CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	Low-power real-time seizure monitoring via AI-assisted sonification of neonatal EEG
Authors	Nguyen, Tien Van
Publication date	2024
Original Citation	Nguyen, T. V. 2024. Low-power real-time seizure monitoring via AI-assisted sonification of neonatal EEG. MRes Thesis, University College Cork.
Type of publication	Masters thesis (Research)
Rights	© 2024, Tien Van Nguyen. - https://creativecommons.org/licenses/by/4.0/
Download date	2026-11-01 16:16:38
Item downloaded from	https://hdl.handle.net/10468/16958

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Ollscoil na hÉireann, Corcaigh
National University of Ireland, Cork



Low-Power Real-Time Seizure Monitoring via AI-Assisted Sonification of Neonatal EEG

Thesis presented by
Tien Van Nguyen

for the degree of
Master of Engineering Science

University College Cork
School of Engineering and Architecture
Department of Electrical and Electronic Engineering

Head of School/Department: Prof. Jorge Oliveira/Prof. Peter Parbrook
Supervisor(s): Dr. Emanuel Popovici and Prof. Andriy Temko

2024

© 2024 Tien Van Nguyen

TABLE OF CONTENTS

Page

LIST OF FIGURES **iv**

LIST OF TABLES **v**

ACKNOWLEDGMENTS **vi**

PUBLICATIONS **vii**

ABSTRACT **viii**

1 Introduction **1**
1.1 Background 1
1.2 Aim and Scope 5
1.3 Research Contributions 6

2 An AI-assisted Sonification Method for Neonatal EEG **7**
2.1 Overview 7
2.1.1 Algorithm Design 8
2.1.2 The Phase Vocoder 9
2.1.3 Machine Learning for Seizure Detection 11
2.2 Digital Signal Processing Block 13
2.2.1 Pre-processing 13
2.2.2 Denoiser 15
2.2.3 Phase Vocoder 16
2.2.4 Stereo Mixer 18
2.3 AI Block 20
2.4 Discussion 22

3 Sonification of Neonatal EEG in Real-time **23**
3.1 Overview 23
3.2 A Real-time Processing Design 24
3.2.1 Data Storage 25
3.2.2 Pre-processing 26
3.2.3 Phase-Vocoder and Denoiser 28
3.3 Real-time Processing Solutions 31
3.3.1 Digital Filters 31

3.3.2 Short-Time Fourier Transform . . . 32
3.3.3 Denoiser . . . 34
3.3.4 Interpolation . . . 34
3.3.5 Stereo Mixer . . . 35
3.4 Validation . . . 36
3.4.1 Dynamic Time Warping . . . 36
3.4.2 Constant-Q Transform . . . 37
3.4.3 Objective Comparison . . . 38
3.4.4 Subjective Comparison . . . 40
3.5 Discussion . . . 41

4 Neural Networks and Quantization 42
4.1 Quantization: Basic Concepts . . . 42
4.1.1 Scheme Categorization . . . 43
4.1.2 Post Training Quantization . . . 45
4.1.3 Quantization Aware Training . . . 46
4.1.4 Batch Normalization Folding . . . 46
4.2 Hardware Aware Quantization . . . 47
4.3 Cross-Validation . . . 48
4.3.1 Strategies . . . 49
4.3.2 Dataset . . . 49
4.3.3 Data Augmentation . . . 51
4.4 Results and Discussion . . . 51

5 A Low-power Implementation 53
5.1 Hardware Platform . . . 53
5.1.1 MAX78002 . . . 54
5.1.2 Evaluation Kit . . . 55
5.2 Software Implementations . . . 56
5.2.1 Multi-threading . . . 57
5.2.2 Concurrency . . . 57
5.2.3 Task Synchronization . . . 58
5.2.4 Versions . . . 59
5.3 Results and Discussion . . . 61
5.3.1 Speed Performance . . . 61
5.3.2 Energy Consumption . . . 65
5.3.3 Neural Network Power Signatures . . . 67

6 Conclusion 70

LIST OF FIGURES

Page

2.1 Original Architecture 8
2.2 Neural Network Architectures 21

3.1 Real-time AI-assisted Sonification Operates in Complement to Visualization. 24
3.2 Multi-threading Design for Real-time Sonification. 25
3.3 Cascaded Filter Frequency Response 27
3.4 Dynamic Range Compressor 27
3.5 Real-time Design of Phase Vocoder 28
3.6 Real-time Design of Digital Filters 31
3.7 Overlap Add using Circular Buffers 33
3.8 Sum with Delays using Circular Buffer 35
3.9 Similarity matrix between two sets of audio files generated by the real-time sonification algorithm (columns) and the baseline (rows). . 39
3.10 Visual Comparison of Audio Spectrograms: (a & c) Audio outputs of the original algorithm. (b & d) Audio outputs of the real-time algorithm. 40
3.11 DTW Alignment of Audio Spectrograms 41

4.1 Dataset Splitting 50
4.2 Before and After Quantization Performances 52

5.1 MAX78002 Evaluation Kit 55
5.2 Versions Comparison 63
5.3 Total Time Consumption by Number of Channels 64
5.4 Power Measurement Sequence 65
5.5 Power Consumption by Number of Channels 66
5.6 Power Measurement Setup 67
5.7 Neural Network Power Signatures 69

LIST OF TABLES

Page

2.1 Polar Coordinates of Electrodes 19

5.1 DSP Tasks and Subtasks Definitions 62
5.2 Speed Performance of The Real-Time Sonification Algorithm 63

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my main supervisor, Dr. Emanuel Popovici, for providing me with the opportunity to pursue this master's degree. Your guidance, support, and encouragement throughout the project have been invaluable, and I am deeply thankful for your mentorship.

I also wish to extend my sincere thanks to my co-supervisor, Prof. Andriy Temko, for his academic support and mentorship. Your insights and expertise greatly contributed to the successful completion of this research.

I am extremely grateful to Qualcomm Technologies for their generous scholarships, which funded my studies and made this journey possible. My appreciation also goes to the Department of Electrical & Electronic Engineering at the School of Engineering and Architecture, University College Cork for providing the necessary resources and an inclusive environment for my research in the Embedded Systems Group. Additionally, I would like to thank the engineering team at Analog Devices/Maxim for their assistance with their device, which was instrumental to the success of my project.

On a personal note, I would like to thank my family—my parents and sister—for their unwavering love, support, and encouragement throughout my academic journey. Finally, I am grateful to my friends for their constant support and for making this experience a memorable one.

LIST OF PUBLICATIONS

Journal Publications

T. Van Nguyen, A. Daly, S. G. Quintana, F. O'Sullivan, A. Temko and E. Popovici, "Low-Power Real-Time Seizure Monitoring using AI-Assisted Sonification of Neonatal EEG", *IEEE Transactions on Emerging Topics in Computing*, pp. 1-11. doi: 10.1109/TETC.2024.3481035

Conference Publications

T. Van Nguyen, A. Daly, F. O'Sullivan, S. G. Quintana, A. Temko and E. Popovici, "A real-time and ultra-low power implementation of an AI-assisted sonification algorithm for neonatal EEG", *9th International Workshop on Advances in Sensors and Interfaces (IWASI)*, Monopoli (Bari), Italy, 2023, pp. 313-318, doi: 10.1109/IWASI58316.2023.10164463

E. Popovici, S. Gomez-Quintana, A. Daly, **T. Van Nguyen**, F. O Sullivan, A. Factor, A. Temko, "Fast and accurate seizure detection in neonatal EEG using AI-assisted sonification and its implementation on digital technologies", *Presented at the 4th International Congress on Mobile Health and Digital Technology in Epilepsy*, Lausanne, Switzerland, 2023, P22, [Online].
Available https://www.mhdtepilepsy.com/content.aspx?e=1501&c=17454

ABSTRACT

Seizures in newborns are most frequently provoked by an acute brain injury or systemic insult, making them predominantly acute symptomatic in nature. Early identification and intervention are essential to reducing the risks of mortality and long-term disabilities. Continuous electroencephalogram (EEG) monitoring remains the gold standard for accurate neonatal seizure detection. This procedure is expensive, time-consuming, and requires a high level of expertise.

Recent advancements in artificial intelligence (AI) and machine learning (ML) have shown substantial promise in expediting seizure diagnosis. These algorithms can provide decision assistance with expert-level accuracy and thus are increasingly used for EEG analysis. However, the lack of transparency and explainability in the decision-making process of ML algorithms hinders their widespread adoption in medical settings.

Researchers at the Embedded.Systems@UCC Group proposed a novel method that integrates machine learning with sonification, an alternative EEG interpretation approach by converting the signals into sound, to address the black-box problem while simultaneously improving seizure detection performance. The study also found that human listeners achieved better performance using AI-assisted audio analysis compared to using the ML seizure detection algorithm alone.

This study presents a real-time processing design that allows the AI-integrated sonification method to operate in parallel with EEG acquisition. The new design eliminates the operational time delay associated with offline processing in the original study. This is particularly beneficial when the diagnosis is time-critical or in a busy environment where medical professionals have a tight schedule.

A low-power implementation of the proposed real-time system is also presented. This implementation, featuring a system-on-chip with a deep neural networks accelerator, has an average power consumption of 13mW. It can be powered using a battery or a general USB port, allowing for a compact design, suitable for deployment in space-constrained environments like the neonatal intensive care units (NICU).

Chapter 1

Introduction

This chapter presents the general context of neonatal seizures and their effects. Sonification techniques used for analyzing electroencephalograms (EEGs) are also discussed. Following this is a brief introduction to an earlier study to which this study is a follow-up. Finally, the main contributions of this work are summarized.

1.1 Background

According to an estimation by the United Nations Inter-agency Group for Child Mortality Estimation (UN IGME), the average neonatal mortality rate worldwide is 18 per 1000 live births (UNIGME, 2021). This rate is on a decreasing trend, but it varies significantly between nations and is often correlated to the national income level with few exceptions. Newborns in developing countries face a higher risk of mortality compared to those in developed countries, often due to the limitation of resources in neonatal care facilities.

Neonatal seizures are the most frequent neurological condition observed in the

neonatal intensive care unit (NICU) with the majority resulting from an acute brain injury, such as stroke, trauma or brain infection, which is referred to as acute symptomatic (R. A. Shellhaas et al., 2017). They are associated with higher mortality rates, as well as an increased risk of cerebral palsy, epilepsy, and intellectual disability (Pisani, Prezioso, and Spagnoli, 2020). Hypoxic-ischemic encephalopathy (HIE) is the most frequent cause of neonatal seizures (Renée Shellhaas et al., 2020; Chakrapani Vasudevan and Malcolm Levene, 2013), responsible for 15% to 20% of neonatal mortality (Pisani, Spagnoli, et al., 2021). Most of these outcomes can be prevented with optimal care and timely treatment. Acute intervention should be initiated as soon as possible (Renée Shellhaas et al., 2020), optimally within 1 hour of seizure onset (Pavel et al., 2022).

Most seizures in neonates have minimal or no clinical manifestation making them difficult to identify (Ramantani et al., 2019). The gold standard for diagnosis and management of neonatal seizures is continuous electroencephalogram (EEG) monitoring. EEG interpretation requires special expertise and training; the procedure is wholly dependent on the availability of experts. As a result, maintaining 24h EEG monitoring is costly and is unaffordable for most NICUs (Boylan, Nathan J. Stevenson, and Vanhatalo, 2013). Due to its ease of application and interpretation, amplitude-integrated EEG (aEEG)— a simplified compressed trend of EEG amplitude using a limited number of channels—has grown in popularity as a complement to conventional EEG (H. C. Glass, C. J. Wusthoff, and R. A. Shellhaas, 2013). However, aEEG also has limitations, including lower sensitivity for detection, especially with short and low amplitude seizures, and only a restricted area of the brain is monitored.

Artificial Intelligence (AI) and Machine Learning (ML) algorithms—now referred to as AI/ML—have shown great effectiveness in detecting seizures with EEG signals (Boonyakitanont et al., 2020; Shoeibi, Moridian, et al., 2022a; Siddiqui et al.,

2020). However, the adoption of AI/ML in medicine has been limited due to the lack of transparency in the decision-making process of these algorithms (Kelly et al., 2019). Studies have been trying to address this black box problem by introducing explainability to AI/ML, making its decision transparent and understandable and, thus, increasing trust between human and machine (Kundu, 2021), (Amann et al., 2020). On the other hand, experts have also been calling for keeping "human in the loop" (Bakken, 2023; Jotterand and Bosco, 2020; Nakao et al., 2022) to provide contextual understanding, intuition, and ethical considerations for better and reliable decision-making, especially in complex or ambiguous situations, which often arise in healthcare and medicine.

Sonification, defined as "*a technique that uses data as input, and generates sound signals (eventually in response to optional additional excitation or triggering)*" by Hermann, 2008, has been studied as an alternative method for analyzing EEG (Väljamäe et al., 2013), as well as other bio-signals (Aldana Blanco, Grautoff, and Hermann, 2020). This technique can highlight anomalies, trends, or irregularities in time-series signals that might be less apparent in visual plots alone, because human ears are adept at detecting subtle changes and patterns in sound. For seizure detection, sonification effectively simplifies data interpretation, allowing non-expert listeners to have comparable performance to that of trained experts (Gomez-Quintana et al., 2022; Parvizi et al., 2018).

Various techniques were developed to audify EEG signals: parameters mapping such as spectral mapping, distance matrix (Hermann et al., 2002), and event-based mapping (Baier, Hermann, and Stephani, 2007); a voice-like sound synthesizer by using EEG signals as modulators (Chafe and Parvizi, 2018). The phase vocoder (PV)—a well-established technique in speech and audio processing (Liuni and Roebel, 2013; Zolzer, 2011)—was first used for EEG sonification in Andriy Temko et al., 2014. The study showed that untrained listeners using audified EEG

performs better than experienced neonatologists using aEEG alone in identifying seizures.

The time compression/stretching rate might affect the sensitivity of sonification methods (Gómez-Quintana et al., 2018). The dominant frequency of EEG signals evolves over time during seizures, resulting in a change of pitch in the audio. This evolution can be slow and indistinguishable to human ears, hence, speeding up the signals using time-scaling can make this change more apparent. However, all previously mentioned studies consider the time-scale factor as a constant, meaning EEG signals are scaled uniformly in time. Conversely, electrographic seizures are usually rare events and can be very short compared to the duration of the EEG recordings. While a high compression factor causes difficulty in detecting short seizures, a low rate results in a long audio duration, which can cause fatigue and strain on listeners, negatively impacting their decision-making ability.

A method for non-uniform time-scaling sonification, based on the phase vocoder, was proposed by Gomez-Quintana et al., 2022. It leverages a seizure detection algorithm to adjust the compression rate of EEG signals based on the probability of seizure occurrence: EEG segments containing seizures are compressed at a slower rate than non-seizure segments. This approach not only draws listeners' attention to seizure events but also reduces the overall audio duration by using a higher compression rate for non-seizure sections. Consequently, this method is referred to as the AI-assisted sonification algorithm. The length of synthesized audio varies depending on the seizure occurrence frequency and typically ranges between a few seconds to half a minute for every hour of EEG signal.

1.2 Aim and Scope

Sonification offers a valuable tool for EEG interpretation, serving as a complement to traditional visualization methods. This study aims to develop a low-power assistant application for seizure identification using the AI-assisted sonification algorithm. This implementation can be powered by batteries or standard USB ports, making it compact and compatible with a variety of EEG acquisition equipment commonly used in neonatal intensive care units (NICUs) without the need for additional setup. Operating near the data source not only mitigates data privacy risks but also conserves power by reducing the need for data transmission.

Microcontrollers are the ideal hardware for this application. In fact, this type of device is getting more popular for Internet of Things (IoT) machine learning applications in healthcare (Diab and Rodriguez-Villegas, 2022), which has been dominated by Raspberry Pi (Amin and Hossain, 2021). This is due to the recent advances in low-power hardware and supporting software (Immonen and Hämäläinen, 2022). Furthermore, their low production cost also makes the system more accessible in underdeveloped regions where expertise and equipment are scarce.

An earlier implementation of this algorithm using a Raspberry Pi was proposed by O'Sullivan et al., 2022. In the same study, a "lite" algorithm was also introduced. However, the device is not suitable for a battery-powered application nor able to run using typical USB ports available on most EEG acquisition systems.

So far, the AI-assisted sonification algorithm is designed to process data only in an offline fashion. This means that the sonification process requires the complete dataset before it can begin and must be restarted each time it finishes. Additionally, the algorithm takes approximately seven minutes to sonify one hour of EEG data on a Raspberry Pi 3B+. This delay can hinder the timely identification

of seizures, thereby reducing the algorithm's practicality in a busy medical environment. To address this issue, a potential solution is to adapt the AI-assisted sonification algorithm to run in real-time, processing data in parallel with EEG acquisition.

1.3 Research Contributions

The novel aspects of this study can be listed into three main contributions:

1. A real-time design for the AI-assisted sonification algorithm is introduced. Differing from the original design that requires EEG signals to be available at the time of sonification, the real-time design produces audio continuously and in parallel with data acquisition. Medical professionals can have access to audio analysis without any waiting periods, saving their time and effort. The new design is verified against the original design with both objective and subjective methods.

2. A low-power implementation of this algorithm is also presented. The hardware platform used in this study is a system-on-chip, featuring an Arm Cortex-M4 processor, a 32-bit RISC-V coprocessor, and a low-power neural network accelerator. While the sonification tasks are performed using the Arm Cortex-M4 processor, the seizure detection algorithm runs on the dedicated accelerator for deep neural networks.

3. This study also performs an analysis, regarding performance, power consumption, and inference speed, of two neural networks for low-power seizure detection. The effects of quantization on these algorithms are also discussed.

Chapter 2

An AI-assisted Sonification Method for Neonatal EEG

This chapter explains the original design of the AI-assisted sonification algorithm. It begins with a review of the phase vocoder technique and various machine learning methods for seizure detection. Finally, it outlines the architecture of two patient-independent classifiers for neonatal seizures, which are utilized in both the original study and the current research.

2.1 Overview

The AI-assisted sonification algorithm proposed by Gomez-Quintana et al., 2022 relies on the phase vocoder (PV) to map EEG signals into the audible frequency range. A machine learning algorithm was used to provide context on the EEG signals. In this section, the overall design of this algorithm is presented, followed by the basics of the phase vocoder and a brief review on AI/ML algorithms for seizure detection.

2.1.1 Algorithm Design

The system design of the AI-assisted sonification algorithm is illustrated in Fig. 2.1. Its two main blocks are defined by their tasks:

1. The Digital Signal Processing (DSP) block comprises a series of signal processing operations that converts EEG signals into sound. This block can be understood as the implementation of the sonification algorithm.

2. The AI block is an automated seizure detection algorithm. It works similar to an attention mechanism, providing probability of seizures on each EEG channel as context for time scaling and attenuation tasks done by the DSP block.

The input to this algorithm is multichannel EEG, following the 10-20 system (Jasper, 1958). It outputs stereo sound (two channels, left and right) with an adaptable duration, which depends on the frequency of seizure occurrence. Typically, the audio generated from one hour of EEG lasts between a few seconds to half a minute.

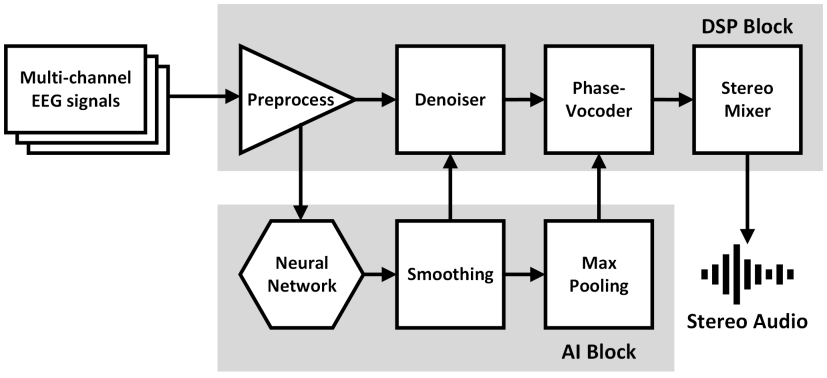


Figure 2.1: Original Architecture

This algorithm is highly adaptable. It accepts an arbitrary number of EEG channels and allows freedom of choice for the AI block. However, one should consider

the hardware limitations as well as the application for a suitable configuration.

2.1.2 The Phase Vocoder

Although the term *phase vocoder* was first coined in J. L. Flanagan and Golden, 1966, the now classical form of PV employing short-time Fourier transform (STFT) analysis and synthesis was introduced later in Portnoff, 1981. This section adopts the notation used by Laroche and M. Dolson, 1999.

The phase vocoder is composed of three steps: analysis, transformation, and re-synthesis. During analysis, a short-time Fourier transform (STFT) is performed over the input signal.

$$X(t_a^u, \Omega_k) = \sum_{n=-\infty}^{\infty} h(n)x(t_a^u + n)e^{-j\Omega_k n}, \tag{2.1}$$

where x and $X(t_a^u, \Omega_k)$ represent the input signal and its STFT representation respectively, t_a^u denotes the analysis time-instants, $h(n)$ indicates the analysis window of length N, and $\Omega_k = \frac{2\pi k}{N}$ is the center frequency of the kth "channel". With an analysis time step (hop size) R_a, then $t_a^u = uR_a$, where the integer u indicates the time index.

Transformations in the time-frequency domain (e.g., phase propagation) of $X(t_a^u, \Omega_k)$ are then performed to yield the synthesis STFT $Y(t_s^u, \Omega_k)$.

$$y(n) = \sum_{u=-\infty}^{\infty} \omega(v - t_s^u)y_u(n - t_s^u) \tag{2.2}$$

$$y_u(n) = \frac{1}{N}\sum_{k=0}^{N-1} Y(t_s^u, \Omega_k)e^{j\Omega_k n} \tag{2.3}$$

Finally, the synthesis time instants t_s^u are given by $t_s^u = uR_s$, where R_s denotes

the synthesis hop size. At each of these time instants, a short-time signal $y_u(n)$ is obtained from the inverse Fourier-transform of the short-time spectra $Y(t_s^u, \Omega_k)$. The output signal $y(n)$ is synthesized by the overlap-add procedure of these short-time segments, weighted by an optional synthesis window $\omega(n)$.

Time-scale Modification

The classical way of using a phase vocoder for time-scaling is to set different hop sizes during analysis and synthesis $R_a \neq R_s$ (Mark Dolson, 1986). This means the synthesis STFT frames are spaced "further/closer" apart than the analysis STFT frames. The time scaling factor α will be defined as

$$\alpha = \frac{R_s}{R_a} \tag{2.4}$$

While the magnitude components of the STFT can simply be kept unchanged, the synthesis phase is constructed by the recursive phase propagation (integration) formula

$$|Y(t_s^u, \Omega_k)| = |X(t_a^u, \Omega_k)|, \tag{2.5}$$

$$\angle Y(t_s^u, \Omega_k) = \angle Y(t_s^{u-1}, \Omega_k) + R_s \omega X(t_a^u, \Omega_k), \tag{2.6}$$

where $\omega X(t_a^u, \Omega_k)$ is the instantaneous frequency of the kth channel, given by

$$\omega X(t_a^u, \Omega_k) = \Omega_k + \frac{1}{R_a}\left[\angle X(t_a^u, \Omega_k) - \angle X(t_a^{u-1}, \Omega_k) - R_a \Omega_k\right]_{2\pi}, \tag{2.7}$$

$[\cdot]_{2\pi}$ denotes the phase *unwrapping* procedure.

Phase-scale Modification

The typical approach to pitch-scale modification using a phase vocoder is to perform time-scaling by the desired factor α, followed by resampling with a new sampling period $\alpha\Delta T$, where ΔT is the original sampling period. The result is a new signal with pitch multiplied by α but having the same duration as the input signal.

Phasiness Problem

Because phase propagation errors are at the heart of many sound quality issues, the correction of the STFT phase is important and also the main challenge in the phase vocoder.

The classical implementation of PV, described above, treats each frequency channel as a separate sinusoid whose phase is computed by accumulating the estimate of its instantaneous frequency and thus preserving the *horizontal phase coherence*.

This approach produces specific artifacts referred to as "*phasiness*" which are generally attributed to the loss of *vertical phase coherence*. A modification of PV was proposed by Laroche and M. Dolson, 1999 addressing this problem. Different improvements were also introduced in later studies (Prusa and Holighaus, 2022).

2.1.3 Machine Learning for Seizure Detection

A variety of machine learning algorithms have been employed in EEG-based seizure detection, categorized into traditional and deep learning methods:

- Traditional Methods: These include algorithms such as Random Forest (RF)

(Mursalin et al., 2017; Wang et al., 2019), Support Vector Machine (SVM) (Shoeb and Guttag, 2010; Subasi, Kevric, and Abdullah Canbaz, 2019; A. Temko et al., 2011; Zhang and Chen, 2017), and K-Nearest Neighbors (KNN) (Mardini et al., 2020). Traditional methods typically involve a feature extraction step prior to classification.

- Deep Learning (DL) Approaches: Popular deep learning models include convolutional neural networks (CNNs) (Nogay and Adeli, 2021; Thuwajit et al., 2022; Zhou et al., 2018), generative adversarial networks (GANs) (Truong et al., 2019; You et al., 2020), recurrent neural networks (RNNs) (Golmohammadi et al., 2017; Petrosian et al., 2000), and autoencoders (AEs) (Abdelhameed, Daoud, and Bayoumi, 2018; Shoeibi, Ghassemi, et al., 2022). CNNs, both 1D and 2D, are the most widely used deep learning architecture in epileptic seizure detection, featured in more than half of related studies (Shoeibi, Moridian, et al., 2022b).

Seizure detection using machine learning approaches can also be categorized based on the type of model used: patient-specific and patient-independent. Most studies follow the patient-specific approach, which tailors the classifiers to individual patients by training on their specific data. However, this approach suffers from data scarcity, as it relies solely on individual patient data. Consequently, patient-specific models may exhibit poor generalization ability. In contrast, patient-independent models aim to create a universal model that can predict seizures across multiple patients. They pool data from various subjects during training. Unfortunately, patient-independent models often perform worse due to significant individual variations across patients.

2.2 Digital Signal Processing Block

The Digital Signal Processing (DSP) block combines all DSP tasks in the AI-assisted sonification algorithm. This section goes into the details of its individual tasks in the same order as in operation: Pre-Processing, Denoiser, Phase Vocoder, and Stereo Mixer.

2.2.1 Pre-processing

The EEG signals picked up from the scalp tend to contain a lot of noise and artifacts which can be classified into two categories: physiological and nonphysiological (White and Cott, 2010). Physiological artifacts originate from sources in the body such as muscles (especially face and neck muscles), the eyes (during blinks and the movements of the eyeballs), and the heart. Other biometric data, for example, electrooculogram (EOG) or electrocardiogram (ECG), if available, can be used to remove the interferences from EEG signals with adaptive filtering (P. He, Wilson, and Russell, 2004; Kim, 2018). Nonphysiological artifacts are generated from sources outside the body. The common sources of this type include line noise (the alternating current supply to the electrical wall outlets), loose electrode contact, and cable movement.

Pre-processing aims to obtain "clean" and relevant neural signals from the EEG data, often by utilizing various DSP techniques. In Gomez-Quintana et al., 2022, this step begins with filtering:

- A 0.5Hz cutoff high-pass filter to reduce very slow oscillations or drifts in the data caused by body sway or skin potentials.

- A 12.8Hz cutoff low-pass filter to separate the frequency region of seizure manifestation in neonatal EEG which is between 0.5 and 13 Hz and predominantly between 0.5 and 6 Hz (Kitayama et al., 2003).
- A 50Hz notch filter to further attenuate line noise in the EEG data which was recorded in Europe with a 50Hz power supply.

Adaptive filtering was used to remove the interference by the heart's electrical activities. By assuming this interference in the EEG channels to be an attenuated and delayed version of the provided ECG channel recording, a parametric model can be derived:

$$y[n] = \hat{x}[n] + \alpha \cdot r[n - \delta]$$

where $y[n]$ represents corrupted EEG signal, $\hat{x}[n]$ is the estimation of true EEG signal, $r[n]$ is ECG reference signal, and (α, δ) are the model parameters.

To reduce the memory consumption and computational load, EEG signals are then down sampled from 256Hz to 32Hz. There was no need for an additional anti-aliasing filter since the new sampling frequency is higher than twice the highest frequency (Nyquist theorem) in the EEG signals after the low-pass filter.

Next, a dynamic range compressor (DRC) (Giannoulis, Massberg, and Reiss, 2012) is applied to remove any big bursts that may exist due to artifacts or electrode disconnections. The DRC reduces the level of signal that crosses a given threshold T_{dB} (in decibel). A ratio R determines the amount of compression applied. The gain signal $g_{dB}(t)$ (in decibel) is given by:

$$g_{dB}(t) = \begin{cases} 0, & e_{dB}(t) < T_{dB} \\ \frac{R-1}{R}(T_{dB} - e_{dB}(t)), & \text{otherwise,} \end{cases} \tag{2.8}$$

where $e_{dB}(t)$ (in decibel) is an ADSR (attack, decay, release, and sustain) envelope of the input signal $x(t)$. The output signal, $y(t)$, is the element-wise product of the gain and input signals: $y(t) = x(t) \odot g(t)$.

Finally, an electrographic seizure has a minimum 2 μV pp voltage and can be up to 150 μV pp (T. N. Tsuchida et al., 2013). Limiting the magnitude of the signals in this range prevents outliers from skewing the signals after normalization.

2.2.2 Denoiser

The Denoiser is based on an acoustic noise suppression technique introduced in Boll, 1979. It attenuates the spectral magnitude of non-seizure (background) EEG regions, thus highlighting seizure regions.

First, the short-time Fourier transform (STFT) of the entire available EEG signal is computed. The STFT matrix is defined as: $\mathbf{X}[m,k] = \text{STFT}\{x[n]\}$. Next, the background spectral profile, $T[k]$, is estimated as the mean Power Spectral Density (PSD) $\mathbf{P}[m,k] = \mathbf{X}[m,k]^2$ of all background regions:

$$T[k] = \sum_m \mathbf{P}[m,k] \frac{1 - p[m]}{\sum_m (1 - p[m])}, \tag{2.9}$$

where $1 - p[m]$ is the complementary seizure probability for the m^{th} column. The "denoised" STFT matrix, $\mathbf{Y}[m,k]$, is then obtained by applying a gain function $\mathbf{G}[m,k]$:

$$\mathbf{G}[m,k] = \begin{cases} 1, & \text{if } \mathbf{P}[m,k] \geq T[k] \\ \frac{\mathbf{P}[m,k]}{T[k]}, & \text{otherwise} \end{cases} \tag{2.10}$$

$$\mathbf{Y}[m,k] = 10^{p[m]-1} \cdot \mathbf{G}[m,k] \cdot \mathbf{X}[m,k] \tag{2.11}$$

In Eq. (2.11), $\mathbf{G}[m,k] \leq 1 \quad \forall(m,k)$ and $10^{p[m]-1}$ are two attenuation factors that reduce the amplitude of the background regions. Finally, the "denoised" time-series EEG signals are obtained by calculating the inverse-STFT of $\mathbf{Y}[m,k]$.

2.2.3 Phase Vocoder

The attenuated signals are then non-uniformly scaled in the time domain by the phase vocoder. Like the previous step, the scaling factor is decided based on the classification result of the automated seizure detection algorithm (the AI block); unlike the previous step, the PV considers the maximum seizure probability across all EEG channels; this is to ensure that the regions in EEG signals are aligned correctly.

Time Scale Modification

Besides the classical approach, the number of STFT frames can also be manipulated, typically by interpolation/decimation, in order to time-scale a signal (Fischman, 1997).

Assume that $X(t_a^m, \Omega_k)$ represents the STFT representation of a finite length input signal x, where $0 \leq m \leq M-1$ is the analysis time-index and M is the total number of analysis time-instants.

Let $Y(t_s^n, \Omega_k)$ be the synthesis STFT, where $0 \leq n \leq N-1$ denotes the synthesis time-index and N is the total number of synthesis time-instants. When the analysis and synthesis hop sizes are the same ($R_a = R_s$), the average time scaling factor, α, is therefore given by

$$\alpha = \frac{N}{M} \tag{2.12}$$

The synthesis STFT magnitude $|Y(t_s^n, \Omega_k)|$ can be obtained by linear interpolation:

$$|Y(t_s^n, \Omega_k)| = |Y(t_s^{n-1}, \Omega_k)| + \frac{1}{\alpha}\frac{\Delta}{\Delta m}|X(t_a^m, \Omega_k)|, \quad \lfloor \alpha m \rfloor \leq n < \lfloor \alpha(m+1) \rfloor \quad (2.13)$$

where $\frac{\Delta}{\Delta m}|X(t_a^m, \Omega_k)| = |X(t_a^m, \Omega_k)| - |X(t_a^{m-1}, \Omega_k)|$ is the difference in magnitude between two consecutive frames.

Since $R_s = R_a$, Eqs. (2.6) and (2.7) can be re-written as follows:

$$\angle Y(t_s^n, \Omega_k) = \angle Y(t_s^{n-1}, \Omega_k) + \omega X(t_a^m, \Omega_k), \quad \lfloor \alpha m \rfloor \leq n < \lfloor \alpha(m+1) \rfloor \quad (2.14)$$

$$\omega X(t_a^m, \Omega_k) = \left[\angle X(t_a^m, \Omega_k) - \angle X(t_a^{m-1}, \Omega_k)\right]_{2\pi} \quad (2.15)$$

As in Section 2.1.2, the horizontal phase coherence is preserved:

$$\omega Y(t_s^n, \Omega_k) = \angle Y(t_s^n, \Omega_k) - \angle Y(t_s^{n-1}, \Omega_k) = \omega X(t_a^m, \Omega_k)$$

where $\omega Y(t_s^n, \Omega_k)$ is the synthesis instantaneous frequency of the k^{th} channel. This reflects the characteristic of PV that keeps frequency content unchanged while modifying the signal's time-scale.

Phase Correction

Equation (2.14) contains a discontinuity at $n = \lfloor \alpha m \rfloor$ as the instantaneous frequency changes from $\omega X(t_a^m, \Omega_k)$ to $\omega X(t_a^{m+1}, \Omega_k)$. This issue is not relevant when α is small, however, when $\alpha >> 1$, it can introduce artifacts. To ensure a "smooth" transition of the instantaneous frequency, an additional term was

added to Eq. (2.14):

$$\angle Y(t_s^n, \Omega_k) = \angle Y(t_s^{n-1}, \Omega_k) + \omega X(t_a^m, \Omega_k) + (\frac{n}{\alpha} - m)\frac{\mathrm{d}}{\mathrm{d}m}\omega X(t_a^m, \Omega_k),$$
$$\lfloor \alpha m \rfloor \leq n < \lfloor \alpha(m+1) \rfloor \quad (2.16)$$

By taking into account the first derivative of the instantaneous frequency $\frac{\mathrm{d}}{\mathrm{d}m}\omega X(t_a^m, \Omega_k)$, the synthesis phase $\omega Y(t_s^n, \Omega_k)$ now has second-order continuity.

Variable Scaling Factor

The average scaling factor, α, is calculated by dividing the lengths of the input and output signals (Eq. (2.12)). The instantaneous factor, α_a^m, however, is a variable and might take a different value at different time stamps. This allows for a non-linear mapping from t_a^m to t_s^n. α_a^m is given by:

$$\alpha_a^m = \frac{F_{out}}{F_{in}} \frac{1}{V_{min}^{P_m} V_{max}^{1-P_m}} \quad (2.17)$$

where $F_{in} = 32Hz$ and $F_{out} = 16000Hz$ are the sampling frequencies of the input and output signals, respectively, $V_{min} = 60$ and $V_{max} = 3600$ are the minimum and maximum compression speeds, and P_m is the maximum seizure probability across all available EEG channels at the m^{th} time-instant.

2.2.4 Stereo Mixer

Each EEG channel produces its audio output separately, resulting in N audio signals for N EEG channels. Based on the interaural time delay (ITD) precedence effect in (A. D. Brown, Stecker, and Tollin, 2014), a model that maps N audio channels into two (left and right) stereo channels was proposed. The audio sources

of the N channels are situated on a 2D plane with an imaginary listener whose left and right ears (L and R) are situated at $(-H, 0)$ and $(H, 0)$, where H is the distance of each ear from the center of the head (the origin of coordinates). The transmission delays of sound sources to the ears can be obtained as:

$$d_{L/R} = \frac{1}{c}\sqrt{\rho^2 + H^2 \pm \rho H \sin\theta} \tag{2.18}$$

where the head's radius $H = 0.1m$ and the speed of sound in air at room temperature is $c = 340m/s$.

Table 2.1: Polar coordinates of electrodes on the left-hemisphere. The right hemisphere pairs have the same values with a negative sign on θ.

8 channels										
chan	F3–C3	C3–O1	T3–C3	C3–Cz						
ρ/H	5.2	6.3	7.4	2.6						
θ	60°	140°	90°	90°						
18 channels										
chan	Fp1–F3	F3–C3	C3–P3	P3–O1	Fp1–F7	F7–T3	T3–T5	T5–O1	Fz–Cz	Cz–Pz
ρ/H	8.0	5.1	5.1	8.0	9.5	9.6	9.6	9.5	2.4	2.4
θ	30°	60°	120°	150°	35°	70°	110°	145°	0°	180°

The Euclidean distance and the azimuthal angle form a set of coordinates, (ρ, θ), specific to each audio source on the 2D plane. Table 2.1 shows these position values which are proportional to the actual physical positions of the electrode pairs of the 8-channel and 18-channel EEG montages.

$$S_{L/R}[t] = \sum_{ch=[1:N]} a_{ch}\left[t - d_{L/R}\right] \tag{2.19}$$

Left and right stereo channels $S_{L/R}[t]$ are calculated as the sum of N audio channels with their corresponding delays $d_{L/R}$.

2.3 AI Block

The AI block operates an EEG-based machine learning seizure detection algorithm. It provides the probability of seizure occurrence on all EEG channels. These outputs are used at different stages of the DSP block: to determine the signal's amplitude gain during the Denoiser, using channel-wise probabilities, and to compute the re-sampling rate of the STFT matrix during the Phase Vocoder, using the cross-channels maximum probability.

There is flexibility in the choice of the seizure detection algorithm for the AI block. Any EEG-based algorithm that provides seizure classification on each EEG channel can potentially be used. However, since there is no easy way to train the model after the sonification algorithm goes into operation, a patient-independent algorithm is preferred.

Figure 2.2 depicts two convolutional neural networks for patient-independent seizure detection. The first model, referred to as VGGNet, is a fully convolutional neural network (FCNN) introduced in O'Shea et al., 2020. Its architecture is based on another neural network for computer vision introduced in Simonyan and Zisserman, 2015. The VGGNet has three feature extraction blocks, followed by a fully convolutional classification block. Each feature extraction block contains three convolution layers with activation (ReLU), normalization, and pooling.

The second model is referred to as ResNet, based on an architecture in K. He et al., 2015. This particular model is proposed in Daly et al., 2021, having an increased receptive field and the ability to learn longer temporal seizure characteristics. It contains an extra feature extraction block and achieved a relative improvement of 21.9% in AUC for EEG-based neonatal seizure detection, compared to VGGNet.

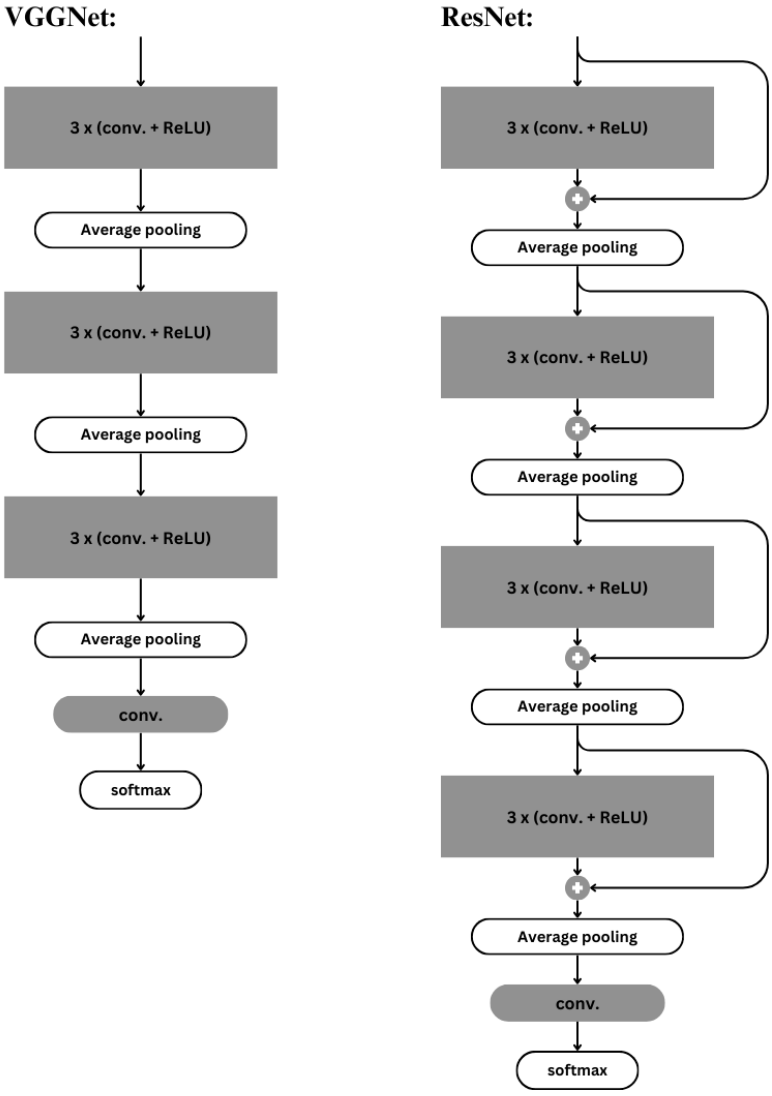


Figure 2.2: Neural Network Architectures

Both neural networks above do not utilize fully-connected (linear) layers for the classification block, instead, a convolution layer is used. This is to prevent the models from learning the cross-channel information, which is highly patient-dependent. As a result, the model is forced to generalize and learn cross-patient features.

The VGGNet was implemented in the original AI-assisted sonification algorithm. Consequently, it is used in validation of a new algorithm design presented in the following chapter. This study analyzes the performances of the two neural networks in a low-power setting, where the effects of quantization on these models will be discussed.

2.4 Discussion

In Gomez-Quintana et al., 2022, an online survey (`http://sergigomezquintana.github.io/EEGsoundSurvey`) where 23 participants were asked to identify seizures using synthesized audios from 79 multichannel EEG recordings (Nathan J Stevenson et al., 2019) was carried out. The surveyed answers showed the same accuracy with the annotations given by experienced EEG specialists who used a full continuous video EEG setup while outperformed that of the ML seizure detection algorithm.

The original AI-assisted sonification algorithm discussed in this chapter works in an offline fashion. This approach requires pre-collected EEG signals and introduces processing time delays, which, if running on a Raspberry Pi, are around seven minutes O'Sullivan et al., 2022. These waiting periods can compromise the algorithm's practicality in clinical settings.

Chapter 3

Sonification of Neonatal EEG in Real-time

A new design to enable the real-time processing capability for the AI-assisted sonification algorithm is proposed in this chapter. To confirm its consistency with the original design, different validation methods, both objective and subjective, were performed.

3.1 Overview

This chapter presents a real-time (online) design for the AI-assisted sonification algorithm to address the drawbacks associated with the original offline processing approach. The new design audifies EEG in parallel with acquisition and generates output accumulatively. Medical professionals are provided with continuous audio analysis in complement to conventional analysis methods such as visualization (Fig. 3.1).

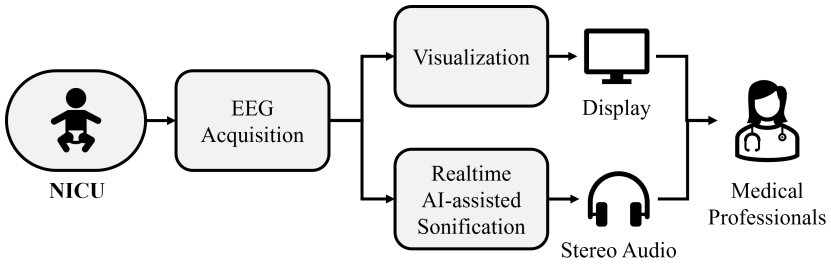


Figure 3.1: Real-time AI-assisted Sonification Operates in Complement to Visualization.

In a real-time (or streaming) DSP system, the analyzed (input) and generated (output) samples can be processed (or generated) continuously in the time it takes to input and output the same set of samples independent of the processing delay. In other words, the mean processing time per sample, including overhead, must not exceed the sampling period, which is the reciprocal of the sampling rate (Kuo, Lee, and Tian, 2006). For EEG, typically sampled at 256Hz, this time limit is $\frac{1}{256}$ seconds.

The adaptation of the sonification algorithm for real-time processing includes changes in its system design and mathematical equations. As a result, a verification step is needed to validate the new algorithm. Subsequent sections will go into detail of these modifications and validation.

3.2 A Real-time Processing Design

Parallelization is a crucial technique to reduce overall computation time and enable more efficient processing. Figure 3.2 presents a multi-threading design for the AI-assisted sonification algorithm where its tasks are set up for concurrent and real-time operation.

The original DSP block is divided into two parallel tasks (or *threads*): "Pre-

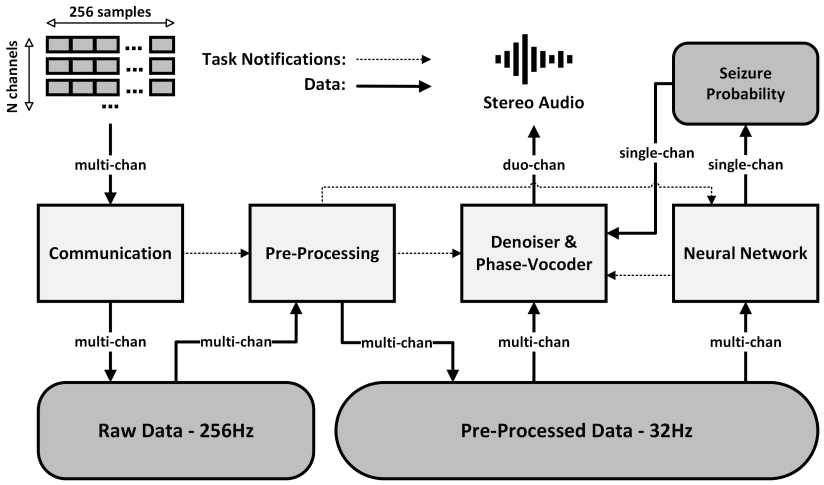


Figure 3.2: Multi-threading Design for Real-time Sonification.

processing", performing the pre-processing step, and "Denoiser & Phase Vocoder", combining two steps, the Denoiser (Section 2.2.2) and the Phase Vocoder (Section 2.2.3). The "Neural Network" thread operates a seizure detection neural network (or the AI block). This design handles streaming input in the "Communication" thread, which stores incoming data into circular buffers.

3.2.1 Data Storage

There are three main circular buffers that are used for data storage in this study:

1. `raw_cb` stores raw EEG at 256Hz. This buffer is populated by the Communication task (`comm_t`), while its data is accessed and read by the Pre-processing task (`pre_processing_t`).

2. `processed_cb` stores pre-processed and down-sampled EEG data at 32Hz. Its data is filled by `pre_processing_t` and can be accessed by the Phase Vocoder (`phase_vocoder_t`) and Neural Network (`neural_network_t`) tasks.

3. `prob_cb` stores the seizure classification results from the neural network accelerator. `phase_vocoder_t` can access these probabilities, either by channel

or channel-wise maximum.

Access to the main circular buffers is shared between tasks but limited to those that either produce or consume the stored data. The implementation of this sharing mechanism will be detailed in Section 5.2.3. Besides the shared circular buffers, each task also contains a number of static and dynamic data structures for intermediate data storage.

3.2.2 Pre-processing

This thread implements the DSP tasks outlined in Section 2.2.1. The "ECG removal" step is not included in this study to reduce the complexity of the implementation. Instead, the EEG recordings are cleaned of ECG interference and stored prior to the start of this algorithm.

The pre-processing thread begins with an attempt to read one segment (a matrix of $N \times 256$ samples) from the `raw_cb` buffer. If there are not enough data available, it enters sleep until being notified. On the other hand, if the read succeeds, the thread continues by performing the following steps:

1. Filtering: A 64th-ordered filter (68 coefficients) results from cascading the high-pass, low-pass, and notch (band-stop) filters. The frequency response of the cascaded filter is shown in Fig. 3.3.

2. Down-sampling: Simple decimation is performed to down-sample filtered data from 256Hz to 32Hz. It keeps every 8th sample and discards the rest. The data segment after this stage has the size of 32 samples. Average pooling can also be used as an alternative method for down-sampling.

3. Compression: The output ($y(t)$) of the DRC is the element-wise product of

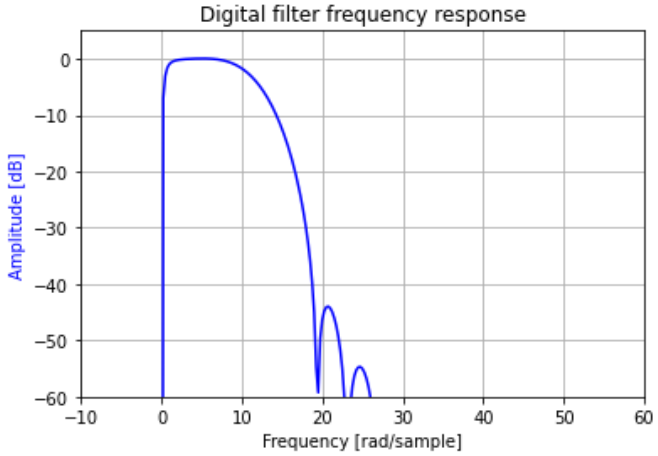


Figure 3.3: Cascaded Filter Frequency Response

a delayed version of the input signal ($x(t + delay)$) and a gain signal ($g(t)$), which is determined by an ADSR envelope ($e(t)$) instead of the input signal directly (Fig. 3.4).

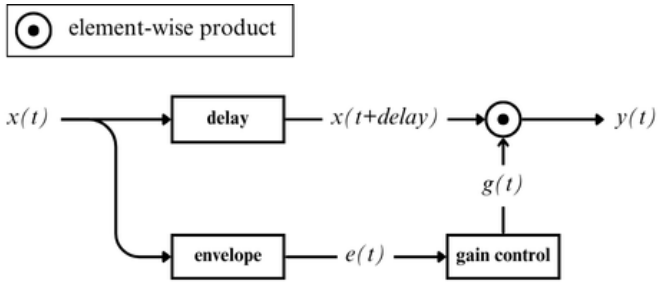


Figure 3.4: Dynamic Range Compressor

Both $x(t + delay)$ and $e(t)$ are obtained by convolution. This is the same as applying two FIR filters. While the filter coefficients for the envelope function are derived from its attack (A), decay (D), release (R), and sustain (S) parameters, the filter coefficients for delay all equals zero, except for the index at $t - delay$ being equal to 1.

Finally, the *pre-processed* output (a matrix of $N \times 32$ samples) is stored in the `processed_cb` circular buffer. Section 3.3.1 presents a custom implementation of filters in real-time processing.

3.2.3 Phase-Vocoder and Denoiser

Originally, the Denoiser and Phase-Vocoder were two separate and sequential steps (see Chapter 2). However, both operations are performed in the frequency domain by computing the Short-Time Fourier Transform (STFT) for the entire EEG signal. This leaves room for optimization.

In this study, these two tasks are integrated into the `phase_vocoder_t` thread. This integration significantly reduces the number of Fourier transforms required, thereby decreasing the overall computational load. This optimization is especially critical for embedded platforms, where computational resources are limited.

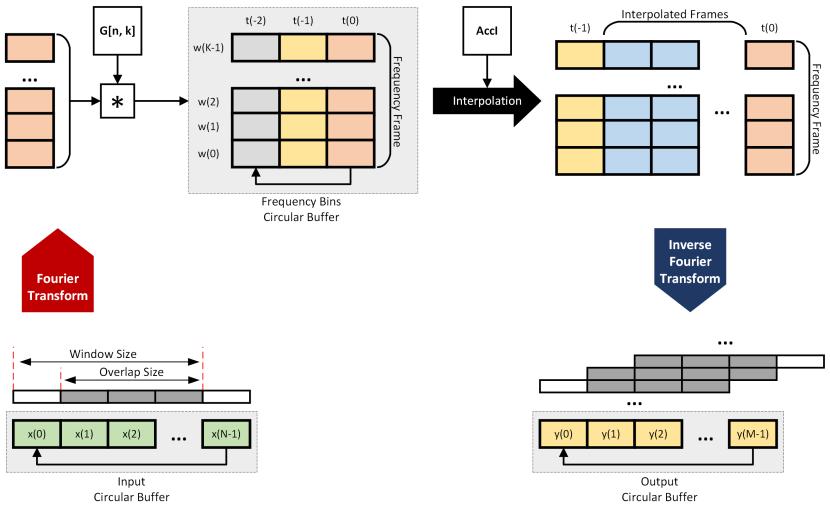


Figure 3.5: Real-time Design of Phase Vocoder

An overview to the operation of `phase_vocoder_t` is given in Fig. 3.5. This thread uses an analysis window of four seconds in length (or $N \times 128$ samples); each window has an overlap of three seconds (or $N \times 96$ samples). The input data segment is analyzed in the frequency domain, which involves magnitude attenuation and interpolation. The newly interpolated frequency frames are then used to synthesize the output data.

Data Retrieval

First, this thread attempts to read data from two circular buffers: `prep_cb`, which contains processed data, and `prob_cb`, which contains seizure probabilities. If both read attempts are successful, `phase_vocoder_t` then performs a fast Fourier transform to convert read data into the frequency domain as formularized in Eq. (3.4). However, if either read attempt fails due to insufficient data in the buffer, `phase_vocoder_t` enters a sleep state. The thread can only be reactivated by the other threads responsible for writing data to the buffer, which are `pre_processing_t` for `prep_cb` and `neural_network_t` for `prob_cb`.

Denoiser

Next, the Denoiser (see Section 2.2.2 and Section 3.3.3 for the real-time implementation) transforms the newly computed frequency bin into the "denoised" frequency frame $X_m^d[k]$. The three most recent columns of the STFT matrix ($X_{-2:0}^d[k]$) are stored using the `STFTMatrixCB` circular buffer.

Interpolation

The interpolation processes of the magnitude $\left|X_m^d[k]\right|$ and phase $\angle X_m^d[k]$ components are performed separately (Listing 3.1). While the synthesis magnitude $|Y_n[k]|$ can be simply derived from interpolating $\left|X_m^d[k]\right|$, the synthesis phase $\angle Y_n[k]$ is obtained via the unwrapped instantaneous frequency $[\omega X_m^d[k]]_{2\pi}$, to preserve the phase coherence in the interpolated frames (see Section 2.2.3). The solution for real-time interpolation is explained in Section 3.3.4.

The new indexes, I_0 and I_{-1}, of current and last data points are updated by

```
if intCount(I_{-1}, I_0) < 1:
    continue
else:
    |Y| = linearInterp(|X^d_{-1}|, |X^d_0|, I_{-1}, I_0) # Interpolate Magnitude
    ωX = difference(∠X^d_{[-2:0]}) # Instantaneous Frequencies
    ωX = phaseUnwrap(ωX) # Phase Unwrapping
    ωY = linearInterp(ωX_{-1}, ωX_0, I_{-1}, I_0) # Interpolate Phase
    ∠Y = cumsum(ωY) # Cumulative Sum
I_{-1} = I_0 − ⌊I_0⌋ # Update I_{-1}
I_0 = I_{-1} + I_tmp # Update I_0
```

Listing 3.1: Real-time Interpolation

cumulative summing a temporal I_{tmp} interpolation factor, which is derived from variables defined in (Gomez-Quintana et al., 2022):

$$I_{tmp} = \frac{F_{out}}{F_{in} \cdot V_{max} \left(\frac{V_{min}}{V_{max}}\right)^{P}} \tag{3.1}$$

where $F_{in} = 32Hz$ and $F_{out} = 16kHz$ are the sampling rates of EEG input (to the Phase Vocoder) and audio output signals; $V_{max} = 3600$ and $V_{min} = 60$ are max and min compression rates, meaning in one second of audio compressed between one minute to one hour of EEG signal; and P is the maximum seizure probability across all channels. Given P ranges between 0 and 1, the temporal interpolation factor I_{tmp} ranges between ~ 0.139 and ~ 8.333.

Audio Synthesis

The audio signals corresponding to the individual channels of EEG are obtained using inverse short-time Fourier transform (iSTFT) with overlap-add (OLA) method (see Section 3.3.2). These signals are not stored using intermediate memory but directly mixed, by adding, into the two circular buffers (left and right stereo channels) for the final stereo audio (see Section 3.3.5).

3.3 Real-time Processing Solutions

The original design utilizes many DSP operations that are only applicable in offline processing, and thus cannot be directly implemented in a real-time system. This section details the approaches to implement these operations in this study.

3.3.1 Digital Filters

The output of a filter is the convolution of the input signal and past outputs with its respective coefficient vectors (**B** and **A**). In Fig. 3.6, past values of input and output, denoted with negative indices (-1, -2, etc.), are stored in two circular buffers. After the operation, the current input (x_0) and output (y_0) samples are stored again in the circular buffers for the next operation.

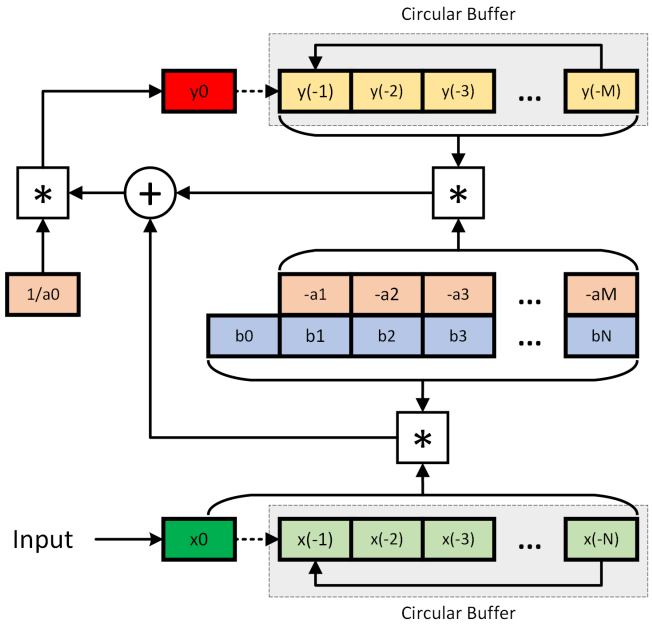


Figure 3.6: Real-time Design of Digital Filters

3.3.2 Short-Time Fourier Transform

The short-time Fourier transform (STFT) of a signal is computed by sliding an analysis window $g(n)$ of length M over the signal and calculating the discrete Fourier transform (DFT) of each segment of windowed data. The window hops over the original signal at intervals of R samples, equivalent to $L = M - R$ samples of overlap between adjoining segments. The DFT of each windowed segment is added to a complex-valued matrix that contains the magnitude and phase for each point in time and frequency. The resulting STFT matrix has

$$k = \lfloor \frac{N_x - L}{M - L} \rfloor$$

columns, where N_x is the length of the signal $x(n)$ and the $\lfloor \rfloor$ symbols denote the floor function. The number of rows in the matrix equals $N_{DFT} = \lfloor \frac{M}{2} \rfloor + 1$. The m_{th} column of the STFT matrix $\mathbf{X}(f) = [X_1(f), X_2(f), X_3(f), \ldots, X_k(f)]$ contains the DFT of the windowed data centered about time mR:

$$X_m(f) = \sum_{n=-\infty}^{\infty} x(n) \cdot g(n - mR) \cdot e^{-j2\pi fn}. \tag{3.2}$$

Fast Fourier transforms (FFTs) are commonly used to replace DFT. The STFT of the $32Hz$ EEG signal is computed with the Hann function, which has a length of $M = 128$, hop size $R = 32$, and overlap size $L = 96$ samples.

In this study, the STFT matrix $\mathbf{X}$ is limited to a few columns of present and past frames, that are strictly necessary for the next step. The m_{th} column of the STFT

matrix is then re-defined as:

$$x_m[n] \triangleq \begin{cases} x[n+mR], & n = [1:M] \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

$$X_m[k] = \text{FFT}_{hann}\{x_m[n]\} \tag{3.4}$$

where $x_m[n]$ is the m_{th} time-series frame with length M. Inverse STFT is calculated using the overlap-add (OLA) method.

$$y_m[n] = \text{IFFT}\{X_m[k]\} \tag{3.5}$$

$$y_m[n] = \sum_i y_{m+i}[n - iL] \tag{3.6}$$

Figure 3.7 shows how overlaps L are simulated by the circular buffers' tracking

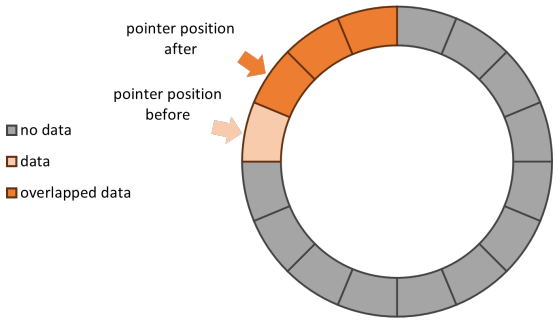


Figure 3.7: Overlap Add using Circular Buffers

pointer. After every read/write, the tracking pointer moves backward L positions, causing the next read/write to overlap with the last operation.

```
1  AccP = 0 # Cumulative complementary probability
2  T_0[k] = 0 # Threshold at frame 0
3  for m = [1:inf]:
4  T_m[k] = (AccP·T_{m-1}[k]+(1-p_m)·P_m[k]) / (AccP+(1-p_m))
5  AccP += (1 - p_m)
```

Listing 3.2: Real-time Denoiser

3.3.3 Denoiser

The implementation of the Denoiser in the original work (see Section 2.2.2) cannot be applied for a real-time system, as it requires calculating the STFT matrix $\mathbf{X}[m,k]$ of the entire EEG signals, which is not available before the acquisition process finishes.

One solution is to compute the spectral profile $T_m[k]$ accumulatively via a cumulative variable $AccP$ (Listing 3.2). The temporal gain, $G_m[k]$, and output frame, $Y_m[k]$, can be calculated using the temporal form of Eqs. (2.10) and (2.11).

3.3.4 Interpolation

In the offline processing design, interpolation with a variable factor is achieved by mapping the data points into a new dimension (x_{new}) using a time-transformation vector. This vector, calculated by performing the cumulative sum of the resampling rates, works similarly to the "Kernel Trick" used in support-vector machines (SVM). The data points were then linearly interpolated at a new sampling rate (f_{new}) on x_{new}.

By choosing $f_{new} = 1$, new data points would be located at integer indexes on x_{new}. The problem becomes finding integers between two real numbers: $I_{-1} \leq N[i] \leq I_0$

while $I_{-1}, I_0 \in \mathbb{R}$ and $N[i] \in \mathbb{N}$. The solution for real-time processing is to keep track of the indexes I_0 and I_{-1} of current (t_0) and last (t_{-1}) data points on x_{new}. More details on calculating these values are presented under Section 3.2.3.

3.3.5 Stereo Mixer

In this work, storing audio signals from all N EEG channels is redundant. Instead, only two circular buffers are used to temporarily store left and right stereo channels. The sum with delays of individual audio signals is simulated during the writing process into these circular buffers.

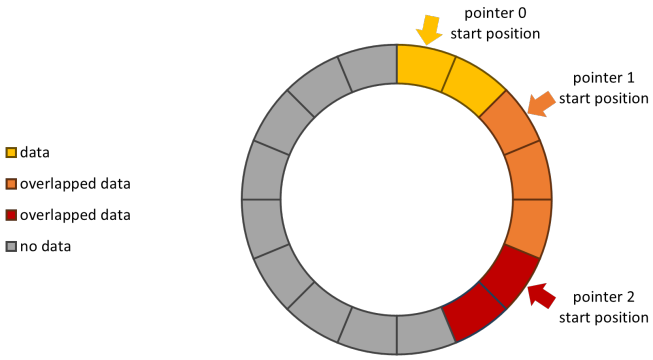


Figure 3.8: Sum with Delays using Circular Buffer

By defining a unique write pointer for each EEG channel, their respective delay $d_{L/R}$ can be specified by the initial position of the write pointer in the respective left/right audio buffer (Fig. 3.8). This design allows audio signals to be mixed and stored without using intermediate storage. During the writing process, the new values are the sum of their old values and written data. While the reading process resets all read values to zero.

3.4 Validation

This study does not arrange another survey for the real-time algorithm (see Section 2.4). To verify the faithfulness of the newly introduced real-time processing design to the original (baseline) algorithm, the audios rendered by the two algorithms are compared both objectively and subjectively. This section uses the same dataset that was used for the survey.

3.4.1 Dynamic Time Warping

Independently introduced in (Vintsyuk, 1968) and (Sakoe and Chiba, 1978), Dynamic Time Warping (DTW) is a similarity measure between two temporal sequences. Its popular applications include speech recognition and music synchronization (Müller, 2021). The equation to calculate the DTW distance is as follows (Tavenard, 2021):

$$\mathrm{DTW}_q(x, y) = \min_{\pi \in \mathcal{A}(x_i, y_j)} \left(\sum_{(i,j)\in\pi} d\left(x_i, y_j\right)^q \right)^{\frac{1}{q}} \tag{3.7}$$

where an alignment path π of length K is a sequence of K index pairs $((i_0, j_0), \ldots, (i_{K-1}, j_{K-1}))$ and $\mathcal{A}(x_i, y_j)$ is the set of all admissible paths between two time-series x and y. To be considered admissible, a path should satisfy the following conditions:

1. Beginning (resp. end) of time series are matched together:
   - $\pi_0 = (0, 0)$
   - $\pi_{K-1} = (n-1, m-1)$
2. The sequence is monotonically increasing in both i and j and all time series

indexes should appear at least once, which can be written:

- $i_{k-1} \leq i_k \leq i_{k-1} + 1$
- $j_{k-1} \leq j_k \leq j_{k-1} + 1$

The local distance $d(x_i, y_j)$ between the data points can be measured using any suitable distance measure; this study used *Squared Euclidean* as the local distance metric.

DTW is a distance *measure* and not a *metric* for it only met two of four requirements (Mueen and Keogh, 2016): Symmetry ($\mathrm{D}(x, y) = \mathrm{D}(y, x)$) and Constancy of Self-Similarity ($\mathrm{D}(x, x) = 0$), but neither Separation ($\mathrm{D}(x, y) = 0$ if $x = y$) nor Triangular Inequality ($\mathrm{D}(x, y) \leq \mathrm{D}(x, z) + \mathrm{D}(y, z)$).

3.4.2 Constant-Q Transform

Constant-Q Transform (CQT), first introduced in (J. C. Brown, 1991), is closely related to the Fourier Transform. Like the Fourier Transform, a constant-Q transform is a series of filters. Yet, unlike the former, it has logarithmically spaced center frequencies $f_k = f_0 \cdot 2^{\frac{k}{b}}$, where b is the number of filters per octave. Choosing the spectral width of the k-th filter as $\delta_k^{cq} = f_{k+1} - f_k = f_k \left(2^{\frac{1}{b}} - 1\right)$ makes the filter domains adjacent and yields a constant ratio of frequency to resolution $Q = \frac{f_k}{\delta_k^{cq}} = \left(2^{\frac{1}{b}} - 1\right)^{-1}$.

What makes the constant-Q transform suited for musical representation is that by an appropriate choice for f_0 (minimal center frequency) and b, the center frequencies directly correspond to musical notes. Another feature of the constant Q transform is its increasing time resolution towards higher frequencies, resembling how we perceive sounds (Blankertz, 1999).

3.4.3 Objective Comparison

Due to the DTW distance being the cumulative distance along the DTW alignment path, the distance between longer signals tends to be larger than that of shorter ones. To avoid this effect, the normalized DTW distance can be used instead, which is the distance divided by $N + M$, where N and M are the lengths of the time series x and y, respectively (Leodolter, Plant, and Brändle, 2021). The distance between two audio signals is defined as the normalized DTW distance between their constant-Q transforms.

As a result, the (n, n)-sized distance matrix $\mathbf{D}$ for two sets of n audio files, X, generated by the real-time algorithm and Y, by the baseline, is defined as:

$$\mathbf{D}\left[i, j\right] = \mathrm{DTW}\left(\mathrm{CQT}(X_i, Y_j)\right), \tag{3.8}$$

with X_i and Y_j are audio files from X and Y, $i, j \in [1 : n]$ are file indices, and $n = 79$ for the Helsinki dataset.

Similarity scores, which range from 0 to 1, with 0 indicating no similarity and 1 meaning two audios are identical, can be calculated from distance measures using any suitable function that maps distance to this range. Some commonly used similarity functions are:

- Exponential: $e^{-\gamma d}$;
- Gaussian: $e^{-\gamma^2 d^2}$;
- Reciprocal: $\frac{1}{\gamma + d}$;

The diagonal values of the similarity matrix $\mathbf{S}(\mathbf{D})$ indicate the similarity between pairs of audio files that were rendered from the same EEG signals; in the ideal case,

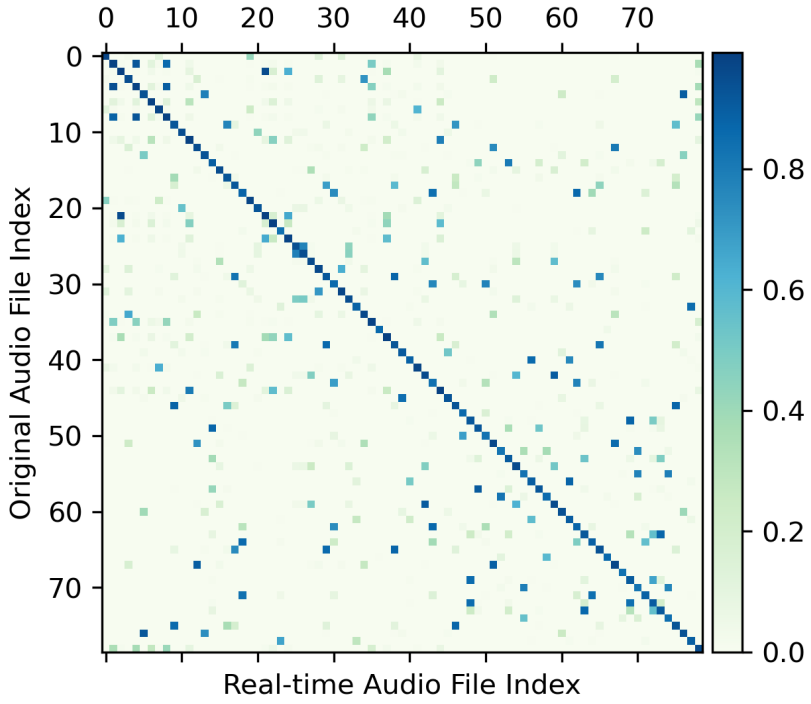
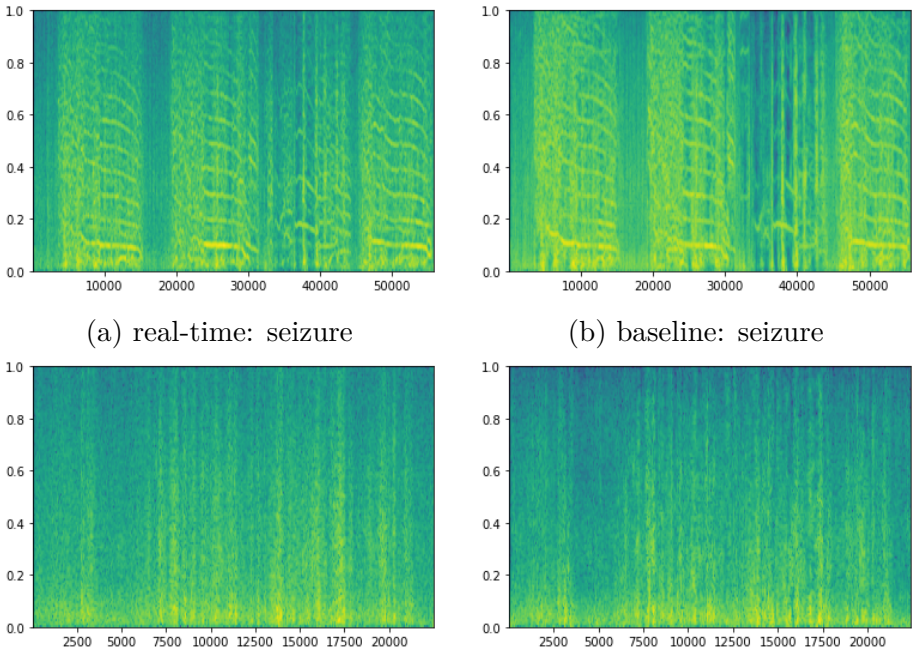


Figure 3.9: Similarity matrix between two sets of audio files generated by the real-time sonification algorithm (columns) and the baseline (rows).

they have a mean equal to 1, meaning the audios from the two sets are identical, or in other words, the real-time algorithm produces the same output as the baseline. On the other hand, non-diagonal values are the similarity scores between different audios and should ideally be 0.

The similarity matrix $\mathbf{S}$ between X and Y audio sets, illustrated in Fig. 3.9, was yielded by fine-tuning the similarity functions and their parameter γ to maximize the clustering separation between its diagonal and non-diagonal values. The chosen function was *Gaussian* with $\gamma = 0.125 \max(\mathbf{D})$. The mean similarity score between the two audio sets is $S_{mean} = 0.9245$.



(a) real-time: seizure

(b) baseline: seizure

(c) real-time: non-seizure

(d) baseline: non-seizure

Figure 3.10: Visual Comparison of Audio Spectrograms: (a & c) Audio outputs of the original algorithm. (b & d) Audio outputs of the real-time algorithm.

3.4.4 Subjective Comparison

Spectrograms are the standard for spectral analysis of audio signals (James L. Flanagan, 1972). Therefore, it is a common approach to compare audio files by their spectrograms. Figure 3.10 shows two pairs of audio files, one with seizures (Figs. 3.10a and 3.10b) and another without (Figs. 3.10c and 3.10d), rendered by the real-time and the baseline algorithm.

One can also use Dynamic Time Warping to find the best alignment of two audio files. Visualization of the wrapping path is an effective way to detect any relative compression/stretching between two audio signals. Figure 3.11 illustrates how the spectrograms of two audio files are aligned using DTW.

Another method to compare these audio signals, which is more intuitive, is to simply listen to them. Audio files and their spectrograms, rendered by the real-

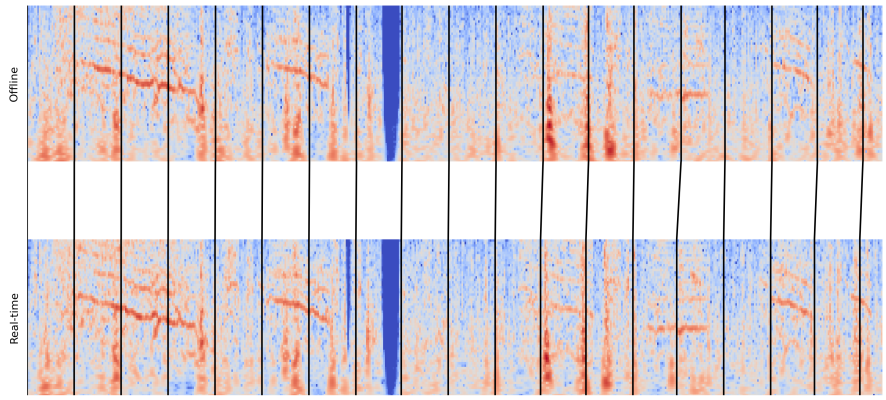


Figure 3.11: DTW Alignment of Audio Spectrograms

time algorithm in this study and the baseline algorithm in (Gomez-Quintana et al., 2022), are available for an interactive comparison at `https://tiennvhust.github.io/listen2eeg.github.io`.

3.5 Discussion

To enable real-time operation, changes to the original AI-assisted sonification algorithm were introduced in this chapter. These modifications include not only system design changes but also approximation of mathematical formulas. The new algorithm is validated against its original design, showing no significant difference in their audio outputs. This chapter also proposed an objective method to contrast the synthesized audios of the two algorithms.

Chapter 4

Neural Networks and Quantization

This chapter evaluates the performance of seizure detection neural networks in 8-bit integer representations. Neural network quantization techniques and validation strategies are also discussed.

4.1 Quantization: Basic Concepts

Quantization of neural networks is the process of converting the numerical components of the model, including weights, activations, and gradients, to lower precision representations. A quantized neural network has a significantly smaller size and lower latency. It also consumes less power due to less computation and memory access. For example, a neural network with parameters converted from 32-bit floats to 8-bit integers would reduce the model size by a factor of 4. And because integer operations are much faster than floating point types, the model's speed and power consumption are also improved.

The main trade-off of quantization is the potential loss in the model's accuracy, which is highly dependent on the network architecture and the method of quantization. However, neural networks are robust to quantization. Studies have shown that quantization often has a relatively small impact on neural networks' accuracy (Coelho et al., 2021; Jacob et al., 2017; Nagel et al., 2021; Rokh, Azarpeyvand, and Khanteymoori, 2023).

4.1.1 Scheme Categorization

Quantization schemes can be categorized in several ways. A common way is by the quantization operator $Q(r)$ that maps the weights and activations to quantized values (Gholami et al., 2021, see).

Uniform Quantization

Uniform quantization refers to the quantization schemes that use a constant quantization step size when mapping, meaning the discrete quantized values (quantization levels) are uniformly spaced. The quantization function in uniform quantization is defined as:

$$Q(r) = \text{Int}(\frac{r}{S}) - Z, \tag{4.1}$$

where Q is the quantization operator, r is a real-valued input (activation or weight), S is a real-valued scaling factor, and Z is an integer zero point. The Int function maps a real value to an integer value through a rounding operation. The scaling factor divides a given range of real values r into a number of partitions:

$$S = \frac{\beta - \alpha}{2^b - 1}, \tag{4.2}$$

where $[\alpha, \beta]$ denotes the clipping range, and b is the quantization bit width. The process of choosing the clipping range is often referred to as *calibration*. *Symmetric* and *asymmetric* quantization refers to the symmetry of the clipping range (whether $-\alpha = \beta$ or $-\alpha \neq \beta$). A popular choice in asymmetric quantization is choosing $\alpha = r_{min}$ and $\beta = r_{max}$. In symmetric quantization, it is also popular to choose the clipping range based on the range of the input value r, $-\alpha = \beta = \max(|r_{max}|, |r_{min}|)$. While asymmetric quantization is often preferred, using symmetric quantization simplifies the quantization function by restricting the zero-point to $Z = 0$.

Dynamic and Static Quantization

Another quantization scheme differentiator is *when* the process of calibration, determining the clipping range $[\alpha, \beta]$, takes place. Unlike constant tensors, such as weights and biases that can be statically calibrated, model input, activations (outputs of intermediate layers), and model output are variable.

In *dynamic quantization*, the activations are stored in floating point and quantized dynamically with their clipping range computed at inference time. This method provides reduced memory usage while minimizing quantization errors and, therefore, often results in a higher accuracy. However, dynamic quantization is not compatible with integer-only hardware devices or accelerators and has a higher latency due to the computation and storage of floating-point activations.

Non-Uniform Quantization

The quantization step size and quantization levels are allowed to be non-uniformly spaced in *Non-Uniform quantization*. The formal definition of non-uniform quan-

tization is shown in Eq. (4.3), where X_i represents the discrete quantization levels and Δ_i the quantization steps.

$$Q(r) = X_i, \quad \text{if } r \in [\Delta_i, \Delta_i + 1). \tag{4.3}$$

A typical nonuniform quantization is to use a logarithmic distribution where the quantization steps and levels increase exponentially. A base-2 logarithm quantization function is shown in Eq. (4.4).

$$Q(r) = \text{sign}(r) \cdot 2^{\lfloor \log_2 (r) \rfloor} \tag{4.4}$$

Logarithmic quantization has denser quantization levels for values close to zero. This allows for a more accurate quantization of deep convolution neural networks (DCNNs), which often have weights following a normal distribution (Park, Ahn, and Yoo, 2017; Rokh, Azarpeyvand, and Khanteymoori, 2023). However, nonuniform quantization schemes are typically difficult to deploy efficiently on general computation hardware such as GPUs and CPUs. As a result, uniform quantization is currently the de-facto method due to its simplicity and efficient mapping to hardware (Gholami et al., 2021).

4.1.2 Post Training Quantization

Post-training quantization (PTQ) is a technique applied after a neural network has been fully trained. In PTQ, the model's weights and activations are converted from high-precision formats, such as 32-bit floating point, to lower precision formats like

8-bit integers. This reduction in precision significantly decreases the model size and improves inference speed and efficiency, making it suitable for deployment on resource-constrained devices like smartphones and embedded systems. PTQ is particularly appealing because it does not require retraining the model, making it a quick and easy method to optimize existing models. However, the reduction in precision can sometimes lead to a degradation in model accuracy, especially if the original model is highly sensitive to small changes in weight values.

4.1.3 Quantization Aware Training

Quantization-aware training (QAT), on the other hand, involves incorporating quantization into the training process itself. During QAT, the model simulates lower precision arithmetic (e.g., 8-bit integers) while still being trained with high precision weights. This approach allows the model to learn to adjust its parameters to mitigate the potential accuracy loss due to quantization. By training the model with quantization in mind, QAT often yields better performance and higher accuracy in the quantized model compared to PTQ. While QAT can be more complex and time-consuming because it requires retraining the model, it provides a more robust solution for scenarios where maintaining high accuracy is critical after quantization.

4.1.4 Batch Normalization Folding

Batch normalization folding involves integrating the batch normalization parameters (specifically, the scale and shift parameters) into the preceding convolution or fully connected layer weights during inference. This process simplifies the model by reducing the number of operations required, as it eliminates the need for separate

batch normalization layers. Consequently, it enhances computational efficiency, decreases memory usage, and can lead to faster inference times without compromising the accuracy of the quantized model.

4.2 Hardware Aware Quantization

An AI microprocessor—the MAX78002, detailed in the next chapter—is used to implement the software developed in this study. Its manufacturer—Analog Devices—provides a software toolkit for deep learning development and quantization, containing customized PyTorch classes (Paszke et al., 2019), designated for models running on the MAX7800x AI microprocessor family (Analog-Devices, 2023). These classes contain additional features to enable support for quantization while ensuring compatibility on their hardware, including "Fused" operators that simulate in-flight pooling and activation, hardware-compatible rounding and clipping, and batch normalization folding (see Section 4.1.4).

An example of implementing a feature extraction block in the ResNet model with MAX7800x AI customized PyTorch classes is shown in Listing 4.1. This block contains three convolution layers and a residual connection.

Another hardware-related issue is that the MAX7800x family can only support arbitrary kernel lengths in 1D operations. This does not cause any substantial changes to the architecture other than the removal of the final maximum pooling layer, which allows the 2D neural networks to take multiple EEG channels as input and detect seizure across channels (O'Shea et al., 2020). The 1D neural networks share the same layers and parameters as their 2D counterparts, with all filters being one-dimensional, and operate on a single EEG channel.

```
class FeBlock(nn.Module):

    def __init__(self, num_channels=1, **kwargs):
        super().__init__()
        self.conv1 = FusedConv1dReLU(in_channels=num_channels, out_channels=32,
            kernel_size=3, **kwargs)
        self.conv2 = FusedConv1dReLU(in_channels=32, out_channels=32, kernel_size=3, **kwargs)
        self.conv3 = Conv1d(in_channels=32, out_channels=32, kernel_size=3, **kwargs)
        self.add = Add()
        self.atv = nn.ReLU(True)
        self.norm = Conv1d(in_channels=32, out_channels=32, kernel_size=1,
            batchnorm='Affine', **kwargs) # Batch normalization layer

    def forward(self, block_input):
        x = self.conv1(block_input)
        x = self.conv2(x)
        x = self.conv3(x)
        skip_conv = block_input[: , :, 3:-3]
        x = self.add(x, skip_conv)
        x = self.atv(x)
        x = self.norm(x)
        return x
```

Listing 4.1: A Feature Extraction Block using Customized PyTorch Classes

4.3 Cross-Validation

K-fold cross-validation is a robust technique that enhances model evaluation by dividing the dataset into k equally-sized folds, ensuring that each fold serves as both a training set and a validation set in different iterations. This method mitigates the risk of overfitting and provides a more accurate estimate of model performance compared to a basic train/test split. By averaging the results across all folds, k-fold cross-validation reduces the variability of the performance metric, leading to more reliable and generalizable insights. Additionally, it ensures that every data point is used for both training and validation, maximizing the use of available data. This comprehensive evaluation is especially beneficial for small datasets, where a single train/test split might not capture the full variability of the data. This section evaluates the performance of the VGGNet and ResNet models in Section 2.3 with 8bit quantization.

4.3.1 Strategies

In ML/AI for medicine, there are two primary cross-validation (CV) strategies: *record-wise* and *subject-wise*. In record-wise CV, the data is randomly split, allowing records from each subject to contribute to both the training and holdout sets. On the other hand, subject-wise CV ensures that the subjects in the training and holdout sets are independent (Saeb, Lonini, Jayaraman, D. Mohr, et al., 2016).

In diagnostic applications, it is recommended to use a subject-wise data split to obtain a better generalization of the model to new subjects. This strategy mimics the clinically relevant use-case scenario of diagnosis in newly recruited subjects (Saeb, Lonini, Jayaraman, D. C. Mohr, et al., 2017). The record-wise split can create dependence between training and test sets because identities are shared across them. This can lead the models to learn to identify subjects, in addition to diagnostic signals, which is called identity confounding (Chaibub Neto et al., 2019).

However, some studies still advocate the use of record-wise data splits, arguing that adopting subject-wise data splits in heterogeneous datasets might lead to model under-fitting and larger classification errors (Little et al., 2017).

4.3.2 Dataset

The same public dataset used in Gomez-Quintana et al., 2022 for its survey and in Section 3.4 for validation of the real-time algorithm is used again in this chapter for neural networks quantization and validation. This dataset of neonatal EEG, referred to as the Helsinki dataset, includes multichannel EEG recordings from 79 term neonates admitted to the NICU, with a median recording duration of 74

minutes. Seizures were annotated by three independent experts, with 39 neonates identified as having seizures and 22 as seizure-free by consensus (Nathan J Stevenson et al., 2019).

Subject-wise k-fold

The 79 multichannel EEG recordings are sorted (subject-wise) into five folds of similar total length (range: 21.9 to 22.8 hours) and a similar percentage of seizures (range: 9.5% to 10%) as shown in Fig. 4.1.

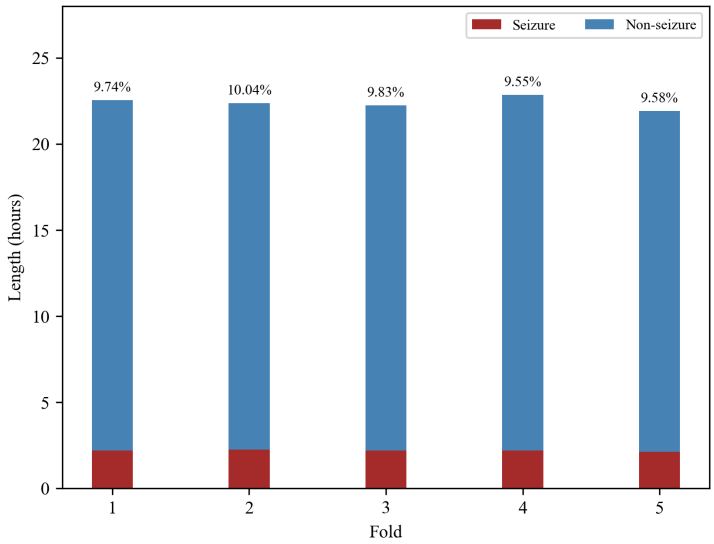


Figure 4.1: Dataset Splitting

Splitting Algorithm

Different algorithms can be exploited to find the optimal k-fold split that has the most balanced distribution of positive labels and fold lengths. A sorting algorithm can quickly find a distribution that balances either positive labels or fold lengths, however, it would struggle to balance both values. An optimal solution for this

```
1 L_target = 1/k Σ_{i=1}^{N} L_i # Mean fold length
2 P_target = 1/k Σ_{i=1}^{N} P_i # Mean positive values number
3 pool = [1:N]
4 for fold in k-folds:
5     i_min = min(i in pool: D_i = |L_fold + L_i − L_target| + |P_fold + P_i − P_target|)
6     fold.push(i_min)
7     pool.pop(i_min)
```

Listing 4.2: Greedy Algorithm

problem can be found using a greedy algorithm. However, finding a solution is not always a guarantee.

Given that the goal is balancing both fold lengths and the percentage of labels, the target values can be defined as the mean values over the dataset. The greedy algorithm finds the best match for each fold by minimizing a distance function, D_i, defined as the Euclidean distance between the current fold and the target (Listing 4.2).

4.3.3 Data Augmentation

Different data augmentation techniques were used during the quantization-aware training. In addition to signal amplitude scaling and sign inverting (Daly et al., 2021), basic computer vision techniques were also applied, such as Cutmix (Yun et al., 2019), and MixUp (Zhang and Chen, 2017).

4.4 Results and Discussion

Both neural networks are evaluated using the Helsinki dataset: 5-fold cross-validation for the ResNet and a 50/50 train/test sets split for the VGGNet

architecture. Figure 4.2 compares the performance of these two algorithms before and after 8-bit quantization.

The ResNet outperformed the VGGNet model in both floating-point and 8-bit integer representations by 2.6% and 3%, respectively. Additionally, the ResNet model in 8-bit format achieved a higher AUC, reaching 97.6%, compared to the VGG model in floating-point format at 95.4%, marking a 2.2% improvement.

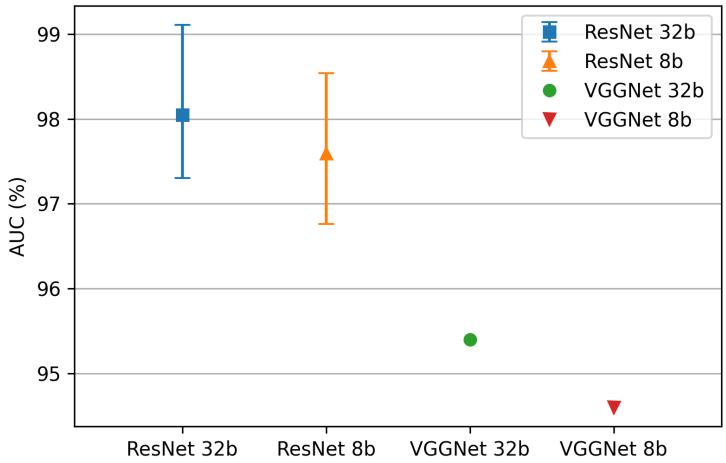


Figure 4.2: Before and After Quantization Performances

The ResNet architecture also demonstrated greater robustness to integer quantization, seeing a 0.4% performance drop, compared to a 0.8% drop observed in VGGNet. This trade-off in classification performance is negligible compared to the reduction in size, power consumption, and gain in computational speed thanks to all operations being performed in 8-bit integers instead of floating point numbers.

Chapter 5

A Low-power Implementation

This chapter discusses the implementation of the real-time AI-assisted sonification algorithm on hardware. It starts with the introduction to an AI-capable microcontroller used in this implementation. Next, it presents a performance analysis of the algorithm, specifically, execution speed and energy consumption. This evaluation involves two software versions of the algorithm coupled with two seizure detection neural networks presented in the last chapter.

5.1 Hardware Platform

The utilized hardware—a microcontroller with an integrated machine learning accelerator—is discussed in this section. This device combines low power consumption with the computational capability necessary for the sonification of EEG signals.

5.1.1 MAX78002

MAX78002 is a new generation of AI microcontroller designed for running neural networks with ultra-low power consumption by Analog Devices. This system-on-chip (SoC) features a dual-core microcontroller with an Arm Cortex-M4 processor up to 120MHz and a 32-bit RISC-V coprocessor up to 60MHz. The Arm Cortex-M4 (CM4) processor is ideal for a high-efficiency, low-power, and low-cost signal processing system. It supports DSP and Single Instruction, Multiple Data (SIMD) instructions with a Floating-Point Unit (FPU). The addition of a 32-bit RISC-V coprocessor (RV32) provides the system with ultra-low power consumption signal processing capability. The microcontroller core has a large on-chip system memory with 2.5 MB flash and up to 384 kB SRAM.

The MAX78002 enables energy-efficient AI capability with a Convolutional Neural Network (CNN) accelerator, consisting of 64 parallel processors with 1.31 MB of SRAM-based storage. It supports network depths up to 128 layers, over 2,340 kB of weight memory (equivalent to over 2 million 8-bit weights), and 1,280 kB for data (activations) memory.

The system clock can be selected from multiple clock sources including an internal phase-locked loop (IPLL) at 100MHz and 200MHz, an internal primary oscillator (IPO) at 120MHz, an internal secondary oscillator (ISO) at 60MHz, and other external clock inputs. The MAX78002 provides six general-purpose, 32-bit timers for timing, capture/compare, or generation of pulse-width modulated (PWM) signals with minimal software interaction.

5.1.2 Evaluation Kit

The MAX78002 Evaluation Kit (MAX78002EVKIT), shown in Fig. 5.1, provides a platform and tools for application development. It is equipped with a power accumulator to track energy consumption on the MAX78002. Two USB connectors provide serial access to the MAX78002, one directly and the other through a USB to UART bridge. A third USB connector allows access to the MAX78002 energy consumption data.

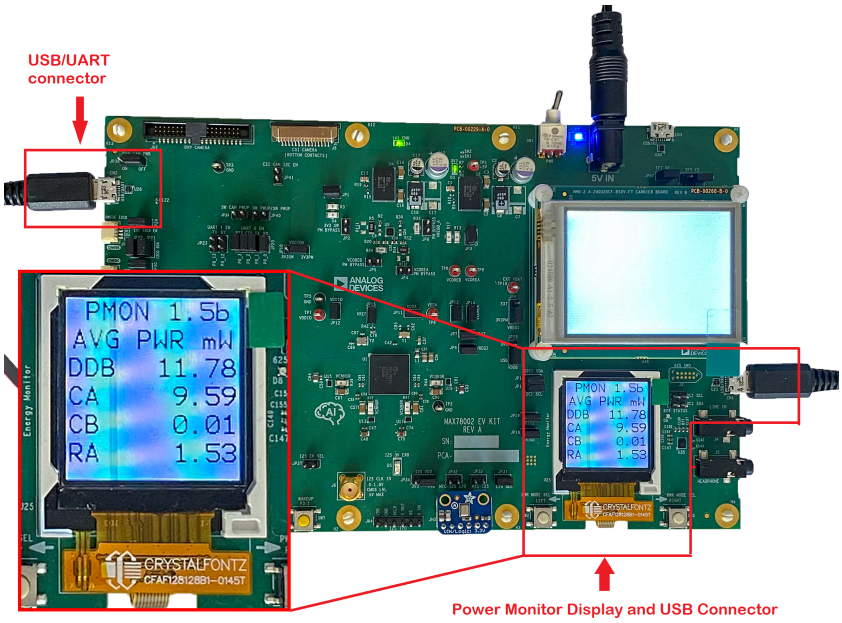


Figure 5.1: MAX78002 Evaluation Kit

Power Monitor

A power monitor (PMON) subcircuit is included in the MAX78002EVKIT providing quick and coarse measurements on four power domains of the MAX78002 (labeled DDB, CA, CB, and RA in Fig. 5.1):

1. DDB: V_{REGI}, V_{BAT}, and V_{DDIOH} are, in the same order, voltage supply to integrated Single-Inductor Multiple-Output (SIMO), external battery power

input, and GPIO supply voltage (high). They are connected at the circuit board level and range between 2.85V to 3.6V (typically 3.3V). The supplementary power supply to the CNN cores from an external unit (U4) is also tracked in this domain.

2. CA: V_{COREA} denotes the input supply voltage to core A and the CNN quadrants [0:3] (typically 1.1V). The supplementary power supply to the CNN cores from an external unit (U4) is also tracked in this domain.

3. CB: V_{COREB} denotes the input supply voltage to core B (typically 1.1V).

4. RA: V_{DDA} and V_{DDIO}, connected at the circuit-board level, are Analog and GPIO supply voltages, respectively.

This subcircuit also allows windowed power measurements with software-triggered events for the CNN cores and the total system energy. This data is displayed on a secondary LCD and can be streamed using a USB connector.

In addition, the evaluation kit provides jumpers with built-in shunts that serve as voltage drop test points for fine power measurements of the MAX78002's domains.

5.2 Software Implementations

This section details the software implementation of the multi-threading design presented in Chapter 3. It covers the C/C++ implementation of multi-threading and task synchronization. Finally, two software versions of the real-time AI-assisted sonification algorithm are introduced.

5.2.1 Multi-threading

The real-time sonification algorithm, proposed in Section 3.2, consists of four concurrent threads: `comm_t`, `pre_processing_t`, `phase_vocoder_t`, and `neural_network_t`. In C++, a thread of execution can be represented by the class `std::thread` (since C++11) or `boost::thread` (by the Boost C++ Libraries).

However, these libraries are not fit for microcontrollers. Instead, multithreading on the MAX78002 was implemented with FreeRTOS Real Time Kernel (RTOS). Unlike General Purpose Operating Systems (GPOS) such as Unix or Windows, Real-Time Operating Systems (RTOS) are deterministic and have a much smaller memory footprint. In return, they are not designed to run on desktop computer class processors.

5.2.2 Concurrency

In this study, the real-time sonification algorithm—a multithread program—used only the Arm Cortex-M4 (CM4) processor, while the RISC-V core was reserved for future developments. This is called *concurrency*, where execution resources are switched between the program's threads, thereby allowing more than one thread to make progress without running simultaneously.

The seizure detection algorithms were running in *parallel* with the sonification process, using the neural network accelerator on the MAX78002. This is handled by `neural_network_t`; it loads the neural networks and input data from SRAM to the accelerator and unloads the final outputs.

5.2.3 Task Synchronization

The real-time sonification algorithm contains three circular buffers that are shared resources (critical sections) between its threads. A *semaphore* is an integer variable used to coordinate the activities of multiple processes in a computer system. They are used for critical sections to enforce mutual exclusion, avoid race conditions, and implement process synchronization.

The process of using semaphores provides two operations: wait (P) and signal (V). The wait operation decrements the value of the semaphore, and the signal operation increments the value of the semaphore. When the value of the semaphore is zero, any process that performs a wait operation will be blocked until another process performs a signal operation.

There are two types of semaphores: *Binary Semaphores*, also known as *mutex* locks, which can only be either 0 or 1; and *Counting Semaphores* can be used to control access to a resource that has a limitation on the number of simultaneous accesses.

In C++, semaphores and mutexes can be initialized with `std::counting_semaphore` (since C++20) and `std::mutex` (since C++11). When using FreeRTOS, direct-to-task notifications can be used in addition to mutexes as a faster and more memory-efficient way to unblock RTOS tasks (Barry, 2020).

Example

Listing 5.1 details an example of how direct-to-task notifications and mutexes were used in this study: The `phase_vocoder_t` attempts to access processed data from the `processed_cb` using a `read` method. If unsuccessful, the thread will

block using `ulTaskNotifyTake`. The signal to unblock this thread is sent from `pre_processing_t` with `xTaskNotifyGive` when it finds the buffer is ready.

The circular buffer class contains a mutex lock to avoid racing conditions. Any process that attempts to access the buffer will have to first obtain a mutex lock with `xSemaphoreTake`. Only when this is successful can it proceed to access the stored data. The mutex lock is released using `xSemaphoreGive` after the session finishes allowing access to other processes that are currently in the queue.

5.2.4 Versions

The proposed real-time sonification algorithm in Section 3.2 comes with three different software implementations:

1. x86-64: This software version was created for the ease of verifying the algorithm against the original design that operates on the same hardware during the developmental process. The programming language used to develop this software is C++17.

2. CM4 version 0: This is the first implementation of the algorithm on the MAX78002. The software is written in C and C++11. Except for hardware compatibility, no changes were made to the algorithm regarding its operation.

3. CM4 version 1: This version optimizes the algorithm's performance on the Arm Cortex-M4 by taking advantage of its floating-point unit (FPU).

In the earlier versions (x86-64 and CM4 version 0), the PocketFFT for C++ library was used to implement the FFT, while the filters have a custom implementation using circular buffers (see Section 3.3.1).

```
1  /* ######################### phase_vocoder_t
       ######################### */
2  void sonification::phase_vocoder::read (size_t channel, float *data)
       {
3      /* If processed_cb data is not ready, hang here until get
           notified */
4      if (!prep->at(channel)->read_ready(SONI_THREAD_ID)) {
5          ulTaskNotifyTake(pdTRUE, portMAX_DELAY);
6      }
7      /* Read data from processed_cb with in-flight Hann windowing
           function */
8      prep->at(channel)->read_window (SONI_THREAD_ID, data,
           window.data());
9  }
10 /* ######################### pre_processing_t
       ######################### */
11     /* If raw_cb is not ready, hang here until get notified */
12     if (!raw->at(0)->read_ready()) {
13         ulTaskNotifyTake(pdTRUE, portMAX_DELAY);
14     }
15     ...
16     /* This part performs the pre-processing steps */
17     ...
18     /* Notify phase_vocoder_t */
19     if (processed->at(0)->read_ready(SONI_THREAD_ID)) {
20         xTaskNotifyGive(this->pv->hdl);
21     }
22 /* ######################### circular buffer class
       ######################### */
23 void read_window (size_t chan, T* output, T* window) {
24     assert (chan < channel_size && output != nullptr && window !=
           nullptr);
25     /* Try lock mutex */
26     if (xSemaphoreTake(mtx, portMAX_DELAY) == pdTRUE) {
27         ...
28         /* Reading data and applying in-flight window function */
29         ...
30         xSemaphoreGive(mtx); /* Release mutex */
31     }
32 }
```

Listing 5.1: Direct-to-task Notifications Example

The final version of the algorithm (CM4 version 1) utilizes the CMSIS-DSP 1.15.0 library to optimize its performance on the MAX78002. It is a comprehensive suite of optimized signal processing functions designed specifically for ARM Cortex-M and Cortex-A processors. The library leverages the SIMD (Single Instruction, Multiple Data) instruction set and Floating-Point Unit (FPU) of the ARM Cortex-M4 to accelerate DSP operations, resulting in significant performance improvements for FFTs and filter computations.

5.3 Results and Discussion

The real-time sonification algorithm and seizure detection neural networks are implemented using the MAX78002 development kit (MAX78002EVKIT). This section evaluates the performance of these algorithms on the hardware: the DSP tasks with the microprocessor cores and neural networks on the accelerator core.

5.3.1 Speed Performance

Measurements Setup

The signal processing threads (`phase_vocoder_t` and `pre_processing_t`) are further divided into sub-tasks for better analysis (see Table 5.1).

The MAX78002 microprocessor operates in ACTIVE mode and is clocked at 120 MHz (internal primary oscillator). Task execution time on the CM4 processor, as well as the neural networks' inference time, were measured using the MAX78002's general-purpose timers.

Table 5.1: DSP Tasks and Subtasks Definitions

Subtasks	**Details**
`phase_vocoder_t`	
Read & Windowing	Read data and perform in-flight hann window function.
Fourier Transform	Calculating the fast Fourier transform (FFT).
Interpolation	Denoiser and interpolation of FFT frames.
Synthesis & Mixing	Computing inverse fast Fourier transforms (iFFT), overlap-add, and mixing channel-wise audios into stereo audio.
Others	Other causes of time consumption include memory allocation, performing max-pooling, and data attenuation.
`pre_processing_t`	
Filters	Digital filters (a band-pass filter and a notch-pass filter with a total of 65 coefficients) and down-sampling.
Soft-limit	Soft limitation and normalization of down-sampled data.
Others	Other causes of time consumption include data reading and writing.

Results

Table 5.2 details the speed performance of the real-time sonification algorithm on the MAX78002 with one and eight EEG channels. The main difference between the two CM4 versions 0 and 1 is the use of the CMSIS-DSP library by Arm for performance optimization. The total time consumption of each thread is indicated in bold font, whereas bottlenecks are highlighted with an underline.

Unlike `pre_processing_t` and `neural_network_t` that have a constant time consumption, `phase_vocoder_t` and its sub-tasks have a varying workload, depending on the number of the interpolated frequency frames, which varies between zero and nine (see Eq. (3.1)). Best and worst scenarios are defined accordingly.

In the best case (BC), `phase_vocoder_t` has a minimal workload. When there are zero new FFT frames created, this thread concludes after calculating the FFT and skipping subsequent tasks. `pre_processing_t` becomes the most time-consuming task with applying filters occupying the most time.

Table 5.2: Execution time of the real-time sonification algorithm by tasks and subtasks, measured by the microcontroller's internal clock.

Tasks	Execution Time (ms)							
	1 Channel				8 Channels			
	version 0		version 1		version 0		version 1	
	BC	WC	BC	WC	BC	WC	BC	WC
`phase_vocoder_t`	**1.9**	**38.3**	**1.2**	**27.9**	**20.9**	**302.6**	**12.1**	**152.3**
- Read & Windowing	0.1	0.1	0.1	0.1	2.0	1.8	2.0	1.9
- Fourier Transform	0.8	0.8	0.3	0.3	10.6	13.1	3.8	3.7
- Interpolation	0.3	4.0	0.3	4.2	2.0	33.0	2.3	34.1
- Synthesis & Mixing	0.2	32.7	0.1	23.0	1.4	250.8	0.6	108.1
- Others	0.5	0.7	0.4	0.3	4.9	3.9	3.4	4.5
`pre_processing_t`	**26.5**		**4.6**		**210.0**		**40.9**	
- Filters	22.7		4.2		182.5		34.5	
- Soft-limit	3.7		0.3		27		4.8	
- Others	0.1		0.1		0.5		1.7	
`neural_network_t`	ResNet				VGGNet			
	-				-			
- Inference Time	0.937				0.248			

In the worst case scenario (WC), for every original FFT frame, there are between eight and nine newly interpolated frames. `phase_vocoder_t` overtakes `pre_processing_t` in the total running time while performing audio synthesis and mixing results in the most computational load.

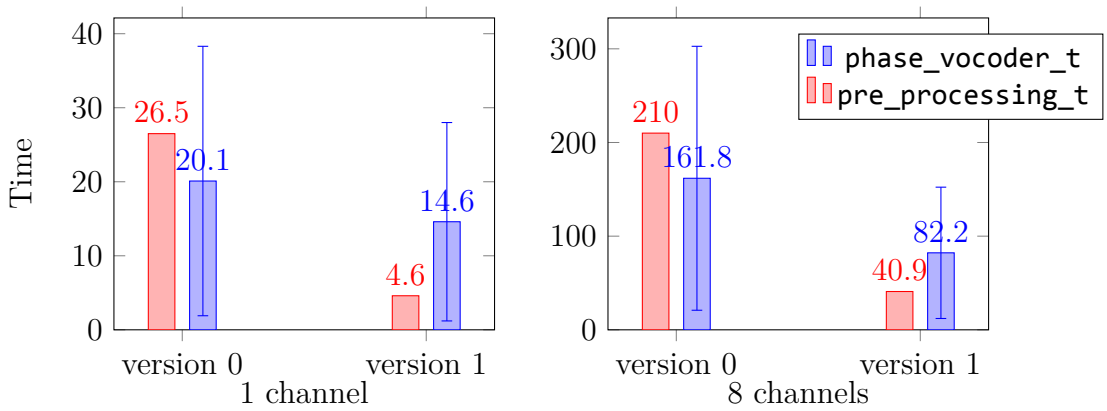


Figure 5.2: Versions Comparison

Figure 5.2 illustrates a running time comparison between the signal-processing tasks of two versions. Version 1, with the use of the floating-point unit (FPU), shows a significant boost in speed performance: `pre_processing_t` and filter op-

erations are approximately 5.5 and six times faster; whereas `phase_vocoder_t` was accelerated from 60% to 100% with FFT operations sped up 2.7 times.

The neural network's inference speed on the MAX78002 was also benchmarked using its timers. The inference time of a single 1D image (size of 512 and 256, respectively) by the ResNet and VGGNet are 937 μs and 248 μs respectively. These measurements were later verified with their power signatures (see Section 5.3.3).

Discussion

The algorithm's total processing time is the time required to complete its most time-consuming task (`phase_vocoder_t`) in the worst case scenario. Figure 5.3 shows a linear relationship between the total running time and the number of EEG channels.

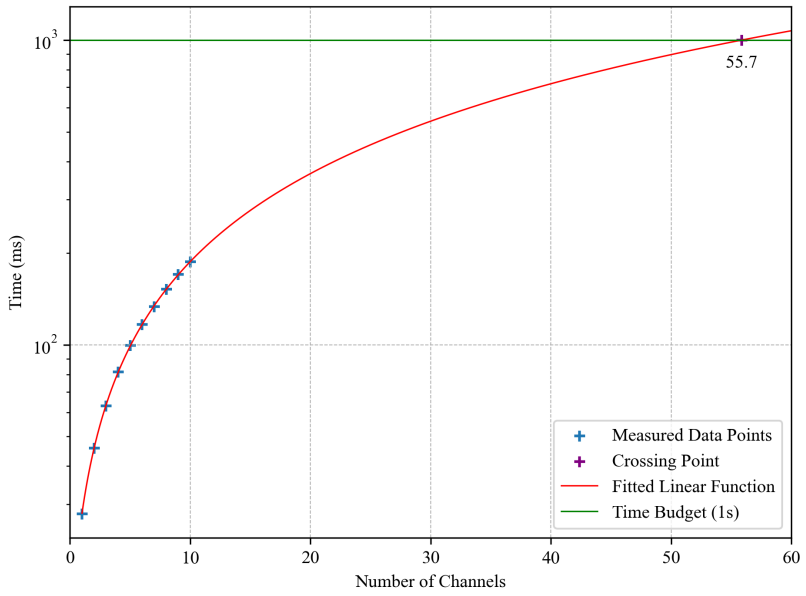


Figure 5.3: Total Time Consumption by Number of Channels

Theoretically, this implementation (Arm Cortex-M4 version 1) can process up to 55 parallel channels of EEG before it crosses the one-second processing time budget for a real-time system. In reality, this number is capped at 13 channels

on the MAX78002 due to a bottleneck in memory consumption. Measurement of memory footprint is challenging due to hidden memory usage by the operating systems and libraries.

5.3.2 Energy Consumption

Measurements Setup

This section evaluates the energy consumption of the real-time sonification algorithm (in two versions) coupling with two different neural network architectures. The MAX78002 operates in ACTIVE mode and clocks at 120MHz.

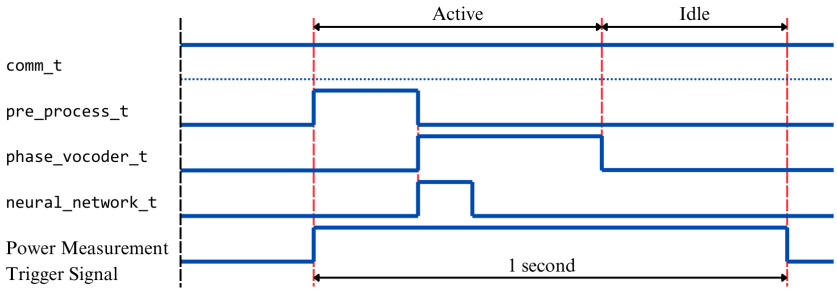


Figure 5.4: Power Measurement Sequence

The integrated power measurement circuit (PMON) on the MAX78002EVKIT was used to capture the total energy dispersed on the four supplies that power the MAX78002. The measurement window, illustrated in Fig. 5.4, lasts for one second, starting when a task notification is sent to `pre_processing_t` signaling a ready state in the raw data buffer.

Results and Discussion

Figure 5.5 displays the measurement results in error bars with lower and upper limits corresponding to best and worst case scenarios, respectively.

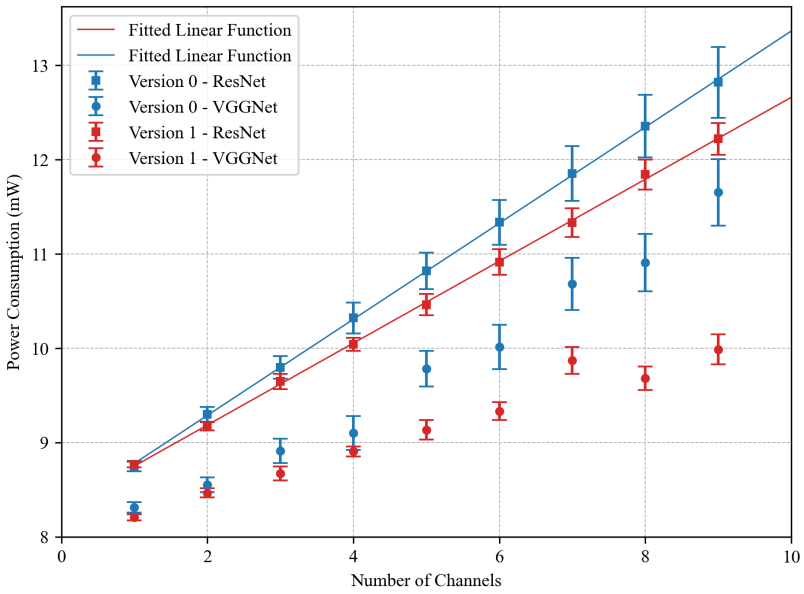


Figure 5.5: Power Consumption by Number of Channels

Overall, the power consumption on the MAX78002 has a linear correlation with the number of operating EEG channels. However, implementations with the VGGNet do not reflect this relationship without large deviations. A possible cause of this is due to the PMON measurement circuit having a low sampling rate (1024Hz). This means that the measurement of events shorter than the sampling period (976 mus) is unreliable, meanwhile, the inference speed of the VGGNet is measured at 248 μs.

The CMSIS-DSP software library was shown to consume less power compared to the older implementation (version 0). This is potentially a result of the Arm Cortex-M4 processor's running time being shortened.

When operating with 8 EEG channels, the algorithm (version 1) consumes approximately 12 mW using the ResNet model. This figure is lower when using the VGGNet, at around 10 mW. In the setting with 18 EEG channels, the algorithm (version 1) is predicted to consume approximately 16 mW.

5.3.3 Neural Network Power Signatures

Measurements Setup

The power consumption of the MAX78002's accelerator can be measured by monitoring the energy consumption on net VCOREA_PM (labeled CA on the PMON display). A voltage test point for this power net was pre-installed on the evaluation kit at jumper J4 with a shunt of 10 $m\Omega$ (R14).

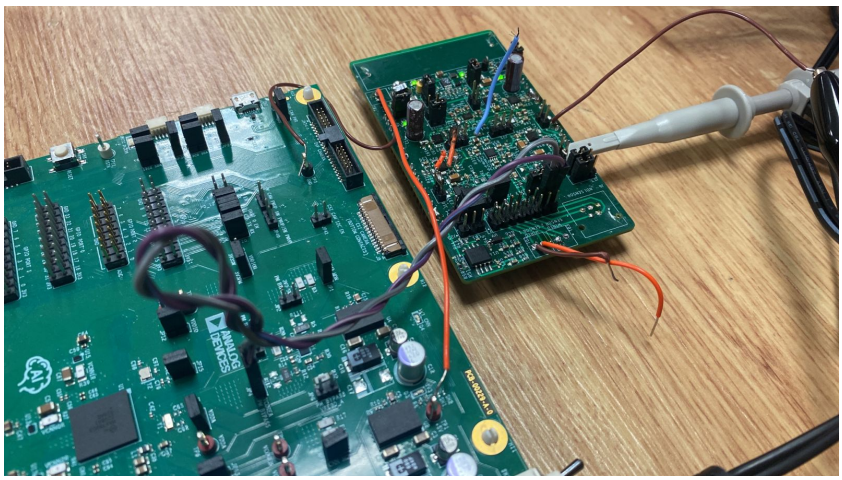


Figure 5.6: Power Measurement Setup

Figure 5.6 shows the measurement setup to capture the power signatures on net VCOREA_PM via jumper J4. This test point is bridged with an amplifier circuit of 497 times gain (in the middle) before connecting to an oscilloscope for fine voltage measurement.

Results and Discussion

Figure 5.7 presents the current (derived from the measured voltage) drawn by the two neural networks. The power signatures of the ResNet are shown in blue (a1 with 8 inferences and a2 with 1), likewise, b1 and b2 show the VGGNet (in

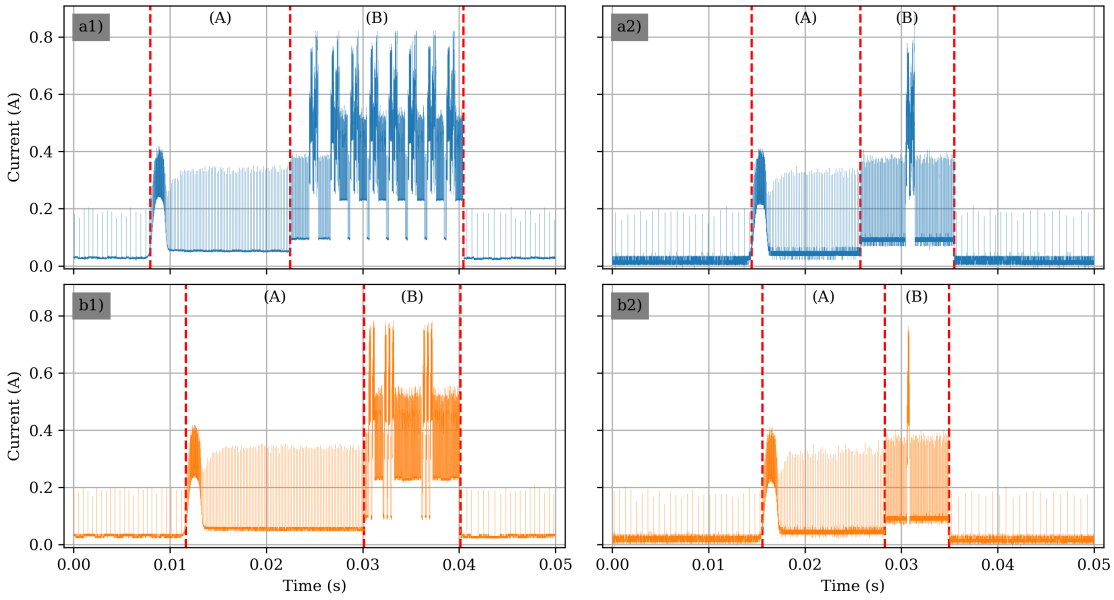
orange).

The accelerator's operation time is divided into two stages (A and B) corresponding to the events of CNN clock enabling, CNN starting, and CNN clock disabling.

- Stage A: initializing state machine, loading biases, and configuring state machine.
- Stage B: loading input data, running inference (peak), and unloading output data.

The summary table provides details on the features of these measurements. Dispersed energy (mJ) is calculated with the supply voltage on net VCOREA_PM at 1.1V.

These power signatures are consistent with the earlier measurements in Table 5.2. The inference time of the ResNet is about 3.8 times longer than that of the VGGNet, caused by its deeper architecture, larger number of parameters, and larger input size. Both neural networks draw a similar current on the MAX78002's accelerator (between 450 and 495 mA). Therefore, the difference in the amount of energy they consumed can be linked to the disparity in their running times.



Summaries

	Total Time (ms)	$A^{(1)}$ (ms)	$B^{(2)}$ (ms)	Peak Time (ms)	$B^{(2)}$ Current (mA)	Peak Current (mA)	$B^{(2)}$ Energy (mJ)	Total Energy (mJ)
a1)	32.444	14.479	17.965	0.922	318.806	464.728	5.727	7.592
a2)	20.991	11.249	9.742	0.927	143.195	453.887	1.395	2.479
b1)	28.864	18.743	10.122	0.240	308.051	494.259	3.118	5.054
b2)	19.352	12.722	6.630	0.247	125.668	493.048	0.833	1.959

(1) Enabling accelerator clock, configuring state machine, and loading CNN parameters

(2) Loading input data, CNN inference (peaks), and unloading output data

Figure 5.7: Neural Network Power Signatures

Chapter 6

Conclusion

Electroencephalography (EEG) is often time-consuming and requires a high level of expertise due to the complexity of the recordings. Nevertheless, it is the gold standard to confirm seizures, a commonly observed neurological symptom in the neonatal intensive care unit, which are linked to higher mortality rates in neonates around the globe.

Sonification—a signal processing approach that converts different types of data into audible sound—has shown promising results in biosignal analysis thanks to the excellent capability of human hearing at detecting subtle changes in sound patterns. The work in Gomez-Quintana et al., 2022 proposed a sonification approach that integrates artificial intelligence to suppress background noise in the EEG recordings, effectively reducing the listening time and cognitive loads for listeners.

This study aims to develop a practical application to assist in EEG diagnostic procedures based on the previous work. We target a battery-powered and inexpensive system that can be widely accessible in underdeveloped regions where neonates

are the most vulnerable due to the scarcity of expertise and equipment.

A real-time processing design was presented as a part of this work, allowing the sonification algorithm to operate and synthesize audio in parallel with EEG acquisition. The new design contains essential modifications without affecting its consistency with the original study. This approach removes the need for human intervention in the process and thus provides better accessibility for medical workers.

Bringing artificial intelligence to the far edge is another step toward realizing the targeted application. The effects of full integer quantization on neural networks were studied. Although the conversion into integer representations results in a slight accuracy loss, it significantly improves the inference speed, memory footprint, power consumption, and hardware adaptability of deep learning models.

An implementation of the proposed system was presented using a low-power and AI-capable microcontroller. Its speed and energy consumption are linearly correlated with the number of input EEG channels. This real-time EEG analysis assisting system can process up to 55 parallel channels while being powered by a small battery.

Future works will focus on improving the implementation for enhanced efficiency and scalability. On the hardware side, we plan to explore more powerful platforms, such as FPGAs, to support processing a larger number of EEG channels. Conversely, a more power-efficient setup could be achieved by enabling parallel processing on the existing 32-bit RISC-V coprocessor of the MAX78002 microcontroller.

In terms of algorithmic design, further optimization can be achieved by dividing existing threads into smaller, more manageable tasks, which may improve paral-

lelism and processing speed. Additionally, we are considering the development of a new filter design with fewer coefficients to minimize computational overhead.

Lastly, optimizing the codebase will be a priority to maximize software efficiency, thereby reducing execution time and resource usage. This software can also be adapted on the emerging Extended Reality (XR) platforms, paving the way for a new approach to EEG analysis.

Bibliography

Abdelhameed, Ahmed M., Hisham G. Daoud, and Magdy Bayoumi (2018). "Epileptic Seizure Detection using Deep Convolutional Autoencoder". In: *2018 IEEE International Workshop on Signal Processing Systems (SiPS)*, pp. 223–228. DOI: 10.1109/SiPS.2018.8598447.

Aldana Blanco, Andrea Lorena, Steffen Grautoff, and Thomas Hermann (June 2020). "ECG sonification to support the diagnosis and monitoring of myocardial infarction". In: *Journal on Multimodal User Interfaces* 14.2, pp. 207–218. ISSN: 1783-8738. DOI: 10.1007/s12193-020-00319-x.

Amann, Julia et al. (Nov. 2020). "Explainability for artificial intelligence in healthcare: a multidisciplinary perspective". In: *BMC Medical Informatics and Decision Making* 20.1, p. 310. ISSN: 1472-6947. DOI: 10.1186/s12911-020-01332-6.

Amin, Syed Umar and M. Shamim Hossain (2021). "Edge Intelligence and Internet of Things in Healthcare: A Survey". In: *IEEE Access* 9, pp. 45–59. DOI: 10.1109/ACCESS.2020.3045115.

Analog-Devices (Dec. 2023). *Model training for ADI's MAX78000 and MAX78002 AI devices*.

Baier, Gerold, Thomas Hermann, and Ulrich Stephani (2007). "Event-based sonification of EEG rhythms in real time". In: *Clinical Neurophysiology* 118.6, pp. 1377–1386. ISSN: 1388-2457. DOI: https://doi.org/10.1016/j.clinph.2007.01.025.

Bakken, Suzanne (June 2023). "AI in health: keeping the human in the loop". In: *Journal of the American Medical Informatics Association* 30.7, pp. 1225–1226. ISSN: 1527-974X. DOI: 10.1093/jamia/ocad091.

Barry, Richard (Sept. 2020). *Decrease Ram footprint and accelerate execution with FreeRTOS notifications*.

Blankertz, Benjamin (Sept. 1999). *Notes on the Constant Q Transform*. https://doc.ml.tu-berlin.de/bbci/material/publications/Bla_constQ.pdf.

Boll, S. (1979). "Suppression of acoustic noise in speech using spectral subtraction". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 27.2, pp. 113–120. DOI: 10.1109/TASSP.1979.1163209.

Boonyakitanont, Poomipat et al. (2020). "A review of feature extraction and performance evaluation in epileptic seizure detection using EEG". In: *Biomedical Signal Processing and Control* 57, p. 101702. ISSN: 1746-8094. DOI: https://doi.org/10.1016/j.bspc.2019.101702.

Boost C++ Libraries (2022). *Boost C++ Libraries*. https://www.boost.org/.

Boylan, Geraldine B., Nathan J. Stevenson, and Sampsa Vanhatalo (2013). "Monitoring neonatal seizures". In: *Seminars in Fetal and Neonatal Medicine* 18.4. Neonatal seizures, pp. 202–208. ISSN: 1744-165X. DOI: https://doi.org/10.1016/j.siny.2013.04.004.

Brown, Andrew D., G. Christopher Stecker, and Daniel J. Tollin (Dec. 2014). "The Precedence Effect in Sound Localization". In: *Journal of the Association for Research in Otolaryngology* 16.1, pp. 1–28. ISSN: 1438-7573. DOI: 10.1007/s10162-014-0496-2.

Brown, Judith C. (Jan. 1991). "Calculation of a constant Q spectral transform". In: *The Journal of the Acoustical Society of America* 89.1, pp. 425–434. ISSN: 0001-4966. DOI: 10.1121/1.400476.

Chafe, Christopher D and Josef Parvizi (Nov. 2018). *Method of sonifying brain electrical activity*. US Patent 10,136,862.

Chaibub Neto, Elias et al. (Oct. 2019). “Detecting the impact of subject characteristics on machine learning-based diagnostic applications”. In: *npj Digital Medicine* 2.1. ISSN: 2398-6352. DOI: 10.1038/s41746-019-0178-x.

Coelho, Claudionor N. et al. (June 2021). “Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors”. In: *Nature Machine Intelligence* 3.8, pp. 675–686. ISSN: 2522-5839. DOI: 10.1038/s42256-021-00356-5.

Daly, Aengus et al. (2021). “Towards Deeper Neural Networks for Neonatal Seizure Detection”. In: *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 920–923. DOI: 10.1109/EMBC46164.2021.9629485.

Diab, Maha and Esther Rodriguez-Villegas (Jan. 2022). “Embedded Machine Learning Using Microcontrollers in Wearable and Ambulatory Systems for Health and Care Applications: A Review”. In: *IEEE Access* PP, pp. 1–1. DOI: 10.1109/ACCESS.2022.3206782.

Dolson, Mark (1986). “The Phase Vocoder: A Tutorial”. In: *Computer Music Journal* 10.4, pp. 14–27. ISSN: 01489267, 15315169.

Fischman, Rajmil (1997). “The phase vocoder: theory and practice”. In: *Organised Sound* 2.2, pp. 127–145. DOI: 10.1017/S1355771897009060.

Flanagan, J. L. and R. M. Golden (1966). “Phase vocoder”. In: *The Bell System Technical Journal* 45.9, pp. 1493–1509. DOI: 10.1002/j.1538-7305.1966.tb01706.x.

Flanagan, James L. (1972). “Techniques for Speech Analysis”. In: *Speech Analysis Synthesis and Perception*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 140–204. ISBN: 978-3-662-01562-9. DOI: 10.1007/978-3-662-01562-9_5.

FreeRTOS (2023). *FreeRTOS Real Time Kernel (RTOS)*. https://www.freertos.org/. Version 11.0.1. Real Time Engineers Ltd.

Gholami, Amir et al. (2021). "A Survey of Quantization Methods for Efficient Neural Network Inference". In: *CoRR* abs/2103.13630.

Giannoulis, Dimitrios, Michael Massberg, and Joshua Reiss (July 2012). "Digital Dynamic Range Compressor Design—A Tutorial and Analysis". In: *AES: Journal of the Audio Engineering Society* 60.

Glass, Hannah C., Courtney J. Wusthoff, and Renée A. Shellhaas (2013). "Amplitude-Integrated Electro-encephalography: The Child Neurologist's Perspective". In: *Journal of Child Neurology* 28.10. PMID: 23690296, pp. 1342–1350. DOI: 10.1177/0883073813488663.

Golmohammadi, M. et al. (2017). "Gated recurrent networks for seizure detection". In: *2017 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, pp. 1–5. DOI: 10.1109/SPMB.2017.8257020.

Gomez-Quintana, Sergi et al. (June 2022). "A method for AI assisted human interpretation of neonatal EEG". In: *Scientific Reports* 12.1, p. 10932. ISSN: 2045-2322. DOI: 10.1038/s41598-022-14894-4.

Gómez-Quintana, Sergi et al. (2018). "On sound-based interpretation of neonatal EEG". In: *2018 29th Irish Signals and Systems Conference (ISSC)*, pp. 1–6. DOI: 10.1109/ISSC.2018.8585349.

He, Kaiming et al. (2015). *Deep Residual Learning for Image Recognition*.

He, P., G. Wilson, and C. Russell (May 2004). "Removal of ocular artifacts from electro-encephalogram by adaptive filtering". In: *Medical and Biological Engineering and Computing* 42.3, pp. 407–412. ISSN: 1741-0444. DOI: 10.1007/BF02344717.

Hermann, Thomas (2008). "Taxonomy and definitions for sonification and auditory display". In.

Hermann, Thomas et al. (2002). "Sonifications for EEG data analysis". In: Georgia Institute of Technology.

Immonen, Riku and Timo Hämäläinen (2022). "Tiny Machine Learning for Resource-Constrained Microcontrollers". In: *Journal of Sensors* 2022.1, p. 7437023. DOI: https://doi.org/10.1155/2022/7437023.

Jacob, Benoit et al. (2017). "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference". In: *CoRR* abs/1712.05877.

Jasper, Herbert H. (1958). "Report of the committee on methods of clinical examination in electroencephalography: 1957". In: *Electroencephalography and Clinical Neurophysiology* 10.2, pp. 370–375. ISSN: 0013-4694. DOI: https://doi.org/10.1016/0013-4694(58)90053-1.

Jotterand, Fabrice and Clara Bosco (Oct. 2020). "Keeping the "Human in the Loop" in the Age of Artificial Intelligence". In: *Science and Engineering Ethics* 26.5, pp. 2455–2460. ISSN: 1471-5546. DOI: 10.1007/s11948-020-00241-1.

Kelly, Christopher J. et al. (Oct. 2019). "Key challenges for delivering clinical impact with artificial intelligence". In: *BMC Medicine* 17.1, p. 195. ISSN: 1741-7015. DOI: 10.1186/s12916-019-1426-2.

Kim, Sung-Phil (2018). "Preprocessing of EEG". In: *Computational EEG Analysis: Methods and Applications*. Ed. by Chang-Hwan Im. Singapore: Springer Singapore, pp. 15–33. ISBN: 978-981-13-0908-3. DOI: 10.1007/978-981-13-0908-3_2.

Kitayama, Masaomi et al. (2003). "Wavelet analysis for neonatal electroencephalographic seizures". In: *Pediatric Neurology* 29.4, pp. 326–333. ISSN: 0887-8994. DOI: https://doi.org/10.1016/S0887-8994(03)00277-7.

Kundu, Shinjini (Aug. 2021). "AI in medicine must be explainable". In: *Nature Medicine* 27.8, pp. 1328–1328. ISSN: 1546-170X. DOI: 10.1038/s41591-021-01461-z.

Kuo, Sen M., Bob H. Lee, and Wenshun Tian (June 2006). "Introduction to Real-Time Digital Signal Processing". In: *Real-Time Digital Signal Processing*. John

Wiley & Sons, Ltd. Chap. 1, pp. 1–47. ISBN: 9780470035528. DOI: https://doi.org/10.1002/0470035528.ch1.

Laroche, J. and M. Dolson (1999). "Improved phase vocoder time-scale modification of audio". In: *IEEE Transactions on Speech and Audio Processing* 7.3, pp. 323–332. DOI: 10.1109/89.759041.

Leodolter, Maximilian, Claudia Plant, and Norbert Brändle (2021). "IncDTW: An R Package for Incremental Calculation of Dynamic Time Warping". In: *Journal of Statistical Software* 99.9, pp. 1–23. DOI: 10.18637/jss.v099.i09.

Little, Max A et al. (2017). "Using and understanding cross-validation strategies. Perspectives on Saeb et al." In: *GigaScience* 6.5, gix020.

Liuni, Marco and Axel Roebel (Aug. 2013). "Phase vocoder and beyond". In: *Music/Technology* VII, pp. 73–89. DOI: 10.13128/Music_Tec-13209.

Mardini, Wail et al. (2020). "Enhanced Detection of Epileptic Seizure Using EEG Signals in Combination With Machine Learning Classifiers". In: *IEEE Access* 8, pp. 24046–24055. DOI: 10.1109/ACCESS.2020.2970012.

Mueen, Abdullah and Eamonn Keogh (2016). "Extracting Optimal Performance from Dynamic Time Warping". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, pp. 2129–2130. ISBN: 9781450342322. DOI: 10.1145/2939672.2945383.

Müller, Meinard (2021). "Music Synchronization". In: *Fundamentals of Music Processing: Using Python and Jupyter Notebooks*. Cham: Springer International Publishing, pp. 119–170. DOI: 10.1007/978-3-030-69808-9_3.

Mursalin, Md et al. (2017). "Automated epileptic seizure detection using improved correlation-based feature selection with random forest classifier". In: *Neurocomputing* 241, pp. 204–214. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2017.02.053.

Nagel, Markus et al. (2021). “A White Paper on Neural Network Quantization”. In: *CoRR* abs/2106.08295.

Nakao, Yuri et al. (July 2022). “Toward Involving End-users in Interactive Human-in-the-loop AI Fairness”. In: 12.3. ISSN: 2160-6455. DOI: 10.1145/3514258.

Nogay, Hidir Selcuk and Hojjat Adeli (Jan. 2021). “Detection of Epileptic Seizure Using Pretrained Deep Convolutional Neural Network and Transfer Learning”. In: *European Neurology* 83.6, pp. 602–614. ISSN: 0014-3022. DOI: 10.1159/000512985.

O’Sullivan, Feargal et al. (2022). “An implementation of an AI-assisted sonification algorithm for neonatal EEG seizure detection on an edge device”. In: *2022 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, pp. 01–04. DOI: 10.1109/BHI56158.2022.9926876.

O’Shea, Alison et al. (2020). “Neonatal seizure detection from raw multi-channel EEG using a fully convolutional architecture”. In: *Neural Networks* 123, pp. 12–25. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2019.11.023.

Park, Eunhyeok, Junwhan Ahn, and Sungjoo Yoo (2017). “Weighted-Entropy-Based Quantization for Deep Neural Networks”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7197–7205. DOI: 10.1109/CVPR.2017.761.

Parvizi, Josef et al. (2018). “Detecting silent seizures by their sound”. In: *Epilepsia* 59.4, pp. 877–884. DOI: https://doi.org/10.1111/epi.14043.

Paszke, Adam et al. (2019). “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *CoRR* abs/1912.01703.

Pavel, Andreea M. et al. (2022). “Neonatal Seizure Management: Is the Timing of Treatment Critical?” In: *The Journal of Pediatrics* 243, 61–68.e2. ISSN: 0022-3476. DOI: https://doi.org/10.1016/j.jpeds.2021.09.058.

Petrosian, Arthur et al. (2000). “Recurrent neural network based prediction of epileptic seizures in intra- and extracranial EEG”. In: *Neurocomputing* 30.1,

pp. 201–218. ISSN: 0925-2312. DOI: https://doi.org/10.1016/S0925-2312(99)00126-5.

Pisani, Francesco, Giovanni Prezioso, and Carlotta Spagnoli (2020). "Neonatal seizures in preterm infants: A systematic review of mortality risk and neurological outcomes from studies in the 2000's". In: *Seizure* 75, pp. 7–17. ISSN: 1059-1311. DOI: https://doi.org/10.1016/j.seizure.2019.12.005.

Pisani, Francesco, Carlotta Spagnoli, et al. (2021). "Seizures in the neonate: A review of etiologies and outcomes". In: *Seizure* 85, pp. 48–56. ISSN: 1059-1311. DOI: https://doi.org/10.1016/j.seizure.2020.12.023.

Portnoff, M. (1981). "Time-scale modification of speech based on short-time Fourier analysis". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 29.3, pp. 374–390. DOI: 10.1109/TASSP.1981.1163581.

Prusa, Zdenek and Nicki Holighaus (2022). *Phase Vocoder Done Right*.

Ramantani, Georgia et al. (July 2019). "Neonatal Seizures—Are We there Yet?" In: *Neuropediatrics* 50. DOI: 10.1055/s-0039-1693149.

Reinecke, Martin (2024). *PocketFFT for C++*. https://github.com/mreineck/pocketfft.

Rokh, Babak, Ali Azarpeyvand, and Alireza Khanteymoori (Nov. 2023). "A Comprehensive Survey on Model Quantization for Deep Neural Networks in Image Classification". In: 14.6. ISSN: 2157-6904. DOI: 10.1145/3623402.

Saeb, Sohrab, Luca Lonini, Arun Jayaraman, David Mohr, et al. (June 2016). "Voodoo Machine Learning for Clinical Predictions". In: *BioRxiv*. DOI: 10.1101/059774.

Saeb, Sohrab, Luca Lonini, Arun Jayaraman, David C. Mohr, et al. (Mar. 2017). "The need to approximate the use-case in clinical machine learning". In: *GigaScience* 6.5, gix019. ISSN: 2047-217X. DOI: 10.1093/gigascience/gix019.

Sakoe, H. and S. Chiba (1978). "Dynamic programming algorithm optimization for spoken word recognition". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26.1, pp. 43–49. DOI: 10.1109/TASSP.1978.1163055.

Shellhaas, Renée et al. (2020). "Neonatal seizures: diagnosis, etiologies, and management". In: *Seminars in neurology*. Vol. 40. 02. Thieme Medical Publishers 333 Seventh Avenue, New York, NY 10001, USA., pp. 246–256. DOI: 10.1055/s-0040-1702943.

Shellhaas, Renée A. et al. (2017). "Profile of neonatal epilepsies". In: *Neurology* 89.9, pp. 893–899. DOI: 10.1212/WNL.0000000000004284.

Shoeb, Ali and John Guttag (Aug. 2010). "Application of Machine Learning To Epileptic Seizure Detection". In: pp. 975–982.

Shoeibi, Afshin, Navid Ghassemi, et al. (2022). "Detection of epileptic seizures on EEG signals using ANFIS classifier, autoencoders and fuzzy entropies". In: *Biomedical Signal Processing and Control* 73, p. 103417. ISSN: 1746-8094. DOI: https://doi.org/10.1016/j.bspc.2021.103417.

Shoeibi, Afshin, Parisa Moridian, et al. (2022a). "An overview of deep learning techniques for epileptic seizures detection and prediction based on neuroimaging modalities: Methods, challenges, and future works". In: *Computers in Biology and Medicine* 149, p. 106053. ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2022.106053.

— (2022b). "An overview of deep learning techniques for epileptic seizures detection and prediction based on neuroimaging modalities: Methods, challenges, and future works". In: *Computers in Biology and Medicine* 149, p. 106053. ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2022.106053.

Siddiqui, Mohammad Khubeb et al. (May 2020). "A review of epileptic seizure detection using machine learning classifiers". In: *Brain Informatics* 7.1, p. 5. ISSN: 2198-4026. DOI: 10.1186/s40708-020-00105-1.

Simonyan, Karen and Andrew Zisserman (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition*.

Software, Arm (2023). *CMSIS-DSP 1.15.0*. https://github.com/ARM-software/CMSIS-DSP.

Stevenson, Nathan J et al. (2019). "A dataset of neonatal EEG recordings with seizure annotations". In: *Scientific data* 6.1, pp. 1–8.

Subasi, Abdulhamit, Jasmin Kevric, and M. Abdullah Canbaz (Jan. 2019). "Epileptic seizure detection using hybrid machine learning methods". In: *Neural Computing and Applications* 31.1, pp. 317–325. ISSN: 1433-3058. DOI: 10.1007/s00521-017-3003-y.

Tavenard, Romain (2021). *An introduction to Dynamic Time Warping*. https://rtavenar.github.io/blog/dtw.html.

Temko, A. et al. (2011). "EEG-based neonatal seizure detection with Support Vector Machines". In: *Clinical Neurophysiology* 122.3, pp. 464–473. ISSN: 1388-2457. DOI: https://doi.org/10.1016/j.clinph.2010.06.034.

Temko, Andriy et al. (2014). "Neonatal EEG audification for seizure detection". In: *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 4451–4454. DOI: 10.1109/EMBC.2014.6944612.

Thuwajit, Punnawish et al. (2022). "EEGWaveNet: Multiscale CNN-Based Spatiotemporal Feature Extraction for EEG Seizure Detection". In: *IEEE Transactions on Industrial Informatics* 18.8, pp. 5547–5557. DOI: 10.1109/TII.2021.3133307.

Truong, Nhan Duy et al. (2019). "Epileptic Seizure Forecasting With Generative Adversarial Networks". In: *IEEE Access* 7, pp. 143999–144009. DOI: 10.1109/ACCESS.2019.2944691.

Tsuchida, Tammy N. et al. (Apr. 2013). "American Clinical Neurophysiology Society Standardized EEG Terminology and Categorization for the Description

of Continuous EEG Monitoring in Neonates: Report of the American Clinical Neurophysiology Society Critical Care Monitoring Committee". In: *Journal of Clinical Neurophysiology* 30.2, pp. 161–173. ISSN: 0736-0258. DOI: 10.1097/wnp.0b013e3182872b24.

UNIGME (2021). *Mortality rate, neonatal (per 1,000 live births)*.

Väljamäe, Anastasiia et al. (2013). "A review of real-time EEG sonification research". In: *International Conference on Auditory Display 2013 (ICAD 2013)*. Lodz, Poland, pp. 85–93.

Vasudevan, Chakrapani and Malcolm Levene (2013). "Epidemiology and aetiology of neonatal seizures". In: *Seminars in Fetal and Neonatal Medicine* 18.4. Neonatal seizures, pp. 185–191. ISSN: 1744-165X. DOI: https://doi.org/10.1016/j.siny.2013.05.008.

Vintsyuk, T. K. (Jan. 1968). "Speech discrimination by dynamic programming". In: *Cybernetics* 4.1, pp. 52–57. ISSN: 1573-8337. DOI: 10.1007/BF01074755.

Wang, Xiashuang et al. (2019). "Detection Analysis of Epileptic EEG Using a Novel Random Forest Model Combined With Grid Search Optimization". In: *Frontiers in Human Neuroscience* 13. ISSN: 1662-5161. DOI: 10.3389/fnhum.2019.00052.

White, Diane M. and C. Anne Van Cott (2010). "EEG Artifacts in the Intensive Care Unit Setting". In: *American Journal of Electroneurodiagnostic Technology* 50.1. PMID: 20361716, pp. 8–25. DOI: 10.1080/1086508X.2010.11079750.

You, Sungmin et al. (2020). "Unsupervised automatic seizure detection for focal-onset seizures recorded with behind-the-ear EEG using an anomaly-detecting generative adversarial network". In: *Computer Methods and Programs in Biomedicine* 193, p. 105472. ISSN: 0169-2607. DOI: https://doi.org/10.1016/j.cmpb.2020.105472.

Yun, Sangdoo et al. (2019). "CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features". In: *CoRR* abs/1905.04899.

Zhang, Tao and Wanzhong Chen (2017). "LMD Based Features for the Automatic Seizure Detection of EEG Signals Using SVM". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 25.8, pp. 1100–1108. DOI: 10.1109/TNSRE.2016.2611601.

Zhou, Mengni et al. (2018). "Epileptic Seizure Detection Based on EEG Signals and CNN". In: *Frontiers in Neuroinformatics* 12. ISSN: 1662-5196. DOI: 10.3389/fninf.2018.00095.

Zolzer, Udo (2011). *DAFX: Digital Audio Effects*. 2nd. Wiley Publishing. ISBN: 0470665998.