition algorithms," in *2020 54th Asilomar Conference on Signals, Systems, and Computers*, 2020, pp. 277–283. DOI: 10.1109/IEEECONF51394.2020.9443456

[15] M. H. Korayem, S. Azargoshasb, A. H. Korayem, and S. Tabibian, "Design and implementation of the voice command recognition and the sound source localization system for human–robot interaction," *Robotica*, vol. 39, no. 10, pp. 1779–1790, 2021. DOI: 10.1017/S0263574720001496

[16] M. D. Fletcher, J. Zgheib, and S. W. Perry, "Sensitivity to haptic sound-localization cues at different body locations," *Sensors*, vol. 21, no. 11, 2021, ISSN: 1424-8220. DOI: 10.3390/s21113770 [Online]. Available: https://www.mdpi.com/1424-8220/21/11/3770

[17] H. Aldarmaki, A. Ullah, S. Ram, and N. Zaki, "Unsupervised automatic speech recognition: A review," *Speech Communication*, vol. 139, pp. 76–91, 2022, ISSN: 0167-6393. DOI: https://doi.org/10.1016/j.specom.2022.02.005 [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167639322000292

[18] S. Chen, R. Tan, Z. Wang, X. Tong, and K. Li, "Voicemap: Autonomous mapping of microphone array for voice localization," *IEEE Internet of Things Journal*, vol. 11, no. 2, pp. 2909–2923, 2024. DOI: 10.1109/JIOT.2023.3294937

[19] M. D. Fletcher, C. A. Verschuur, and S. W. Perry, "Improving speech perception for hearing-impaired listeners using audio-to-tactile sensory substitution with multiple frequency channels," *Scientific Reports*, vol. 13, p. 13 336, Aug. 2023, Received 03 March 2023, Accepted 11 August 2023, Published 16 August 2023. DOI: 10.1038/s41598-023-40509-7 [Online]. Available: https://doi.org/10.1038/s41598-023-40509-7

[20] M. D. Fletcher, S. W. Perry, I. Thoidis, et al., "Improved tactile speech robustness to background noise with a dual-path recurrent neural network noise-reduction method," *Scientific Reports*, vol. 14, p. 7357, Mar. 2024. DOI: 10.1038/s41598-024-57312-7 [Online]. Available: https://doi.org/10.1038/s41598-024-57312-7

[21] M. Huzaifah, "Comparison of time-frequency representations for environmental sound classification using convolutional neural networks," *arXiv preprint arXiv:1706.07156*, 2017.