

LB/DON/62/2020

DCM 04/57 ✓

AN INTELLIGENT HARDWARE SYSTEM FOR REAL-TIME INFANT CRY DETECTION AND CLASSIFICATION

U.P. Pradeep Dushyantha Pathirana

179411X



LIBRARY
UNIVERSITY OF MORATUWA, SRI LANKA
MORATUWA

Dissertation submitted in partial fulfillment of the requirement of the
degree of Master of Science in Artificial Intelligence

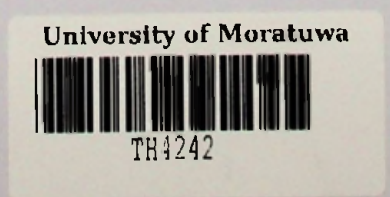
Department of Computational Mathematics

University of Moratuwa

Sri Lanka

January 2020

519.6 "2020
004.8 (043)



TH 4242 + CD-ROM
is not working

TH 4242

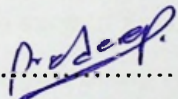
Declaration

I declare that this dissertation does not incorporate, without any acknowledgment, any material previously submitted for a degree or a diploma in any university and to the best of my knowledge and belief, it does not contain any material previously published or written by any other person or myself except where due reference is made in the text. I also hereby give consent for my dissertation if accepted, to be made available for photocopying and for interlibrary loans, and for the title and summary to be available to outside organizations.

Name of Student

Signature of Student

U.P. Pradeep Dushyantha Pathirana

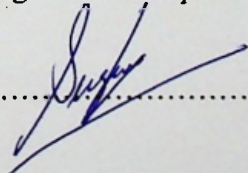
.....


The above candidate has carried out research for the Master's dissertation under my supervision.

Supervisor

Signature of Supervisor

Dr. Sagara Sumathipala

.....


Acknowledgment

I would like to extend my heartfelt gratitude to my project supervisor Dr. Sagara Sumathipala for guiding me through the research. He extended his fullest corporation whenever I sought for advice and guidance to make this project a success. On the same note, I would also like to thank him for his guidance and support as the course coordinator during the entire course period. Similarly, I would like to extend my sincere gratitude to the entire academic staff of the Department of Computational Mathematics for their guidance throughout the course period. Further, I must thank the non-academic staff for their continuous support during the entire course in providing the learning environment to the best of their ability.

I would like to thank the entire management and research team of Synergen Technology Labs (Pvt) Ltd, for their permission to extend this project as my research project and providing me the required resources for its successful completion.

Abstract

Cry, the universal communication language of the infants encodes vital information about the physiological and psychological health of the infant. Experienced caregivers can understand the cause of cry based on the pitch, tone, intensity, and duration. Similarly, pediatricians can diagnose hearing impairments, brain damages, and asphyxia by analyzing the cry signals, providing a non-invasive mechanism for early diagnosis in the first few months. Hence, automated cry classification has gained great importance in the fields of medicine and baby-care. With the emergence of the concept of the Internet of Things coupled with Artificial Intelligence, baby monitors have recently gained huge popularity due to features like sleep analysis, cry detection, and motion analysis through multiple sensors. Since cry classification involves audio processing in real-time, most of the solutions have either complex and costly designs or distributed computing, which leads to privacy concerns of the users. This research presents a low-cost intelligent hardware system for real-time infant cry detection and classification. The proposed solution presents the selection of the hardware to suit the requirements of audio processing while adhering to financial constraints and the firmware design, which includes voice activity detection, cry detection, and classification. This proposes the use of the multi-agent system as a resource management concept while proving that AI concepts can also be extended to resource-limited hardware platforms as the novelty. Firmware and algorithm are designed to maintain the accuracy figures above 90% while processing the audio signal at a higher rate than its production to maintain stability. A voice activity detector was designed to filter human voice through temporal features while cry detection and classification were respectively based on Artificial Neural Network and K-Nearest Neighbor algorithm trained with a spectral-domain feature vector called Mel Frequency Cepstral Coefficients (MFCC). Evaluations under diverse conditions showed accuracy figures of 96.76% and 77.45% in cry detection and classification, respectively.

Keywords: Cry Detection, Cry Classification, Voice Activity Detection

Contents

1	INTRODUCTION	1
1.1	Prolegomena	1
1.2	Research Problem.....	2
1.3	Aims and Objectives	2
1.4	Background and Motivation.....	3
1.5	Problem in Brief	4
1.6	The Novelty in The Proposed Solution	6
1.7	Structure of the Thesis.....	6
1.8	Summary	7
2	CURRENT TRENDS AND ISSUES IN INFANT CRY DETECTION	8
2.1	Introduction	8
2.2	Baby Monitors.....	8
2.3	Cry Detection and Classification in Baby Monitors	9
2.4	Recent Works in Cry Detection and Classification.....	9
2.5	Baby Monitors with Cry Detection	14
2.6	Summary	15
3	INCORPORATED TECHNOLOGIES	16
3.1	Introduction	16
3.2	Multi-Agent Technology	16
3.3	Artificial Neural Network	19
3.4	K-Nearest Neighbour	21
3.5	Summary	22
4	HYPOTHESIS	23
4.1	Introduction	23
4.2	Hypothesis.....	23
4.3	Input.....	23
4.4	Output.....	24

4.5	Process.....	24
4.6	Users.....	24
4.7	Summary	25
5	HARDWARE AND FIRMWARE DESIGN.....	26
5.1	Introduction	26
5.2	Hardware Design.....	26
5.3	Firmware Design	28
5.4	Audio capturing and Noise filtering.....	30
5.5	Temporal Feature Extraction and Voice Activity Detection.....	31
5.6	Spectral Feature Extraction and Cry detection.....	31
5.7	Cry Classifier.....	32
5.8	Summary	32
6	IMPLEMENTATION OF CRY CLASSIFIER	34
6.1	Introduction	34
6.2	Hardware Selection	34
6.3	Firmware Design	36
6.4	Summary	63
7	EVALUATION OF THE SYSTEM.....	65
7.1	Introduction	65
7.2	Test Scenarios.....	65
7.3	Test Setup.....	65
7.4	Definitions of the Evaluation Parameters.....	66
7.5	Performance of the Cry Detector.....	68
7.6	Performance of the Cry Classifier	69
7.7	Summary	73
8	CONCLUSION AND FUTURE WORK	74
8.1	Introduction	74
8.2	Conclusion.....	74
8.3	Limitation and Related Future Works	74

REFERENCES.....	76
APPENDIX A: Source Codes.....	79
The Preprocessor Agent.....	79
Selection of The Best KNN Classifier for Cry Detection.....	81
Selection of The Best ANN Classifier for Cry Detection.....	82
Cry Classifier.....	83

List of Figures

Figure 4.1 Flow diagram of the process.....	24
Figure 5.1 Hardware architecture.....	27
Figure 5.2 Firmware design	30
Figure 6.1 OSD3358 microprocessor.....	35
Figure 6.2 Hardware prototype	35
Figure 6.3 Pulse density modulated sinusoid.....	38
Figure 6.4 Frequency response of the compensation filter	40
Figure 6.5 Raw cry signal vs. filtered cry signal.....	40
Figure 6.6 Song filtered by compensation filter.....	41
Figure 6.7 Comparison of Raw signal vs. Down-sampled signal	42
Figure 6.8 Bandwidth of raw signal	42
Figure 6.9 Bandwidth of the down-sampled signal	43
Figure 6.10 Short Time Energy of cry signal.....	45
Figure 6.11 Short-Time Zero-Crossings of a cry signal	46
Figure 6.12 Short-Time Zero-Crossing of a noise signal.....	47
Figure 6.13 Block diagram of the Voice Activity detector.....	48
Figure 6.14 Input and output of the Voice Activity Detector	49
Figure 6.15 MFCC vector of the cry signal	49
Figure 6.16 MFCC vector of the noise signal.....	50

List of Tables

Table 6.1 Accuracy of KNN cry detector with uniform weights	53
Table 6.2 Accuracy of KNN cry detector weighted with distance	53
Table 6.3 Accuracy of ANN cry detector with sigmoid and Tanh functions	55
Table 6.4 Accuracy of ANN cry detector with ReLU and Identity functions	55
Table 6.5 Accuracy of KNN cry classifier with uniform weights	58
Table 6.6 Accuracy of KNN cry classifier weighted by distance	59
Table 6.7 Accuracy of ANN cry classifier with sigmoid and Tanh functions	60
Table 6.8 Accuracy of ANN cry classifier with ReLU and Identity functions	60
Table 7.1 Output of the cry detector under various test scenarios	68
Table 7.2 Accuracy of the cry detector under various test scenarios	68
Table 7.3 Confusion matrix for cry frame classification	70
Table 7.4 Performance evaluation of cry frame classification	70
Table 7.5 Confusion matrix for cry-event classification	71
Table 7.6 Performance evaluation of cry-event classification	71

List of Abbreviations

ANFIS	Adaptive Neuro-Fuzzy Inference System
AI	Artificial Intelligence
ANN	Artificial Neural Network
BFCC	Bark Frequency Cepstral Coefficients
CIC	Cascaded Integrator Comb
CNN	Convolution Neural Network
CDHMM	Continuous Density Hidden Markov Model
FFT	Fast Fourier Transform
FN	False Negative
FP	False Positive
HMM	Hidden Markov Model
IoT	Internet of Things
MLP	Multi-Layer Perceptron
KNN	K-Nearest Neighbor
LPC	Linear Predictive Coding
LPCC	Linear Predictive Cepstral Coefficients
MFCC	Mel Frequency Cepstral Coefficients
PDM	Pulse Density Modulation
PHMM	Probabilistic Hidden Markov Model
PNN	Probabilistic Neural Network
PPG	Photoplethysmogram
PRU	Programmable Realtime Unit
RBNN	Radial Basis Neural Network
RDS	Respiratory Distress Syndrome
RNN	Recurrent Neural Network
SCP	Secure Copy Protocol
SIDS	Sudden Infant Death Syndrome
SoC	System on Chip
SSH	Secure Shell
STFT	Short-Time Fourier Transform
SVM	Support Vector Machine

TDNN	Time Delay Neural Network
TN	True Negative
TP	True Positive
WT	Wavelet Transform

1 INTRODUCTION

1.1 Prolegomena

The concept of “Internet of Things” (IoT) has brought different new fields into light leading to a wealth of information about different aspects of its users. Fields like telemedicine, elderly care, and remote baby monitoring have gained a boost from the traditional solutions due to the introduction of new features supported by the concepts such as Artificial Intelligence (AI) and distributed computing. Busy lifestyles of new generations have forced them to seek help from these technologies to take care of themselves, parents, children, and loved ones. Nowadays, young parents show a new tendency to purchase baby monitors to keep them updated about their infants’ vital information such as heart rate, oxygen saturation, body temperature, and sleep cycles. Well established believes such as sleep deprivation, abnormal changes in vitals could lead their infants to long-term health hazards, have made them nervous and interested in these different solutions. Similar to sleep cycles and other vitals, the cries of an infant should be given the same importance since they act as the only communication mechanism with their parents. Researchers have shown that they encode information not only about the basic needs such as hunger, pain, discomfort but also about health issues such as hearing impairments, brain damages, and asphyxia.

Generally, experienced parents and caregivers can understand the causes of the infant's cries after analyzing the pitch, tone, intensity, and duration. Similarly, experienced pediatricians and nurses can diagnose the hearing impairments and brain damages of infants by analyzing cry signals. Medically, cry analysis has paved a non-invasive way of diagnosing health problems, at a very early age when the infant cannot even spell a word.

A significant number of researches have been conducted to detect and classify these infant cry signals under different classes related to the basic needs and health problems. Similarly, cry detection and classification have been attempted in baby monitors recently as a value addition to the existing features. This presents its own challenges

such as manufacturing cost, resource limitations regulated by the selling price, and real-time audio processing if the design is thought of as a stand-alone device. A solution coupled with distributed computing technologies may pose the challenges of recurrent cost of the servers (annual subscription fee), privacy concerns of audio streaming, and excess use of user's bandwidth for audio streaming, etc. Hence, this research focuses on the design and implementation of low-cost intelligent hardware design for real-time cry detection and classification.

1.2 Research Problem

This research work focuses on implementing a low-cost hardware solution for infant cry detection and classification in real-time in robust environments. The hardware solution should be a low-cost design to meet a competitive selling price in the market. Similarly, the solution should be a stand-alone solution to avoid privacy concerns of the user regarding the audio streaming, excess use of the user's bandwidth for audio streaming, and the recurrent server cost due to the continuous use of the servers for audio processing in a distributed computing environment. But due to the constraints of the cost, hardware resources available would be limited. Cry detection covers the scope of discrimination between infant cry-events and other audio events. Similarly, cry classification covers the scope of identification of the most likely cause of the cry. Real-time processing requires to process the audio feed at the rate of production to be able to notify the caregivers whenever the baby cries. The solution should function accurately in noisy environments with an acceptable level of accuracy to prove its reliability in robust environments.

1.3 Aims and Objectives

The aims of the research were defined as follows.

- To get the hands-on experience of the practical use of theories and technologies of Artificial Intelligence (AI).

- To evaluate the feasibility of the practical use of AI technologies in the resource-limited hardware platforms with acceptable accuracy levels. Proof of concept (POC)
- Design a hardware solution which can support stand-alone real-time audio processing
- Design a firmware solution that can support real-time audio processing to support cry detection and cry classification in noisy conditions.

The objectives of the following research were identified as

- Design and implementation of an audio processing algorithm to detect infant cries with an accuracy level above 95% and to predict the most-likely cause of the cry with an accuracy level above 75%.
- Implementation of the algorithms in a stand-alone hardware platform to monitor infants continuously in real-time.
- Device functionality in the domestic noisy conditions with an accuracy above 95% for cry detection and 70% in cry classification.

1.4 Background and Motivation

The author currently works as a senior research engineer and a product manager in Synergen Technology Labs (Pvt) Ltd, which focuses on designing and manufacturing electronic products and software related to healthcare-related IoT products. The author leads a project related to design and manufacture a baby monitor, capable of measuring infants' heart rate, blood oxygen saturation, sleep cycles, and body temperature along with the room conditions like temperature and humidity. As a value addition to the system and to gain a competitive advantage, infant cry detection and classification was identified to be a useful feature. Yet the existing baby monitors do not support this feature due to the limited resources in the hardware layer and the recurrent cost involved in cloud solutions. Yet, in the market analysis to identify the user requirements and the key features parents would seek after, they highlighted the concern of knowing when and why their baby is crying. Due to the busy lifestyles and their general practice

to keep the infants in a separate room away from parents, they always have concerns about the health of the baby. Therefore, they even showed an interest in paying extra money to own a device with cry detection. Hence, the requirement of cry detection was highlighted.

In the literature review, it was noticed that there is a significant number of researches conducted recently related to infant cry detection and classification. The scope of the above researches covers the medical fields to support pediatricians and cry detection in domestic environments. Yet, almost all the researchers have attempted to solve these problems in resource-heavy environments such as computers and distributed systems. Further, none of them described the practical applicability in real-time audio processing. A very few exceptions were there, which focused on the hardware designs along with the required algorithms. Yet, the designs were not cost-effective due to the higher component costs of their complex designs.

Hence, due to the minimal number of researches available to support the implementation of a baby monitor with cry detection in real-time in a stand-alone hardware device, the author was motivated to conduct this research as a proof of concept of the cry classification in a stand-alone hardware device.

1.5 Problem in Brief

The use of baby monitors is a popular concept in western countries to monitor newborn babies. These devices cover lots of parameters related to infant health. The parameters like heart rate, blood oxygen saturation, and body temperature have been identified as the critical parameters. Generally, baby monitors with these three parameters only can be marketed easily, covering a good market share. But, irrespective of the similar importance of cry, cry detection only will not be such an influential force to penetrate the market. But the combination of all the above parameters could prove to be a very influential force for market penetration.

Nevertheless, the inclusion of cry detection and classification also creates its own challenges. This requires continuous and real-time audio processing. To support additional

computational requirements, the inclusion of additional computational units may increase the cost. Hence, adding dedicated hardware resources unnecessarily without optimizing may exceed the optimal market price. Therefore, the hardware resource management is highly important. This research focuses on the optimal hardware and firmware designs to manage the computational resources at hand. But someone may suggest using distributed computing to tackle the problem. But this creates more problems than it solves. Audio recording is compulsory, and it cannot be avoided. Yet, audio streaming will create doubts in the users' minds about privacy. This is a severe issue in western countries. Similarly, this possesses the problems of continuous use of ethernet adding recurrent cost to the user due to higher bandwidth requirements to stream audio. Further, due to the cloud resources, a recurrent server cost will also be added to the manufactures. Hence, this should be covered either by increasing the price or by introducing a monthly subscription. Both these approaches help in losing customers than gaining. So, the optimal solution in terms of marketability should be a stand-alone solution.

Cry detection feature should cover the scope of detecting the cry-events under the domestic conditions. This should be able to discriminate the audio activities like noise, lullabies played to help the baby fall asleep, and the other human voice activities like speaking to the baby, baby laughing, etc. Typically, parents often play lullabies/white noise in the infants' rooms to help them fall asleep. These white noises should not cause any problem to the above functionalities. Further, cry classification refers to the identification of the most likely cause of the cry. Typically, in domestic conditions, the causes may include hunger, belly-pain, burps, tiredness, and discomfort. But the most important and frequent causes are belly-pain, burps, and hunger. But new parents find it hard to understand the causes due to the lack of prior experience about the subtle variations of the cries, which encode the cause of cry. Hence, this research work should ultimately produce hardware and firmware designs to detect the infants' cries in real-time and inform the parent with the most likely cause of the cry.

1.6 The Novelty in The Proposed Solution

Literature reviews exposed that there is a minimal number of researches towards cry detection in hardware devices. But these researchers also have proposed complex hardware designs that involve at least two computing units such as Raspberry PI boards. Otherwise, they have sought the help of cloud computing for audio processing. In this research, the focus is to design a stand-alone hardware device that can record and process the audio in real-time to produce the expected results.

Most of the researchers have experimented with solutions in resource-heavy environments. Even the hardware designs proposed in these literatures have complex designs to include more computation power. Hence, resource management is not a concern. But this needs to face the challenge of resource management to tackle the resource limitation imposed by the manufacturing cost. Hence, this proposes the use of Multi-Agent Technologies to manage the resources effectively and solve the problem collaboratively, sharing knowledge.

These researches did not speak about real-time audio processing for cry detection. Most of these works have covered the medical field. The proposed solution focuses on real-time audio processing in noisy conditions for domestic use.

1.7 Structure of the Thesis

This chapter presented a basic overview of the research explaining the problem in brief and then discussing the aims and objectives. Chapter 2 discusses the current trends and issues in infant cry detection and classification, in general, highlighting the conclusions of the literature review. Chapter 3 is dedicated to a comprehensive analysis of the technologies used in the proposed solution. The hypothesis is presented in chapter 4. Chapter 5 elaborates on the proposed solution in detail, presenting the hardware and firmware design diagrams. Chapter 6 will walk the readers through the implementation of the proposed solution. This would provide a comprehensive step by step details analysis of the effect of each action on the outcome. Chapter 7 evaluates the proposed solution against the objectives defined in chapter 1 to decide whether the aims of the research were attained. Chapter 8 summarizes the overall conclusions drawn from the

evaluation and describes the issues and limitations. This would further discuss the future works to mitigate the issues and limitations found.

1.8 Summary

In this chapter, the background information about the importance of the research problem was highlighted discussing the product marketability and, more importantly, the user requirements. Further, the challenges of the implementation were discussed to highlight the scope of the problem at hand to resolve. After discussing the aims and objectives, the author discussed the source of motivation. Next, the problem was briefly elaborated, explaining the keywords such as cry detection, classification, real-time, etc. according to the context of this application. And this further discussed the reasons behind some constrains like low-cost and stand-alone. This includes the explanation of why a stand-alone solution is preferred over a distributed system. Finally, the novelty of the proposed solution was discussed, highlighting the key issues of the existing solutions, such as the need for resource management, real-time audio processing, and simplicity of the solution.



2 CURRENT TRENDS AND ISSUES IN INFANT CRY DETECTION

2.1 Introduction

Chapter 1 introduced the research problem and the related aims and objectives of solving the problem. This chapter discusses recent trends in cry detection and classification. The current trends in baby monitors will be discussed first to form the basis to explain the user requirements of automated cry detection and classification. Next, this will present the details of the literature review covering topics like research areas, the information in cry, temporal and spectral features of interest, and the different classifiers proposed by the previous researchers. Research areas covered in this literature cover both medical and commercial scopes. This only covers the important features in detail, while other minor features are mentioned briefly under the respective topics.

2.2 Baby Monitors

The evolution of the baby monitors has a long history starting from a listening device at the bed-head of the infant to a very sophisticated and intelligent device. Nowadays, these baby monitors are capable of monitoring the infants' behaviors either through cameras or wearable devices. Baby monitors with cameras record the video to analyze them for the movements of the baby. But baby monitors with wearable devices are capable of monitoring parameters like heart rate, body temperature, and oxygen saturation to estimate the health of the baby and warn if normal pattern breaks. These are becoming very popular because parents are conscious about the health of their babies rather than their movements. Sudden Infant Death Syndrome (SIDS) is a serious cause of death of the infants. But the reasons behind the SIDS are yet to be revealed. Hence, this has brought fear to the new parents about the life of their babies and forced them to seek the help of the technologies available.

These baby monitors capture the photoplethysmogram (PPG) signal to estimate the heart rate and blood oxygen concentration. Additionally, some baby monitors would measure body temperature through dedicated sensors. Although for all the parameters, there are dedicated sensors, none of them are capable of measuring these parameters directly. Hence, every vital is associated with a set of dedicated algorithms for noise filtering, parameter estimation, and outlier removal. This requires a significant level of computational power, which will result in a higher selling price. Similar to the above vital parameters, the cry is now thought of as important information about the physical and psychological health of the baby.

2.3 Cry Detection and Classification in Baby Monitors

The very first version of the baby monitors was quite similar to the walkie-talkie. The sole purpose was to listen to the baby from the receiver with the mom and identify when the baby is crying and awake so that the parents can attend to their child accordingly. Here the intelligent decision making was at the hand of the listener. Hence, the listener was supposed to listen continuously. But with the inclusion of the more intelligent features, cry detection lost its importance due to human effort required. Similarly, due to remote monitoring, listening to the baby requires continuous audio streaming, which adds a recurrent cost. But recently, with the developments in hardware resources for stand-alone mini-computers, a few researchers have started to think that an automated cry detection is a viable option.

2.4 Recent Works in Cry Detection and Classification

The literature review highlighted the following important details were highlighted by the researchers in their literature reviews and conclusions.

2.4.1 Coverage of the Research Works

In simple terms, cry detection and classification can be thought of as the identification of when and why the baby is crying. The cry of an infant may depend on multiple



reasons varying from the physical to mental reasons generally. For example, physical reasons may include hunger, belly pain, discomfort, and tiredness, whereas mental or psychological reasons may include loneliness, and fear, etc. Similarly, pediatricians look into the cry audio from a different angle to identify the symptoms such as the hearing losses, and asphyxia due to the deprivation of oxygen to the brain. Recent research works have been targeted at both medical and commercial applications of cry detection. But the majority of the research works are towards the medical field.

2.4.2 Details of A Cry

It is unarguably true that the cry acts as the primary mode of communication for the infants. Yet, cry does not seem to carry much of vital information. But, experienced care-givers will easily derive more information about the cry than a novice. Audio analysis has proven that there are noticeable differences in some features which can be helpful in identifying these causes of cry. Amulya A. Dixit and Nagaraj V. Dharwadkar have successfully researched and concluded that cries due to pain, discomfort, hunger, and sleep could be accurately classified with a proper audio feature analysis[1]. Asphyxia is a medical condition occurred due to the shortage of oxygen supply to the brain. It has been proven that cry signals can identify the asphyxia with more than 94% accuracy[2]. Similarly, hearing impairment is a common defect found in new-borns. But it is very hard to diagnose since the infants do not respond to what they will hear in the first few months. But experimentally, it has been proven that an infant with such a hearing impairment cries differently compared to a normal baby. What makes the cry different is the difference in the feedback of its own cry to the ears. It was experimentally proven that the ratio between the dominant and fundamental frequencies differs for infants with hearing impairments compared to a healthy baby[3].

2.4.3 Characteristics of A Cry

Fundamentally, the cry of an infant still falls under the superclass of the human voice. Therefore, it holds some unique qualities of the human voice. The frequency range of

a cry signal is from 200Hz to 500Hz with the average frequency of 320Hz and 400Hz for males and females, respectively[4]. Zero Crossing rate is a parameter defined to evaluate continuous signals in time-domain. This calculates the rate at which the signal crosses the time axis. Since the bandwidth of human voice is low compared to white noise, it has a low zero-crossing rate[5]. Similarly, the human voice has higher signal energy compared to white noise generally. Although a cry-event that lasts over a few seconds seems to be a continuous effort of the infant, it has some periodic pauses. Normally, a cry is a burst of short cries spaced between the pauses of 500-700ms[6]. These pauses are due to respiration.

2.4.4 Audio Features

When someone comments about the voice of a speaker, the critic speaks about the spectral domain features more than the temporal domain features. Temporal domain features do not capture the variations in pitch and tone. In the spectral domain, the signal is decomposed into different frequencies to analyze which frequencies are significant. Fundamental and dominant frequencies are two features that can reveal the details about the hearing impairments[3]. Cry detection and classification can be thought of as pattern matching. But, matching sample by sample is not effective. Hence, the signal will be represented by some spectral domain features.

2.4.4.1 Linear Predictive Coding (LPC)

Linear Predictive Coding (LPC), as the name suggests, represents the future sample as a linear combination of the immediate past samples. This analysis uses the Z transformation to analyze the spectral features of a discrete signal. Linear Predictive Cepstral Coefficients (LPCC) maps the LPC coefficients to the cepstral domain features[5].

2.4.4.2 Short-Time Fourier Transform (STFT)

Fourier Transform converts the temporal signal into the spectral domain signal by estimating the frequencies present in the signal. Since the signal is subjected to

significant changes over time, spectral-domain analysis over a large window does not provide an accurate picture of the signal. Hence STFT is proposed to suppress the significant changes over time since it shortens the time window of interest. STFT has also shown promising results in some research works[1].

2.4.4.3 Mel Frequency Cepstral Coefficients (MFCC)

Mel Frequency Cepstral Coefficients (MFCC) is another representation of the audio signal for voice recognition applications. These coefficients model the human perception of a given audio signal with respect to the human auditory system. Using MFCC as the feature vectors, classifiers have been successfully trained to predict the cry reasons such as Respiratory Distress Syndrome (RDS) as well[7].

2.4.4.4 Bark Frequency Cepstral Coefficients (BFCC)

Bark Frequency Cepstral Coefficients (BFCC) is another representation similar to MFCC. This maps the power of the signal to the human perception of the loudness of the signal. This has shown better accuracy figures compared to MFCC in some instances[5].

2.4.4.5 Empirical Mode Decomposition (EMD)

Empirical Mode Decomposition (EMD) is a special technique to convert the signal into the constituent frequencies being in the time domain. This is a computationally-intensive process compared to the Fast Fourier Transform (FFT). This extracts different frequency components similar to what FFT produces. These components have been successfully tested as the feature vector to detect and classify cries[8].

2.4.4.6 2D spectrogram

The 2D spectrogram is a representation of the signal in both time and frequency. This presents the variations of the spectral domain features along the time axis. Hence this

gives a clear image of the variation of the important frequencies with time. The 2D spectrogram is a popular feature vector in deep learning in signal processing applications. This has shown good performances in cry detection and classification when combined with deep learning concepts like Convolution Neural Networks (CNN)[4], [6],[9].

2.4.5 Audio Classifiers

Cry detection and classification both require the intelligence to predict output based on past observations/training data. Hence, different researchers have proposed different classifiers with the combination of the above features to classify the inputs accurately. These classifiers differ from lazy learning algorithms to deep learning algorithms. In contrast, some researchers have proposed the use of classifiers like Fuzzy Logic, and Hidden Markov Model (HMM). Yet, most of the researchers proposed the use of the following classifiers.

2.4.5.1 K-Nearest Neighbors (KNN)

Researchers have shown interest in experimenting with the KNN algorithm in cry detection and classification. KNN classifier with MFCC as the feature vector has shown better performances [10]. KNN had shown the accuracies in the range of 60-70% in cry classification when it was trained with features such as Mel Frequency Cepstral Coefficients (MFCC), Bark Frequency Cepstral Coefficients (BFCC) and Linear Predictive Coding (LPC)[5].

2.4.5.2 Artificial Neural Network (ANN)

Research works suggest that Artificial Neural Networks also performs better in this domain. ANN classifiers trained with MFCC and BFCC have also shown the accuracy figures of 60.45% and 76.47%, respectively, in cry classification[5], [7].

2.4.5.3 Convolution Neural Network (CNN)

Convolution Neural Network is a popular choice with the 2D spectrogram as the feature vector. This suits the resource-heavy environments. CNN has yielded 82% accuracy in cry detection in adverse domestic conditions[9]. To benchmark the performance, this was compared with a linear regression model, which yielded 81% in the best case.

2.4.5.4 Other Classifiers

Apart from the above-highlighted classifiers, the use of the following classifiers was also highlighted in the literature reviews.

- Recurrent Neural Network (RNN)
- Probabilistic Neural Network (PNN)
- Time Delay Neural Network (TDNN)
- Radial Basis Neural Network (RBNN)
- Hidden Markov Model (HMM)
- Probabilistic Hidden Markov Model (PHMM)
- Continuous Density Hidden Markov Model (CDHMM)
- Adaptive Neuro-Fuzzy Inference System (ANFIS)
- Support Vector Machine (SVM)
- Fuzzy Support Vector Machine (FSVM)

2.5 Baby Monitors with Cry Detection

There are a few numbers of research attempts towards building the baby monitors capable of cry detection. Prithvi Raj et al. has successfully experimented with cry detection using a baby monitor design, which included two Raspberry PI boards and one Arduino board [11]. There is another attempt in which, instead of cry detection, noise detection has been chosen to serve a similar purpose[12]. This concept works on the assumption that the infants' rooms are generally very quiet. This involves a comparatively simple hardware design.

2.6 Summary

This chapter discussed the concept of baby monitors and their importance. It described the birth of the concept of baby monitors as a listening device for infants' cries and then evolving as a sophisticated device. Due to the human effort required, cry detection started to lose importance, and recently, with automatic cry detection, this started to gain attention. Next, findings of the literature review were discussed in detail, highlighting the importance of cry detection, information encoded in cries, characteristics, spectral-domain features, and classifiers. This further showed frequent use of Mel-Frequency Cepstral Coefficients as the feature vector. Even though the different researchers had suggested different types of classifiers with a wide variety of characteristics, the most popular and effective classifiers are KNN and ANN. To conclude, the previous efforts to implement cry detection in baby monitors were discussed. It revealed that complex designs had been proposed while another research had suggested looking at the problem in the angle of noise detection instead of cry detection.

3 INCORPORATED TECHNOLOGIES

3.1 Introduction

In this chapter, the author explains the background and potential of each technology highlighted in this research work. First, the concept of Multi-Agent Systems (MAS) will be explained with its advantages and practical applications in resource management. The next two sections are dedicated to cover the background and the applications of the K-Nearest Neighbor algorithm and Artificial Neural Network as the main classifiers of interest for the audio recognition applications. These sections will walk the readers through the basic concepts, important parameters/techniques related to the accuracy, and their typical applications.

3.2 Multi-Agent Technology

A Multi-Agent System is a computing system composed of multiple intelligent agents working towards a common goal of solving a complicated or impossible problem for a single agent [13]. These agents can act independently and autonomously on behalf of the user in either standalone or distributed environments. Hence the multi-agent system (MAS) is commonly defined as a loosely coupled network of software agents that interact to solve problems that are beyond the individual capacities or knowledge of each problem solver.

3.2.1 Advantages of a Multi-Agent Approach

- Distribute computational resources across a network of interconnected agents. Due to resource limitations, performance bottlenecks, and critical failures, traditional systems may fail some times. Since multi-agent systems are decentralized, they do not suffer a single point of failure problem.

- Support the interaction with the existing legacy systems. Building wrappers/interfaces will allow interaction with the legacy systems.
- Model real-world problems in terms of autonomous agents. This is proving to be a more natural way of simulating task allocation and team planning of human team works.
- provides solutions in situations where expertise is spatially and temporally distributed.
- Enhance overall system performance in computational efficiency, reliability, robustness, maintainability, responsiveness, flexibility, and reuse.

3.2.2 Resource Management Using Multi-Agent Systems

Resource management in the distributed systems is a critical requirement for uninterrupted service of good quality. In distributed systems, memory usage, CPU usage, network availability, and network traffic act as the critical factors affecting the performance. Hence, the researchers have been actively working on solving the problem of optimal resource management. To manage the resources in distributed systems, one of the famous concepts tried by most researchers is the multi-agent-based approach due to its own distributed nature. Irrespective of the distributed nature, these agents are working with a common goal to solve the problem at hand. In distributed systems, nodes can be geographically separated, and for each node, a set of agents monitoring different aspects of requirements were proposed by some researchers[14].

Similar to distributed computing, efficient energy management in distributed systems also face the challenges of the same nature. With the introduction of edge computing and distributed computing, microcontrollers, and microprocessors in a distributed network of devices face the problem of providing the optimal performance without hindering the battery life of each device. For example, in smart cities, there are sensor networks monitoring multiple parameters such as the concentrations of carbon monoxide and hazardous gasses due to vehicles and factories, and weather readings. Nodes in these sensor networks are collecting data regularly to provide continuous readings. Hence efficient communication is a critical role given that they are some times, especially at night, operating on battery power. Multi-agent based solutions for efficient

energy management in distributed networks of electronic devices have also been successfully experimented[13].

Similar to resource management, coordination among the different entities in a given system is a critical requirement. Coordination provides an error-free service leading to proper resource management. Previously, resource management was seen as an attempt to provide the minimum requirements for uninterrupted service. This can be extended beyond the computational requirements as well. For example, in power grids, coordination can help in-phase synchronization, which eventually reduces financial losses[15]. In cellular networks also, maximization of the network utilization and balancing cell usage similar to load balancing the distributed computing help in overall performance[16].

3.2.3 Key Features of Multi-Agent Systems in Resource Management

- Identify the critical resources and deploy independent agents to monitor resource availability from time to time. Define criteria to assess the severity of the resource shortage or over usage to trigger the actions to redirect the incoming requests[14],[16].
- Depending on the computational complexity and priority of the pending tasks, the agents were instructed to pick the next job/task. For the discussion making, algorithms like Highest response ratio, highest priority first, Deadline scheduling and etc. were introduced. These approaches have been successfully tested in resource-limited hardware systems like raspberry pi boards[17].
- In distributed computing, with multiple nodes capable of performing the same task, the multi-agent system within each node monitors their different types of resource availabilities and calculates the overall resource availability of each node to place a bid and win the next available task/job[13].

3.2.4 Success in Multi-Agent System in Resource Management

Multi-agent based approaches for resource management have been successfully tested in both resource-heavy environments like distributed computing as well as in resource-limited platforms like Raspberry pi boards. Further, these have proved that this approach can boost the performance significantly by sharing and coordinating the available resources at crisis times.

3.3 Artificial Neural Network

An artificial neural network (ANN) is an AI concept inspired by the biological neural network of the brain. The human brain learns different tasks through observation. Similarly, the ANN classifier learns from examples. Hence, this is known as a supervised learning algorithm. Researchers have tried ANN classifiers in audio recognition applications and succeeded in achieving better performances with feature vectors like Mel Frequency Cepstral Coefficients and Bark Frequency Cepstral Coefficients.

3.3.1 Artificial Neural Networks in Audio Recognition Applications

ANN is a widely used classifier in audio processing applications, including infant cry detection and classification. Typical applications of ANN related to audio processing are as follows.

- Human-machine interfacing is a hot research area to reduce the communication gap between humans and machines (humanoid robots). These robots can accept the commands from the natural languages through the auditory sensors if they can interpret the vocal command. For these applications, ANN is used to classify the vocal commands extracting Mel-Frequency Cepstral Coefficients as the feature vector[18], [19].
- Authentication systems based on the audio fingerprint is a famous application of ANN in this domain. In these applications, the main aim is to identify and authenticate the user. Hence, these applications target understanding the key features which discriminate the speaker's voice from the others[20], [21].

- Researchers have successfully experimented with ANN in audio-visual emotion recognition using both audio and visual inputs of a speaker. These are trained with features of the facial images and vocal outputs. Given that the people express the same emotion differently, it is hard to mathematically model a generic detector. With the ability to generalize and memorize, ANN classifiers are capable of building a general mathematical model to detect the emotions of the people[22].

3.3.2 Training and Prediction with Artificial Neural Networks

ANN classifiers are trained with labels data sets. To train the classifier, training, testing, and validation data sets are prepared with an almost equal number of examples of each class of outputs. The training data set is used to train the classifiers, while the testing-data set is used to evaluate the classifier against unseen examples in the training phase. The validation data set is used to evaluate the different architectures of the classifiers and select the best out of them. The overall accuracy of the classifier depends on the number of neurons, learning rate, training iterations, initial weights, activation function, and some other factors[23]. Hence, the training phase involves lots of effort to select the best classifier. Training the classifier can be mathematically described as the estimation of best weight vectors producing the best accuracy against test and validation data sets. Since the values of the initial weights are randomly initialized, training the same classifier with the same data set multiple times can end up in different accuracy figures. Hence, as rule of thumb, general practice is to train the classifier with the same configuration using the same training data set multiple times and select the best for each configuration. A significant number of researchers have chosen Mel-frequency cepstral coefficients as the feature vector since it is known as one of the best representations of the human auditory system.

3.4 K-Nearest Neighbour

K-Nearest neighbor is known to one of the best classifiers in pattern recognition. Audio recognition can also be viewed as a pattern recognition problem. Hence, researchers have often chosen KNN as a classifier in their respective applications. Since KNN does not involve a learning phase prior to the prediction, KNN is also recognized as a lazy algorithm. Further, since KNN does not make any assumption of the statistical distributions of the training data, this is known as a non-parametric classifier. Hence, this sometimes out-performs the other classifiers. Since the prediction involves the series of calculations of the similarity measure against each training example, the time complexity of the prediction of KNN grows exponentially against the training data set.

3.4.1 K-Nearest Neighbour Classifier in Audio Recognition Applications

- Auditory suspicious event detection is an interesting application where suspicious events like screaming, gunshots, explosions, police sirens are to be detected in real-time to trigger security alerts. MFCC can still be the main choice for the feature vector since it still is capable of modeling the human perception of these suspicious events[24].
- Speaker recognition is another application of the KNN classifiers in this domain. In KNN, to filter the K-number of nearest neighbors, the similarity measure plays a critical role. Fisher Kernel is one of the efficient mathematical models for the similarity measure[25].
- One research conducted on a non-trivial application presents that the guitar model can be identified by the audio processing with a KNN classifier. As the feature vectors, MFCC and LPCC have been chosen. In this research, multiple classifiers like Support Vector Machine (SVM), Bayesian Classifiers have been tried along with KNN. KNN has outperformed the other classifiers[26].

- In surveillance applications, researchers have successfully experimented with KNN for sound classification[27],[28].
- Nocturnal cough and snore detection is a medical diagnosis to follow the progress of respiratory diseases and quality of sleep. Hence, this requires to process the audio feed continuously and process in real-time with a KNN classifier[29].

3.4.2 Prediction with K-Nearest Neighbor Classifier

The performance of the KNN classifier depends on the mathematical equation for the similarity measure and the number of neighbors of interest (K). As the similarity measure, typically, Euclidean distance and Manhattan distance are chosen. Further, the fisher kernel is also another choice associated with a probabilistic approach for similarity measure. The choice of K is normally decided after multiple trials of different K values.

3.5 Summary

This chapter discussed the technologies intended to experiment in the proposed solution. Hence, the Multi-agent system was discussed within the scope of resource management. Next, Artificial neural network and K-Nearest Neighbor classifiers were discussed with the conclusions drawn from the literature review to understand the technologies behind these classifiers.

4 HYPOTHESIS

4.1 Introduction

Chapter 3 discussed the technologies and theories helpful in constructing the proposed solution. With the basic understanding of these concepts, Chapter 4 starts the discussion of the proposed solution. This chapter presents the hypothesis and defines the input, output, and process. Proper understanding of the inputs and outputs gives a better understanding of the process. Finally, this discusses the users of the solution to conclude the discussion.

4.2 Hypothesis

Stand-alone hardware design can resolve the problems of user privacy issues, recurrent cost of the servers, and excess use of the user's bandwidth. A firmware design based on Multi-Agent Technology can support continuous audio processing to detect and classify cry-events in real-time. ANN and KNN can provide the required intelligence and performance in the resource-limited hardware environment for real-time prediction.

4.3 Input

The input to the system is a continuous audio signal. This audio stream may contain

- Infant cries
- Infant laughs
- Adult voices (parents speaking to babies)
- Lullabies/white noise
- Silent environment

4.4 Output

The output of the system is a message with the information

- Whether the baby is crying
- If the baby is crying, what is the most likely cause

4.5 Process

This records noise present in the vicinity of the device and produces a continuous audio stream for analysis. The audio stream is preprocessed to remove noise. Given the higher probability of observing a quiet room, the signal will be filtered to select the human voices. The intention of discarding the irrelevant information of the signal as much as possible in the early stages is to release the hardware without being over-utilized. If the human voice is detected, classifiers will evaluate whether it is a crying event. If not, the classifier will discard these signal frames. If it is a cry, the second classifier will analyze the relevant features of the signal frame and decide the cause.

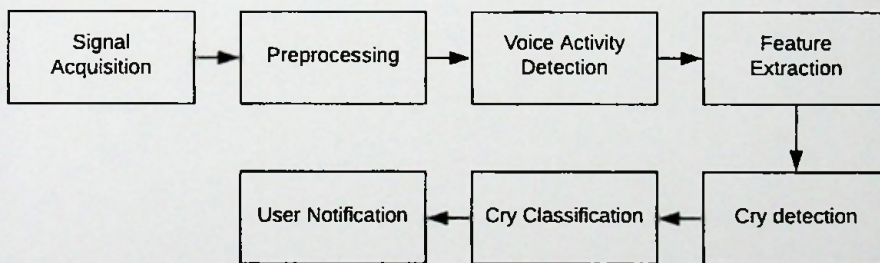


Figure 4.1 Flow diagram of the process

4.6 Users

The target users of the proposed solution are the parents and caregivers. This can help pediatricians and the nurses to detect the hearing impairments in medical fields as a screening device if the classifiers were trained with an appropriate data set.

4.7 Summary

Chapter 4 discussed the hypothesis of the proposed solution. The hypothesis was formulated based on MAS concepts, ANN, KNN classifiers, and digital signal processing techniques for continuous audio processing. Next, the inputs and outputs of the system were defined. With the help of identified inputs and outputs, the process was defined. Finally, the users of the proposed solution were identified and highlighted that there is a possibility of using this as a clinical device for the initial screening of the babies with hearing impairments if the classifiers were trained with an appropriate data set. In the next chapter, the design of the proposed design will be discussed in detail.

5 HARDWARE AND FIRMWARE DESIGN

5.1 Introduction

This chapter explains the design of the proposed solution in detail. In the design of a hardware solution, designers should focus on two critical inter-dependent aspects called hardware design and firmware design. In the process of designing a complete hardware solution, the cooperation between hardware and firmware plays a critical role. Similarly, the research problem can be divided into two inter-connected subdomains called hardware design and firmware design. Hardware design covers the scope of identification, selection, and integration of the required hardware features. In this chapter, hardware design will be elaborated in detail, covering the architecture and the expected functionalities of each module. Next, this chapter will discuss the firmware design. Firmware designs should meet both the functional and non-functional requirements. As the functional requirements, firmware design covers the scope of controlling peripherals, data flow, and algorithm execution to realize the given features. In addition to the execution of above-designated functionalities, firmware should meet the non-functional requirements like stability, scalability, robustness, and optimality. This chapter will further explain the use of Multi-agent technology as the foundation to realize both functional and non-functional requirements.

5.2 Hardware Design

Figure 5.1 shows the architecture of the hardware system. Dashed lines represent the electrical connectivity, while the solid lines represent the communication between different modules. Generally, hardware systems are designed with either microcontroller or microprocessor, depending on the computational complexity associated with the functionalities. Given the price comparisons, microcontrollers are preferred over the microprocessors. But, due to the higher computational complexity and the memory

requirements in audio processing, a microprocessor with approximately 1GHz clock would be required.

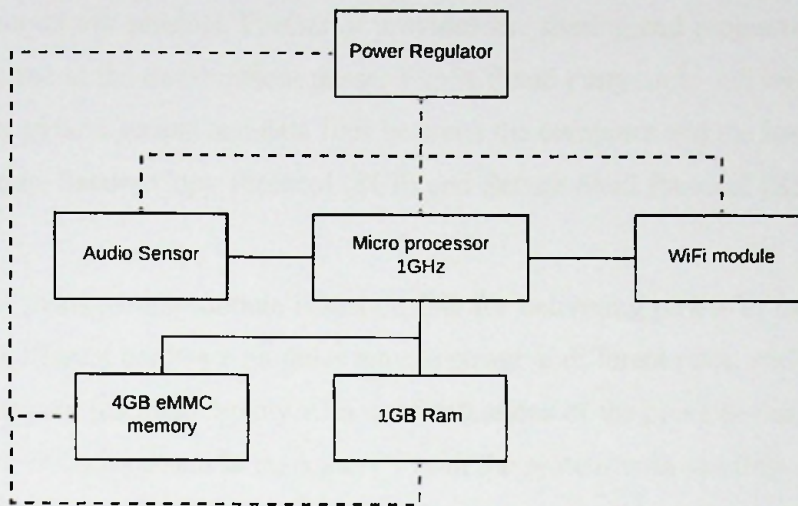


Figure 5.1 Hardware architecture

Microprocessors are not designed with in-built memory or Random-Access Memory (RAM), allowing the designers to integrate them externally according to the required specifications. Hence, with the above microprocessor, 4GB flash memory will be integrated as the system memory. This is a non-volatile memory to store the operating system and the firmware. Similarly, an external RAM of 1GB or more will be integrated into the microprocessor. Since the selected microprocessor executes an Operating System (OS) and an audio processing application, a minimum of 1 Gigabyte (GB) RAM is mandatory. Embedded Linux is one of the popular operating systems (OS) in the embedded development environments due to its stability and performance. Hence, in this application Embedded Linux operating system is selected. As the programming language, in Embedded Linux environments, python is popular due to its execution speed, easy implementation/fast prototyping, and the availability of digital signal processing (DSP) libraries.

Additionally, the design includes an audio sensor to record the audio signal. The audio sensor samples the audio signal at a predefined sample rate and transmits the audio

samples to the microprocessor via the dedicated communication channel. As a cry detector, parents should be notified whenever the system detects a crying event. Hence, a Wi-Fi module is used to connect to the home router and to establish a connection with the server via ethernet. Further, it provides file sharing and programming interfaces required in the development phase. WinSCP and Putty tools will be used to exchange the python scripts and data files between the computer and the hardware system. For this, Secure Copy Protocol (SCP) and Secure Shell Protocol (SSH) will be useful.

The power management module is responsible for delivering power to the hardware modules. Different hardware modules require power at different rates, and instantaneous current may fluctuate rapidly with the fluctuations of the processor usage. Hence proper power management is mandatory to run the system with stability. The power management module will receive the power from an external 5volt source and feed the power to the internal modules while protecting them from voltage surges, fluctuations, etc.

5.3 Firmware Design

In hardware designs, firmware inside the micro-controllers and microprocessors are designed considering both functional and non-functional requirements. In this application, as functional requirements, firmware should deliver features such as real-time audio processing, cry detection, cry classification, and user notifications. Therefore, it is imperative to manage the limited resources efficiently to achieve the above computationally heavy features. Similarly, considering the non-functional features such as the ease of updating/upgrading, scalability, and robustness, the firmware will be built based on the concept of Multi-Agent System (MAS).

5.3.1 Multi-Agent System as the Foundation

Since the hardware design involves multiple modules with limited resources, it is mandatory to control each module efficiently. Since firmware designs based on Multi-Agent Systems can improve the performance by distributing the agents across the

system, Multi-Agent System is selected as the foundation of the firmware design. Hence, distributing agents across hardware modules to achieve common goals with efficient communication can solve the problem of resource management. Similarly, firmware designs typically end up in being very complicated due to a large number of independent features. Creating a new cluster of agents to implement each independent feature provides the ease of scalability and maintainability. Since new agents can easily be added to the existing multi-agent system with minimal effort, the scalability of the existing solution is relatively effortless. Further, solving complicated problems demands to break down the problem into sub-problems. This is similar to breaking down the problem and solving the subproblems by a team of human experts sharing their knowledge with each other. Hence, the concepts of Multi-Agent System provide the foundation for firmware design to achieve a list of complicated features by breaking them down to a list of small and relatively simple problems.

Within the scope of this project, the importance of the following agents is identified for the indicated tasks.

- Sensor Agent:
 - responsible for controlling the sensor and recording the audio input
- Preprocessor Agent:
 - responsible for preprocessing the audio signal for noise filtering.
- Classifier Agent:
 - responsible for feature extraction, cry detection, and classification.
- Auditor Agent:
 - responsible for evaluating parameters and sending notifications to the user.
 - When multiple parameters are simultaneously monitored, this agent should have the overall knowledge about all the parameters.
- Network Agent:
 - responsible for communicating with the server to update estimated parameters and to send notifications when required
- Manager Agent:
 - responsible for the life cycle management of the above agents.

Due to the unavailability of stable, light-weight, and efficient Multi-Agent platforms for the embedded environments, a simple version of Multi-Agent System would be implemented with the help of Processors and Queues in multiprocessor package in Python. Within the scope of this project, implementation of a fully-fledged Multi-Agent System was not the main focus. Hence, the following implementation covers only the fundamentals of a Multi-Agent System required for the given application. Therefore, the above-identified manager agent plays a critical role in the life-cycle management of the other agents.

5.3.2 Firmware Design Architecture

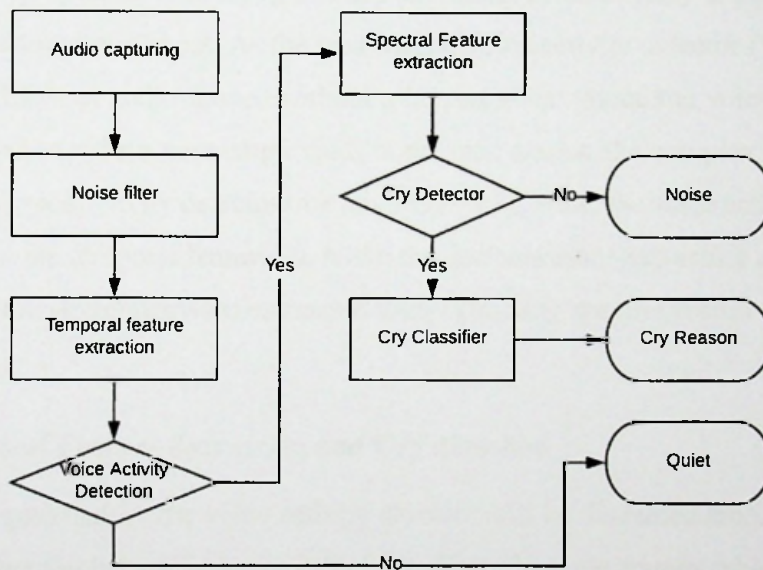


Figure 5.2 Firmware design

Figure 5.2 presents the firmware design of the proposed solution to detect and classify infant cry-events.

5.4 Audio capturing and Noise filtering

The first step of audio capturing includes the configuration of the audio sensor and continuous recording of the audio signal. Next, the recorded audio signal is passed



through the noise filters to remove the noise present in the audio sensor due to the inherent sensor noise and the ambient noise. Since the audio signal is recorded continuously, the audio signal should be processed at the rate of production at least to maintain the equilibrium and to notify when the infant cries. The audio signal will be sampled at 8kHz, and for audio analysis, the signal will be treated as a sequence of blocks of audio samples.

5.5 Temporal Feature Extraction and Voice Activity Detection

Typically, the infants' rooms are either quiet or filled with white noise to help the baby fall asleep. Hence, it is infrequent to encounter a human voice in the infants' rooms. Therefore, the systematic filtering of irrelevant information as early as possible would lead to a performance boost. As the next step, a voice activity detector (VAD) is employed to filter out audio frames without a human voice. Since the voice activity detector is used to reduce the complexities in the later stages, the complex implementations of the voice activity detectors are not preferred. Hence, the voice activity detector will analyze the temporal features to make the decision since extracting temporal features are more straightforward compared with extracting spectral features.

5.6 Spectral Feature Extraction and Cry detection

Frames disqualified by the voice activity detector will be discarded since they cannot contain either the human voice or infant cry. Therefore, the frames which were only qualified under the criteria defined in the voice activity detector will progress forward. Frames passed through the voice activity detector will be used for feature extraction. As the feature vector, Mel-Frequency Cepstral Coefficients (MFCC) will be calculated for each frame. These MFCC feature vectors would be used to train the classifier in the cry detector. Only the frames with the human voice and infant cries are passed through the cry detector for training and then prediction. The cry detector is a classifier based on either the Artificial Neural Networks (ANN) or the K-Nearest Neighbor algorithm (KNN). Based on the accuracy and computational complexity, the final decision of the exact classifier will be made.

Although the audio signal is analyzed as a sequence of frames of few milliseconds, cry-events will last more than a few milliseconds. Hence, non-overlapping time-windows of a few seconds (approximately 5 seconds) are analyzed to identify cry and noise events. The width of the audio event should be reasonably large to encapsulate an infant cry-event. Similarly, it should not be unnecessarily large to introduce a delay in event detection. The exact period of the audio event will be determined experimentally at the implementation. If the majority of the frames within the given period are classified as cry frames, the system will identify the event as a "Cry" event and will try to figure out the most probable cause of cry by using the cry classifier in the next phase. Similarly, if the majority of the frames represent the noise frames, the system would declare the event as a "Noise" event and avoid further processing.

5.7 Cry Classifier

Cry classifier holds the responsibility of identifying the most likely cause of the infant cry. This classifier is required only upon the detection of a Cry-event. When the cry detector detects a Cry-event, the cry detector filters the frames with the label of Cry and passes them to the next stage. Frames labeled as Noise will not proceed to the cry classifier. As the cry classifier, classifiers based on ANN or KNN will be trained. Based on the accuracy and computational complexity, the final classifier will be selected experimentally at the implementation. To train the classifier, as the labels of the inputs, the most frequent causes of infant cries were identified as Belly Pain, Hunger, and Tiredness.

5.8 Summary

In this chapter, the design of the solution was discussed in detail. The original problem was broken down into two sub-problems called hardware design and firmware design. Hardware design covers the scope of identification, selection, and integration of the required hardware modules. In the selection of hardware, the cost of the hardware components plays a critical role. Further, hardware design architecture was discussed in detail with the importance of each module. Next, the firmware design of the

proposed solution was analyzed. An approach based on the multi-agent system was proposed as the foundation of the firmware solution to achieve the functional requirements like audio recording, audio processing, cry classification, and non-functional requirements like scalability and robustness. To achieve the functional requirements, this introduced the use of a voice activity detector to filter audio frames with human voice only for further processing. As the next step, the cry detector was introduced to detect cry frames and cry-events with the use of Mel Frequency Cepstral Coefficients (MFCC) as the feature vector. As the final step, upon detection of a cry-event, the cry classifier would come into effect to identify the most likely reason behind each cry-event. The types of classifiers used as the cry detector and cry classifier will be experimentally decided based on the accuracy and time complexities. Primary candidates for the above classifiers are ANN and KNN. This design is expected to reduce the use of computationally heavy classifiers to increase performance by systematically filtering the data consumed by each classifier

6 IMPLEMENTATION OF CRY CLASSIFIER

6.1 Introduction

The previous chapter discussed the design of the proposed solution. This chapter discusses the implementation of the proposed solution. This chapter breaks down the implementation into two sections for hardware design and firmware design. Implementation of the hardware design covers the scope of selection and integration of the hardware modules. The implementation of the firmware design covers the implementation of the multi-agent system and the algorithms to support real-time cry detection and classification. In each phase of the implementation, there are unique challenges to be dealt with, to reach the final goal. Hardware design faces the challenge of selecting the best hardware configuration to meet the resource requirements for audio processing while minimizing the manufacturing cost. Due to the limitation of resources and the requirement of real-time audio processing, the firmware should be designed to utilize the hardware resources efficiently. Under firmware design, the author explains the use of the concepts of Multi-Agent Systems (MAS) to achieve these goals. Next, the author will walk the readers through the algorithms responsible for noise filtering, voice activity detection, cry detection, and cry classification and their implementations. Generally, the choice of classifiers in machine learning depends on the accuracy figures, and rarely computational complexity also matters.

6.2 Hardware Selection

The scope of the hardware design of the proposed solution covers the steps from audio signal acquisition to cry classification and notifying the caregivers. Since audio processing requires a significant amount of processing power and memory, selecting a microprocessor is subjected to constraints of both performance and price. Typically, a microprocessor with at least 1GHz clock with 1GB RAM is required for audio processing applications. OSD3358 microprocessor (Beagle-Bone Board) was chosen to

satisfy the requirements of both price and performance. Compared to the other microprocessors within the same range of performance and price, this has two additional microcontrollers(200MHz) called Programmable Real-Time Units (PRUs) in the same System on Chip (SoC). The microprocessor can delegate selected tasks to PRUs to reduce the workload.

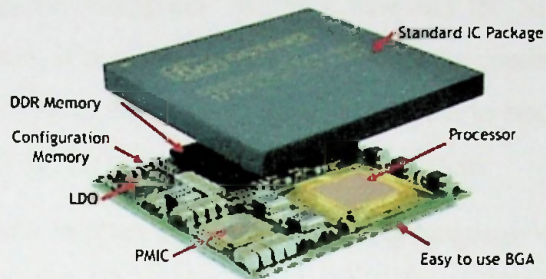


Figure 6.1 OSD3358 microprocessor

MP34DT01 microphone was integrated into one of the Programmable Realtime Units (PRUs) to record the audio signal transmitted through the Pulse Density Modulation (PDM) protocol. The Signal to Noise Ratio(SNR), which is the standard measure of the noise in the signal is 63dB. To communicate with the server, it is mandatory to maintain an ethernet connection. Wi-Fi connectivity is the best choice since it provides portability compared to wired connections. WL18MODGB Wi-Fi module was selected to communicate with the server through Ethernet. Figure 6.2 shows the integration of the hardware modules selected above.

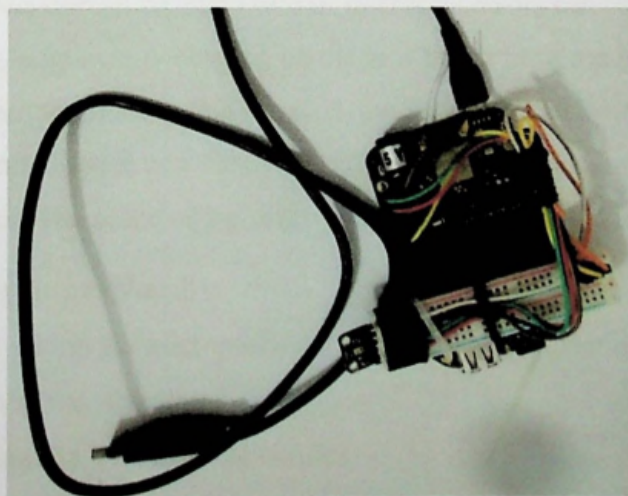


Figure 6.2 Hardware prototype

6.2.1 Hardware Setup

The design of the hardware system requires to set up the environment to program the hardware. The first step to program the microprocessor is to install the operating system. Linux operating system was installed using an SD (Secure Digital) card. For programming purposes, WinSCP software was installed to share the files between the hardware system and the Computer. WinSCP supports the Secure Copy Protocol (SCP) to share the files. If not, one may choose to use the command prompt to share the files using SCP. Further, Putty software was installed to issue commands via the Secure Shell protocol (SSH) to control the hardware system.

Python version 2.7 was selected as the programming language since it was the latest stable version that supports all the required libraries. To support the computations, the following python libraries were also installed.

- NumPy for mathematical computations
- SciPy for mathematical computations and machine learning algorithms
- Scikit-learn for KNN classifier.
- Python Speech Features for audio feature extraction.

6.3 Firmware Design

Firmware is commonly known as the software which controls the hardware devices to achieve desired functionalities. Firmware of microprocessors can be written in different programming languages depending on the processing power at hand and the complexity of the functionalities. In this research, the firmware was written in Python due to its better execution speed and the availability of audio processing libraries for embedded electronics. The scope of the firmware design includes the following features.

- Control of the peripheral
- Data flow within the microprocessor and with the peripherals
- Execution of the algorithms
- Communication with external entities

Since the anomalies should be notified in real-time, the design of the firmware plays a critical role in realizing continuous real-time audio processing in the resource-limited hardware design. Hence, the firmware was designed based on a multi-agent system to avoid resource blocking and over-utilization. Further, this gives the flexibility of upgrading and expanding system functionalities with ease since the agents are loosely coupled with each other. This allows adding more agents to build a complex system without affecting the existing functionalities, given that the system has the required resource. The current design of the firmware, which covers the scope of cry detection and classification only, includes the following agents.

6.3.1 Sensor Agent

The sensor agent resides in the programmable real-time unit (PRU) of the microprocessor. This is responsible for configuring the microphone and recording the audio signal continuously. This agent controls the relevant General-Purpose Inputs and Outputs (GPIO's) to configure and read the audio signal produced by the sensor.

6.3.1.1 Signal Acquisition

The signal acquisition includes the steps of sensor configuration and continuous recording of the audio signal. The sensor communicates with the PRU through a two-wire communication protocol called Pulse Density Modulation (PDM). The two wires or the channels include the input clock signal and the output (Data out). The output data rate of the sensor depends on the clock signal provided by the PRU. The sensor generates an output pulse at the falling edge of the input clock signal. PRU was configured to feed a 1MHz clock to the sensor. This produces an output PDM signal of 1 megabit, which will later be converted to a 64kHz audio signal with a 16bit sample width. As the name suggests, the density of the pulses represents the amplitude of the signal. A frame of a higher density of 1's represents a large positive value, while 0's represents a large negative value. Similarly, a frame with an even distribution of 0's and 1's indicates an output close to zero. This is quite similar to the biological

phenomena where the number of neurons firing together is proportionate to the intensity of the sensory input.

Figure 6.3 represents the PDM signal corresponding to a sinusoidal signal. It is noticeable that at the peak of the sinusoidal, the density of 1's has reached its maximum while at the minimum of the sinusoidal, the density of 0's has reached its maximum. Demodulation of the PDM signals is achieved by averaging the signal. Since the low pass filters have the same averaging effect on the high-frequency signals, low-pass filters could be used to demodulate the Pulse Density Modulated signal.

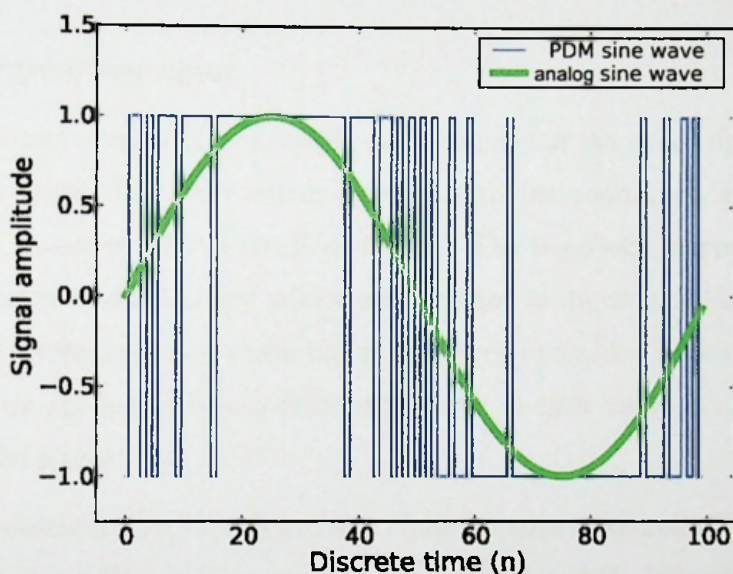


Figure 6.3 Pulse density modulated sinusoid

Hence, the sensor agent filters the input bitstream through a low-pass filter to demodulate the audio signal. Since the signal was demodulated by the PRU and the rest of the audio processing is supposed to occur in the microprocessor, the demodulated signal should be transmitted to the microprocessor.

The PRU is designed to communicate with the microprocessor through the hardware interrupts and the shared memory. Given the nature of the signal, shared memory is used for this purpose. 128kB of the shared memory is allocated for the data exchange purposes. The allocated space is sufficient for holding 1024 milliseconds of data. The allocated block was subdivided into 2 equal spaces of 64kB. The PRU writes to one of

the blocks while the microprocessor reads from the other block. Hence, the PRU and the microprocessor access these blocks alternatively. It is mandatory to synchronize the read/write operations to avoid data corruption due to concurrent access to shared memory. Hence, the PRU interrupts the microprocessor as soon as it finishes writing the data to acknowledge the presence of a new data block. This gives a 512-millisecond time window for the microprocessor to read the data before the PRU overwrites them. Each block carries 32,768 samples of 2 bytes. Access to the shared memory by the microprocessor is controlled by the preprocessor agent since it has the first use of the received audio signal.

6.3.2 Preprocessor Agent

The preprocessor agent is responsible preprocessing of the audio signal captured by the sensor agent. The preprocessor agent accepts the continuous audio signal as a stream of frames of 32,768 samples (512ms). The frequency response of the audio sensor introduced a frequency selective attenuation to the recorded audio signal. The different frequencies of the audio signal have been attenuated differently. Hence, the preprocessor agent applied a compensation filter to each frame to reverse the above effect of the sensor.

The compensation filter is designed as a Finite Impulse Response (FIR) filter. Hence, the output of the filter only depends on a few samples of the immediate past. Unlike in the Infinite Impulse Response (IIR) filters, the current output of the above filter does not depend on the previous outputs. Figure 6.4 shows the response of the filter against the normalized frequency of the input signal. Hence, according to figure 6.4, higher frequencies close to the sampling frequency will be significantly attenuated compared to the lower frequencies.



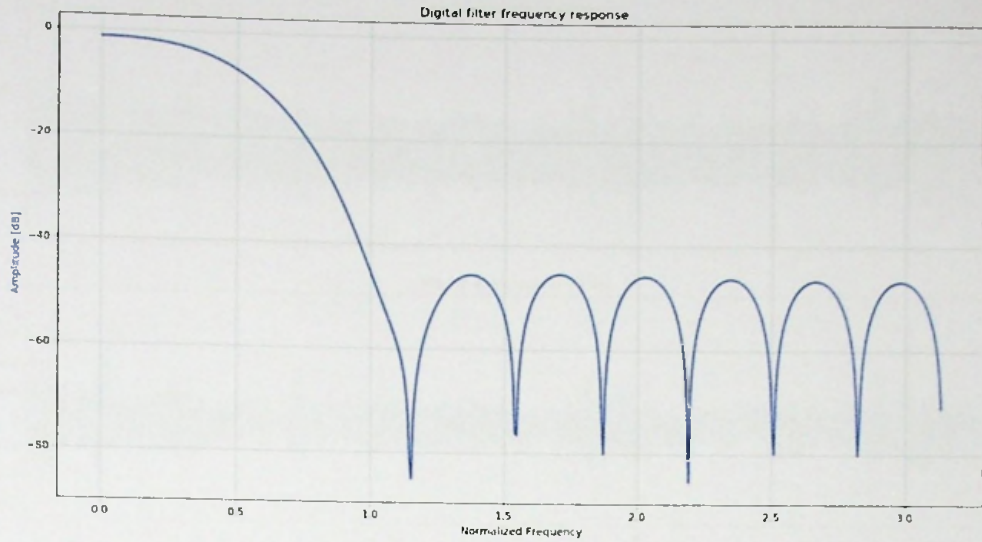


Figure 6.4 Frequency response of the compensation filter

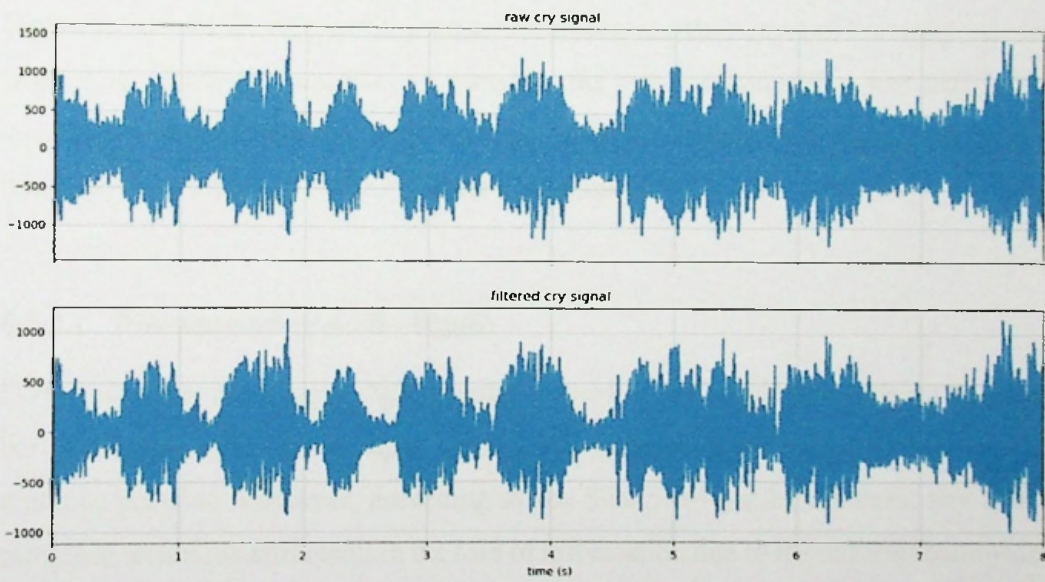


Figure 6.5 Raw cry signal vs. filtered cry signal

Figure 6.5 shows the output of the sensor, which is subjected to frequency selective attenuation and the compensation filter output. The above-recorded signal contains a cry signal.

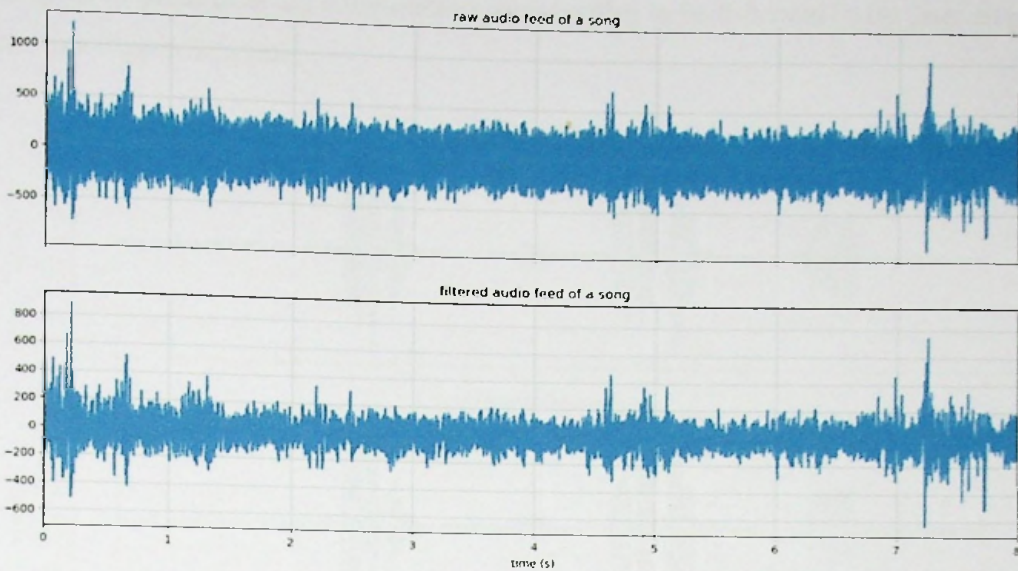


Figure 6.6 Song filtered by compensation filter

Figure 6.6 shows the output of the sensor for a song and the output of the compensation filter. The quality of the audio generated by the compensation filter was verified by comparing the input and output of the compensation filter with the actual audio signal manually. Refer the appendix A for the source codes.

6.3.2.1 Downsampling Audio Signal

Down-sampling is a unique digital signal processing technique that reduces the number of samples of the given input signal while preserving the information content as much as possible. However, according to the Shannon-Nyquist theorem, any down-sampling technique will result in the loss of information due to the reduced bandwidth. But as the researchers suggest, the minimum bandwidth requirement for infant cry detection is 500Hz. Hence, according to the Shannon-Nyquist theorem, the minimum required sampling rate is 1kHz capture the 500Hz bandwidth. Generally, reducing the bandwidth unnecessarily may result in loss of information, which helps to identify the speaker uniquely. In this particular application, this may result in loss of information that would support the cry classification. Hence, as the standard practice suggests, the down-sampling frequency was selected as 8kHz. Hence, 64kHz signal was down-

sampled to 8kHz to avoid computation complexities to be followed in the later stages due to a higher data rate.

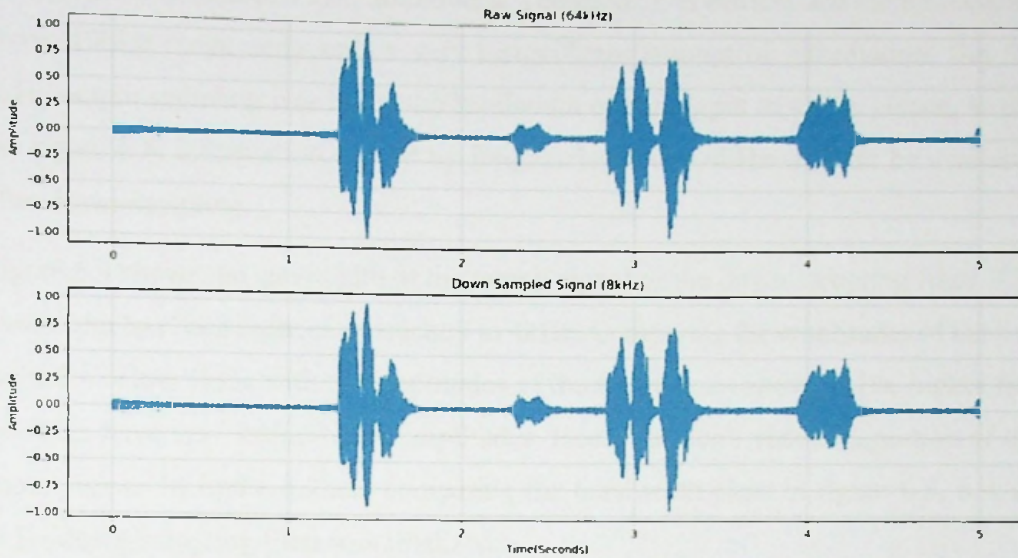


Figure 6.7 Comparison of Raw signal vs. Down-sampled signal

Figure 6.7 shows the input and output of the down-sampling algorithm. From the human eye, it is nearly impossible to identify any distortion that occurred to the signal due to the down-sampling. Hence, bandwidths of the input and output audio signals were analyzed to estimate the distortion mathematically. The Fast Fourier Transform (FFT) of the signals was calculated to estimate the bandwidth of each signal.

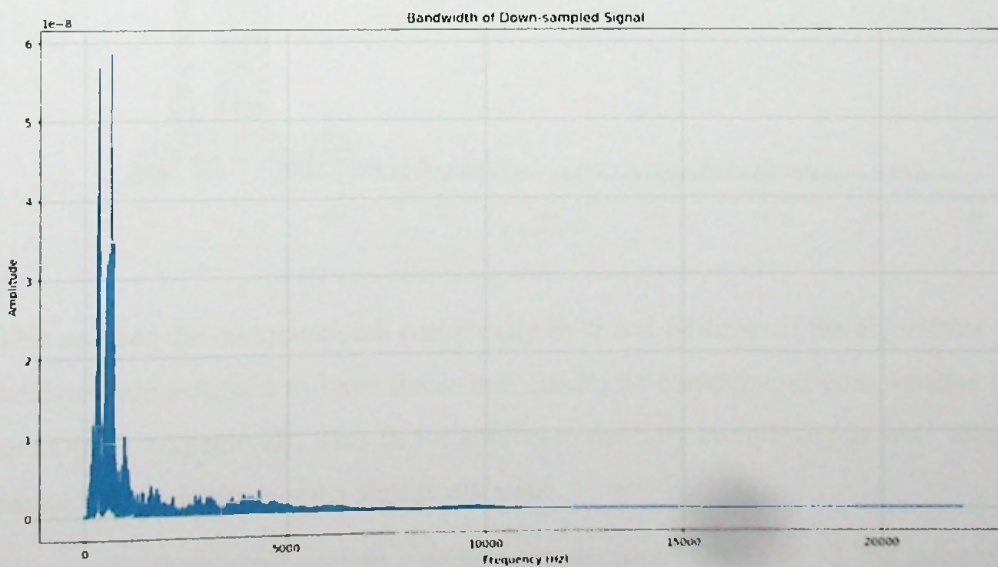


Figure 6.8 Bandwidth of raw signal

Figure 6.8 shows the bandwidth of the input signal to the down-sampling filter. The horizontal axis of the graph represents the constituent frequencies of the signal, while the vertical axis indicates their amplitudes. Therefore, it is evident that the frequencies above 10kHz range carry only a very insignificant amount of information. But the 8kHz output sampling rate limits the bandwidth of the output to 4kHz. Hence, as per the figure 6.8, information carried by frequencies above 4kHz will not be available after down-sampling.

Figure 6.9 shows the bandwidth of the output signal of the down-sampling filter. The bandwidth has been reduced drastically to 4kHz. Comparing the amplitudes of the frequencies below 1kHz with the amplitudes of the frequencies above 4kHz, higher frequencies have very insignificant amplitudes. Hence, through visual inspection of the audio signals in figure 6.7 and comparing the bandwidth plots in figure 6.8, 6.9, an 8kHz down-sampling filter was finalized.

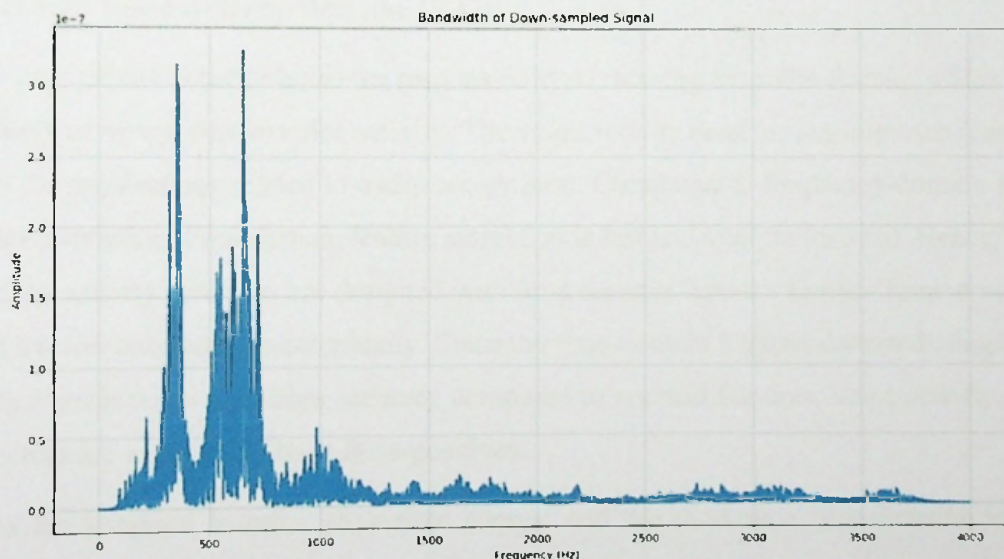


Figure 6.9 Bandwidth of the down-sampled signal

This reduces the computational complexity by 8 and 64 times if the algorithms to be followed are assumed to have linear and quadratic computation complexities ($O(n)$ and $O(n^2)$), respectively. This factor applies to memory complexity as well, although memory complexity is not a significant issue.

The preprocessor agent accepts a continuous stream of blocks of 32,768 samples and applies the compensation filter before down-sampling them to 8kHz. Hence the output

of the agent is another stream of blocks of 4096 samples, which corresponds to 512 milliseconds at a sampling rate of 8kHz. The output is fed to the classifier agent since, as the next few steps, a series of algorithms are applied to the signal by the classifier agent to detect infant cries.

6.3.3 Classifier Agent

The classifier agent is responsible for audio processing to detect and classify the infant cries. This covers the entire scope of temporal feature extraction, voice activity detection, spectral feature extraction, cry detection, and classification, respectively. The classifier agent receives the audio frames of 4096 samples from the preprocessor agent and processes them in the order in which it receives.

6.3.3.1 Voice Activity Detector (VAD)

Voice activity detector holds the responsibility of filtering the audio frames, which are likely to have a human voice activity. The voice activity detector is a common feature in the applications related to audio recognition. Compared to frequency-domain feature extraction, time-domain feature extraction is fast and straight forward. Hence, the voice activity detectors are designed with time-domain features to filter human voice at a lower computation complexity. Since the time-domain features cannot distinguish the human voice with high accuracy compared to spectral features, voice activity detectors are expected to have false-positives.

As the temporal features, short-time energy, and the short-time zero-crossing were identified to be effective for human voice detection.

6.3.3.2 Short-Time Energy

Short-time energy (STE) is defined as the average sample energy of the signal in a short time period. Mathematically STE can be defined as.

$$E(n) = \frac{1}{N} \sum_{i=0}^{N-1} [w(m)x(n-m)]^2$$

where $w(m)$ represents coefficients of a windowing function (Hamming window) of length N to minimize the maximum side-lobes in the power spectral density (PSD) estimation. Due to the truncation of the continuous audio signal into frames for audio analysis, the signal has abrupt changes at both ends of the frame. Mathematically, sudden changes in the signal appear due to the presence of high-frequency components. Hence, distortion occurred due to the truncation of the signal to acquire a signal frame, introduces high frequencies that were not available in the original signal. The use of the hamming window reduces the abrupt changes in the signal due to truncation.

Figure 6.10 shows the variation of the short-time energy estimated for an infant cry signal. STE has a significant variation across the cry signal highlighting the pauses and cry instances. In general, human voice contains higher STE compared to white noise signals and unvoiced signals. Hence, STE has been widely used in speech detectors as a parameter to identify the human voice.

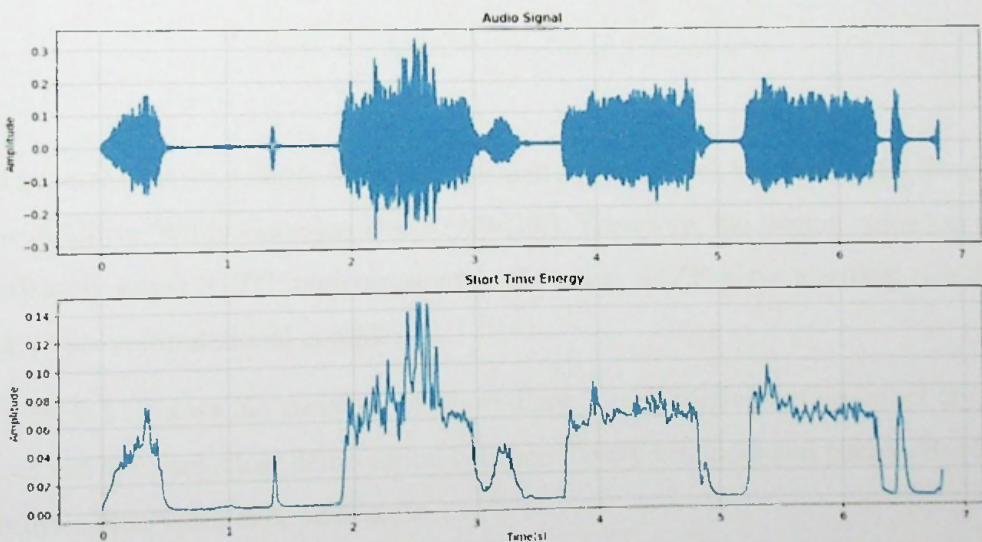


Figure 6.10 Short Time Energy of cry signal

6.3.3.3 Short-Time Zero-Crossings

Short-time zero-crossings (STZC) is defined as the rate of change of sign over a short period of time. Mathematically STZC is defined as:

$$Z(n) = \frac{1}{2N} \sum_{m=0}^{N-1} |\text{sign}(x(n-m)) - \text{sign}(x(n-m-1))|$$

Where

$$\text{sign}(x(m)) = \begin{cases} 1 & x(m) \geq 0 \\ -1 & x(m) < 0 \end{cases}$$

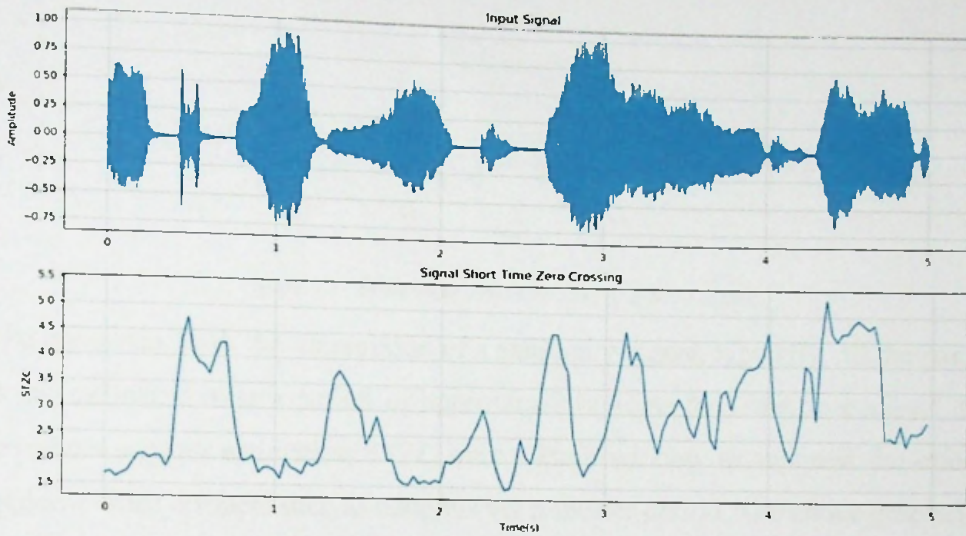


Figure 6.11 Short-Time Zero-Crossings of a cry signal

In general, Human voice is distributed around 200Hz, which is significantly lower than the Additive White Gaussian Noise (AWGN). Therefore, the human voice has a significantly lower STZC rate compared to the noise. STZC plays a critical role due to the lesser computational complexity.

Figure 6.11 shows the variation of Short-Time Zero-Crossings of an infant cry. This does not have any clear demarcation between the cry instances and pauses. But STZC shows a clear difference with noise.

Figure 6.12 shows the variation of Short-Time Zero-Crossing of a white noise signal. After comparing the two figures in 6.11 and 6.12, it can be concluded that Short-Time Zero Crossing has the ability to distinguish audio clips with the human voices.

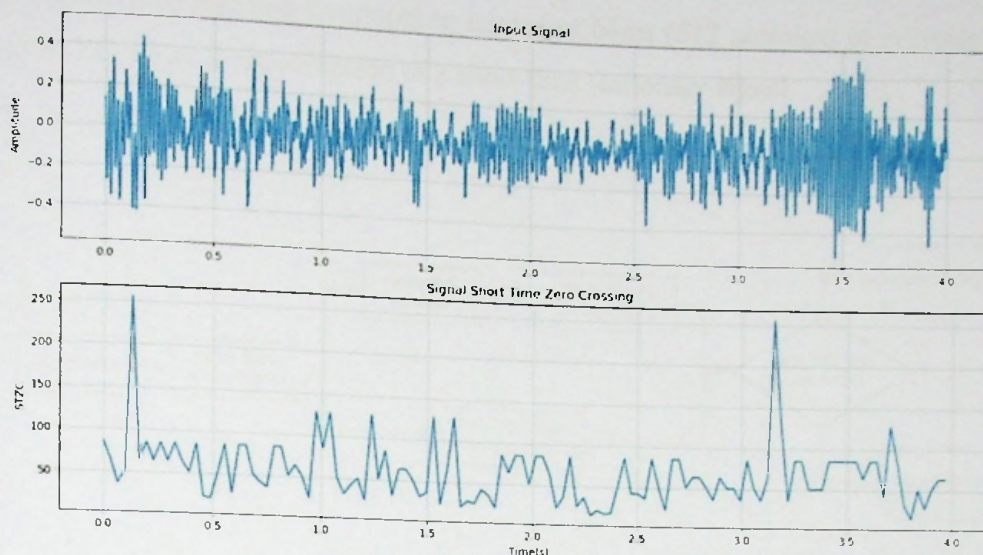


Figure 6.12 Short-Time Zero-Crossing of a noise signal

To be consistent with the assumption of a stationary signal, typically, all the parameters are estimated over a period of approximately 16ms to 32ms. In contrast, some researchers suggest estimating STZC over a large window to suppress the effect of impulsive voice artifacts such as coughs over a shorter period. The above graphs illustrate the variations of the respective parameters over the frames of 64ms.

6.3.3.4 Voice Activity Detection

For each block of 4096 samples (512ms) received by the classifier agent, the VAD algorithm calculates Short-Time Energy (STE) and Short-Time Zero-Crossings (STZC) to predict the presence of the human voice. The threshold for STE was experimentally determined to filter signals with the minimum required energy level. Human voice generally contains higher short-time energy compared to white noise and unvoiced audio. But voice activity detection based on STE only may introduce false positives since ambient noise may also contain frames with higher energy. Hence, Short-Time Zero-Crossings was also coupled with STE. STZC is low in human voice since the human voice has a lower bandwidth compared to white noise. Hence, the threshold for STZC was experimentally determined such that audio signals below the threshold would contain human voice. Blocks of 4096 samples sent by the preprocessor agent

are sub-divided into non-overlapping frames of 64ms (512 samples) to estimate STE and STZC under the assumption of a short-time stationary signal.

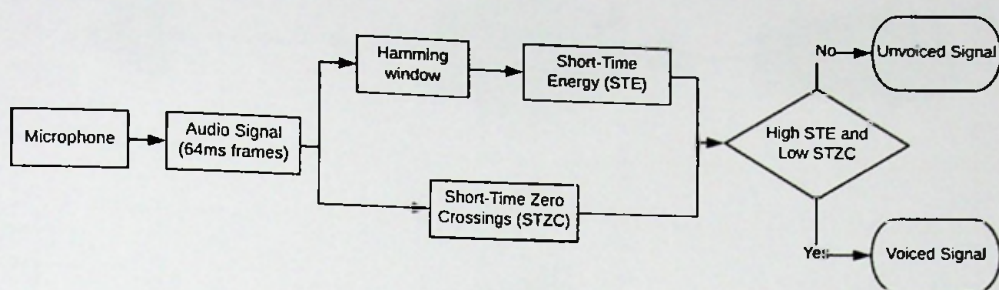


Figure 6.13 Block diagram of the Voice Activity detector

Figure 6.13 shows the algorithm of the voice activity detector. Each 64ms frame that does not comply with the above conditions will not proceed forward for the feature extraction. They will be labeled as ‘Quiet’ frames. The frames which qualify under the above conditions only will proceed forward to the feature extraction phase of the cry detection. Although the frames labeled as “Quiet” are discarded, a number of such frames detected in the immediate past will be memorized to include them in the final vote to decide the output.

Figure 6.14 compares the input and output of the voice activity detector. VAD has discarded some frames which are unlikely to have any human voice. Similarly, for better visualization purposes, disqualified frames have been replaced with empty frames. From the above graphs, it is evident that a significant percentage of the frames have been discarded even within a cry signal covering the pauses and idle spikes.

6.3.3.5 Feature Extraction

As the spectral feature vector, MFCC was chosen based on its success highlighted in the literature review. The frames filtered by the VAD only will reach to the spectral

feature extraction phase. Feature extraction is also based on the assumption that the input signal is a short-time stationary signal.

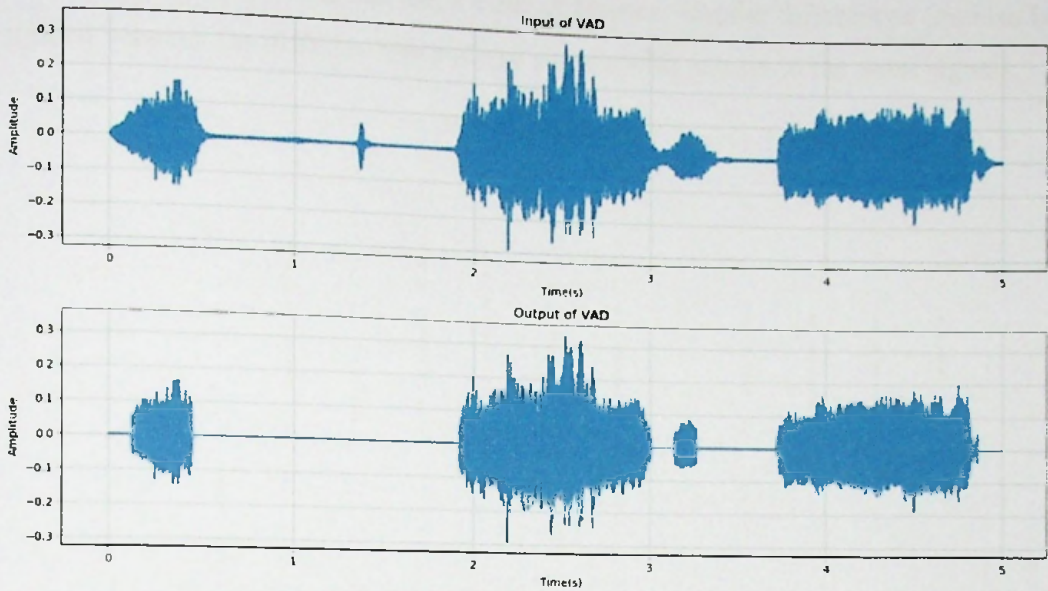


Figure 6.14 Input and output of the Voice Activity Detector

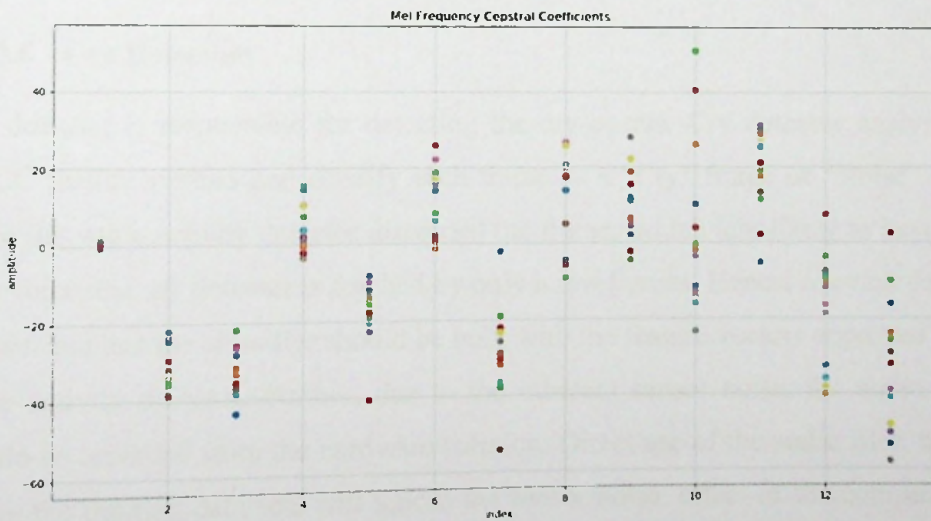


Figure 6.15 MFCC vector of the cry signal

Hence, each frame of 64ms will also become the input for the spectral feature extraction. But, the number of frames reaching the feature extraction phase has been significantly reduced by the voice activity detector. In a resource-limited environment, this filtering has a significant impact on realizing real-time processing.

Figure 6.13 shows the scatter plot of feature vectors extracted from an infant cry signal. Figure 6.14 shows a similar scatter plot of noise audio. The distribution of the vector components of the two signals has a clear difference. Similar differences can also be noticed between the different types of cry signals with respect to the noise signals.

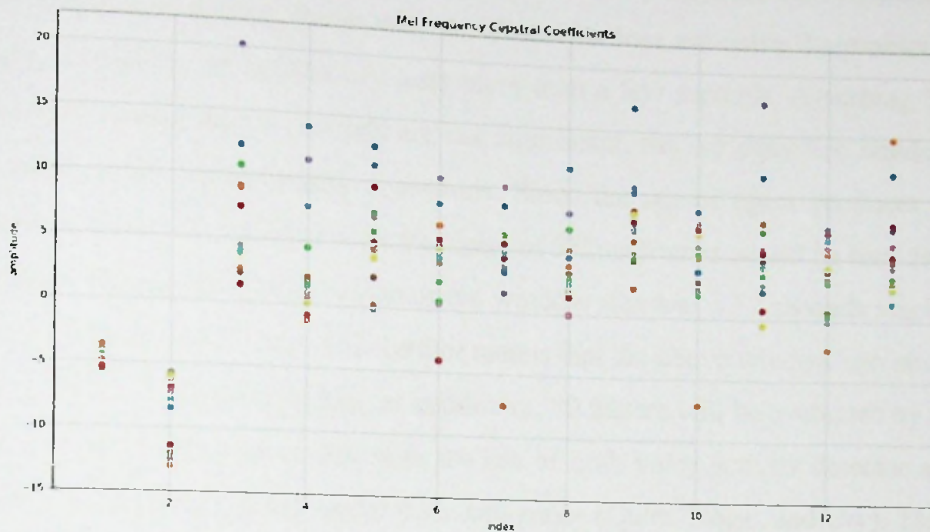


Figure 6.16 MFCC vector of the noise signal

6.3.3.6 Cry Detection

Cry detector is responsible for detecting the cry-events. Cry detector analyses the MFCC feature vectors and classify each frame as a "Cry" frame or "Noise" frame. Since the voice activity detector discarded the frames which less likely to have a human voice, the cry detector is reached by only a few frames. Hence, relevant data sets to train and test the classifier should be built with the feature vectors approved by the voice activity detector. Further, due to the inherent sensor noise, the audio signal should be recorded from the hardware solution. Direct use of the audio files, and deriving the training data sets will ignore the sensor noise, effect of the compensation filter and the down-sampling filter. Similarly, training the classifier in a computer and transferring it to the microprocessor could introduce problems if the differences of the OS environments (32bit/ 64bit) were disregarded. As per the suggestions of the majority of the researchers, two classifiers were initially selected. Hence, classifiers based

on Artificial Neural Networks and K Nearest Neighbor algorithm will be experimentally evaluated.

Although the cry detector classifies the frames of 64ms based on the corresponding MFCC feature vector, the cry events have a significantly large time window compared to the frame size. Hence, frame-wise cry detection does not solve the problem completely. Typically, an infant's cry lasts more than a few seconds. Assuming that the cry-events of less than 5 seconds are not significant, the cry detection window was estimated to be approximately 5 seconds. Since the sensor agent produces 512ms frames, deciding the window to fit multiples of 512ms frames would be easy for computations. Hence, the final decision of the window size was 5.12 seconds since it can absorb 10 frames of 512ms. This further means that the above window can absorb 80 frames of 64ms. This means that, at maximum, 80 frames will be evaluated by the cry detector within this period. But with the use of both voice activity detector and cry detector, 80 frames may fall under three categories (Quiet, Noise, and Cry). This window of 5.12 seconds is a non-overlapping window.

- Quiet frames : discarded by voice activity detector
- Noise frames : approved by voice activity detector but rejected by cry detector
- Cry frames : approved by both voice activity detector and cry detector

The label of the audio event over the 5.12 seconds is decided as follows based on the majority votes.

$$probability(cry) = \frac{a}{a + b + c}$$

$$probability(noise) = \frac{b}{a + b + c}$$

$$probability(quiet) = \frac{c}{a + b + c}$$

Where a , b , and c denotes the number of frames labeled as Quiet, Noise, and Cry, respectively. If the cry probability is more than 50%, the audio event is labeled as a Cry-event.

6.3.3.7 Selection of the Classifier for Cry Detection

As per the literature review, the choices for the classifier as the cry detector are KNN and ANN. Hence, as the next step, the classifier with the best performance in terms of accuracy and time complexity should be selected. The data sets for training, testing, and validation purposes were constructed from the features extracted from the frames selected by the VAD. To compare the classifiers with each other, the accuracy of the classification was defined as follows

$$\text{accuracy} = \frac{\text{number of correct classification}}{\text{number of total classifications}}$$

Where a correct classification refers to the instances where the predicted class is also the actual class.

6.3.3.8 K-Nearest Neighbor algorithm as the cry detector

K-Nearest Neighbor (KNN) algorithm was trained and tested as the cry detection classifier in the computer first to evaluate its practical use in this scenario. First, Euclidian and Manhattan distances were chosen as the similarity measure. Since cry detection is a binary classification, the number of neighbors (K) was selected to be an odd number to guarantee a winner in the final selection. Refer the Appendix A for source codes.

Table 6.1 shows the accuracy figures of the KNN algorithm trained with Euclidian and Manhattan distances against the number of neighbors (K) selected. In this experiment, all the neighbors were assigned uniform/equal weights. It shows that the accuracy of the detector decreases when k increases. Accuracy figures have decreased when the input data is normalized in comparison with the raw input. Further, neither of euclidian distance nor manhattan distance has clearly outperformed the other.

Table 6.1 Accuracy of KNN cry detector with uniform weights

Neighbors(k)	Euclidian Distance		Manhattan Distance	
	Raw	Normalized	Raw	Normalized
1	94.592	92.954	94.485	93.244
3	94.592	92.954	94.485	93.244
5	91.351	90.545	91.103	90.534
7	91.351	90.545	91.103	90.534
9	90.300	89.669	90.159	89.690
11	90.300	89.669	90.159	89.690
13	89.872	89.335	89.573	89.093
15	89.872	89.335	89.573	89.093
17	89.535	88.745	89.204	88.725
19	89.535	88.745	89.204	88.725

Table 6.2 Accuracy of KNN cry detector weighted with distance

Neighbors(k)	Euclidian Distance		Manhattan Distance	
	Not normal-	Normalized	Not normal-	Normalized
1	94.592	92.954	94.485	93.244
3	94.592	92.954	94.485	93.244
5	94.964	92.689	94.819	90.534
7	94.960	91.834	94.795	90.534
9	95.022	91.479	94.898	89.690
11	94.981	90.986	94.874	89.690
13	94.981	90.796	94.826	89.093
15	94.960	90.458	94.781	89.093
17	94.947	90.172	94.788	88.725
19	94.895	89.866	94.750	88.725

Similarly, Table 6.2 shows the accuracy figures of the KNN algorithm trained by assigning higher weights to the closer neighbors than the far away. In contrast to the above, this shows similar accuracy figures irrespective of the value of K when the

features are not normalized. In all the above cases, the normalization process has reduced accuracy. Hence, in cry detection, feature normalization will be ignored. As the similarity measure, Euclidian distance shows the best results when the neighbors are assigned weights based on the distance.

Hence, the best choice of a KNN classifier for cry detection would be a classifier trained under the following constraints.

- The number of neighbors (k) should be 9.
- Mel Frequency Cepstral Coefficients should be used as the feature vector.
- Input feature vectors should not be normalized.
- Euclidian distance should be the similarity measure.
- Neighbors should be weighted based on the proximity (inverse of distance).

According to the above experimental results, this would yield 95% accuracy in cry detection.

6.3.3.9 Artificial Neural Network as the cry detector

ANN algorithm was evaluated with the training and testing data in the computer to evaluate the practical use of it as the cry detector. The ANN classifier was experimented with Sigmoid, ReLU, Identity, and Tanh activation functions and different architectures to decide which suits better. Since the weight vectors are initialized randomly at the start, the accuracy of the classifier has a considerable effect on the initial values. Hence, each classifier model was trained multiple times and selected the best. Table 6.3 and 6.4 summarize the accuracy figures of the ANN classifier with Sigmoid, Tanh, ReLU, and Identity activity function. Each of the following records was obtained after training each classifier model 10 times with random initializations and selecting the best out of 10. Refer the Appendix A for source codes.

Table 6.3 Accuracy of ANN cry detector with sigmoid and Tanh functions

Architecture	Sigmoid (%)		Tanh (%)	
	Raw	Normalized	Raw	Normalized
()	67.37	65.91	67.57	65.19
(13)	89.28	87.90	90.01	87.96
(13,6)	91.00	89.14	90.97	88.58
(13,10,6)	90.92	89.53	92.49	89.18
(13,10,4)	91.08	88.92	92.09	88.98
(13,10,10,4)	90.61	88.62	92.44	89.55
(13,10,6,4)	89.59	88.99	91.94	89.27
(13,13,10, 6,4)	89.90	89.33	92.59	89.66
(13,13,13,10,4)	89.49	89.97	93.23	89.59
(13,13,13,13,4)	89.48	89.24	93.34	89.79
(13,13,13,13,10,4)	90.42	88.38	93.41	89.95
(13,13,13,13,13,4)	90.65	86.64	93.59	89.56
(13,13,13,13,13,10,4)	89.43	78.07	93.37	89.62

Architectures defined in table 6.3 and 6.4 describe the number of neurons in the hidden layers. Since the input feature vector has 13 parameters, the input layer has 13 neurons. Since the cry detector is a binary classifier, the output layer has a single layer. Hence, () denotes an ANN with the input layer of 13 neurons and the output layer of 1 neuron, while (13,6) denotes two extra hidden layers of 13 neurons followed by 6 neurons.

Table 6.4 Accuracy of ANN cry detector with ReLU and Identity functions

Architecture	ReLU		Identity	
	Raw	Normalized	Raw	Normalized
()	67.71	65.91	68.03	65.96
(13)	89.87	88.85	68.09	66.84
(13,6)	91.51	89.08	68.26	67.17
(13,10,6)	92.20	89.56	68.72	67.37
(13,10,4)	91.72	89.14	68.71	67.35
(13,10,10, 4)	92.23	89.40	68.35	67.37

(13,10,6, 4)	91.76	89.60	68.52	67.70
(13,13,10, 6, 4)	92.27	90.16	68.40	67.42
(13,13,13,10, 4)	92.36	89.58	68.35	67.18
(13,13,13,13, 4)	92.85	89.50	69.90	68.06
(13,13,13,13,10, 4)	92.58	90.08	69.01	68.69
(13,13,13,13,13, 4)	92.48	89.31	69.37	68.64
(13,13,13,13,13,10, 4)	92.37	89.97	69.74	67.79

Comparing the results obtained for each of the above cases, it is clearly evident that the identify function does not perform adequately compared to the other activation functions. Normalized data also has a performance lag with respect to the original data. This phenomenon was visible in the performance analysis of the KNN algorithm too. ReLU, Sigmoid, and Tanh functions are performing when the architecture is becoming complex. Yet, all of these three functions tend to show a very insignificant improvement of the accuracy against the increasing complexity of the architecture. Hence, it can be concluded that the increasing complexity of the architecture does not yield a sufficient improvement of the accuracy beyond a certain point (Saturation point).

Hence, the specification of the best choice of the ANN classifier for a cry detector is

- The activation function should be Tanh (Hyperbolic Tan).
- Features should not be normalized.
- The architecture of the hidden layers should be (13,13,13,10,4).
 - Higher-order architectures were discarded because the increasing complexity does not justify the selection of them over the above-selected architecture due to the very insignificant superiority.

Note that the above ANN classifiers were trained with early stopping enabled and a constant learning rate of 0.0001.

6.3.3.10 KNN vs. ANN

The above two experiments with KNN and ANN classifiers proved that optimum KNN classifier yields 95% accuracy while ANN yields 93.23%. Selected two classifiers were again tested in the hardware platform to evaluate which classifier outperforms the other.

- Prediction time of KNN per feature vector: 30ms
- Prediction time of ANN per feature vector: 0.1ms

Given that an audio event of 5.12 seconds, contains 80 feature vectors, prediction time is also a critical factor. KNN would take 2.4 seconds to label 5.12 seconds of data, while ANN takes only 8ms. Given that KNN consumes 47% of the given time, only decide whether the audio event is a cry or not.

Irrespective of the marginally superior performance of KNN over ANN, the ANN classifier was selected as the cry detector due to extensive time complexity compared to ANN.

6.3.3.11 Cry Classification

In contrast, to cry detection, cry classification is a multi-class classification. This categorizes the cause of the cry into one of the three reasons. As proof of concept, the training data set was formulated to include the most-likely causes of the cries. These reasons include hunger, belly pain, and hunger. The input to the cry classifier is the frames labeled as cry frames by the cry detector, and the remaining frames with the other labels will be ignored here onwards. Hence, the construction of the data sets for training, testing, and validation required to execute the cry detector on top of the audio frames and filter the frames with cry labels. Similar to cry detector, KNN and ANN classifiers will be tested for the performance to select which suits the best in this case.

6.3.3.12 Selection of the Classifier for Cry Detection

Similar to the selection of the classifier for cry detection, the choices for the cry classifier are also the KNN and ANN. Hence, out of these two, the classifier with the best



performance in terms of accuracy and time complexity would be selected. The data sets for training, testing, and validation purposes were constructed from the features extracted from the frames, which were labeled as cry frames by the cry detector. These cry frames were produced by playing cry recordings with known reasons. To compare the classifiers with each other, the accuracy of the classification was defined as follows

$$\text{accuracy} = \frac{\text{number of correct classification}}{\text{number of total classifications}}$$

Where a correct classification refers to the instances where the predicted class is also the actual class. Compared to the cry detector, the cry classifier produces three labels.

6.3.3.13 KNN as the Cry Classifier

KNN algorithm was trained and tested as the cry classifier on the computer first to evaluate its performance. First, Euclidian and Manhattan distances were chosen as the similarity measure. Since cry classification involves three types of output labels, the number of neighbors (K) was selected to avoid the multiple of 3 to increase the probability of having a winner in the final selection. Neighbors were weighted based on the distance as one of the test cases, while the other test case covered the uniform weights. Refer the Appendix A for source codes.

Euclidian distance with uniform weight and 5 neighbors provides the maximum accuracy when the data is not normalized. The normalized data produce slightly lower accuracies compared to the data without normalization. Similarly, all the different configurations of KNN guarantee similar accuracy figures with slight variations.

Table 6.5 Accuracy of KNN cry classifier with uniform weights

Neighbors(k)	Euclidian Distance		Manhattan Distance	
	Raw	Normalized	Raw	Normalized
1	78.40	77.49	78.40	77.73
2	78.40	77.49	78.40	77.73
4	78.40	64.28	65.88	64.96
5	78.62	66.07	67.83	66.63
7	78.25	65.89	67.41	65.99

8	78.18	62.91	64.44	63.32
10	77.83	63.04	63.30	62.24
11	77.60	63.22	63.77	62.49
13	77.16	61.73	62.59	60.93
14	76.73	60.99	61.57	60.38
16	76.44	59.69	61.17	59.64
17	75.96	59.89	61.23	59.97
19	75.70	59.58	60.66	59.55

Table 6.6 Accuracy of KNN cry classifier weighted by distance

Neighbors(k)	Euclidian Distance		Manhattan Distance	
	Raw	Normalized	Raw	Normalized
1	78.29	77.49	78.40	77.73
2	78.29	77.49	78.40	77.73
4	78.29	77.49	78.40	77.73
5	78.30	77.67	78.62	77.76
7	77.97	77.86	78.25	77.67
8	78.13	77.87	78.18	77.80
10	77.67	77.94	77.83	77.90
11	77.27	77.80	77.60	77.77
13	76.62	77.79	77.16	77.24
14	76.35	77.87	76.73	77.42
16	76.10	77.93	76.44	77.63
17	75.78	77.73	75.96	77.65
19	75.15	77.63	75.70	77.56

Best candidate of the KNN algorithm for cry classifier is as follows

- The similarity measure is the Euclidian distance.
- The number of neighbors is 5.
- Features without normalization.
- Neighbors were weighted uniformly.

6.3.3.14 ANN as a Cry Classifier

Similar to the KNN classifier, the ANN classifier was also trained and tested as the cry classifier on the computer first to evaluate its performance. Different architectures of the ANN classifier with different activation functions were tested with and without data normalization. These classifiers show variations in the accuracy figures due to random initialization. Hence, all the classifiers were tested 10 times and selected best as the candidate for each category.

Table 6.7 Accuracy of ANN cry classifier with sigmoid and Tanh functions

Architecture	Sigmoid (%)		Tanh (%)	
	Raw	Normalized	Raw	Normalized
()	34.65	34.44	33.83	34.25
(13)	35.10	35.40	41.56	39.25
(13,6)	35.11	35.08	45.64	43.45
(13,10,6)	34.79	35.22	48.55	45.04
(13,10,4)	34.03	33.82	43.61	47.40
(13,10,10,4)	34.50	33.71	49.08	48.45
(13,10,6,4)	34.90	34.18	50.03	47.73
(13,13,10, 6,4)	33.75	28.83	49.60	50.01
(13,13,13,10,4)	34.29	33.82	49.69	52.51
(13,13,13,13,4)	34.16	33.40	51.84	50.70
(13,13,13,13,10,4)	26.78	26.78	50.85	51.85
(13,13,13,13,13,4)	26.96	29.35	51.55	56.00
(13,13,13,13,13,10,4)	26.78	26.78	52.42	53.51

Table 6.8 Accuracy of ANN cry classifier with ReLu and Identity functions

Architecture	Relu (%)		Identity (%)	
	Raw	Normalized	Raw	Normalized
()	34.21	34.15	33.82	34.64
(13)	45.53	41.92	35.59	39.30
(13,6)	46.11	45.96	34.32	34.54

(13,10,6)	49.42	47.09	34.71	34.71
(13,10,4)	49.43	47.95	34.39	34.62
(13,10,10,4)	50.20	50.79	34.62	34.40
(13,10,6,4)	50.25	47.02	34.33	34.15
(13,13,10,6,4)	49.21	47.49	34.49	35.22
(13,13,13,10, 4)	50.74	48.66	34.54	33.98
(13,13,13,13,4)	52.03	51.48	33.79	34.10
(13,13,13,13,10,4)	50.70	51.88	34.19	34.11
(13,13,13,13,13,4)	51.55	48.84	34.99	34.10
(13,13,13,13,13,10,4)	52.62	47.51	34.08	34.72

ANN classifiers under any of the above test scenarios do not provide the required performance. Yet, it seems to increase the accuracy when the complexity of the architecture increases when the ReLu function is selected as the activation function and features are not normalized. Yet, due to increasing the complexity of the architecture, this was not further tested since this would not fit the hardware platform.

Note that the above ANN classifiers were trained with early stopping enabled and a constant learning rate of 0.0001.

6.3.4 KNN Vs. ANN

Due to the absence of a competitive ANN classifier, the KNN classifier was selected as the cry classifier.

6.3.5 Auditor agent

The auditor agent is responsible for auditing the different agents which detect different types of anomalies. When the system grows large covering different types of anomalies, some anomalies may need higher priority and quick attention from the caregivers compared to others. For example, assume that the device is monitoring the infant vitals (heart rate, temperature, and oxygen saturation) and cry-events. These events are monitored independent agents, and hence, anomalies in each parameter are unknown to

each other. But the system should not give priority to infant cry-event when an abnormal heart rate is simultaneously detected by the system since parents respond to the above scenarios differently. Hence, the auditor agent is introduced to decide which anomalies are to be notified and in which order they should be handled. This simulates the financial auditors who audit all the financial records to provide their recommendations. The importance of the auditor agent is highlighted when the system grows larger, monitoring multiple parameters of infant health. Within the scope of cry detection and classification, the auditor receives messages from the classifier agent and the manager agent only.

6.3.6 Network agent

The network agent holds the responsibility of sending messages and notifications to the caregivers through the server. These messages are generated by the auditor agent, and under some special conditions when the system grows large, few more agents may also contribute to the content generation for notifications. Within the scope of the cry classification, the network agent is included in the system to build the complete system but not fully implemented since the objectives of this research do not cover the implementation of the network agent. In a real use case, the network agent holds the following responsibilities.

- Device authentication and authorization. Market ready devices are required to implement the device authentication and authorization as a security feature of the device. This restricts the use of the backend system by unauthorized devices and manipulating the data records.
- Device logs and statistics. In hardware devices, it is a general practice to write the firmware to generate device logs from time to time, depending on the conditions it hits. It helps in developing statistical figures of different use-cases and bugs etc. and identifying new updates and upgrades required in the next versions.
- Device configurations. Users may configure the devices based on their preferences through the mobile application. When it happens, the device should

reflect the changes in hardware in real-time. To achieve this network agent should receive the user configurations.

- **Firmware updates.** From time to time, developers may require to release firmware updates due to algorithm improvements, bug fixing, and upgrading the system with new features. In firmware updates, upon the consent of the users, the firmware is to be updated automatically. This requires a major contribution from the network agent to download the new firmware without getting corrupted.
- **Remote debugging/troubleshooting.** Users may complain about the malfunctions about the devices sometimes. In such events, access to the hardware device remotely may help the technical support team to resolve the problems. Remote Secure shell (SSH) connection with proper security measures to restrict unauthorized access to the system, can help to investigate and resolve the problem.

6.3.7 Manager agent

Typically, multi-agent systems platforms are built with the features of life cycle management. In this research, existing multi-agent platforms are not chosen due to the limitations in hardware resources. Therefore an implementation of a basic multi-agent system was preferred using python processes, queues, and pipes. Hence, the “manager” agent was introduced to the system to manage the life cycles of the other agents. This holds the responsibility of monitoring the states of the other agents and initializing the agents which were killed by the system due to reasons like exceptions. This agent monitors the states of the other agents twice a second to detect any abnormalities. Upon detection of any agent died due to any exception, the manager agent will reinitialize the dead agent to minimize the effect.

6.4 Summary

This chapter covered the scopes of hardware selection, firmware design, and algorithms in detail. This chapter further explained the terms and conditions that governed

the decisions each phase. Hardware selection had to compromise between the resource requirements for audio processing and the overall manufacturing cost. The main focus of the firmware design was to achieving real-time audio processing by avoiding resource over-utilization. Hence, Multi-Agent System (MAS) based firmware design was preferred since it also provided the flexibility of updating and upgrading the firmware with a minimal effect on the other components. Next, this chapter discussed noise filtering, voice activity detection, feature extraction, cry detection, and cry classification in detail under the algorithms. It further explained the choice of each algorithm was based on both the accuracy and computational complexity in prediction. Noise filtering deals with the preprocessing of the signal suppressing the noise while voice activity detector filters the audio events with the human voice. Similarly, the cry detection classifier filters the cry-events from the non-cry-events. The voice activity detector and cry detector play a critical role in reducing the computation complexity. As the final step, the cry classifier would be executed upon the reception of the cry frames from the cry detector. This finally produces results indicating when and why the infant is crying.

7 EVALUATION OF THE SYSTEM

7.1 Introduction

Chapter 6 described the implementation of the proposed solution in detail. Within the scope of the implementation, different classifiers were evaluated to decide which classifier suits best at which position. The scope of the evaluation of these classifiers covered only the scope of each module. In this phase, this chapter evaluates the proposed solution as a single unit. To evaluate the solution, the test conditions are defined first. As the next step, the author will explain the test setup. Finally, the analysis of the test results will follow to conclude this chapter.

7.2 Test Scenarios

The proposed solution was tested under different scenarios, which are likely to occur in normal operational conditions. The proposed solution should perform cry detection and classification accurately when the input audio feed includes the following audio activities.

- Infant cries
- Infant laughs
- Adult voices
- Lullabies/white noise
- Multimedia devices
- Quiet environment

7.3 Test Setup

The hardware device is kept on a table in a room with a separation of 3m from the audio player. This is to simulate the typical placement of a baby monitor in domestic

use. Then the computer played different audio files and recorded the response to calculate the accuracy in each test scenario independently. The results were collected to evaluate the accuracy of cry detection and cry classifier separately.

7.4 Definitions of the Evaluation Parameters

The following standard parameters were estimated according to the context of the proposed solution to evaluate the performance of the solution.

7.4.1 True Positives (TP)

True positives are defined as the correct classifications of members of class A as members of class A. In this context, with respect to cry signals, the true positives (TP) are the classifications of cries as cries.

7.4.2 True Negatives (TN)

True negatives are defined as the correct classifications of non-members of class A as non-members of class A. In this context, with respect to cry signals, the true negatives (TN) are the classifications of noise as noise.

7.4.3 False Positives (FP)

False positives are defined as the incorrect classifications of non-members of class A as members of class A. In this context, with respect to cry signals, the false positives (FP) are the classifications of noise as cries.

7.4.4 False Negatives (FN)

False negatives are defined as the incorrect classifications of members of class A as non-members of class A. In this context, with respect to cry signals, the false negatives (FN) are the classifications of cries as noise.

7.4.5 Precision

Precision has been defined as the ratio of the true positives to the total predicted positives.

$$\textit{precision} = \frac{TP}{TP + FP}$$

This estimates the probability of the system to be correct when it is predicting positive results.

7.4.6 Specificity

Specificity has been defined as the ratio of the true negative to the total actual negatives.

$$\textit{specificity} = \frac{TN}{TN + FP}$$

This estimates the probability of the system detecting negatives as negatives correctly.

7.4.7 Sensitivity

Sensitivity has been defined as the ratio of the true positives to the total actual positives.

$$\textit{sensitivity} = \frac{TP}{TP + FN}$$

This estimates the probability of the system detecting the positives as positives.

7.4.8 F1 Measure

F1 measure provides a meaning to the above parameters since it forms the balance between the above scores. F1 is a weighted value defined as follows to have a range between 0 and 1.



$$F1 \text{ measure} = \frac{2 \times \text{precision} \times \text{sensitivity}}{\text{precision} + \text{sensitivity}}$$

7.5 Performance of the Cry Detector

The test setup for the cry detector tests the performance of both the voice activity detector and the cry detector. These results should also give an overview of the success of the cry detector in different conditions. This test produces 3 types of outputs called Quiet, Noise, and Cry. This test produces the Cry and Noise labels only after consulting the cry detecting classifier. The quiet label is produced without any involvement of the cry detector. Hence the percentage of each label gives an approximation about the use of the classifier.

Table 7.1 Output of the cry detector under various test scenarios

Audio event	Quiet (%)	Noise (%)	Cry (%)
Quiet	216	1	0
Noise	8	205	7
Music	0	252	4
Laugh (baby)	0	320	2
Adult(male)	0	222	2
Adult(female)	0	200	2
Adult(stammer)	0	255	37
Cry	2	5	205

Table 7.1 summaries the test results of the cry detector under different test scenarios.

Table 7.2 Accuracy of the cry detector under various test scenarios

Audio event	Quiet (%)	Noise (%)	Cry (%)
Quiet	99.54	0.46	0.00
Noise/Lullabies	3.64	93.18	3.18
Music	0.00	98.44	1.56
Laugh (baby)	0.00	99.38	0.62
Adult(male)	0.00	99.11	0.89

Adult(female)	0.00		
Adult(stammer)	0.38	99.01	0.99
Cry	0.94	85.55	14.07
		2.36	96.69

Table 7.2 shows the accuracy as a percentage based on the results shown in table 7.1. The above results confirm that the combination of the cry detector and voice detector classifies the noise with an accuracy above 96% except when the speaker has the stammer. The reason behind the low accuracy for stammer might be the high percentage of pauses present in the voice. The success rate of cry detection is 96.69%, which is beyond the threshold set at the beginning.

Summary of the cry detector evaluation is as follows:

- True Negatives (noise detecting as noise) : 96.83%
- True Positives (cry detecting as cry) : 96.69%
- False negatives (cry detecting as noise) : 3.31%
- False positives (noise detecting as cry) : 3.16%
- Overall accuracy : 96.76%

7.6 Performance of the Cry Classifier

The evaluation of the cry classifier was based on the cry audio files played by the audio player in the vicinity. Since the cry classifier is getting activated if and only if the cry detector detects a cry, these test results are dependent on the performance of the cry detector as well. This test evaluates the accuracy figures of the cry classifications of the actual cry-events. This means that the classification of the noise as cry-events were not considered. Hence, this evaluates the conditional probabilities of successful cry classifications, given that cry is already detected.

Table 7.3 Confusion matrix for cry frame classification

Events	Belly Pain	Hungry	Tiredness
Belly Pain	1887	315	220
Hungry	528	2680	248
Tiredness	480	529	1551

Table 7.4 Performance evaluation of cry frame classification

Events	Belly Pain (%)	Hungry	Tiredness
Precision	65.18	76.05	76.20
Specificity	83.25	83.06	92.04
Sensitivity	77.91	77.55	60.59
F1 measure	70.98	76.79	67.66

Table 7.3 shows the results obtained when the frame-wise performance of the cry classifier was evaluated. These statistics summarized the behavior of the classifier when cry frames were fed. The evaluation scores calculated based on the results are shown in Table 7.4

7.6.1 Classification of the Frames of Cries Due to Belly Pain

The precision of the belly pain cries shows a comparatively low score. Yet, in comparison with the average classification accuracies in the literature reviews, this is an average score. This suggests that there is a probability of 65.18% for the baby to have belly pain when the system predicts so. In contrast, this has a fairly good ability to rule out belly pains when it is not. This is indicated by the specificity score of 83.25%. The sensitivity of 77.91% shows that the solution is sensitive to belly pains. The F1 measure gives a fairly average value highlighting low precision.

7.6.2 Classification of the Frames of Cries Due to Hunger

The precision of the hungry cries shows a good score of 76.05%. This suggests that there is a good probability of 76.05% for the baby to be hungry when the system predicts so. Similarly, this has a fairly good ability to rule out hunger when it is not. This is indicated by the specificity score of 83.06%. The sensitivity of 76.79% shows that the solution is sensitive to hunger, as well. The F1 measure gives a fairly good score since both sensitivity and precision have good scores.

7.6.3 Classification of the Frames of Cries Due to Tiredness

The precision of the tiredness shows a good score of 76.20%. This suggests that there is a good probability of 76.20% for the baby to be tired when the system predicts so. Similarly, this has a very good ability to rule out tiredness when it is not. This is indicated by the specificity score of 92.04%. The sensitivity of 60.59% shows that the solution is not sensitive to tiredness, although it has a higher probability of being correct when it predicts. The F1 measure gives a fairly average score since the sensitivity is low, though the precision has a good score.

Table 7.5 Confusion matrix for cry-event classification

Events	Belly Pain	Hungry	Tiredness
Belly Pain	244	7	0
Hungry	8	114	1
Tiredness	89	28	99

Table 7.6 Performance evaluation of cry-event classification

Events	Belly Pain (%)	Hungry	Tiredness
Precision	71.55	76.51	99.00
Specificity	71.39	92.51	99.73
Sensitivity	97.21	92.68	45.83
F1 measure	82.43	83.82	62.69

Table 7.5 shows the results obtained when the event-wise performance of the cry classifier was evaluated. These statistics summarized the behavior of the classifier when cry frames were fed. The evaluation scores calculated based on the results are shown in Table 7.6

7.6.4 Classification of the Events of Belly Pain Cries

The precision of the belly pain cries shows a comparatively good score. This suggests that there is a probability of 71.55% for the baby to have belly pain when the system predicts so. Similarly, this also has a fairly good ability to rule out belly pains when it is not as well. This is indicated by the specificity score of 71.39 %. The sensitivity of 97.21% shows that the solution is very sensitive to belly pains. The F1 measure produces a fairly good value due to comparatively good scores of precision and sensitivity.

7.6.5 Classification of the Events of Hungry Cries

The precision of the hungry cries shows a comparatively good score. This suggests that there is a high probability of 76.51% for the baby to be hungry when the system predicts so. Additionally, this has a very good ability to rule out hunger when it is not. This is indicated by the specificity score of 92.51%. The sensitivity of 92.68% shows that the solution is very sensitive to hunger, as well. The F1 measure gives a fairly good score since both sensitivity and precision have good scores.

7.6.6 Classification of the Events of Tiredness Cries

The precision of the tiredness shows a good score of 99.0%. This suggests that there is a very high probability of 99.0% for the baby to be tired when the system predicts so. Similarly, this has a very good ability to rule out tiredness when it is not. This is indicated by the specificity score of 92.04%. The sensitivity of 60.59% shows that the solution is not sensitive to tiredness, although it has a higher probability of being

correct when it predicts. The F1 measure gives a fairly average score since the sensitivity is low, although the precision has a good score.

The overall accuracy of the cry classifier is estimated to be 77.46%, and it exceeds the threshold of 70% defined as one of the objectives based on the literature review.

7.7 Summary

This chapter evaluated the performance of the system in terms of the two types of outputs the solution should provide. This evaluated the performance of the cry detector first and found out that the overall performance of the combination of the voice activity detector and cry detector is good. Next, the author evaluated the frame-wise performance of the cry classifier. It showed that the performance of the system in detecting tired babies is comparatively low. Finally, the event-wise performance of the cry classifier was evaluated, and it also showed a similar performance lag in detecting tiredness. Overall other probable causes were detected well. Finally, it was concluded that the performance of the system exceeds the minimum requirements defined as the objectives

8 CONCLUSION AND FUTURE WORK

8.1 Introduction

Chapter 7 discussed the performance of the system after evaluating the solution under multiple scenarios. It was concluded that the overall performance of cry detection is very good, and cry classification shows fairly good performance. In chapter 8, the author discusses the limitations of the system and future works related to the improvements. With that, the author concludes the literature.

8.2 Conclusion

With the evaluation of the overall system, it can be concluded that the overall performance of cry detection has exceeded the threshold defined in the objectives. This showed accuracy figures of more than 96%, where the threshold based on the literature review and practical use case was set to be 95%. Similarly, the cry classifier also exceeded the expected threshold, recording 77% of accuracy. This research was conducted as a proof of concept to evaluate the feasibility of real-time audio processing in the hardware platform. Execution of the system and observing results in real-time proved the hypothesis that the firmware design based on the multi-agent concept can provide the required level of performance. This also proved that these hardware devices could support AI-based approaches to feed more intelligence.

8.3 Limitation and Related Future Works

During the implementation of the proposed solution and the evaluation, the following issues were identified. Further, the author believes that these issues can be resolved in the future as an extension of this research.

8.3.1 Low Accuracy in Cry Classification for Some Classes

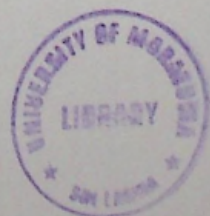
Classification of the cry signals showed some low accuracy figures in predicting tiredness as the most likely cause of cry. The literature review also proved that even though cry detection showed the accuracies above 90%, cry classification has proven to be a hard problem. So, the author believes that there is a space for improvements for the cry classification.

8.3.2 Validation of the Cry Source

The current design does not validate the source of the cry. The system does not know whether the cry sound is coming from an actual infant or any audio player. Similarly, as a baby monitor, it doesn't know the baby who is detected to be cry is the designated subject for monitoring. For example, in an environment where multiple babies are being monitored simultaneously using multiple babies, one baby can trigger all the baby monitors. The author firmly believes that this problem can be solved using the data observed by the wearables in the baby monitors. These wearables measure the motion and other biological parameters. These parameters tend to differ when the infants are in motion. Hence, this can be used to validate the cry since cries are generally associated with feet movements. Vital parameters and motion patterns might differ from one cause to another. Hence, this approach might help in improving the accuracy of detection and classification of the cry as well.

8.3.3 Unavailability of a Multi-Agent Platform for Embedded Systems

Unavailability of a light-weight Multi-agent development platform for embedded systems was noticed at the implementation of the solution. Yet, this has proven its worth in different aspects. Further, since the firmware of the proposed solution was designed borrowing the concepts of MAS, the author believes that MAS has a unique importance in embedded developments as well.



REFERENCES

- [1] A. A. Dixit and N. V. Dharwadkar, "A Survey on Detection of Reasons Behind Infant Cry Using Speech Processing," in *2018 International Conference on Communication and Signal Processing (ICCSP)*, Chennai, 2018, pp. 190–194, doi: 10.1109/ICCSP.2018.8524517.
- [2] R. Sahak, W. Mansor, L. Y. Khuan, A. Zabidi, and A. I. M. Yassin, "Detection of asphyxia from infant cry using support vector machine and multilayer perceptron integrated with Orthogonal Least Square," in *Proceedings of 2012 IEEE-EMBS International Conference on Biomedical and Health Informatics*, Hong Kong, 2012, pp. 906–909, doi: 10.1109/BHI.2012.6211734.
- [3] G. Jr. Varallyay, Z. Benyo, A. Illenyi, Z. Farkas, and L. Kovacs, "Acoustic analysis of the infant cry: classical and new methods," in *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, San Francisco, CA, USA, 2004, vol. 3, pp. 313–316, doi: 10.1109/IEMBS.2004.1403155.
- [4] S. Sharma and V. K. Mittal, "A qualitative assessment of different sound types of an infant cry," in *2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON)*, Mathura, 2017, pp. 532–537, doi: 10.1109/UPCON.2017.8251106.
- [5] L. Liu, Y. Li, and K. Kuo, "Infant cry signal detection, pattern extraction and recognition," in *2018 International Conference on Information and Computer Technologies (ICICT)*, DeKalb, IL, 2018, pp. 159–163, doi: 10.1109/IN-FOCT.2018.8356861.
- [6] M. J. Kim, Younggwon Kim, Seungki Hong, and H. Kim, "ROBUST detection of infant crying in adverse environments using weighted segmental two-dimensional linear frequency cepstral coefficients," in *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, San Jose, CA, USA, 2013, pp. 1–4, doi: 10.1109/ICMEW.2013.6618321.
- [7] J. Saraswathy, M. Hariharan, S. Yaacob, and W. Khairunizam, "Automatic classification of infant cry: A review," in *2012 International Conference on Biomedical Engineering (ICoBE)*, Penang, Malaysia, 2012, pp. 543–548, doi: 10.1109/ICoBE.2012.6179077.
- [8] L. Abou-Abbas, L. Montazeri, C. Gargour, and C. Tadj, "On the use of EMD for automatic newborn cry segmentation," in *2015 International Conference on Advances in Biomedical Engineering (ICABME)*, Beirut, Lebanon, 2015, pp. 262–265, doi: 10.1109/ICABME.2015.7323302.
- [9] Y. Lavner, R. Cohen, D. Ruinskiy, and H. IJzerman, "Baby Cry Detection in Domestic Environment using Deep Learning," p. 5, 2016.
- [10] K. Srijiaranon and N. Eiamkanitchat, "Application of neuro-fuzzy approaches to recognition and classification of infant cry," in *TENCON 2014 - 2014 IEEE*

- Region 10 Conference*, Bangkok, Thailand, 2014, pp. 1–6, doi: 10.1109/TENCON.2014.7022296.
- [11] P. R. Myakala, R. Nalumachu, S. Sharma, and V. K. Mittal, “A low cost intelligent smart system for real time infant monitoring and cry detection,” in *TENCON 2017 - 2017 IEEE Region 10 Conference*, Penang, 2017, pp. 2795–2800, doi: 10.1109/TENCON.2017.8228337.
- [12] M. P. Joshi and D. C. Mehetre, “IoT Based Smart Cradle System with an Android App for Baby Monitoring,” in *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, Pune, 2017, pp. 1–4, doi: 10.1109/ICCUBEA.2017.8463676.
- [13] I. Galanis, D. Olsen, and I. Anagnostopoulos, “A multi-agent based system for run-time distributed resource management,” in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, Baltimore, MD, USA, 2017, pp. 1–4, doi: 10.1109/ISCAS.2017.8050298.
- [14] H. M. Kelash, M. Amoon, G. M. Ali, and H. M. Faheem, “A Social Agent Interface for Resource Management in Distributed Systems,” in *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-LAWTIC'06)*, Vienna, Austria, 2005, vol. 2, pp. 390–395, doi: 10.1109/CIMCA.2005.1631500.
- [15] T. Nagata, Y. Ueda, and M. Utatani, “A multi-agent approach to smart grid energy management,” in *2012 10th International Power & Energy Conference (IPEC)*, Ho Chi Minh City, 2012, pp. 327–331, doi: 10.1109/ASSCC.2012.6523287.
- [16] M. Karani, N. Giri, and S. Bodhe, “Resource Management for Cellular Network Using Socially Intelligent Multi Agent System,” in *2010 International Conference on Recent Trends in Information, Telecommunication and Computing*, Kochi, Kerala, 2010, pp. 231–233, doi: 10.1109/ITC.2010.73.
- [17] I. Burlachenko, “Management of energy efficient distributed computer systems with self-contained remote modules using multi-agent system,” in *2015 IEEE 35th International Conference on Electronics and Nanotechnology (ELNANO)*, Kyiv, Ukraine, 2015, pp. 512–514, doi: 10.1109/ELNANO.2015.7146940.
- [18] A. Al-Abdullah *et al.*, “Artificial Neural Network for Arabic Speech Recognition in Humanoid Robotic Systems,” in *2019 3rd International Conference on Bio-engineering for Smart Technologies (BioSMART)*, Paris, France, 2019, pp. 1–4, doi: 10.1109/BIOSMART.2019.8734261.
- [19] N. Joshi, A. Kumar, P. Chakraborty, and R. Kala, “Speech controlled robotics using Artificial Neural Network,” in *2015 Third International Conference on Image Information Processing (ICIIP)*, Wagnaghat, India, 2015, pp. 526–530, doi: 10.1109/ICIIP.2015.7414829.
- [20] M. H. Hammad, A. Mohammed, and M. E. Eldow, “Design an electronic system use the audio fingerprint to access virtual classroom using Artificial Neural Networks,” in *2015 International Conference on Computer, Communications, and*

- Control Technology (I4CT)*, Kuching, Sarawak, Malaysia, 2015, pp. 192–195, doi: 10.1109/I4CT.2015.7219564.
- [21] A. N. Vasquez, D. M. Ballesteros, and D. Renza, “Speaker verification with fake intonation based on Neural Networks,” in *2019 7th International Workshop on Biometrics and Forensics (IWBF)*, Cancun, Mexico, 2019, pp. 1–5, doi: 10.1109/IWBF.2019.8739173.
- [22] K. Lu and X. Zhang, “Audio-Visual Emotion Recognition Using Neural Networks Learned with Hints,” in *2013 Seventh International Conference on Image and Graphics*, Qingdao, China, 2013, pp. 515–519, doi: 10.1109/ICIG.2013.109.
- [23] A. N. Vasquez, D. M. Ballesteros, and D. Renza, “Speaker verification with fake intonation based on Neural Networks,” in *2019 7th International Workshop on Biometrics and Forensics (IWBF)*, Cancun, Mexico, 2019, pp. 1–5, doi: 10.1109/IWBF.2019.8739173.
- [24] B. D. Barkana, N. John, and I. Saricicek, “Auditory Suspicious Event Databases: DASE and Bi-DASE,” *IEEE Access*, vol. 6, pp. 33977–33985, 2018, doi: 10.1109/ACCESS.2018.2848269.
- [25] S. Jiang, H. Frigui, and A. W. Calhoun, “Speaker Identification in Medical Simulation Data Using Fisher Vector Representation,” in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, Miami, FL, USA, 2015, pp. 197–201, doi: 10.1109/ICMLA.2015.187.
- [26] D. Johnson and G. Tzanetakis, “Guitar model recognition from single instrument audio recordings,” in *2015 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, Victoria, BC, Canada, 2015, pp. 370–375, doi: 10.1109/PACRIM.2015.7334864.
- [27] Y. Li and G. Liu, “Sound classification based on spectrogram for surveillance applications,” in *2016 IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, Beijing, China, 2016, pp. 293–297, doi: 10.1109/ICNIDC.2016.7974583.
- [28] C. Megha and V. K. Reddy, “Robust Classification of Abnormal Audio using Background-Foreground Separation,” in *2017 14th IEEE India Council International Conference (INDICON)*, Roorkee, 2017, pp. 1–6, doi: 10.1109/INDICON.2017.8487922.
- [29] S. Vhaduri, T. V. Kessel, B. Ko, D. Wood, S. Wang, and T. Brunschwiler, “Nocturnal Cough and Snore Detection in Noisy Environments Using Smartphone-Microphones,” in *2019 IEEE International Conference on Healthcare Informatics (ICHI)*, Xi’an, China, 2019, pp. 1–7, doi: 10.1109/ICHI.2019.8904563.

APPENDIX A: Source Codes

The Preprocessor Agent

```
import os
import sys
import mmap
import pypruss
import numpy as np
import Queue as queue_except
from multiprocessing import Process, Queue
from scipy.signal import firwin2, lfilter, resample
import wave

class AudioRecord(Process):

    def __init__(self, queue):
        super(AudioRecord, self).__init__()
        self.__exit_audio_record = False
        self.__frame_size = 32768
        self.__queue = queue

    @staticmethod
    def __configure_pins():
        os.system('sudo sh Audio/Record/Config/config_pins.sh')

    @staticmethod
    def __map_memory():
        memory_address = pypruss.ddd_addr()
        memory_length = pypruss.ddd_size()
        host_memory = pypruss.map_extmem()

        data = np.array([memory_address, memory_length], dtype=np.uint32)
        pypruss.pru_write_memory(1, 0, data)
        return [memory_address, memory_length, host_memory]

    def run(self):
        self.__configure_pins()

        pypruss.init()
        pypruss.open(1)

        pypruss.pruinc_init()

        address = self.__map_memory()

        pypruss.exec_program(1, 'Audio/Record/Config/cic_pru1.bin')

        with open("/dev/mem", "r+b") as f:
            ddr_mem = mmap.mmap(f.fileno(), address[1], offset=address[0])

        first_half = True

        h = self.__get_compensation_filter()

        last_frame = np.zeros(self.__frame_size, dtype=np.float32)

        signal_factor = 1000.0
        padding_window_size = 1000

        while True:
            pypruss.wait_for_event(1)
            pypruss.clear_event(1, pypruss.PRU1_ARM_INTERRUPT)
```



```

if first_half:
    raw_u32 = np.frombuffer(DDR_mem[:address[1] / 2], dtype=np.uint32)
    first_half = False
else:
    raw_u32 = np.frombuffer(DDR_mem[address[1] / 2:], dtype=np.uint32)
    first_half = True

frame = np.copy(raw_u32)
frame = frame.astype(np.float32)

con_frame = np.concatenate((last_frame[-padding_window_size:], frame))
last_frame[:] = frame

filtered_frame = lfilter(h, 1, con_frame)

filtered_frame = filtered_frame[padding_window_size:]
filtered_frame = filtered_frame - np.mean(filtered_frame)
filtered_frame = filtered_frame/signal_factor

filtered_frame = resample(filtered_frame, self.__frame_size/8)
try:
    self.__queue.put_nowait(filtered_frame)
except queue_exceptions.Full:
    pass

```

```

DDR_mem.close()

```

```

@staticmethod

```

```

def __get_compensation_filter():

```

```

    eps = sys.float_info.epsilon

```

```

    H = 4

```

```

    M = 1

```

```

    R = 16

```

```

    Fs = 64000

```

```

    f = np.linspace(0, 1.0, Fs/2)

```

```

    hf = (np.sin(np.pi * M * f) / (np.sin(np.pi * f / R) + eps)) ** (2 * H)

```

```

    hf = hf[1:]

```

```

    hf_db = 10 * np.log10(np.abs(hf))

```

```

    hf_db = hf_db - np.max(hf_db)

```

```

    hf = 10 ** (hf_db / 10.0)

```

```

    gf = 1.0 / hf

```

```

    Fc = 8000

```

```

    L_fc = int(Fc/2.0)

```

```

    gf[L_fc:] = 0

```

```

    gf = np.concatenate((np.array([0]), gf))

```

```

    n = 20

```

```

    h = firwin2(n, f, gf)

```

```

    return h

```

Selection of The Best KNN Classifier for Cry Detection

```
from research.researchknn import ResearchKNN
import os
```

```
if __name__ == '__main__':
```

```
    train_file = 'train_detector' # training data set
    test_file = 'test_detector' # testing data set
    classes = 2 # number of classes
    neighbors = range(1,20) # choices of K

    weights = ['uniform', 'distance'] # similarity measures
    r_values = [1, 2] # corresponding values for Minkowski distance
    normalize = [False, True]
```

```
    # create directories to save classifiers and results.csv
    if not os.path.exists('architecture'):
        os.mkdir('architecture')
```

```
    if not os.path.exists('architecture\\classifiers'):
        os.mkdir('architecture\\classifiers')
```

```
    path = 'architecture'
```

```
    for weight in weights:
```

```
        for r_value in r_values:
            distance = 'manhattan' if r_value == 1 else 'euclidean'
```

```
            file_name = '{0}\\{1}_{2}.csv'.format(path, weight, distance)
            output_file = open(file_name, 'w')
```

```
            for k in neighbors:
                if k % classes == 0:
                    continue
```

```
            save_as = 'architecture\\classifiers\\detector_{0}_{1}_{2}_r.pickle'.format(weight, distance, k)
            summary, accuracy_1, time_taken_1 = ResearchKNN.train_knn_classifier(train_file=train_file,
                                                                              test_file=test_file,
                                                                              classifier_name=save_as,
                                                                              train=True,
                                                                              normalize=False,
                                                                              neighbors=k,
                                                                              r=r_value,
                                                                              weights=weight)
```

```
            save_as = 'architecture\\classifiers\\detector_{0}_{1}_{2}_n.pickle'.format(weight, distance, k)
            summary, accuracy_2, time_taken_2 = ResearchKNN.train_knn_classifier(train_file=train_file,
                                                                              test_file=test_file,
                                                                              classifier_name=save_as,
                                                                              train=True,
                                                                              normalize=True,
                                                                              neighbors=k,
                                                                              r=r_value,
                                                                              weights=weight)
```

```
            result = '{0},{1:.5f},{2:.5f}'.format(k, accuracy_1, accuracy_2)
            output_file.write('{0}\n'.format(result))
            output_file.flush()
            print result
```

```
            output_file.close()
            os.system("shutdown /s /t 1") # finally shutdown
```


Selection of The Best ANN Classifier for Cry Detection

```
import os
from research.researchann import ResearchANN

...
n = normalized
r = raw/not normalized
...
marker = '\n-----{0}_{1}-----\n'
if __name__ == '__main__':

    train_file = 'train_classifier'
    test_file = 'test_classifier'

    instances = 10
    functions = ['identity', 'relu', 'logistic', 'tanh']
    models = [((), 13, (13, 6), (13, 10, 6), (13, 10, 4), (13, 10, 10, 4), (13, 10, 6, 4),
              (13, 13, 10, 6, 4), (13, 13, 13, 10, 4), (13, 13, 13, 13, 4), (13, 13, 13, 13, 10, 4),
              (13, 13, 13, 13, 13, 4), (13, 13, 13, 13, 13, 10, 4)]

    for function in functions:
        if not os.path.exists('architecture'):
            os.mkdir('architecture')

        if not os.path.exists('architecture\\{0}'.format(function)):
            os.mkdir('architecture\\{0}'.format(function))

        for iteration in range(1, instances + 1):

            path = 'architecture\\{0}\\iteration_{1}'.format(function, iteration)
            if not os.path.exists(path):
                os.mkdir(path)

            output_file = open('{0}\\model_{1}.csv'.format(path, function), 'a')

            print marker.format(function, iteration)

            print marker.format(function, iteration)

            for model in models:
                "run a new instance without data normalization"
                save_as = '{0}\\detector_r_{1}.pickle'.format(path, model)
                summary, accuracy_1, time_taken_1 = ResearchANN.train_ann_classifier(train_file=train_file,
                                                                                    test_file=test_file,
                                                                                    classifier_name=save_as,
                                                                                    train=True,
                                                                                    normalize=False,
                                                                                    model=model,
                                                                                    activation=function)

                "run a new instance with data normalization"
                save_as = '{0}\\detector_n_{1}.pickle'.format(path, model)
                summary, accuracy_2, time_taken_2 = ResearchANN.train_ann_classifier(train_file=train_file,
                                                                                    test_file=test_file,
                                                                                    classifier_name=save_as,
                                                                                    train=True,
                                                                                    normalize=True,
                                                                                    model=model,
                                                                                    activation=function)

            result = '{0},{1:.5f},{2:.5f}'.format(model, accuracy_1, accuracy_2)

            output_file.write('{0}\n'.format(result))
            print result
            output_file.flush()
            output_file.close() # close the file

os.system("shutdown /s /t 1") # finally shutdown
```

Cry Classifier

```
import json
import pickle
import numpy as np
from scipy import stats
from math import ceil, log
from Utils.util import Util
from Utils.logs import Logs
from multiprocessing import Process
from python_speech_features import mfcc
from Audio.Analysis.Features.ZeroCrossing import ZeroCrossing
from Audio.Analysis.Features.ShortTimeEnergy import ShortTimeEnergy

class CryDetection(Process):

    __PROCESSING_INTERVAL = 10
    __STZC_THRESHOLD = 10
    __STE_THRESHOLD = 0.05
    __SAMPLE_FREQUENCY = 8000
    __WINDOW_WIDTH = 512
    __NOISE_PROBABILITY = 1

    def __init__(self, queue):
        super(CryDetection, self).__init__()
        self.__queue = queue
        self.__log = Logs()

    def run(self):
        detector_labels = ['cry', 'noise', 'quiet']
        classifier_labels = ['belly_pain', 'hungry', 'tired']

        with open('Resources/Classifiers/cry_detector.pickle', 'rb') as model_file:
            cry_detector = pickle.load(model_file)

        with open('Resources/Classifiers/cry_classifier.pickle', 'rb') as model_file:
            cry_classifier = pickle.load(model_file)

        frame_index = 0
        segment = np.array([], dtype=np.float32)
        classes = detector_labels[0:-1]

        window_len = 0.064 #in seconds
        window_step = window_len #in seconds
        fft_length = int(2 ** ceil(log(window_len*self.__SAMPLE_FREQUENCY, 2)))

        frames_per_iteration = int((4096*CryDetection.__PROCESSING_INTERVAL)/(self.__SAMPLE_FREQUENCY*window_len))

        noise_frames = 0

        cry_records = [0,0,0]
        vad_records = [0,0,0]

        iteration = 0
        while True:
            try:
                frame = self.__queue.get()

                for i in range(0, len(frame), self.__WINDOW_WIDTH):
                    sub_frame = frame[i : i + self.__WINDOW_WIDTH]
                    zeros = ZeroCrossing.extract(sub_frame)
                    energy = ShortTimeEnergy.extract(sub_frame)

                    if zeros <= self.__STZC_THRESHOLD and energy >= self.__STE_THRESHOLD:
                        segment = np.concatenate((segment, sub_frame))
                    else:
                        noise_frames += int(len(sub_frame)/self.__WINDOW_WIDTH)
```




```

frame_index += 1

if frame_index == CryDetection.__PROCESSING_INTERVAL:
    if len(segment) >= 2*self.__SAMPLE_FREQUENCY:
        start = Util.get_epoch()
        features = mfcc(segment, self.__SAMPLE_FREQUENCY, winlen=window_len,
                        winstep=window_step, nfft=fft_length,
                        highfreq=4000)

        reduced_features = []
        for feature in features:
            vector = feature
            vector = feature / np.sum(vector)
            reduced_features.append(vector)

        features = np.array(reduced_features)
        predictions = cry_detector.predict(features)
        annotation_count = [0]*len(classes)
        annotation_count[-1] = noise_frames

        i = 0
        cry_features = []
        for prediction in predictions:
            annotation_count[prediction] += 1
            if prediction == 0:
                cry_features.append(features[i])
            i += 1

        vad_output = annotation_count.index(max(annotation_count))
        probability = float(max(annotation_count))/sum(annotation_count)
        time_taken = Util.time_elapsed(start)
        cry_category = detector_labels[vad_output]

        vad_records[vad_output] += 1

        if vad_output == 0:
            cry_predictions = cry_classifier.predict(cry_features)
            cry_output = stats.mode(cry_predictions)[0][0]
            cry_category = classifier_labels[cry_output]
            cry_records[cry_output] += 1

    else:
        time_taken = 0
        probability = float(noise_frames)/frames_per_iteration
        vad_output = detector_labels.index('quiet')
        vad_records[vad_output] += 1
        cry_category = 'quiet'

iteration += 1
frame_index = 0
noise_frames = 0
segment = np.array([], dtype=np.float32)

report = json.dumps({'category': cry_category,
                    'probability': probability,
                    'timeTaken': time_taken,
                    'records': self.__get_record(vad_records, cry_records)})

print('{0:03d}'.format(iteration), '{0}'.format(report))

```

```
except Exception as ex:
    self.__log.e('Exception',ex)

@staticmethod
def __get_record(vad_records, cry_records):
    return {
        'cry':{
            'bellyPain' : cry_records[0],
            'hungry'    : cry_records[1],
            'tired'     : cry_records[2]
        },
        'noise': {
            'cry'       : vad_records[0],
            'noise'    : vad_records[1],
            'quiet'    : vad_records[2]
        }
    }
}
```

