PERSONALIZATION OF NOISE REDUCTION AND COMPRESSION

FOR HEARING ENHANCEMENT

by

Nasim Taghizadeh Alamdari

APPROVED BY SUPERVISORY COMMITTEE:

Nasser Kehtarnavaz, Chair

Edward Lobarinas

Carlos Busso-Recabarren

Raja Rajasekaran

To my parents, who sacrificed their priorities and

supported me unconditionally to have a life I love today.

PERSONALIZATION OF NOISE REDUCTION AND COMPRESSION FOR HEARING ENHANCEMENT

by

NASIM TAGHIZADEH ALAMDARI, BS, MS

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT DALLAS

May 2021

ACKNOWLEDGMENTS

First, I would like to thank my family for all their love and encouragement. This dissertation is dedicated to my parents who supported me unconditionally.

I express my gratitude to the Department of Electrical and Computer Engineering of The University of Texas at Dallas, for giving me the opportunity to research and pursue a PhD. My solemn gratitude to my PhD advisor, Dr. Nasser Kehtarnavaz, who has been a philosopher and guide to me. While being my supervisor in this academic period, he has enriched and trained me with vast knowledge and competitive skills.

I am very much grateful to my committee members Dr. Edward Lobarinas, Dr. Carlos Busso, and Dr. Raja Rajasekaran for their feedback and for being members of my committee. Special thanks to Dr. Edward Lobarinas for his time and lab support during this research.

I am also indebted to my colleagues Tina Campbell, Dr. Celia Escabi, and Ali Salman for their valuable help. My appreciation also goes to my graduate student peers Arian Azarang, Haoran Wei, Fatemeh Saki, Abhishek Sehgal, and Neha Dawar for always being there for me and making this journey joyful.

February 2021

PERSONALIZATION OF NOISE REDUCTION AND COMPRESSION

FOR HEARING ENHANCEMENT

Nasim Taghizadeh Alamdari, PhD
The University of Texas at Dallas, 2021

Supervising Professor: Dr. Nasser Kehtarnavaz, Chair

Noise reduction and dynamic range compression constitute two main signal processing modules for hearing enhancement in modern digital hearing aid devices. This dissertation covers personalization solutions for both of these two modules. First, in contrast to the great majority of previous works that are designed based on pre-collected datasets, in this dissertation, a personalized noise reduction solution is developed for field deployment which is capable of dealing with unseen noisy audio environments. More specifically, a deep learning-based approach is devised to improve speech denoising in real-world audio environments by not requiring the availability of clean speech signals as reference. A fully convolutional neural network is designed based on two noisy realizations of the same speech signal, one used as the input and the other as the target of the network. The results of extensive experimentations are reported to show the superiority of the developed personalized deep learning-based speech denoising approach over existing approaches. In support of personalized noise reduction, a personalized noise classification is also developed in this dissertation by performing the noise classification in an unsupervised

manner. Second, in contrast to the existing prescriptive compression strategies used in hearing aids which are devised based on gain averages from a group of users, a personalized compression solution is developed in this dissertation via a human-in-the-loop deep reinforcement learning approach. The developed approach is designed to learn a specific user's hearing preferences in order to set compression ratios based on the user's preference feedbacks. Both simulation and subject testing results are reported to show the superiority of the developed personalized deep learning-based compression over conventional prescriptive compression. In addition, four smartphone apps in support of the above two modules are developed in this dissertation which include the real-time implementation of noise classification, noise reduction, and dynamic range compression.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

According to the World Health Organization (WHO), more than 450 million people around the world have some level of hearing loss and it is estimated that by 2050, this number would grow to more than 900 million people. Many who suffer from hearing loss are prescribed to use hearing aids.

Two main functions or signal processing components in digital hearing aids is noise reduction/suppression and dynamic range compression. The hypothesis that is examined in this dissertation is whether personalization of noise reduction and personalization of compression would lead to improved hearing as compared to the approaches that are commonly being used for noise reduction and compression.

In recent years, deep learning approaches have been increasingly used for noise reduction in contrast to conventional approaches such as Wiener filtering. The existing deep learning approaches normally attempt to establish the nonlinear relationship between noisy and clean speech signals via a deep neural network to recover clean speech signals. A major assumption made in these approaches is the availability of noise-free (clean) speech signals for training deep neural networks. In practice, the problem of noise reduction or speech denoising is more challenging due to the fact that clean speech signals are not known or available when operating in the field or in actual audio environments. In addition, the generalization capability of a trained network for all types of speakers and noise environments poses limitations in practice. This dissertation addresses the personalization of noise reduction by easing the requirement of having access to clean speech signals.

In order to have a more effective noise reduction, the ability to adapt to different noise types in an on-the-fly manner is needed. In most existing noise classifiers, the classification is carried out in a supervised manner to select a noise type or class among several previously trained noise classes. The training process of such classifiers involves first carrying out data collection and then running an offline training algorithm on the collected data. A major shortcoming of supervised noise classifiers is that they are not capable of coping with noise environments for which no training is done. As a result, the personalization of such classifiers to the noise environments encountered by a specific user cannot be achieved with ease since the training process needs to be repeated when a new noise environment is encountered. This dissertation addresses the personalization of noise classification as part of the personalization of noise reduction.

In addition to noise reduction, the other main signal processing component of digital hearing aids is dynamic range compression. The compression process involves squeezing or fitting sound into the residual audibility range of a hearing aid user. In hearing aid fitting, so-called compression curves are set up by adjusting gains across a number of frequency bands based on a user's audiometric profile. It has been reported that up to half of individuals using fitted hearing aids prefer amplification or compression settings different than the prescription provided. Considering that suprathreshold hearing perception varies from person to person and that acoustic environments encountered vary from person to person, having an approach to personalize dynamic range compression can improve hearing. This dissertation also addresses the personalization of dynamic range compression or amplification for hearing aids.

In essence, the contributions of this dissertation lie in the development of machine learning solutions to personalize the two main modules of hearing aids: noise reduction and compression.

Since hearing perception varies widely among hearing aid users, it is hypothesized that the personalization of these modules leads to gaining more benefits out of hearing enhancement devices.

The chapters of this dissertation are organized based on two journal and four conference papers that have already been published, with each chapter or paper addressing the dissertation objective from a different perspective. Each chapter provides an abstract, an introduction, a methodology, a discussion of the results obtained, and a conclusion.

More specifically, Chapter 2 presents a personalized noise reduction solution by integrating a voice activity detector as part of a Wiener filter noise reduction speech processing pipeline. This solution has been implemented on smartphones as a real-time low-latency app to allow its operation in the field.

Chapters 3 and 4 cover the details of an unsupervised noise classifier apps developed for the purpose of identifying different noisy environments in an on-the-fly manner as part of the noise reduction pipeline. The personalization of noise reduction is achieved by allowing users to adjust the gains across five frequency bands in noisy environments encountered by a user which are identified and labeled by the unsupervised noise classifier.

Chapter 5 discusses a deep learning-based speech denoising approach, named noisy2noisy, that is developed to enable personalization of noise reduction and thus to achieve improved hearing in real-world environments. The developed approach performs speech denoising without requiring the availability of noise-free speech signals for training, which in practice are not available for various speakers and noise environments that get encountered in the field.

Chapter 6 describes the details of a developed open-source and easy-to-use web-based software tool for dynamic range compression. As part of this tool, a smartphone app is developed to allow the field testing of prescriptive compression strategies.

Chapter 7 introduces a human-in-the-loop deep reinforcement learning framework for personalization of dynamic range compression used in hearing aids. The feedbacks from a user is placed in the learning loop for personalization of compression. A combination of a convolutional neural network and a bidirectional long short-term memory recurrent neural network is utilized to model a user's preferences in those audio environments that are of interest to the user. Then, the compression ratios in different frequency bands are set via deep reinforcement learning.

Finally, Chapter 8 summarizes the dissertation contributions and mentions possible future extensions.

CHAPTER 2

A REAL-TIME PERSONALIZED NOISE REDUCTION SMARTPHONE APP FOR HEARING ENHANCEMENT[*]

Authors – Nasim Alamdari, Shashank Yaraganalu , and Nasser Kehtarnavaz

The Department of Electrical and Computer Engineering,

The University of Texas at Dallas

800 West Campbell Road

Richardson, Texas 75080-3021

[*] ©(2018) IEEE. Reprinted, with permission, from (Nasim Alamdari, Shashank Yaraganalu, and Nasser Kehtarnavaz, "A Real-Time Personalized Adaptive Noise Reduction Smartphone App for Realistic Audio Environments", In IEEE Signal Processing in Medicine and Biology Symposium (SPMB), pp. 1-5. IEEE, 2018.

2.1 Abstract

This paper presents the development of a personalized noise reduction app that is designed to run in real-time with low-latency on smartphone platforms for hearing enhancement purposes. The personalization is achieved by using an unsupervised noise classifier together with a personalized gain adjustment. After applying a Wiener filtering noise reduction, gains in five frequency bands are adjusted by the user to achieve personalized noise reduction depending on the noise environment identified by the classifier. The other signal processing modules of the app include voice activity detection and compression. Publicly available datasets of speech signals and commonly encountered noise signals are used to test the effectiveness of the app. The results obtained by computing a widely used objective speech quality measure indicate the effectiveness of the app for noise reduction.

2.2 Introduction

The use of smartphones for medical applications has been steadily growing in recent years. One of these applications involves enabling a more effective hearing in noisy environments, in particular for those suffering from hearing loss. About 5% of the world's population suffer from some form of hearing loss [1]. By simply running apps on smartphones and interfacing those apps with hearing aids, either in a wired or wireless manner, it is possible to provide this population with an enhanced hearing experience.

A commercial example where smartphones are used to enable better hearing in noisy environments is the so-called Live Listen feature offered by Apple [2]. When this feature is enabled, an enhanced hearing is experienced by hearing aid users. A noise reduction app running on smartphones can be

used both by those having normal hearing and by those suffering from hearing loss. These days Bluetooth enabled hearing aids, e.g. Oticon Opn [3] or Starkey Halo [4], can be connected to smartphones to receive smartphone processed sound signals captured via their microphones.

In [5], an adaptive noise reduction smartphone app was developed by our research lab by integrating three signal processing modules encountered in digital hearing aids consisting of a voice activity detection (VAD) module, a Wiener filtering noise reduction with postfiltering module, and a compression module. This noise reduction app provided an improved performance over an earlier noise reduction app reported in [6] where the noise estimation was carried out once without any adaptation to changes in the signal-to-noise ratio (SNR) that is often experienced in realistic noise environments. By separating noise frames from speech activity frames, the noise estimation was conducted continuously and adaptively thus the variations in the SNR was taken into consideration when operating in realistic audio environments.

The work presented in this paper involves adding a personalization capability to the noise reduction app in [5] and making its code open source. This personalization is done by adding an unsupervised noise classifier to select a personalized set of gains in five frequency bands for those noise environments that are encountered by or are of interest to a specific user. The unsupervised nature of the classifier allows the handling of big data of noise signals that are associated with various noise environments.

The rest of the paper is organized as follows: Section 2.3 provides an overview of the modules that are integrated to enable a personalized noise reduction speech processing pipeline. In Section 2.4, the implementation aspects of the app are discussed. The noise reduction results obtained while

running the app in real-time are then presented in Section 2.5, followed by the conclusion in Section 2.6.

2.3 Personalized Noise Reduction Pipeline

Figure 2.1 shows the block diagram of the developed personalized noise reduction speech processing pipeline. This pipeline consists of both i/o and signal processing modules. The i/o modules include input and output circular buffering, lowpass filtering, downsampling, upsampling, interpolation filtering, and amplification. The signal processing modules include VAD, unsupervised noise classification, Weiner filtering noise reduction with postfiltering, personalized gain adjustment, and compression.

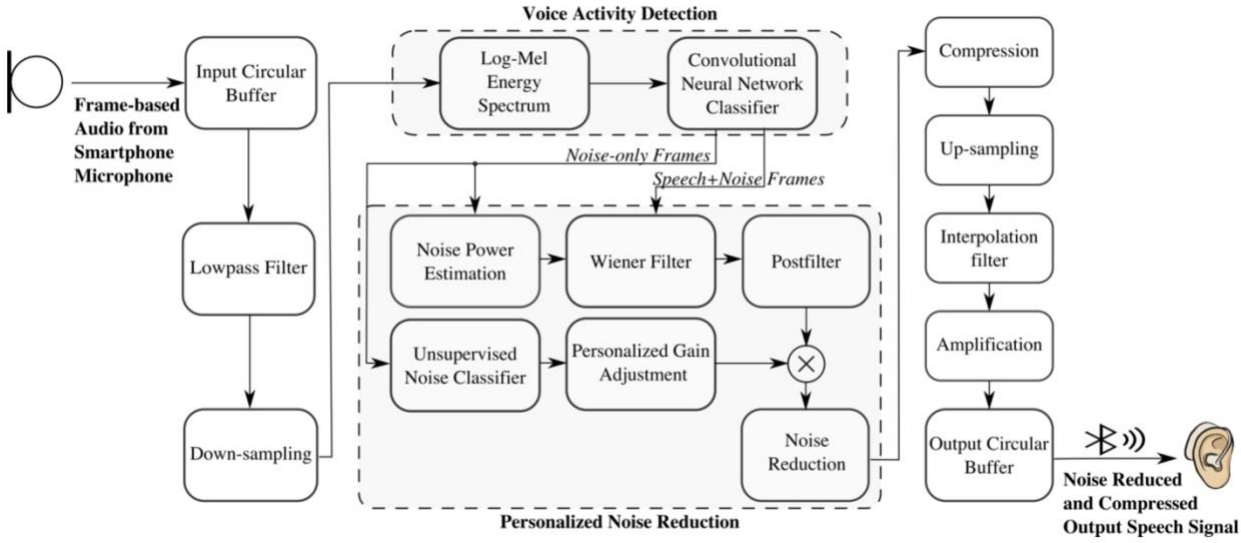


Fig. 2.1. Block diagram of the real-time low-latency personalized noise reduction (PNR) app.

In order to capture and playback audio frames with the lowest latency offered by the i/o hardware of smartphones, as discussed in [7], an input and an output circular buffer are used to process a desired audio frame size. To enable computational efficiency or real-time throughput, the sampling rate is reduced from the lowest latency sampling frequency of 48kHz to 16kHz before frames are

passed through the signal processing modules. This is done by first using a lowpass filter and then by performing a factor 3 down-sampling. To playback processed audio frames through the smartphone speaker, the sampling rate is increased back to 48kHz to retain the lowest latency. This is done by up-sampling or placing 2 zeros between consecutive samples followed by an interpolation lowpass filter.

The first signal processing module is VAD. This module is used to separate noise only frames from frames containing speech or from noisy speech frames. When the VAD output denotes noise-only frames or pure noise, noise estimation is carried out during these frames and a moving average of the noise power is computed to take into consideration variations of the SNR for noise reduction during the speech activity or noisy speech frames. The VAD used in the pipeline is the one developed in [8], which comprises two submodules: one submodule involves formation of images out of log-mel short-time Fourier transforms (STFT) or log-mel spectrograms and the other submodule involves a convolutional neural network (CNN) classifier.

The output of the VAD corresponds to three states of speech+noise, noise, and quiet. The quiet state is considered when the audio signal power is below a user specified sound pressure level (SPL) denoting a quiet audio environment for the user. To avoid fluctuations between speech activity frames of spoken sentences, a smoothing buffer is considered in the developed app so that a sufficient number of frames are seen for switching from the speech+noise state to the noise or quiet state.

The noise reduction module is the one reported in [5] where a Wiener filter is used by carrying out an estimation of the speech and noise powers. To reduce the musical noise artifact introduced by Wiener filtering, a postfilter is utilized as described in [6].

The noise reduction module is followed by the compression module which is a key signal processing component in digital hearing aids. Based on compression curves for five frequency bands, this module brings the sound pressure level into the hearing range of those suffering from hearing loss. The compression module consists of the multiband Dynamic Range Compression (DRC) in [9] in which five compression curves are used corresponding to the five frequency bands of [0Hz-500Hz], [500Hz-1000Hz], [1000Hz-2000Hz], [2000Hz-4000Hz], and above 4000Hz. The reason for using different frequency bands is that hearing loss is different in different frequency bands. Therefore, different amounts of compression need to be applied in different frequency bands. To personalize the app for a specific user, an unsupervised noise classifier module together with a personalized gain adjustment module are added to the above speech processing pipeline. These modules are described next.

2.3.1 Unsupervised Noise Classifier

As part of the personalization of the app, an unsupervised noise classifier is included in the pipeline to identify the noise environments encountered by a specific user in which he/she is having difficulty hearing. These noise environments are classified or identified in an online manner without the need to carry out any supervised training. The unsupervised noise classifier module in [10] is used here due to its effectiveness in realistic noise environments. This unsupervised classifier fuses the decisions of two ART2 (adaptive resonance theory 2) unsupervised classifiers. One classifier uses subband features as described in [11] and the other uses mel-frequency spectral coefficients (MFSC) features as described in [8]. It is worth mentioning that the unsupervised noise classifier app developed in [11] may also be used here.

2.3.2 Personalized Gain Adjustment

For each noise environment identified by the unsupervised noise classifier, the user is given the option to set the gains in the five frequency bands that are used in the compression module. This gain adjustment module is similar to an equalizer by which frequencies are suppressed or amplified depending on the hearing preference of the user for having a better hearing in that noise environment. These gains can be set whenever a new noise environment is encountered and identified by the unsupervised noise classifier. In other words, this module implements the following gain adjustment equation:

$$G_{PF,noise\ type}(f,k) = G_{PF}(f,k).\alpha_{noise\ type}(f) \quad (1)$$

where $G_{PF}(f,k)$ denotes the noise suppression gain at kth frame obtained by Wiener filtering at frequency bin f, $\alpha_{noise\ type}(f)$ denotes the gain entered by a specific user for the noise type identified by the unsupervised classifier, and $G_{PF,noise\ type}(f,k)$ denotes the gains that enable the personalization of the noise reduction outcome. To obtain $\alpha_{noise\ type}(f)$ for every frequency f, the gains across the center frequencies of the five bands are set by the user between 0.1 and 2.0 via five slider bars. Then, a cosine function interpolation is carried out based on these five gains so that a gain is generated for every frequency.

2.4 Implementation Aspects

Both Android and iOS versions of the personalized noise reduction app are developed in this work. All the codes corresponding to different i/o and signal processing modules are coded in C and then are incorporated into the software shells developed in [12] for running them as apps on Android and iOS smartphones. The two smartphones of Google Pixel and iPhone 8 are used in the

experiments reported in the next section. It is to be noted that the coding is done in a modular way. As a result, each of the signal processing modules can be easily replaced by other modules implementing the same signal processing functions.

The i/o audio setup is performed based on the CoreAudio API [13] for iOS smartphones and based on the Superpowered SDK [14] for Android smartphones. The duration of audio frames is 25ms with 50% overlap between consecutive frames. In general, iPhones exhibit a lower audio latency (10ms-15ms) as compared to Android smartphones. For instance, the Google Pixel Android smartphone have an audio latency up to 40ms. This latency varies among different Android smartphones.

In order to capture audio frames with the lowest audio latency, signal samples need to be acquired 64 samples at a time with the sampling frequency of 48kHz on iPhones. This means that frames need to get processed within 64/48kHz = 1.3ms to allow real-time throughput or for no frames getting skipped. For Google Pixel, 192 samples need to be acquired at a time with the sampling frequency of 48kHz. This means that frames need to get processed within 192/48kHz = 4.0ms to allow real-time throughput or for no frames getting skipped.

Figure 2.2 illustrates the main graphical user interface (GUI) page of the personalized noise reduction (PNR) app. This page shows two switches for turning off and on the functions of noise reduction and compression. Each of the noise reduction and compression settings activates a new page for setting parameters associated with these functions. The other entries on the app main page include the frame processing time, the output state of the VAD, and the unsupervised noise classifier outcome. The buttons appearing at the bottom of the page denote Live Audio for running

the app in actual noise environments and Process File for processing noisy sound files stored in memory.

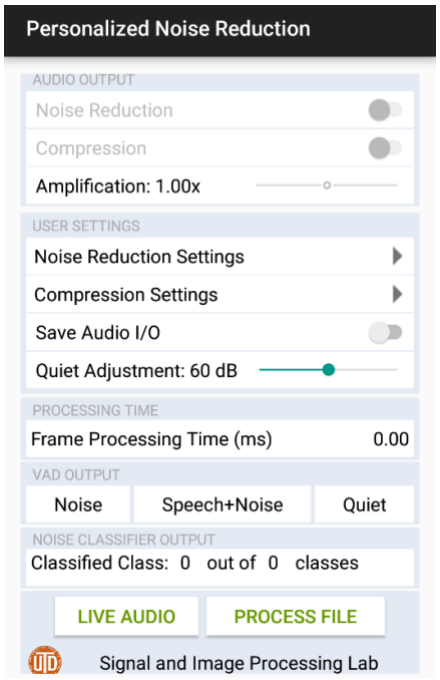


Fig. 2.2. Main GUI page of the personalized noise reduction app.

Figure 2.3 shows the personalization page of the app. On this page, the gains in the five frequency bands per noise class can be adjusted by the user. Similar to an equalizer, the gains for a noise environment of particular interest to a specific user can get adjusted on-the-fly while the app runs in real-time for the purpose of making the noise reduction more effective for that user. The so-called vigilance parameters of the ART2 classifiers can also be set on this page. The amount of time waited before a new noise class is created is an entry on this page for the purpose of adding stability of classification and avoiding the creation of noise classes for transient types of sounds. In addition, the decision smoothing time indicated on this page adds stability to the classification outcome by avoiding fluctuations in realistic noise environments.

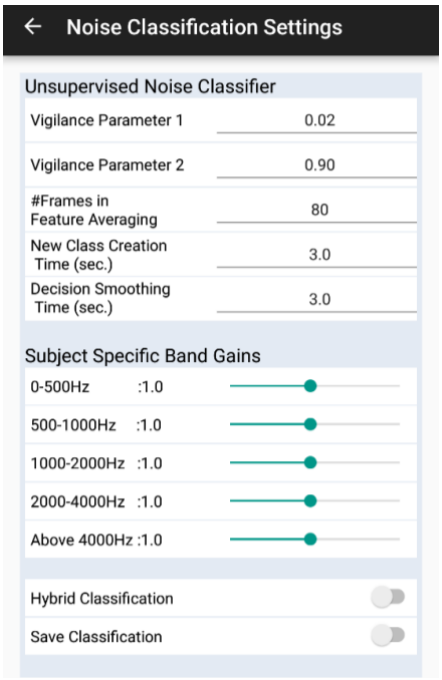


Fig. 2.3. Noise classification settings and personalized gains adjustment GUI page.

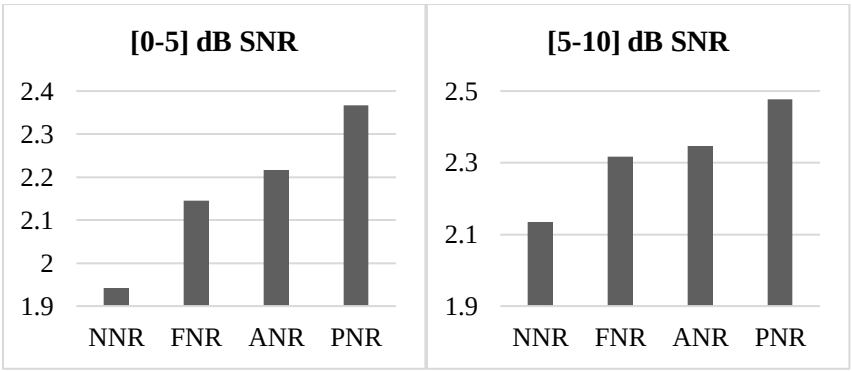
Furthermore, two switches called Hybrid Classification and Save Classification are included for the operation of the unsupervised classifier in its hybrid mode and for saving classifier parameters, respectively. The hybrid mode of classification enables the classifier to use the previously identified noise classes during previous runs of the app and their corresponding gains as specified by the user instead of starting from scratch or with no noise class to begin with.

The app operates in two modes. In its first mode, the app allows the user to set the personalized gains in the five frequency bands to his/her liking in the noise environments in which he/she is having difficulty hearing speech. In its second mode, the app performs personalized noise reduction whenever the user encounters those noise environments. The app has the ability to add to the number of noise classes when a new noise environment is encountered which is different than the previously encountered noise environments or classes.

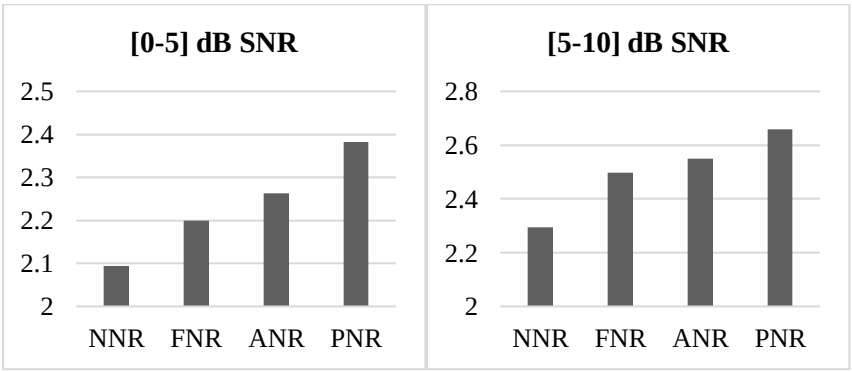
2.5 Results and Discussion

Two public domain datasets, one dataset consisting of clean speech files and the other dataset consisting of environmental noise files, were used to evaluate the performance of the developed personalized noise reduction app. The PN/NC speech dataset [15] was used here which consists of 3600 speech files with each file being about 2 seconds long. This dataset incorporates 20 different speakers (10 females, and 10 males) from two American English regions of the Northern Cities (NC) and the Pacific Northwest (PN), reading the IEEE "Harvard" sentences. These files were concatenated to create a 2-hour long speech file. Next, three bothersome noise files in the 2018 TUT Urban Acoustic Scenes dataset [16] consisting of babble, street traffic, and tram noises were selected. These noise files were recorded by three mobile devices. All the noise files (a total of 120 files) having different SNRs in the range of [0-10] dB were then concatenated and added to the aforementioned speech file to create a 2-hour long noisy speech file.

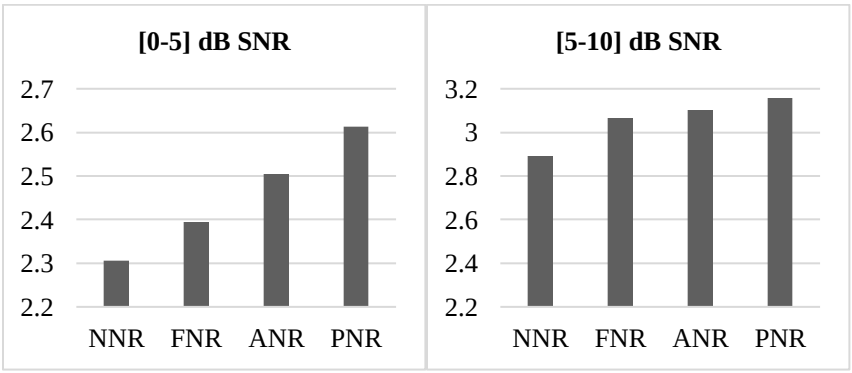
In order to compare the result of no noise reduction (NNR), the fixed noise reduction (FNR) app reported in [6], and the adaptive noise reduction (ANR) app reported in [5] with the developed personalized noise reduction (PNR) app, the Process File button was used to run each testing file through the apps. By activating the save Audio I/O switch, and then by pressing the Process File button, the noisy speech file was processed, and the output of the app was saved. The above 2-hour long noisy speech file was passed through each of the NNR, FNR, ANR, and PNR apps. In order to assess the performance of the personalized noise reduction app, the widely used measure of Perceptual Evaluation of Speech Quality (PESQ) [17] was then computed. Figure 2.4 shows the obtained PESQ measure averaged across the 2-hour long files for each noise type and for two different SNR ranges: [0-5] dB, and [5-10] dB.



(a) PESQ for speech corrupted with babble noise with different SNRs in the range of [0-5] dB (left), and in the range of [5-10] dB (right)



(b) PESQ for speech corrupted with street traffic noise with different SNRs in the range [0-5] dB (left), and in the range of [5-10] dB (right)



(c) PESQ for speech corrupted with tram noise with different SNRs in the range [0-5] dB (left), and [5-10] dB (right)

Fig. 2.4. PESQ measure for no noise reduction (NNR), fixed noise reduction (FNR), adaptive noise reduction (ANR), and personalized noise reduction (PNR) apps.

As can be seen from Figure 2.4, the PESQ values of the personalized noise reduction (PNR) app was found to be consistently higher than the other apps in all the three noise environments and in both of the SNR ranges. The processing time of the developed PNR app per frame is about 1.4ms on Google Pixel, which is less than 4ms frame time, and 0.75ms on iPhone 8, which is less than 1.3ms frame time. These processing times are comparable to the processing times of the ANR app, which are 0.73ms on iPhone 8 and 1.25ms on Google Pixel. Both versions of the app run in real-time with no frames getting skipped. The CPU and memory utilization of the app as obtained by the Android Studio IDE [18] and Xcode IDE [19] are listed in Table 2.1. As can be seen from this table, the CPU utilization is comparable to a typical smartphone app. For Android smartphones, it is worth mentioning that in order to measure the correct CPU utilization, the sustained performance mode of the Superpowered package is required to be changed from the default mode of active to de-active as otherwise an erroneous CPU utilization would be obtained. A video clip of the personalized noise reduction app running in real-time can be viewed at this link: www.utdallas.edu/~kehtar/PersonalizedNRapp.mp4.

Table 2.1. CPU and memory utilization of the developed personalized noise reduction app.

Personalized noise reduction app		
App version	**CPU**	**Memory**
Android	13%	78 MB
iOS	11%	2 B

2.6 Conclusion

A personalized noise reduction smartphone app running in real-time with low-latency has been developed in this paper. The personalization of the app is made possible by allowing the user to

adjust five gains in five frequency bands for bothersome noise environments to that user that are identified in an online manner. Two public domain datasets of speech and noise signals were used to evaluate the app for noise reduction. The results of an objective speech quality measure have indicated the effectiveness of noise reduction when using this personalized noise reduction app as compared to the previously developed noise reduction apps. This app is made open source for public use through the GitHub code hosting service.

2.7 References

[1] World Health Organization, http://www.who.int/mediacentre/factsheets/fs300/en/

[2] Apple, https://support.apple.com/en-us/HT203990

[3] Oticon, https://www.oticon.com/solutions/opn

[4] Starkey Hearing Technologies, https://www.starkey.com/hearing-aids/technologies/halo-2-made-for-iphone-hearing-aids

[5] T. Chowdhury, A. Sehgal, and N. Kehtarnavaz, "Integrating Signal Processing Modules of Hearing Aids into a Real-Time Smartphone App," Proceedings of IEEE 40th International Conference on Engineering in Medicine and Biology, Honolulu, HI, July 2018.

[6] A. Bhattacharya, A. Sehgal, and N. Kehtarnavaz. "Low-latency smartphone app for real-time noise reduction of noisy speech signals," Proceedings of IEEE Symposium on Industrial Electronics Symposium, Edinburgh, Scotland, June 2017.

[7] A. Sehgal and N. Kehtarnavaz, "Utilization of two microphones for real-time low-latency audio smartphone apps," Proceedings of IEEE International Conference on Consumer Electronics, Las Vegas, NV, Jan 2018.

[8] A Sehgal, N Kehtarnavaz, "A Convolutional Neural Network Smartphone App for Real-Time Voice Activity Detection," IEEE Access, vol. 6, pp. 9017-9026, 2018.

[9] MathWorks, https://www.mathworks.com/help/audio/examples/multiband-dynamic-range-compression.html

[10] N. Alamdari, and N. Kehtarnavaz, "A Real-Time Smartphone App for Unsupervised Noise Classification in Realistic Audio Environments," to appear in Proceedings of IEEE International Conference on Consumer Electronics, Las Vegas, NV, Jan 2019.

[11] N. Alamdari, F. Saki, A. Sehgal, and N. Kehtarnavaz, "An unsupervised noise classification smartphone app for hearing improvement devices," Porceedings of IEEE Signal Processing in Medicine and Biology Symposium, Philadelphia, PA, December 2017.

[12] N. Kehtarnavaz, S. Parris, A. Sehgal, Smartphone-Based Real-Time Digital Signal Processing, Morgan and Claypool Publishers, 2015.

[13] Apple, https://developer.apple.com/documentation/coreaudio

[14] Superpowered, http://superpowered.com

[15] D. McCloy, P. Souza, R. Wright, J. Haywood, N. Gehani, and S. Rudolph, PN/NC Corpus Version 1.0, https://depts.washington.edu/phonlab/resources/pnnc/pnnc1/

[16] IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events, http://dcase.community/challenge2018/task-acoustic-scene-classification

[17] P. Loizou, Speech Enhancement: Theory and Practice, CRC Press, Second Edition, 2013.

[18] Google, https://developer.android.com

[19] Apple, https://developer.apple.com

CHAPTER 3

A REAL-TIME SMARTPHONE APP FOR UNSUPERVISED NOISE CLASSIFICATION IN REALISTIC AUDIO ENVIRONMENTS*

Authors – Nasim Alamdari, and Nasser Kehtarnavaz

The Department of Electrical and Computer Engineering,

The University of Texas at Dallas

800 West Campbell Road

Richardson, Texas 75080-3021

3.1 Abstract

This paper presents a real-time unsupervised noise classifier smartphone app which is designed to operate in realistic audio environments. This app addresses the two limitations of a previously developed smartphone app for unsupervised noise classification. A voice activity detection is added to separate the presence of speech frames from noise frames and thus to lower misclassifications when operating in realistic audio environments. In addition, buffers are added to allow a stable operation of the noise classifier in the field. The unsupervised noise classification is achieved by fusing the decisions of two adaptive resonance theory unsupervised classifiers running in parallel. One classifier operates on subband features and the other operates on mel-frequency spectral coefficients. The results of field testing indicate the effectiveness of this unsupervised noise classifier app when used in realistic audio environments.

3.2 Introduction

In our previous works [1-4], a number of smartphone apps for various components of the speech processing pipeline of digital hearing aids have been developed. One of these components is noise reduction since a major complaint of hearing aid users is the difficulty of hearing in noisy audio environments [5]. In order to have a more effective noise reduction smartphone app, the ability to adapt to different noise types in an on-the-fly manner is needed. In other words, a noise classifier is required in order to achieve a more effective noise reduction.

In most existing noise classifiers, the classification is carried out in a supervised manner to select a noise type or class among several previously trained noise classes. The training process of such classifiers involves first carrying out data collection and then running an offline training algorithm

on the collected data. In [1], a supervised noise classifier smartphone app was developed. A major shortcoming of supervised noise classifiers is that they are not capable of coping with noise environments for which no training is done. As a result, the personalization of such classifiers to the noise environments encountered by a specific user cannot be achieved with ease since the training process needs to be repeated when a new noise environment is encountered. To address this shortcoming of supervised classifiers, unsupervised noise classifiers can be considered so that a new noise class is generated whenever a new noise environment is encountered.

Among many unsupervised classifiers reported in the literature, only few are designed to be able to operate in real-time or in an online streaming manner without having access to the entire data. In [6], a real-time unsupervised classifier for environmental noise signals was developed based on the online frame-based clustering (OFC) introduced in [7]. In [3], we developed a smartphone app implementing this real-time unsupervised noise classifier.

The previously developed unsupervised noise classifier app in [3] has two limitations when deployed in realistic audio environments. One limitation is that it does not include a voice activity detection (VAD) to separate the presence of speech from the absence of speech. For proper operation in real-world audio environments, the classification should be carried out in the absence of speech or for pure noise frames. Another limitation is that it does not include buffers to allow a stable operation in the field by avoiding frequent switching between noise classes and by not reacting to transient noises that are not part of a sustained background noise environment.

In this paper, an unsupervised noise classifier smartphone app is developed which overcomes the above limitations and thus provides an improved outcome compared to the app reported in [3] when operated in real-world audio environments. More specifically two improvements are made

in this work. One improvement is the development of a more stable unsupervised classifier compared to the app in [3] by fusing the decisions from two adaptive resonance theory (ART2) [8] unsupervised classifiers. The other improvement involves the inclusion of the VAD developed in [4] with the unsupervised classifier.

The rest of the paper is organized as follows. In section 3.3, the components of the developed unsupervised noise classifier app are described. Then, in section 3.4, the real-time implementation aspects of the app are discussed. The experimental results are reported in section 3.5 followed by the conclusion in section 3.6.

3.3 Components of Developed Unsupervised Noise Classifier

Figure 3.1 shows a block diagram of integrating a voice activity detection (VAD) module with the unsupervised noise classifier. Input audio signals are captured frame by frame from the smartphone microphone using the i/o processing modules described in [9].

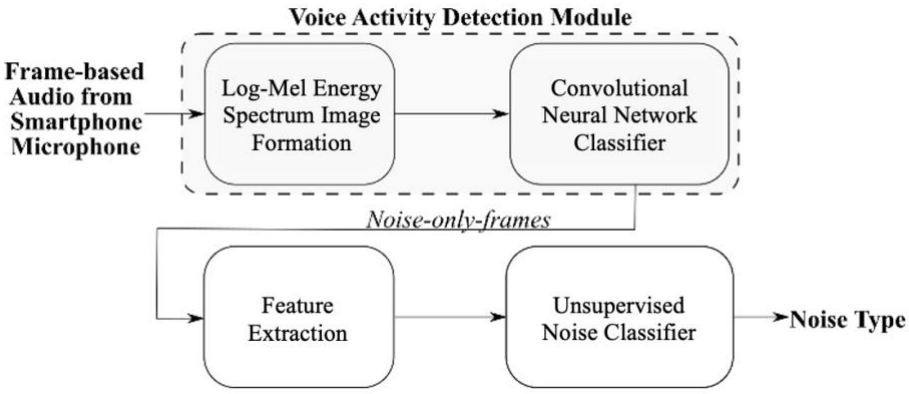


Fig. 3.1. Block diagram of integrating voice activity detection with the unsupervised noise classifier.

Before carrying out noise classification, the VAD module reported in [4] is activated to separate frames with speech activity from pure noise or noise only frames. This VAD consists of two sub-

modules: an image formation sub-module and a convolutional neural network (CNN) sub-module. The first sub-module forms images by computing log-mel energy spectrogram over a time duration. The second sub-module then uses these images as its inputs to separate the presence of speech from the absence of speech. The subsections that follow describe the components of the developed unsupervised noise classifier app.

3.3.1 Feature Extraction

Two feature sets previously used for noise classification are considered here. The first set consists of band-periodicity (BP) and band-entropy (BE) features used in [1]. Based on *B* non-overlapping subbands of a frame, the BP features are computed as follows:

$$BP_b = \frac{1}{N}\sum_{n=1}^{N} \rho_{b,n}\,,\ b = 1, \dots, B \tag{1}$$

where $\rho_{b,n}$ denotes the correlation coefficient between the *n*th frame and its adjacent frames in band *b* and *N* denotes the number of frames in the time duration [*t*-*T*, *t*]. Likewise, the BE features are computed as follows:

$$BE_b = \frac{1}{N}\sum_{n=1}^{N} H_{b,n}\,,\ b = 1, \dots, B \tag{2}$$

where $H_{b,n}$ denotes the entropy of the *n*th frame in band *b*.

The second set of features is mel-frequency spectral coefficients (MFSC) as described in [4]. The conversion from a frequency *f* to *m*th mel-frequency and its inverse are expressed as follows:

$$B(f) = 2595\, log_{10}\left(1 + \frac{f}{700}\right) \tag{3}$$

$$B^{-1}(m) = 700\left(10^{\frac{m}{2595}} - 1\right) \tag{4}$$

Based on *N* triangular overlapping bandpass filters, *N+2* mel-filters are used covering the following equally-spaced frequency ranges indicated by $\hat{m}$:

$$\hat{f}(n) = \frac{(K+1) * B^{-1}(\hat{m}(n))}{f_s}, n = 0, \dots, N+1 \quad (5)$$

where *K* denotes the number of bins in fast Fourier transform (FFT), and *fs* is the sampling frequency. The amplitude of the *n*th filter at frequency bin *k* is given by

$$H_n(k) = \begin{cases} 0 & , k < \hat{f}(n-1) \\ \frac{k - \hat{f}(n-1)}{\hat{f}(n) - \hat{f}(n-1)} & , \hat{f}(n-1) < k \leq \hat{f}(n) \\ \frac{\hat{f}(n+1) - k}{\hat{f}(n+1) - \hat{f}(n)} & , \hat{f}(n) < k \leq \hat{f}(n+1) \\ 0 & , k > \hat{f}(n+1) \end{cases}, \quad (6)$$

$$k = 1, \dots, \frac{K}{2} \text{ and } n = 1, \dots, N.$$

Then, the MFSC coefficients are computed as follows:

$$MFSC(n) = \log\left(\sum_{k=0}^{K} H_n(k) * |F(k)|^2\right), \quad (7)$$

$$n = 1, \dots, N$$

where $|F(k)|^2$ denotes the FFT power spectrum. Log-mel energy spectrum images are then formed by concatenating the coefficients over time. Figure 3.2 shows the log-mel energy spectrum images for three different background noises. As can be seen from this figure, the noise patterns appear different in these images.

3.3.2 Adaptive Resonance Theory (ART2) Classifier

The adaptive resonance theory 2 (ART2) is a computationally efficient unsupervised classifier which allows the processing of data samples to take place in an online manner by reacting to one

data sample (in our case one frame) at a time. An overview of the processing steps involved in ART2 is stated next. More details appear in [8].

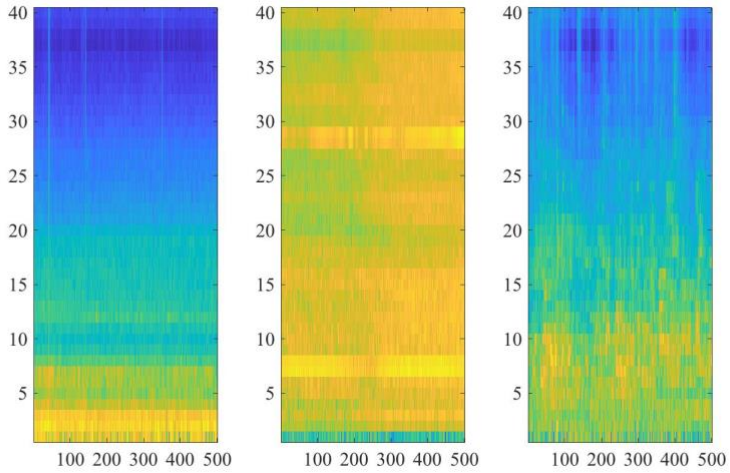


Fig. 3.2. Sample log-mel energy spectra of different noise types; x-axis represents frames in time, and y-axis represents mel frequency spectral coefficients: (a) driving car noise, (b) machinery noise (vacuum cleaner), and (c) babble noise.

Let us consider some classes have already been created. When a new data sample or frame is received, its distance from the existing class centroids or representatives are computed using the city block distance $\|x - w_j\|$, where x denotes the feature vector associated with the new data sample or frame and w_j denotes the centroid or representative of class j. Then, the closest or winner class denoted by j^* is identified. This is followed by carrying out a so-called vigilance test to see whether the winning class j^* passes this distance test $\|x - w_j\| < \rho$, where ρ is called the vigilance parameter. If the test is passed, the centroid or representative of the winning class is modified as follows:

$$w_{j^*}^{new} = \frac{x + w_{j^*}^{current} P_{j^*}^{current}}{(P_{j^*}^{current} + 1)} \tag{8}$$

where $P_{j^*}^{current}$ denotes the number of current data samples in class j. If the test is failed, a new class or cluster is created with the centroid or representative of $w_k = x$.

3.3.3 Decision-Level Fusion of Two ART2 Classifiers

For the purpose of gaining stability when operating in realistic audio environments, two ART2 classifiers are considered in the app. In the results section, an analysis based on three separate datasets is reported indicating the effectiveness of placing two ART2 classifiers in parallel, one operating on subband features and the other operating on MFSC features. Based on this analysis, the fusion structure shown in Figure 3.3 was implemented as a real-time smartphone app.

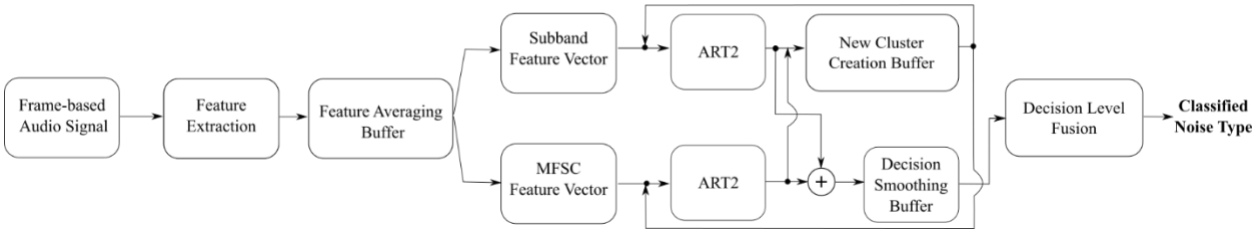


Fig. 3.3. Block diagram of the developed smartphone app implementing decision-level fusion of two ART2 unsupervised classifiers.

3.4 Real-time Smartphone App

This section covers the real-time implementation aspects of the developed smartphone app based on the unsupervised classification fusion structure illustrated in Figure 3.3. Both iOS and Android versions of the app are developed. All the components of the unsupervised classifier are coded in C and incorporated into the Objective-C software environment of iOS smartphones or the Java software environment of Android smartphones using the shells provided in [10].

The audio i/o setup is done using CoreAudio API [11] and Superpowered SDK [12] for iOS and Android smartphones, respectively. All the components are coded in a modular manner by sharing common supporting files so that they can be easily replaced by other similar components.

In modern smartphones, the lowest audio latency is achieved for 48kHz sampling frequency. iOS smartphones or iPhones exhibit 10-15ms audio latency. Audio latency of Android smartphones varies from one phone to another and is normally higher. For example, Google Pixel2 Android smartphones have an audio latency of 40ms. In order to capture an audio frame with the lowest audio latency, audio signals need to be read at the preferred buffer size of 64 samples at a time at 48kHz sampling frequency when using iOS smartphones which corresponds to 64/48kHz = 1.3ms. When using Android smartphones (Google Pixel2 here), audio signals need to be read at the preferred buffer size of 96 samples at a time at 48kHz sampling frequency which corresponds to 96/48kHz = 2ms. This means that the time available to process an audio frame captured every 12.5ms (a 50% overlap is considered for 25ms duration frames) is 1.3ms on iOS smartphones and 2ms on Android smartphones. If frame processing time exceeds these times, frame skipping occurs and real-time throughput cannot be achieved.

The feature extraction is done using a circular buffer to ensure that the classification can run synchronously with the lowest latency i/o setup. Audio frames of duration 25ms are captured at the sampling frequency of 48kHz or 1200 samples. Since the frequency range of speech and most environmental noises lie below 8 kHz, frames are downsampled from 48 kHz to 16 kHz to gain computational efficiency or real-time throughput by reducing the FFT size from 2048 to 512. For computing subband features, the FFT is divided into 4 bands generating a total of 8 subband

features. For computing MFSC features, 40 bandpass filters are used. The lower and upper frequencies are set to 0 and 8000 Hz, respectively.

As stated earlier, the motivation behind developing the app in this paper is to address the limitations associated with the previously developed unsupervised noise classifier smartphone app. The newly developed app avoids noise classification during the presence of speech and creates a new noise class in an online manner when a new noise environment is encountered without allowing fluctuations among the previously created noise classes. The app is also designed to operate in a hybrid mode, that is by saving noise environments in its memory when it is stopped and then these noise environments are used as the initial classes when the app is run again a next time. Figure 3.4 illustrates the graphical user interface (GUI) of the developed smartphone app that include frame processing time in milliseconds, noise power or noise pressure level in decibel (dB), two vigilance parameters associated with the two ART2 classifiers (vigilance 1 and vigilance 2), number of frames used for feature averaging, new class creation time or buffer in seconds, and decision smoothing time or buffer in seconds.

Considering that there are fluctuations of features from one frame to next, feature averaging over a number of frames is carried out in order to create a more stable set of features before using them as the input to the classifier. The default value is 80 consecutive overlapped frames. This means that the class decision rate is one decision per second. A new class creation time or buffer is added for the purpose of creating new noise classes in sustained noise environments and avoiding creation of new noise classes for transient noises that occur for short time durations. A majority-voting decision smoothing time or buffer is also added to allow the classifier to have a smoother

and thus a more stable decision outcome by avoiding fluctuations when moving from one noise environment to another.

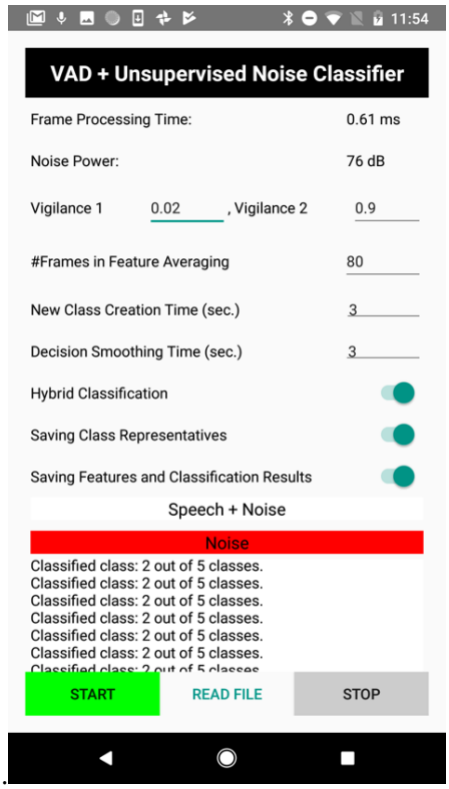


Fig. 3.4. Graphical user interface (GUI) of the unsupervised noise classifier smartphone app (Android version).

The vigilance parameters and the stability times can be adjusted in the GUI to fit the classifier to the noise environments that a specific user encounters or is of interest to a specific user. The adjustment process can be viewed as an online calibration or training phase. Note that no offline training takes place and the calibration or training is done entirely online.

Two switches, named *saving class representatives* and *hybrid classification*, are also included in the GUI for the purpose of operating the app in its hybrid mode. By turning on the switch *saving class representatives*, all the existing noise classes and their parameters are saved. By turning on the switch *hybrid classification*, instead of beginning the classifier from scratch or no class, the

app uses the previously online trained or saved classes as its initial classes when it is run a next time. These switches allow the app to be personalized for the noise environments of interest to a specific user.

3.5 Experimental Results and Discussion

In this section, the analysis that was performed to obtain an effective classification structure for conducting the unsupervised classification is first presented. Then, the parameters identified to serve as the default values for realistic audio environments are specified based on the online calibration or training that was performed. Finally, the field testing and real-time processing results of the developed unsupervised noise classifier app are reported together with a comparison with the previously developed app in [3].

The following three noise datasets were analyzed to identify an effective noise classifier structure: (i) Dataset1: noise sound files from three noise classes in the DCASE 2018 task B dataset [13] recorded by a smartphone consisting of street traffic, airport, traveling by bus. (ii) Dataset2: Noise sound files recorded by our research group with a smartphone consisting of babble in dining hall, construction machinery, and driving car. (iii) Dataset3: Noise sound files from three noise classes in the DCASE 2018 task B dataset recorded by a different smartphone consisting of metro, babble in metro-station, and outdoor park. As illustrated in Table 3.1, the best outcome was found when considering two ART2 classifiers in parallel and by fusing their decisions. This is the structure that was implemented as a smartphone app in this work.

Table 3.2 shows the outcome of the online calibration that was conducted to set the default values of the entries in the app GUI for a stable operation in actual or realistic noise environments. As

indicated in this table, these default values are 80 frames for feature averaging, allowing 3 seconds before creating a new class, and carrying out a majority voting decision over 3 seconds.

Table 3.1. Analysis of different unsupervised classification approaches.

Approaches	Features	Dataset 1	Dataset 2	Dataset 3
Decision-level ART2 Fusion (in Parallel)	Subband+MFSC	90.1%	95.2%	97.2%
Decision-level ART2 Fusion (in Series)	Subband+MFSC	78.3%	95.6%	95.4%
Single ART2	Subband+MFSC	75.6%	69.4%	65.2%
Single ART2	MFSC	76.0%	71.2%	65.1%
Single ART2	Subband	66.7%	60.3%	82.3%

Table 3.2. Default setting of the smartphone app parameters.

Vigilance 1	Vigilance 2	#Frames in Feature Averaging	New Class Creation Time (sec)	Decision Smoothing Time (sec)
0.01-0.03	0.6-1.0	80	3.0	3.0

Next, field testing was conducted in realistic noise environments. Figures 3.5 and 3.6 show the outcome for two sample field testing runs corresponding to three and four class scenarios, respectively. As can be seen from these figures, the developed smartphone app outperforms the previously developed smartphone app in [3]. This is due to the stability modules built into the new app as well as avoiding classification in the presence of speech activity. An item to note here is that the unsupervised classifier in [3] requires the online training of two dependent parameters which is more time consuming to do as compared to the online training of two independent vigilance parameters in the developed app. Furthermore, as illustrated by dashed lines in Figure 3.6, the creation of a new class takes 3 seconds which is one half of the time of the app in [3]. For

these two field test runs shown here, the overall classification rate of the app in [3] was found to be 76.3%, while the overall classification rate of the developed app in this work was found to be 96.1%.

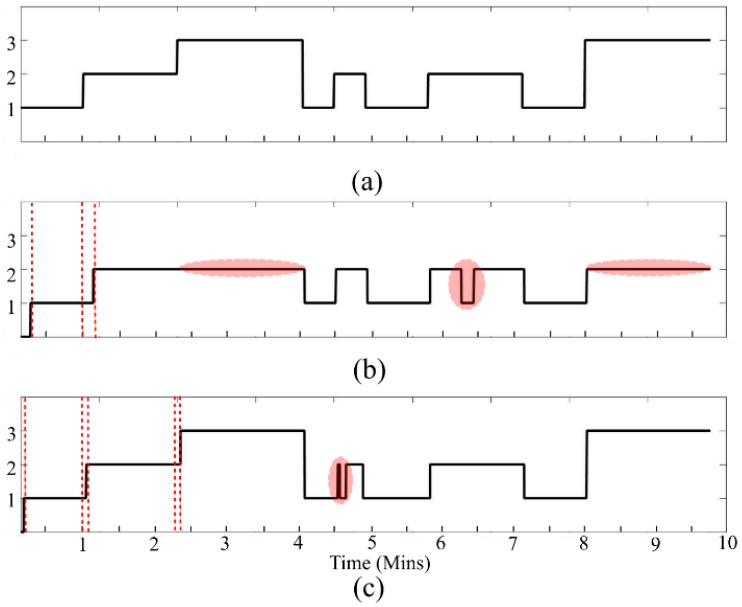


Fig. 3.5. Actual field testing results in three noise environments of street (label 1), dust blower (label 2), babble in activity center (label 3) (x-axis denotes time in minutes): (a) actual clusters or ground truth specified by the user, (b) unsupervised noise classification outcome by the app in [3], and (c) unsupervised noise classification outcome by the developed ART2 fusion app - dashed lines represent the duration taken for generating new classes, and blobs denote misclassifications.

3.5.1 Real-Time Processing

As mentioned earlier, for the real-time operation of the app or for no frames getting skipped, the processing time per frame needs to remain below 1.3ms for iOS smartphones and below 2ms for Android smartphones (Google Pixel2).

The unsupervised noise classifier app developed here takes on average 0.65ms on both Android Google Pixel2 and on iPhone8 smartphones. The CPU and memory consumptions of the app varies between the two versions due to the differences between Android and iOS operating systems. The

CPU and memory consumptions of the Android version obtained by the Android Studio [14] are 26% and 70MB, respectively, and of the iOS version obtained by the Xcode IDE [15] are 15% and 24MB.

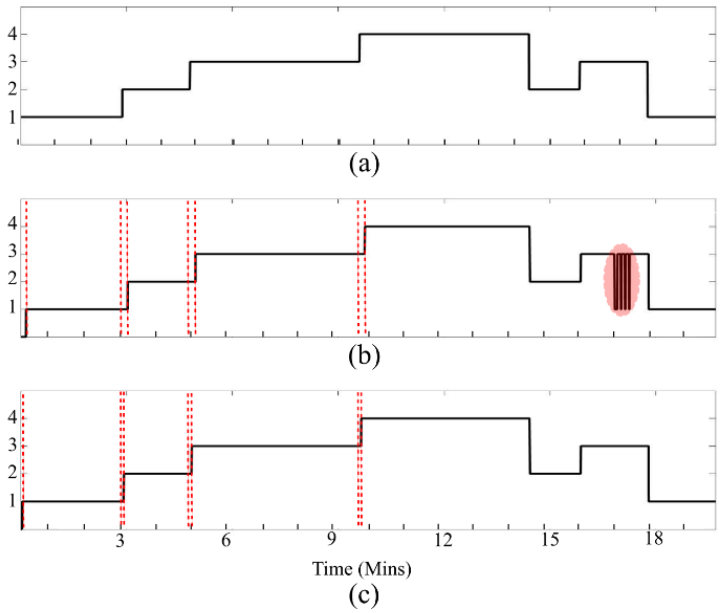


Fig. 3.6. Actual field testing results in three noise environments of babble in dining hall (label 1), driving car (label 2), machinery (label 3), and traffic (label 4) (x-axis denotes time in minutes): (a) actual clusters or ground truth specified by the user, (b) unsupervised noise classification outcome by the app in [3], and (c) unsupervised noise classification outcome by the developed ART2 fusion app - dashed lines represent the duration taken for generating new classes, and blobs denote misclassifications.

A video clip of the developed unsupervised noise classifier smartphone app running in real-time can be viewed at this link: www.utdallas.edu/~kehtar/UnsupervisedNoiseClassifierApp-ART2Fusion.mp4.

3.6 Conclusion

In this paper, a real-time smartphone app for unsupervised noise classification for deployment in realistic noise environments has been developed. The decisions from two ART2 unsupervised

noise classifiers that operate independently in parallel are fused together to gain stable noise classification outcomes in the field. The app is designed in such a way that the training is done online without the need to carry out any offline training, thus allowing the app to be personalized to the noise environments encountered by a specific user. In our future work, we plan to use this unsupervised noise classifier as part of a noise reduction app for hearing improvement purposes.

3.7 Acknowledgment

This work was supported by the National Institute of the Deafness and Other Communication Disorders (NIDCD) of the National Institutes of Health (NIH) under the award number 1R01DC015430-01. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH.

3.8 References

[1] F. Saki, A. Sehgal, I. Panahi, and N. Kehtarnavaz, "Smartphone-based real-time classification of noise signals using subband features and random forest classifier," Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 2204-2208, Orlando, FL, 2016.

[2] A. Bhattacharya, A. Sehgal, and N. Kehtarnavaz, "Low-latency smartphone app for real-time noise reduction of noisy speech signals," Proceedings of IEEE Conference on Industrial Electronics, pp. 1280-1284, Edinburgh, Scotland, 2017.

[3] N. Alamdari, F. Saki, A. Sehgal, and N. Kehtarnavaz , "An unsupervised noise classification smartphone app for hearing improvement devices," Proceedings of IEEE Signal Processing in Medicine and Biology Symposium, Philadelphia, PA, 2017.

[4] A. Sehgal, and N. Kehtarnavaz, "A Convolutional Neural Network Smartphone App for Real-Time Voice Activity Detection," IEEE Access, vol. 6, pp. 9017-9026, 2018.

[5] K. Strom, "The HR 2006 dispenser survey," Hearing Review, vol. 13, pp. 16-39, 2006.

[6] F. Saki, and N. Kehtarnavaz, "Real-time unsupervised classification of environmental noise signals," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 25, pp. 1657-1667, 2017.

[7] F. Saki, and N. Kehtarnavaz, “Online frame-based clustering with unknown number of clusters,” Pattern Recognition, vol. 57, pp. 70-83, 2016.

[8] S. Kung, Digital Neural Networks, pp. 80-85, Prentice Hall, 1993.

[9] A. Sehgal, and N. Kehtarnavaz, “Utilization of two microphones for real-time low-latency audio smartphone apps,” Proceedings of IEEE International Conference on Consumer Electronics, pp. 1-6, Las Vegas, NV, 2018.

[10] N. Kehtarnavaz, S. Parris, and A. Sehgal, Smartphone-Based Real-Time Digital Signal Processing, Morgan and Claypool Publishers, 2015.

[11] Apple, https://developer.apple.com/documentation/coreaudio

[12] Superpowered, http://superpowered.com

[13] IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events, http://dcase.community/challenge2018/task-acoustic-scene-classfication

[14] Google, https://developer.android.com

[15] Apple, https://developer.apple.com

CHAPTER 4

AN UNSUPERVISED NOISE CLASSIFICATION SMARTPHONE APP FOR HEARING IMPROVEMENT DEVICES*

Authors – Nasim Alamdari, Fatemeh Saki, Abhishek Sehgal, and Nasser Kehtarnavaz

The Department of Electrical and Computer Engineering,

The University of Texas at Dallas

800 West Campbell Road

Richardson, Texas 75080-3021

4.1 Abstract

This paper presents an app for running a previously developed unsupervised noise classifier in real-time on smartphone/tablet platforms. The steps taken to enable the development of this app are discussed. The app is utilized to carry out field testing of the unsupervised classification of actual encountered noise environments without any prior training and without specifying the number of noise classes or clusters. Two objective measures of cluster purity and normalized mutual information are considered to examine the performance of the app in the field with the user acting as the identifier of the ground truth classes. The results obtained indicate the effectiveness of this real-time smartphone app for carrying out the environmental noise classification in an unsupervised manner.

4.2 Introduction

According to the World Health Organization, over 5% of the world's population or 360 million people suffer from disabling hearing loss [1]. Hearing improvement devices such as hearing aids and cochlear implants are used to cope with disabling hearing loss. The performance of these devices is adversely affected in the presence of background noise. For this reason, modern hearing aids and cochlear implants use a noise reduction module. Several signal processing pipelines have been developed in the literature that take into consideration the noise type as part of the noise reduction module, e.g. [2, 3]. In these pipelines, a supervised noise classifier is used to identify the noise type in order to adjust the parameters of the noise reduction module depending on the noise type.

The use of a supervised classifier requires a training process, which in turn demands the collection of noise data from noise environments. Once such a classifier is trained based on the collected data, then the operation or utilization of the classifier can begin. Among its trained classes, the classifier would select the noise class which is closest to an observed noise type. In general, there are two limitations associated with supervised classifiers. The first limitation is that they require training, that is one needs to go through a training process based on a previously collected dataset. The second limitation is that, in practice, different users may encounter different noise types. In other words, a different set of noise types may be encountered for which the classifier is not trained. To address these limitations, this paper presents the development of a smartphone app for the real-time implementation of a previously developed unsupervised noise classification algorithm. Such a smartphone app enables the testing of various noise reduction modules to be conducted with ease in the field or in realistic noise environments.

In [4], a clustering or unsupervised classification algorithm, named OFC (Online Frame-Based Clustering), was developed by our research group which requires no training. In this algorithm, the classification is conducted in an unsupervised manner, that is new classes or clusters get generated as needed depending on how different observed samples are to the previously identified classes. This algorithm is capable of generating clusters in an on-the-fly manner without the need to specify the number of clusters, which is a requirement in a typical clustering algorithm such as k-means and its variations. In [5], the OFC algorithm was applied to the problem of environmental noise classification and its effectiveness was demonstrated by examining the cluster purity and normalized mutual information measures.

The rest of the paper is organized as follows: Section 4.3 provides an overview of the previously developed unsupervised classification algorithm. Section 4.4 covers the details of the smartphone implementation done in this work towards generating a real-time low-latency app. The experimental results of field tests based on the developed app are then reported in section 4.5. Finally, the conclusion is stated in section 4.6.

4.3 Overview of the Unsupervised Noise Classifier

This section provides an overview of the unsupervised noise classifier introduced in [4, 5]. This classifier has been implemented as a smartphone app in this work. Figure 4.1 illustrates a block diagram of the unsupervised classifier algorithm. After signal framing, feature extraction is carried out to obtain subband features based on band-periodicity and band-entropy characteristics of signal frames. Frames of an input signal are captured with a sampling frequency of f_s, which are then divided into B non-overlapping subbands.

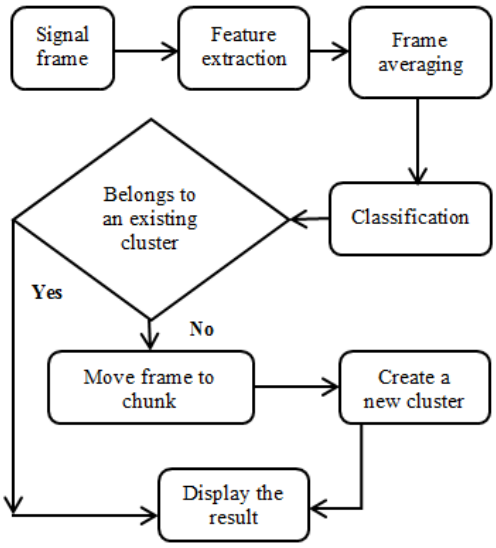


Fig. 4.1. Block diagram of the unsupervised classification algorithm introduced in [5].

Periodicity in a subband is characterized by the maximum value of normalized correlations $\rho_{b,n}$ between an *n*th frame and its adjacent frame, that is:

$$BP_b = \frac{1}{N} \sum_{n=1}^{N} \rho_{b,n} \ , b = 1, \dots, B \quad (1)$$

where *N* denotes the number of frames over this duration [*t-T, t*] with *t* representing current time. The band-entropy features are computed similar to the band-periodicity features by replacing normalized correlations with entropy as noted below

$$BE_b = \frac{1}{N} \sum_{n=1}^{N} H_{b,n} \, , b = 1, \dots, B \quad (2)$$

where $H_{b,n}$ denotes Shannon entropy of the *n*th frame in band *b*. Subband features are averaged over a number of frames and fed into a classifier to see whether a frame belongs to any of the existing clusters or not. If there is no match with any existing cluster or class, the frame is moved to a buffer called chunk for the detection of a possible new cluster. When the chunk gets full, an evaluation is performed to see whether a new cluster needs to be created. Features corresponding to frames in the chunk are checked for statistical similarity to identify the largest so called micro-cluster, that is the largest number of similar and connected features in the chunk. This micro-cluster is then identified and used to create or establish a new cluster. This process is achieved by using a Support Vector Data Description (SVDD) classifier [6]. SVDD is a one-class support vector machine classifier where Gaussian kernels define a sphere boundary around the samples of that class. Two parameters that influence the outcome of SVDD are standard deviation associated with Gaussian kernels and fraction rejection which defines the percentage of the target class samples to be regarded as outliers.

4.4 Real-Time Implementation of Smartphone App

In this section, the key issues associated with the implementation of the smartphone app are covered.

4.4.1 Software Tools and Libraries

All the components of the feature extraction and the clustering algorithm were coded mostly in C with some parts in MATLAB. MATLAB codes were converted to C using the MATLAB Coder utility as per the guidelines described in [7]. To implement the SVDD classifier, the dlib library [8] was used. Similar to [9], a circular buffer was used to maintain synchronous operation between the input/output audio frame and the clustering frame. The codes were organized as an app using the shells developed in [10] for iOS and Android smartphones. For iOS smartphones, the shell is coded in Objective-C and for Android smartphones, the shell is coded in Java. In this paper, we have reported the results for the iOS version due to the lower audio latency of iOS devices or iPhones (10-15ms) as compared to Android smartphones. For example, for Google Pixel Android smartphone, the audio latency is 40ms. It should be noted that since different Android smartphone manufacturers use different i/o hardware, audio latency of Android smartphones varies from phone to phone and in many cases, it is higher than 40ms.

4.4.2 Audio Latency

Latency is the amount of time it takes for an audio signal to get captured by the smartphone microphone, get converted by the smartphone analog-to-digital converter to a digital signal, and then get converted back to an analog signal to be played on the smartphone speaker. The hardware of modern smartphones generates the lowest latency at the sampling frequency of 48kHz. To

achieve the lowest latency audio implementation on iOS devices, the audio signal must be read from the microphone and played from the speaker at 64 samples per frame at 48kHz. The GCC compiler level optimization level 2 (-O2) is used to lower the processing time associated with a frame. Similar to [5], the feature extraction is done for a frame overlap size of 12.5ms or 600 samples. A circular buffer is used to reach this overlap size. This buffer ensures that the clustering algorithm can run synchronously with the lowest latency i/o setup.

4.4.3 Hybrid Classification

As stated earlier, the motivation behind developing this app is to address the limitations associated with supervised noise classifiers. The app creates a new cluster when a new noise environment is encountered. The app is also designed to operate in a hybrid classification mode, that is it examines the previously saved clusters or previously encountered noise environments before creating a new cluster. Figure 4.2 illustrates the settings screen of the developed smartphone app that consists of the number of decisions in chunk, the total number of clusters as an upper limit, the SVDD parameters (sigma and fraction rejection), the frame overlap size, and the classification decision rate. Basically, the chunk reflects decisions that do not match any existing clusters which are then used to create a new cluster. The switches Hybrid Classification and Saving Classification Data seen in Figure 4.2 are used when the app is desired to be operated in its hybrid mode. By activating the switch Saving Classification Data, all the encountered noise classes or clusters and their parameters are saved. By activating the switch Hybrid Classification, instead of starting from scratch or no cluster, the app uses the previously saved clusters as its starting point or initial clusters.

4.4.4 Feature Extraction

To extract the subband features, a frame size of 25ms or 1200 samples is considered by concatenating a current and a previous overlapped frame. Since the features correspond to frequencies lying below 8 kHz for speech processing, frames are down sampled from 48 kHz to 16 kHz. This reduces the computation time significantly because the FFT size gets reduced from 2048 to 512. For each signal frame, the FFT is computed and divided into 4 bands, thus generating a total of 8 subbband features. Similar to [5], the decision rate is considered to be 0.5 sec or 500ms long. This corresponds to one decision per 40 overlapped frames. If desired, the app allows changing this decision rate.

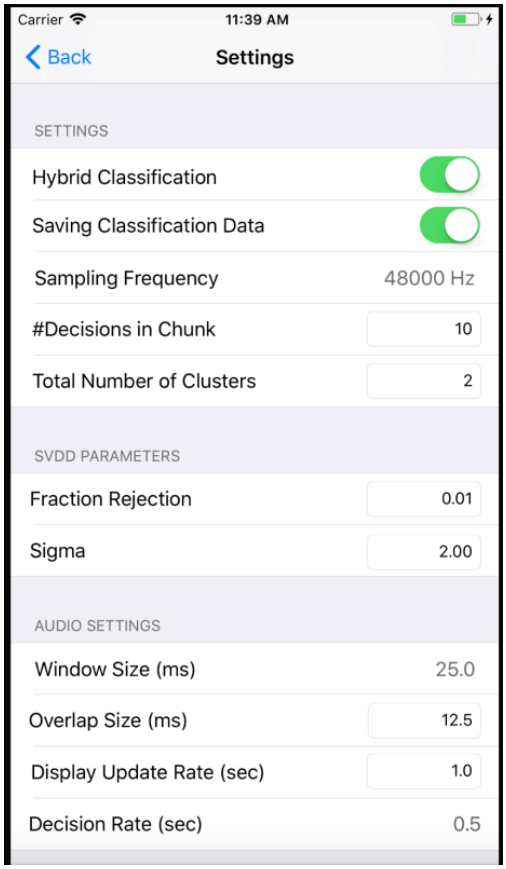


Fig. 4.2. Settings screen of the developed smartphone app.

4.5 Experimental Results

4.5.1 Parameter Settings

First, a study was conducted in various realistic noisy environments to set the default or nominal values of the standard deviation and fraction rejection of the SVDD classifier. These values were set to 0.01 and 2.0, respectively, by observing the correctness of creating a new cluster when the smartphone was physically taken into new noise environments.

In order to examine the performance of the app, similar to [4], the two measures of cluster purity [11] and normalized mutual information (NMI) [12] were computed. The cluster purity measure indicates the purity of clusters in comparison to the actual or ground truth clusters and is expressed as

$$Purity = \frac{\sum_{l=1}^{U} \left|\hat{V}_l\right|}{U} \times 100 \quad (3)$$

where U denotes the total number of clusters, $\hat{V}_l$ is the number of samples with the dominant class label in cluster l, and V_l is the total number of samples in cluster l. The NMI measure indicates to what degree the detected clusters are similar to the actual or ground truth clusters. It should be noted that in practice the ground truth clusters are unknown. Mutual information (MI) between the ground truth cluster set (X) and the detected cluster set (Y) is obtained as follows:

$$MI(X,Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p(x).p(y)}\right) \quad (4)$$

where p(x) and p(y) denote the marginal probability density functions associated with the cluster sets and p(x,y) their joint density function. Normalized mutual information (NMI) is then computed this way

$$NMI(X,Y) = \frac{MI(X,Y)}{\sqrt{E(X).E(Y)}} \quad (5)$$

where E(X) and E(Y) are the entropies of the cluster sets.

4.5.2 Field Testing

In the field tests conducted, the ground truth was provided by the user, that is the user specified whether there was a change in the environmental noise. The real-time field testing was performed on an iPhone 7 and on an iPad4. When these iOS platforms were taken to a new noise environment, a new cluster was created after the chunk got filled.

The plots shown in figure 4.3 exhibit the outcomes of four sample field test runs in terms of the classification rate and new cluster creation. The first plot corresponds to the three audio environments of driving car, restaurant, and vacuum cleaner. The second plot corresponds to the four audio environments of quiet, driving car, outdoor a/c compressor, and restaurant. The third plot corresponds to the three audio environments of office, street, and machinery consisting of kitchen vent motor. The fourth plot corresponds to the five audio environments of quiet, train, inside airplane, street, and driving car. Table 4.1 provides the cluster purity, the NMI, the actual number of clusters, and the number of clusters detected by the app for these four sample field test runs.

As seen from figure 4.3, whenever the audio or noise environment changed to a new audio or noise environment which had not been encountered before, it took 5 seconds (illustrated by vertical lines) to detect that noise environment as a new cluster. It is worth stating that this time can be adjusted by changing the number of decisions in the chunk. The chunk size is set to 10 decisions in the experimentations reported here with each decision taking 0.5 sec for a total time of 5 sec to create

a new cluster. It should be noted that this time is only for the first time a new noise environment or cluster is encountered. Furthermore, in hearing devices, the reaction to noise environments is dampened on purpose as it is not desired to react too quickly to noises that are not sustained or do not last long.

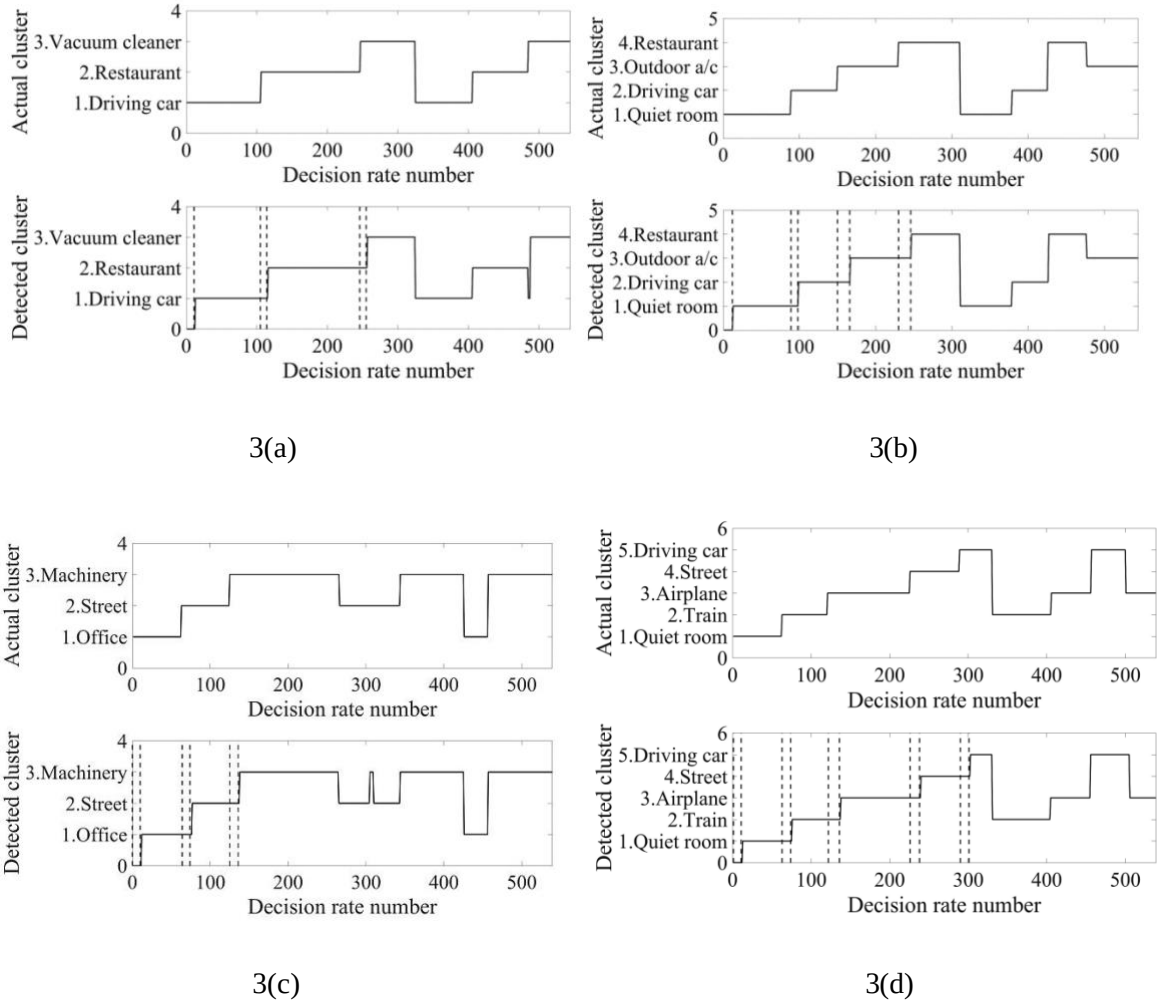


Fig. 4.3. Comparison between the actual noise classes or clusters and the detected noise classes by the developed unsupervised classifier app during four sample field test runs, the cluster decision rate number indicates a decision every 500ms: In 3(a), label 1 denotes driving car, label 2 restaurant, and label 3 vacuum cleaner; in 3(b), label 1 denotes quiet room, label 2 driving car, label 3 outdoor a/c, and label 4 restaurant; in 3(c), label 1 denotes office, label 2 street, and label 3 machinery; in 3(d), label 1 denotes quiet room, label 2 train, label 3 inside airplane, label 4 street, and label 5 driving car.

In other words, the unsupervised classifier is designed to react to sustained noise environments that last at least 5 sec and not to transient noises that last less than 5 sec. To create a reliable cluster for the first time, i.e. the first time that a new noise environment is encountered, enough information needs to be captured to create a new noise cluster in a reliable way.

If a noise environment has been encountered before, the hybrid mode of operation performs the classification with no delay since the noise has been seen before and a reliable cluster has already been created. Thus, strictly speaking, since the new cluster creation time only occurs during the first encounter of a noise type, clustering decisions during such times should not be considered as errors. In Table 4.1, the measures are listed with and without the new cluster creation time (denoted by I and II, respectively).

4.5.3 Real-Time Processing

The processing time for each frame takes 0.7ms on average, which is well below the frame overlap time of 12.5ms thus allowing the app to run in real-time without any frames getting skipped. This timing includes all the computation including feature extraction and classification. The GUI of the developed app is updated every 1 sec. Note that the clustering decision is made at the rate of 500ms or every 40 frames. Table 4.2 shows the CPU consumption and the memory utilization as well as the energy impact of the app running on iPhone7 as provided by the Xcode IDE [13].

A video clip of the developed app running in real-time can be viewed at this link: http://www.utdallas.edu/~kehtar/UnsupervisedClassifierApp.mp4 .

Table 4.1. Clustering outcome of the unsupervised classifier app for four sample field test runs in terms of cluster purity, normalized mutual information, actual number of clusters, and number of detected clusters; I rows correspond to with new cluster creation time and II rows correspond to without new cluster creation time.

	Cluster Purity	**NMI**	**Actual number of clusters**	**Number of detected clusters**
Environment set 1: driving car, restaurant, vacuum cleaner				
I	93.2%	0.81	3	3
II	98.8%	0.96	3	3
Environment set 2: quiet room, driving car, outdoor a/c, and restaurant				
I	89.5%	0.79	4	4
II	99.3%	0.99	4	4
Environment set 3: office, street, machinery				
I	90.0%	0.76	3	3
II	96.8%	0.89	3	3
Environment set 4: quiet, train, airplane, street, driving car				
I	84.0%	0.74	5	5
II	95.9%	0.88	5	5

Table 4.2. CPU consumption, memory utilization, and energy impact of the developed smartphone app as provided by Xcode IDE.

Unsupervised classifier app		
CPU consumption	*Memory utilization*	*Energy impact*
15%	23 MB	Low

4.6 Conclusion

In this work, a smartphone app has been developed in order to perform unsupervised environmental noise classification in real-time. This app allows one to detect audio or noise classes in the field on a portable device without needing to have any prior knowledge of the audio or noise classes or without any training. The obtained field-testing results indicate the effectiveness of the app in creating a new cluster when a new noise environment is encountered. In our future work,

we plan to incorporate this app as part of a signal processing pipeline running on smartphone platforms and interface it with a Bluetooth equipped hearing aid.

4.7 Acknowledgements

This work was supported by the National Institute of the Deafness and Other Communication Disorders (NIDCD) of the National Institutes of Health (NIH) under the award number 1R01DC015430-01. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH.

4.8 References

[1] http://www.who.int/mediacentre/factsheets/fs300/en/

[2] V. Gopalakrishna, N. Kehtarnavaz, T. Mirzahasanloo, and P. Loizou, "Real-time automatic tuning of noise suppression algorithms for cochlear implant applications," *IEEE Transactions on Biomedical Engineering*, vol. 59, pp. 1691-1700, June 2012.

[3] I. Panahi, N. Kehtarnavaz, and L. Thibodeau, "Smartphone-based noise adaptive speech enhancement for hearing aid applications," *Proceedings of IEEE International Conference Engineering in Medicine and Biology*, Orlando, August 2016.

[4] F. Saki and N. Kehtarnavaz, "On-line frame-based clustering with unknown number of clusters," *Pattern Recognition Journal*, vol. 57, pp. 70-83, September 2016.

[5] F. Saki and N. Kehtarnavaz, "Real-time unsupervised classification of environmental noise signals," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 25, pp. 1657-1667, August 2017.

[6] D. Tax and R. Duin, "Support vector data description," *Machine Learning Journal*, vol. 54, pp 45–66, Jan 2004.

[7] N. Kehtarnavaz and F. Saki, *Anywhere-Anytime Signals and Systems Laboratory: From MATLAB to Smartphones*, Morgan and Claypool Publishers, 2017.

[8] D. King, "Dlib-ml: a machine learning toolkit," *Machine Learning Journal*, vol. 10, pp. 1755–1758, 2009.

[9] https://github.com/michaeltyson/TPCircularBuffer

[10] N. Kehtarnavaz, S. Parris, and A. Sehgal, *Smartphone-Based Real-Time Digital Signal Processing*, Morgan and Claypool Publishers, 2015.

[11] F. Cao, M. Estert, W. Qian, and A. Zhou. "Density-based clustering over an evolving data stream with noise," *Proceedings of the SIAM International Conference on Data Mining*, pp. 328-339, 2006.

[12] A. Strehl and J. Ghosh, "Cluster ensembles---a knowledge reuse framework for combining multiple partitions," *Journal of Machine Learning Research*, vol. 3, pp. 583-617, Dec 2002.

[13] Apple, https://developer.apple.com

CHAPTER 5

IMPROVING DEEP SPEECH DENOISING BY NOISY2NOISY SIGNAL MAPPING*

Authors – Nasim Alamdari, Arian Azarang, and Nasser Kehtarnavaz

The Department of Electrical and Computer Engineering,

The University of Texas at Dallas

800 West Campbell Road

Richardson, Texas 75080-3021

5.1 Abstract

Existing deep learning-based speech denoising approaches require clean speech signals to be available for training. This paper presents a deep learning-based approach to improve speech denoising in real-world audio environments by not requiring the availability of clean speech signals as reference in training mode. A fully convolutional neural network is trained by using two noisy realizations of the same speech signal, one used as the input and the other as the target of the network. Two noisy realizations of the same speech signal are generated by using a mid-side stereo microphone. Extensive experimentations are conducted to show the superiority of the developed deep speech denoising approach over the conventional supervised deep speech denoising approach based on four commonly used performance metrics as well as a subjective testing.

5.2 Introduction

Speech denoising is extensively studied in the literature. The existing speech denoising methods can be categorized into two main groups: (1) Conventional methods – These methods involve estimating the noise to achieve denoising. Examples of these methods are spectral subtraction [1] and Wiener filtering [2, 3]. (2) Deep learning-based methods – These more recent methods attempt to model the nonlinear relationship between noisy and clean speech signals via a deep neural network (DNN), e.g. [4-8]. These methods have allowed dealing with non-stationary audio environments [9] and can be further divided into two categories: direct mapping (mapping-based) methods, e.g. [10-12], which mostly use the log-power spectra as the input and output of a DNN and masking-based methods, e.g. [13-15], which estimate a mask to carry out denoising. In addition to a single-channel supervised speech enhancement (SE), there have been works on

supervised multi-channel SE, e.g. [16-18]. In these works, it is shown that as the number of channels increases, the performance of SE is improved. For example, in [16] a fully convolutional neural network (FCN) and a Sinc FCN were used for multi-channel speech enhancement, and in [17], multiple recordings were applied directly in the time domain. However, these works did not address the performance in the field or when subjected to unseen conditions (e.g., unseen speakers with no clean speech signals being available).

A major assumption made in existing deep learning-based methods is the availability of clean speech signals to conduct training. In practice, the problem of speech denoising is more challenging due to the fact that clean speech signals are not known or available when operating in the field or in real-world audio environments. In addition, the generalization capability of current DNN-based approaches is limited for unseen speakers and varying signal-to-noise ratios. In other words, existing supervised speech denoising (SD) models rely on the availability of clean speech signals for training. As a result, SD models are highly dependent on whether actual field signals match training signals.

In [19], an attempt was made to perform denoising without using clean speech signals. The assumption made in [19] was to have simultaneous access to both noise-only and speech+noise signals. In practice, such simultaneous access to noise only and speech+noise signals is very difficult to achieve and only one of these signals is available at any given time.

In this work, a deep speech denoising approach is introduced to ease the major assumption of availability of clean speech signals in the existing deep speech denoising solutions. This is made possible by using a mid-side stereo microphone to fine-tune single-channel speech denoising models. As a result, the introduced approach can be deployed in real-world audio environments or

in the field in which clean speech signals are not available. Not requiring to have clean speech signals is what differentiates this paper from the existing deep speech denoising papers.

The rest of the paper is organized as follows: In section 5.3, the introduced deep speech denoising approach and its implementation aspects are presented. In Section 5.4, an overview of the datasets examined is provided. A comprehensive set of experimentations and their results are then reported in Section 5.5. Finally, the paper is concluded in Section 5.6.

5.3 Developed Deep Speech Denoising Approach

The denoising approach developed in this paper builds upon the commonly practiced denoising approach of supervised training. A deep neural network is first initialized based on the public domain datasets for which clean speech signals are available. Then, the network is further trained in a clean-reference-free manner based on only noisy speech data or in the absence of clean speech signals.

As input to a deep neural network, various audio representation schemes have been utilized. Among them, mel frequency cepstral coefficient (MFCC) [20] has been widely used. Also, log-mel spectrogram coefficient (MFSC) has been used by omitting the discrete cosine transform (DCT) compression from the MFCC computation. These frequency-domain representations focus on the magnitude spectrum and the phase spectrum is often left unprocessed [21]. Similar to recent studies [22-25], raw waveform or time-domain signals are considered here as input to a deep neural network instead of frequency-domain representations.

In supervised deep learning-based speech denoising, a network is trained to perform noise reduction by considering clean speech signals as its output or target signals. A speech+noise mixture or noisy speech signal $y(t)$ can be expressed as

$$y(t) = x(t) + n(t) \tag{1}$$

where $x(t)$ and $n(t)$ denote clean speech and additive noise signals, respectively. A reasonable assumption which is often made is that clean speech and noise signals are uncorrelated and noise is zero mean [26]. A network is then trained and used to generate an estimate of the denoised speech signal $\hat{x}_i = f(y_i; \Theta)$ based on the noisy speech signal y_i as its input. The index i is used here to indicate signal frames.

After the initialization of the network in a supervised manner, the clean speech-free training is conducted by considering noisy speech signals as both the input and the output of the network without knowing clean speech signals. In contrast to supervised denoising, this clean speech-free training makes it more suitable for field deployment as in the field clean speech signals are not available.

As illustrated in figure 5.1, in the supervised speech denoising approach (labeled SSD here), training pairs (y_i, x_i) are used to minimize the network loss function in which y_i is the noisy speech input frame and x_i is the corresponding clean speech target frame. The weights of the network are obtained by solving the following optimization problem

$$arg \min_{\Theta} \sum_i \mathcal{L}(f(y_i; \Theta) = \hat{x}_i, x_i) \tag{2}$$

where $\mathcal{L}(.)$ denotes a loss function, normally the mean squared error (MSE) function, and Θ denotes the network parameters or connection weights.

In the clean speech-free approach, see figure 5.1, two noisy realizations of clean speech signals are used during training. In other words, the input and the target are considered to be two noisy versions of the same speech signal instead of the target being the clean speech signal as in the

supervised approach. This means that the network is trained to solve the following optimization problem instead of the optimization problem in Eq. (2)

$$arg \min_{\Theta} \sum_i \mathcal{L}(f(y_i; \Theta) = \hat{x}_i, y_i') \tag{3}$$

In Eq. (3), y_i' denotes another noisy realization of the same speech signal instead of the clean speech signal x_i. In other words, training pairs (y, y') consisting of two noisy realizations of the same speech signal are used to train the network, $y_i = x_i + n_i$ and $y_i' = x_i + n_i'$.

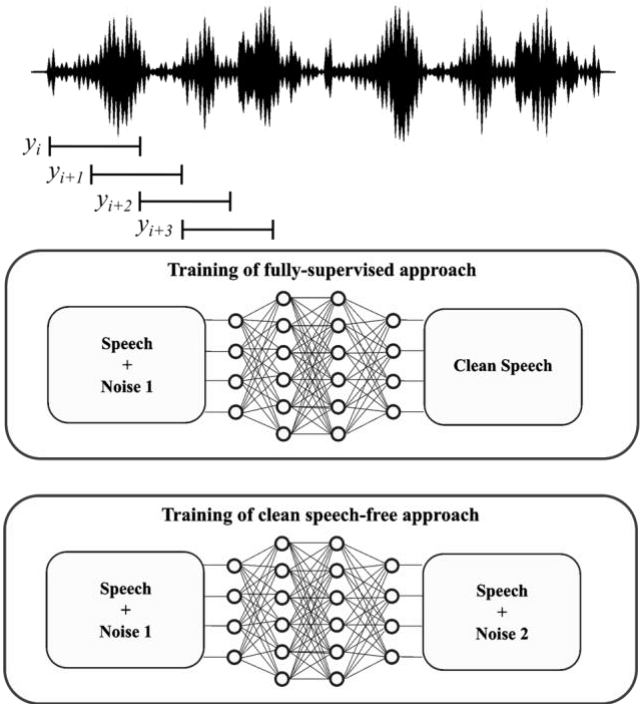


Fig. 5.1. Illustration of the training difference between the supervised deep speech denoising (top) and the clean-free deep speech denoising (bottom).

This approach is inspired from the image denoising work reported in [27], named Nosie2Noise, where image denoising was achieved without using any clean image data. During training, if clean speech target signals are replaced with noisy speech signals with the expected value being equal to clean speech signals ($\mathbb{E}[y_i] \cong x_i$), the final weights of the network would remain more or less the same provided that two assumptions are met as pointed out in [27]. The first assumption is that

the noise to be zero mean which is a reasonable assumption to make noting that in practice noise is often observed to have zero mean. The second assumption is that the two noisy signals y and y'need to be decorrelated or ideally uncorrelated. This assumption is met here by using a mid-side microphone during field training which is discussed next.

5.3.1 Mid-Side Microphone Signals

A mid-side microphone is a stereo microphone whose two signals are generated based on the difference in loudness instead of time-delay. The relationship between the right-left channels and the mid-side microphone signals is given by [28]:

$$y_{\mathrm{L}} = y_{\mathrm{M}} + y_{\mathrm{S}}$$
$$y_{\mathrm{R}} = y_{\mathrm{M}} - y_{\mathrm{S}} \quad (4)$$

where the mid microphone y_{M} is usually faced toward the speech source signal and the bi-directional side microphone y_S with 90-degree rotation with respect to mid, captures signals that is dominated by the background noise signal. Noise signals normally arrive at the side-microphone along different paths and they have similar magnitudes but different phases, that is in the frequency-domain $|N_{\mathrm{R}}| = |N_{\mathrm{L}}|$ and $N_{\mathrm{R}} = e^{j\phi} N_{\mathrm{L}}$ [28]. As shown in Eq. (4), for the left channel, the side signal is added to the mid signal with positive polarity; and for the right channel, the side signal is added to the mid signal with negative polarity. As a result, the right and left signals become decorrelated [29].

Note that a low-quality separation of speech and background noise signals can be achieved by summing the right and left channels. Figure 5.2 shows sample speech signals that are captured by a mid-side stereo microphone (Zoom iO7) in an actual audio environment of cafeteria babble noise.

The first two signals from the top show the right y_{R} and the left y_{L} channels, respectively. The other two signals correspond to the side signal $\mathrm{y_S}$ and the mid signal y_{M}, which are obtained from Eq. (4).

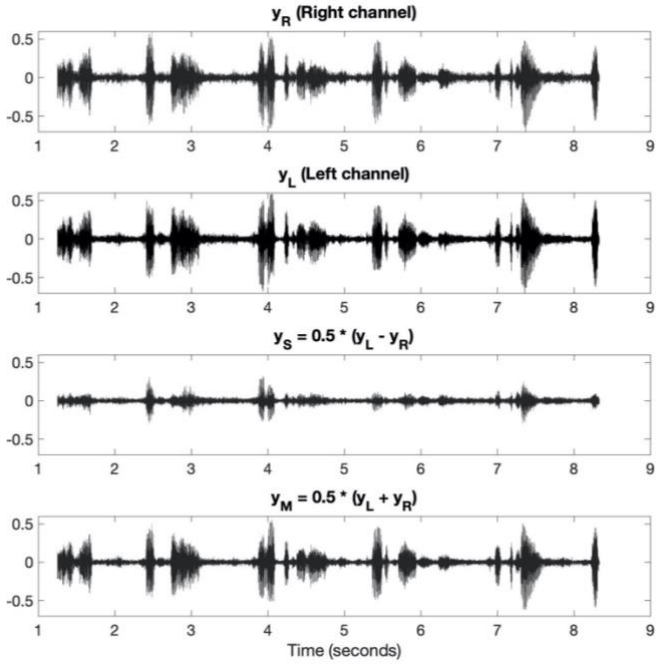


Fig. 5.2. Sample field audio signals captured by a mid-side stereo microphone.

5.3.2 Architecture of Deep Neural Network

The training of the speech denoiser network is performed in a frame-based manner. As illustrated in Fig. 5.1, noisy speech signals are partitioned into 20ms frames through a Hanning window with 50% overlap between adjacent frames. A deep neural network is then used to obtain the denoised speech signal by using $y_i = x_i + n_i$ as its input and $y_i' = x_i + n_i'$ as its target, where y_i and y_i' denote two noisy realizations of the same speech captured by a mid-side stereo microphone.

Next, the architecture of the deep neural network used is mentioned. The network considered is a fully convolutional neural network (FCNN). The FCNN architecture is similar to a conventional convolutional neural network (CNN) architecture. The only difference is that the fully-connected

layers are omitted in FCNN. As noted in [25], FCNNs can model the temporal attributes of time series data using 1D convolution layers. The FCNN architecture used here is similar to the one described in [25]. This architecture incorporates 6 convolution layers. The number of filters and filter size are 55 and (30,1) for the first through the fifth convolution layers, respectively. For the last convolution layer, only one filter of size (1,1) is used followed by the hyperbolic tangent activation function.

The training is speeded up by using the batch normalization in [30] and the Leaky Rectified Linear Units (LeakyReLU) activation function is applied after each convolution layer except for the last layer. In this architecture, MSE is used as the loss function. To train the network, the Adam optimization algorithm described in [31] is used, and the size of the minibatch and the initial learning rate are set to 128 and 0.0004, respectively. The training is normally performed for 25 epochs. As mentioned earlier, the main difference between FCNN and CNN is that there is no fully-connected layer in the output of FCNN. Also, the max-pooling layers are removed. As a result, in FCNN, the output frame depends only on the neighboring input frames. A depiction of the FCNN architecture is provided in Fig. 5.3.

5.3.3 Single Channel Operation or Testing

It needs to be noted that only for training of the clean speech-free network, two channels are used. During the actual operation or testing of the developed deep speech denoising approach, only a single channel is used to feed noisy speech signal frames into the trained deep neural network with the output being denoised speech frames.

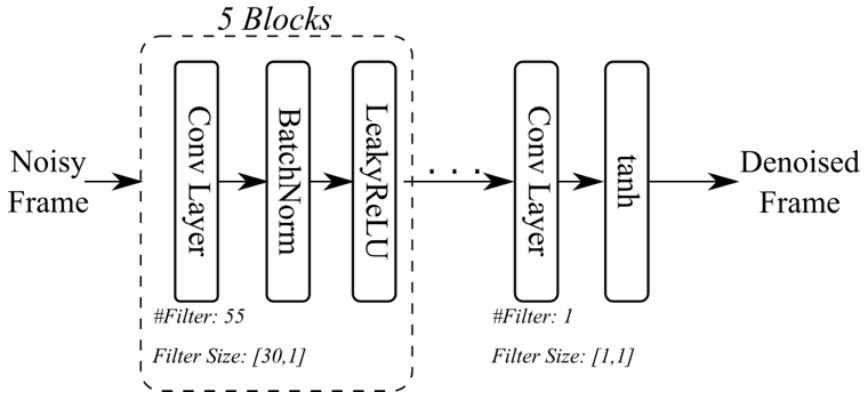


Fig. 5.3. FCNN architecture.

Fig. 5.4 shows the block diagram of the implementation pipeline for field-deployment of the developed deep speech denoising solution. Captured audio signals go through audio framing and averaging to provide the input to a voice activity detection (VAD) module. This module separates noisy speech frames from the absence of speech or from noise-only frames. Noise-only frames are then used to identify different noise types via an unsupervised noise classifier. There have been studies on noise classification, VAD and their utilization in speech denoising, for example [32-37]. Similar to these studies, a VAD together with an unsupervised noise classifier are used to first select an appropriate denoising model based on the noise type. Here, the VAD is the one described in [32] and the unsupervised noise classifier is the one developed in our previous work [33].

Before performing noise classification, the VAD module described in [32] is activated to distinguish frames with speech activity from noise-only frames. This VAD module contains two components: log-Mel energy spectrogram image formation over a time duration as the input of the CNN model; and a convolutional neural network (CNN) model to classify audio frames to either the presence of speech or its absence. Readers are referred to [32] for more details on the VAD used.

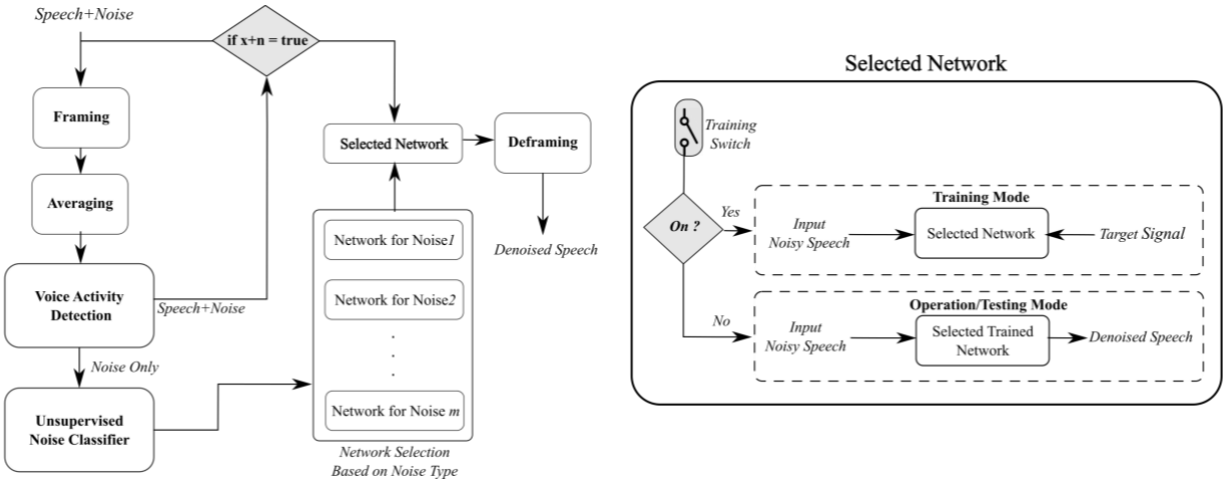


Fig. 5.4. Implementation pipeline of the developed deep speech denoising.

Then, the noise classifier in [33] is activated using noise-only frames. This noise classifier categorizes background noise environments based on decision fusion of two Adaptive Resonance Theory 2 (ART2) classifiers [33]. ART2 is an unsupervised classifier that is computationally efficient and processes frames in an online manner. The two ART2 classifiers operate independently and their decision with equal weight are fused to provide the noise classification outcome. One ART2 classifier uses forty log-Mel frequency spectrogram feature vectors and the other ART2 considers eight subband features (consisting of four band entropy, and four band periodicity feature vectors) as its input. Readers are referred to [33] for more details on the unsupervised noise classifier.

The identification of different noise types allows a bank of fully convolutional neural networks (FCNNs) to be trained, each network for a particular noise type with the number of noise types capped by the user. The selection of a FCNN model for a particular noise type among the bank of the FCNN models is done by the unsupervised noise classifier.

When a new noise type nnew is encountered by the unsupervised noise classifier, a new FCNN can get trained based on speech signals that are corrupted by nnew. Then, this new network gets

added to the network bank. In other words, among all the candidate networks, only one FCNN network will be selected based on the noise type during actual operation or testing in the field. The right part of Fig. 5.4 shows the process of training and operation/testing when a network is selected based on an identified noise type. Once the training switch is activated, see Fig. 5.4, noisy speech frames that belong to the right and left channels are used as the input and the target of the selected network for the clean speech-free training. When the training switch is turned to the off mode, the network goes to the testing or operation mode. The denoising outcome is then played back through the speaker. It is important to note that the introduced approach uses the conventional supervised training as its initial condition. That is why it is labeled here as hybrid speech denoising (HSD).

5.4 Public Domain Datasets

Three widely used public domain speech datasets, IEEE [38], TIMIT [39], and VCTK [40] are considered for the experimentations reported in the next section. The IEEE Corpus consists of 3600 speech audio files by 20 speakers (10 females and 10 males) in which each file is about 2 seconds long. The speakers are from two American English regions of the Pacific Northwest (PN) and the Northern Cities (NC) reading the IEEE "Harvard" sentences. The TIMIT Corpus (Acoustic-Phonetic Continuous Speech) consists of 630 speakers from eight major American English dialects, each reading ten sentences. The VCTK Corpus (Centre for Speech Technology Voice Cloning Toolkit) contains audio files of 109 English native speakers with different accents. About 400 sentences are read by each speaker.

The above clean speech signals are corrupted by the noise signals from the UrbanSound8K dataset [41] consisting of various noise files each lasting 4 seconds. In this work, based on the urban sound taxonomy described in [41], the following four most commonly encountered noise signals are

considered: (i) babble (e.g., restaurant, cafeteria), (ii) wind, (iii) engine, and (iv) driving car. All noisy speech files are sampled at 48 kHz and normalized to have absolute unit maximum.

5.5 Experimental Results and Discussion

Four commonly used objective performance metrics were considered to assess the effectiveness of the developed HSD approach. These metrics include: perceptual evaluation of speech quality (PESQ) [42], short-time objective intelligibility (STOI) [43], segmental SNR (SSNR) [44], and log spectral distance (LSD). For the PESQ and STOI metrics, the ranges are [-0.5, 4.5] and [0, 1], respectively, in which the upper bound of the ranges corresponds to the ideal values. Higher SSNR means better performance and in case of LSD, the ideal value is 0.

Three sets of experiments were conducted. The first two sets of experiments were performed using the public domain datasets. In the first two sets of experimentations, two decorrelated noisy realizations of the same clean speech signals are simulated in order to be able to compute performance metrics and thus compare the HSD approach with the conventional SSD approach. To perform the comparison in a fair manner, the same procedure depicted in Fig. 5.4 was used for the SSD approach. The third set of experiments was performed in the field by carrying out a subjective testing.

5.5.1 Cross-Corpus Simulated Experiments: Unseen Speakers

In a real-world setting, a network should be able to cope with unseen speech signals or speaker(s). In this set of experiments, the clean speech signals of unseen speakers, denoted by dataset 2 in Fig. 5.5, were considered to be unavailable to reflect a real-world setting. As shown in Fig. 5.5, the SSD approach could only get trained with dataset 1 for which clean speech signals were available

and was then tested on dataset 2. Unlike the SSD approach, the HSD approach is capable of coping with unseen speech signals or speaker(s) by keeping the training switch on for unseen speaker(s) since it does not need clean speech signals for its hybrid training.

In this set of experiments, the SSD training was done for 2340 utterances of dataset 1 and the HSD training was done for 1440 utterances of dataset 1 and 900 utterances of dataset 2. More specifically, the length of the training data for both SSD and HSD remained the same in order to have a fair comparison of the two approaches. For training, the noise level was varied to generate three SNR levels of -5 dB, 0 dB, and 5 dB. Then, 360 utterances with SNR of 0 dB were used for testing.

The following experiments were conducted: (i) the network was trained with the IEEE corpus and tested with either the TIMIT or VCTK corpus; (ii) the network was trained with the TIMIT corpus and tested with the IEEE or VCTK corpus; and (iii) the network was tested with the IEEE or TIMIT corpus when the VCTK corpus was used for training. The results of these cross-corpus experiments were averaged and are reported in Table 5.1 as well as in Figs. 5.6 and 5.7, where x+n denotes unprocessed noisy speech signals.

In these experiments, HSD had the ability to turn its training switch to on when untrained speech signals were encountered, see Fig. 5.5. As mentioned earlier, HSD uses supervised training first to initialize the network based on public domain clean speech signals, and then uses the clean speech-free approach based on noisy speech signals in the field. By turning on the training switch, the network associated with a noise type could continue to be trained for unseen speakers. In other words, aforementioned 1440 utterances from dataset 1 were considered for the supervised part, and 900 utterances from dataset 2 were considered for the clean speech-free training part. In

addition to SSD and HSD, the Wiener filtering outcome is also reported here. It is worth noting that similar to HSD, SSD was also conducted based on the models or networks corresponding to different types of noise. The comparison was done for the same type of noise. As shown in Table 5.1, the performance metrics of HSD were found substantially better than those of SSD and Wiener filtering across different noise types.

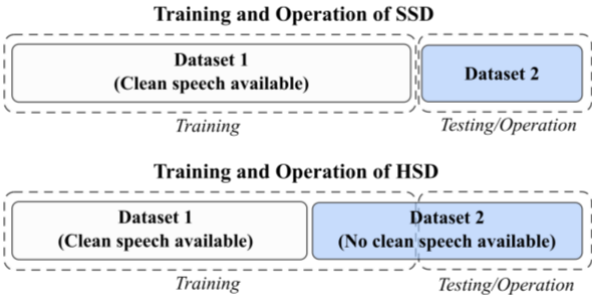


Fig. 5.5. Training and testing of cross-corpus experimentations.

The cross-corpus results when the other two datasets were used for training are provided in the form of bar charts in Figs. 5.6 and 5.7. Fig. 5.6 shows the results when the networks were trained on the VCTK corpus and were tested on the TIMIT or IEEE corpus. Fig. 5.7 reveals the results when the networks were trained on the TIMIT corpus and were tested on the IEEE or VCTK corpus. From these figures, it can be seen that the HSD approach provided superior performance over the SSD approach. For example, as shown in Fig. 5.6, when the network was trained on the VCTK corpus, for all the noise types, the HSD achieved better speech quality compared to the SSD approach.

Since many results are reported in this section, it helps to summarize the key findings below:

- Table 5.1 provides the outcome when dataset 1 in Fig. 5.5 was the IEEE corpus and dataset 2 was the TIMIT/VCTK corpus. The PESQ and STOI metrics for all the noise types were improved using the HSD approach compared to the SSD approach.

Table 5.1. Performance metrics (frame averaged ± standard deviation) for different noise types when the training set is from the IEEE corpus and the testing set is from the VCTK or TIMIT corpus having different clean speech signals than the IEEE corpus, the highest values are bolded.

	Training from IEEE corpus				
Noise Type		PESQ	STOI	LSD	SSNR
Babble	x + n	1.37 ± 0.10	0.61 ± 0.04	1.95 ± 0.11	-3.51 ± 1.05
	Wiener	1.39 ± 0.12	0.50 ± 0.05	1.67 ± 0.13	-2.88 ± 0.85
	SSD	1.43 ± 0.10	0.62 ± 0.04	1.46 ± 0.12	-0.96 ± 0.69
	HSD	**1.48** ± 0.12	**0.67** ± 0.05	**1.29** ± 0.09	**0.18** ± 0.61
Wind	x + n	2.04 ± 0.20	0.85 ± 0.03	1.13 ± 0.08	-2.78 ± 0.76
	Wiener	2.26 ± 0.28	0.71 ± 0.05	1.13 ± 0.06	-2.99 ± 0.51
	SSD	2.42 ± 0.24	0.86 ± 0.03	1.13 ± 0.12	2.56 ± 0.31
	HSD	**2.43** ± 0.24	**0.88** ± 0.03	**1.06** ± 0.07	**3.19** ± 0.55
Engine	x + n	1.91 ± 0.46	0.79 ± 0.12	1.37 ± 0.45	-0.38 ± 2.71
	Wiener	1.94 ± 0.46	0.64 ± 0.07	1.61 ± 0.45	-3.03 ± 1.68
	SSD	1.92 ± 0.43	0.75 ± 0.11	1.23 ± 0.28	0.74 ± 1.66
	HSD	**2.11** ± 0.45	**0.82** ± 0.11	**1.05** ± 0.26	**2.62** ± 2.00
Driving	x + n	1.77 ± 0.21	0.79 ± 0.04	1.39 ± 0.16	-4.02 ± 1.70
	Wiener	2.07 ± 0.21	0.66 ± 0.03	1.32 ± 0.12	-2.78 ± 0.76
	SSD	2.03 ± 0.24	0.81 ± 0.04	1.46 ± 0.20	1.30 ± 0.69
	HSD	**2.10** ± 0.26	**0.84** ± 0.04	**1.24** ± 0.14	**2.04** ± 0.86

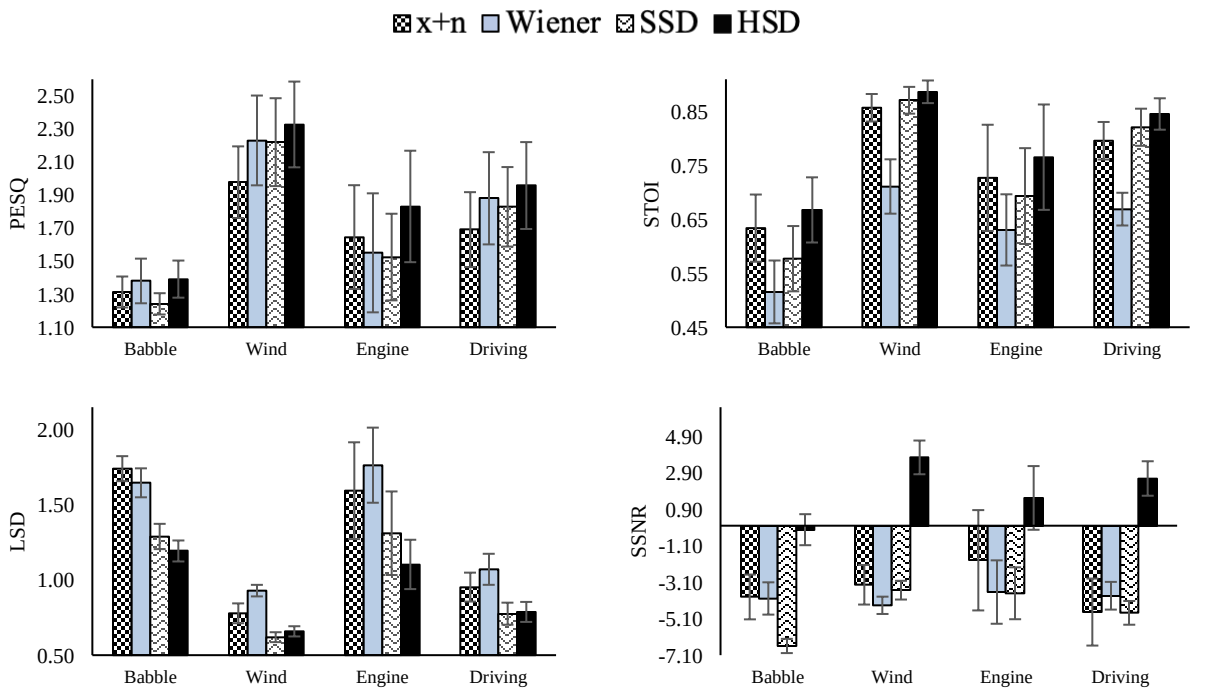


Fig. 5.6. Performance metrics for different noise types when the training set is from the VCTK corpus and the testing set is from the IEEE or TIMIT corpus.

- Fig. 5.6 depicts the outcome of the developed deep speech denoising when dataset 1 was the VCTK corpus and dataset 2 was the IEEE/TIMIT corpus. Across all the noise types, HSD outperformed SSD, except for the LSD metric in the presence of wind and driving noises.
- When the TIMIT corpus was selected as dataset 1 (Fig. 5.7), HSD outperformed SSD across all the noise types.

In general, the results shown in the figures and tables in this section reveal that SSD failed to improve speech intelligibility compared to unprocessed noisy speech signals x+n in many cases. For instance, in Fig. 5.6, SSD did not improve the PESQ and STOI metrics in babble and engine noises. Although the results of Wiener filtering outperformed the SSD results with respect to speech quality, Wiener filtering failed to improve speech intelligibility across the noise types compared to the raw noisy speech signals. That is why in the next set of experimentations, only the deep neural network solutions were considered for further analysis. Basically, the results reported in this section indicate that HSD is capable of performing effective speech denoising for unseen speech signals or speaker(s) while SSD or Wiener filtering are not.

5.5.2 Cross-Corpus Simulated Experiments: Unseen SNRs

The effect of SNR variation is examined in this section. In these experiments, the trained networks were tested with unseen SNRs (3 and -3 dB) to mimic a real-world setting. Unseen SNRs refer to the mismatch between the SNRs used for training and testing. Here, the IEEE corpus was used for training and the VCTK corpus was considered to act as unseen speech signals or speakers. Similar to the previous experiments, SNRs of -5, 0, and 5 dB were used for training. The average results of these experiments across all the noise types are shown in Fig. 5.8 for the PESQ and STOI

metrics. Other SNRs exhibited a similar behavior. As can be seen from this figure, substantial improvements in quality and intelligibility were achieved by the hybrid speech denoising.

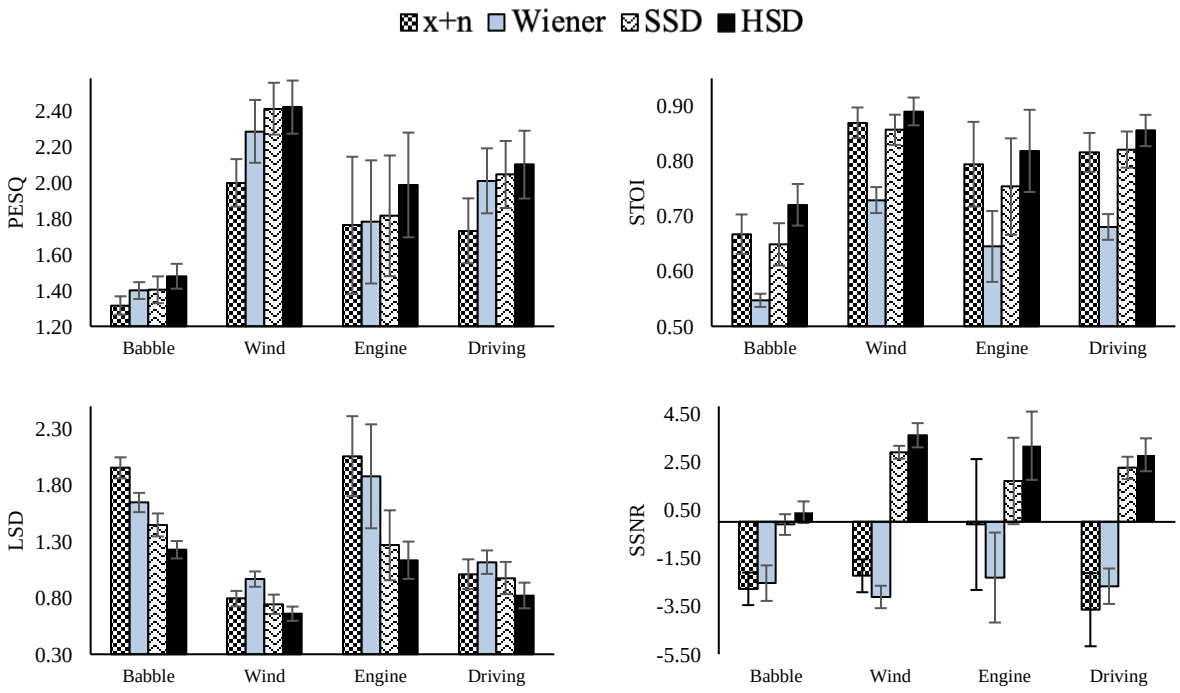


Fig. 5.7. Performance metrics for different noise types when the training set is from the TIMIT corpus and the testing set is from the VCTK or IEEE corpus.

To visually see the effectiveness of HSD, an utterance of a clean speech signal as well as its noisy version, its SSD denoised version, and its HSD denoised version are exhibited in Fig. 5.9. A low-frequency noise corrupted the clean speech signal in which the transient from vowel to another vowel cannot be followed visually, see Fig. 5.9(b). Although both SSD and HSD could reduce the noise, SSD also removed some structure of the clean speech and thus introduced speech distortion. In this figure, the two white arrows point to two regions of the spectrogram where the difference in the speech denoising is visually noticeable.

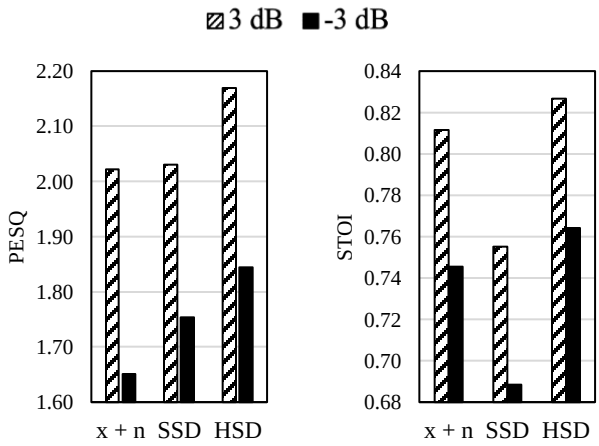


Fig. 5.8. Average PESQ and STOI over different noise types for unprocessed noisy speech (x+n) and denoised speech signals by SSD and HSD approaches.

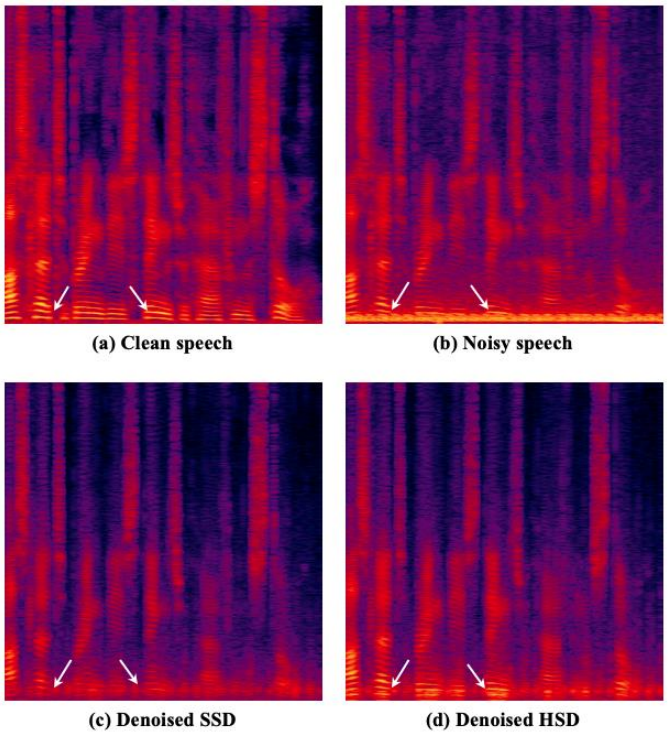


Fig. 5.9. Spectrograms of a sample speech signal corrupted by engine noise (a) clean speech, (b) noisy speech, (c) denoised speech by SSD, and (d) denoised speech by HSD.

5.5.3 Field Testing

In addition to the above experiments, a field testing was conducted. This was done by using the mid-side stereo microphone Zoom iQ7 [45] connected to an iPhone allowing to capture two decorrelated signals in two channels at a sampling frequency of 48 kHz. Noisy speech signals were captured in actual audio environments or in the field for comparing the SSD and HSD approaches. The experiments performed, involved a subject reading sentences in an actual cafe environment. The recording was done for one hour.

In this set of experiments, for SSD, the network was trained by combining all the three datasets of IEEE, VCTK, and TIMIT, see Fig. 5.10. For HSD, 15 minutes of the two-channel field data additionally were used for training as the input and the target of the network. The spectrograms of a sample speech signal from the field and the corresponding denoised signals using SSD and HSD are shown in Fig 5.11. From this figure, it can be seen that under realistic conditions, SSD was not able to achieve effective denoising due to its lack of generalization capability. The last spectrogram in Fig. 5.11(d) shows the spectrogram of the denoised speech by HSD which appears cleaner.

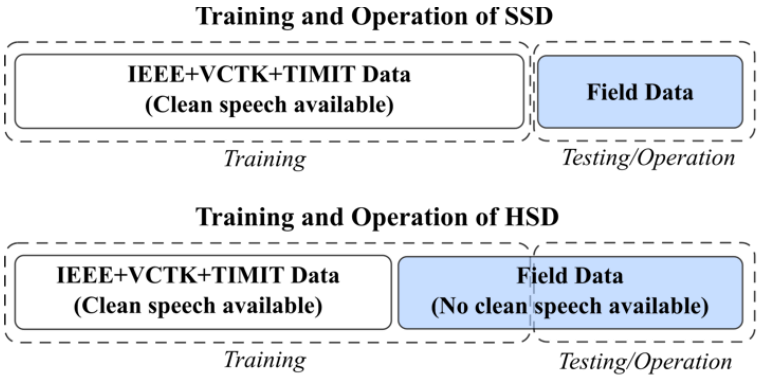


Fig. 5.10. Training and testing in the field.

Although performance metrics cannot be computed in the absence of clean speech signals during field testing, a subjective testing was conducted based on the commonly used Mean Opinion Score (MOS) measure [46]. Twelve normal hearing subjects were asked to score their preference for four signals: (1) unprocessed noisy speech signal, (2) summation of the right and left channels of the mid-side microphone signals (Mid signal), (3) denoised signal using SSD, and (4) denoised signal using HSD. Due to the unprecedented COVID-19 pandemic, the subjective testing was conducted virtually rather than in-person using a video conference utility. The above four audio signals for 15 speech passages were played randomly to each subject. Then, the subjects were asked to score their preference in this range [1(bad), 2(poor), 3(fair), 4(good), 5(excellent)] in terms of noise suppression as well as preservation of speech intelligibility. Fig. 5.12 shows the outcome of the subjective testing averaged over the 12 subjects. As can be seen from this figure, the SSD achieved the lowest averaged score and the developed HSD approach was preferred over the other signals due to its ability to take into consideration the unseen conditions in the field. A portion of a sample input noisy speech signal and its corresponding SSD denoised signal and HSD denoised signal are posted at this link www.utdallas.edu/~kehtar/FieldTestingResults.html for readers to hear the superiority of the HSD approach over the SSD approach when operating in the field.

5.6 Conclusion

A novel hybrid deep learning-based solution has been developed in this paper for the purpose of denoising noisy speech signals in real-world audio environments by not requiring to have clean speech signals. This solution involves the use of two noisy realizations of the same speech signal as the input and the target of a fully convolutional neural network. Extensive experimentations have been conducted to compare the developed hybrid approach with the commonly used

supervised approach in which clean speech signals are used as the target. The effectiveness of this approach has been established by showing that it generates improved outcomes in terms of four commonly used objective performance metrics as well as a subjective testing.

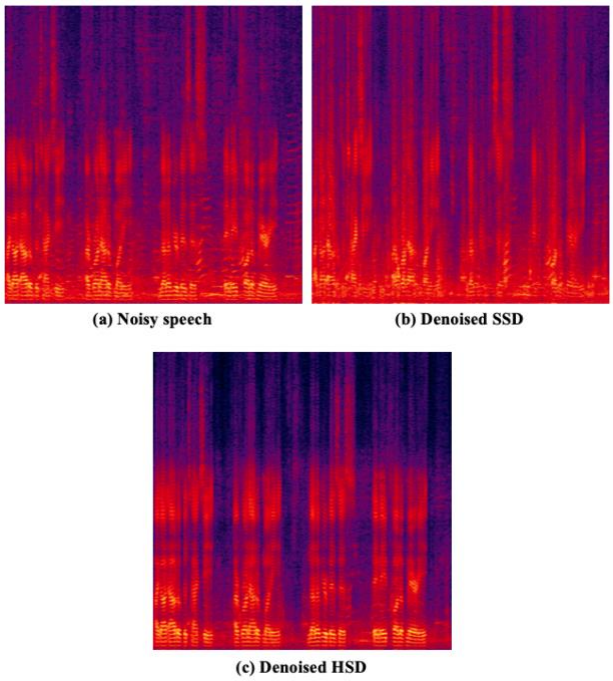


Fig. 5.11. Spectrograms of a sample speech signal in the field corrupted by cafeteria babble noise (a) noisy speech, (b) denoised speech by SSD, and (c) denoised speech by HSD.

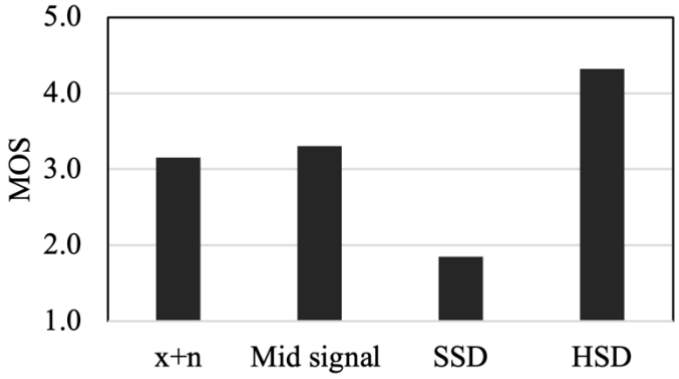


Fig. 5.12. Field subjective testing outcome in terms of Mean Opinion Score (MOS).

The developed deep speech denoising method is capable of adapting to unseen speaker(s) without the need to have clean speech signals. In summary, the developed hybrid deep speech denoising approach allows speech denoising to be carried out in the field as it does not rely on the availability of clean ground-truth speech signals.

5.7 References

[1] Berouti M, Schwartz R, Makhoul J, Enhancement of speech corrupted by acoustic noise. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 1979;4:208-11.

[2] Sreenivas T, Kirnapure P, Codebook constrained Wiener filtering for speech enhancement. IEEE Trans. Speech Audio Process. 1996;4(5), 383–89.

[3] Alamdari, N, Yaraganalu S, and Kehtarnavaz N, A real-time personalized noise reduction smartphone app for hearing enhancement, In: Proceedings of IEEE Conference on Engineering in Medicine and Biology Society, 2018: 1-5.

[4] Xu Y, Du J, Dai L, Lee C, An experimental study on speech enhancement based on deep neural networks, IEEE Signal Process. lett., 2014;21(1):65-8.

[5] Xu Y, Du J, Dai L, Lee C, A regression approach to speech enhancement based on deep neural networks, IEEE/ACM Trans. Audio, Speech, Lang. Process., 2015;23(1):7-19.

[6] Lu X, Tsao Y, Matsuda S, Hori C, Speech enhancement based on deep denoising autoencoder, In: Proceedings of INTERSPEECH, 2013:436-40.

[7] Pascual S, Bonafonte A, Serra J, SEGAN: speech enhancement generative adversarial network, arXiv preprint arXiv:1703.09452 (2017), doi:10.21437/Interspeech.2017-1428.

[8] Kumar A, Florencio D, Speech enhancement in multiple-noise conditions using deep neural networks, arXiv preprint: 1605.02427 (2017) Available: https://arxiv.org/abs/1605.02427.

[9] Erdogan H, Hershey J, Watanabe S, Le Roux J, Deep recurrent networks for separation and recognition of single-channel speech in nonstationary background audio, in New Era for Robust Speech Recognition, 2017:165-186, Springer, Cham.

[10] Han K, He Y, Bagchi D, Fosler-Lussier E, Wang D, Deep neural network based spectral feature mapping for robust speech recognition In: Proceedings of INTERSPEECH, 2015:2484-88.

[11] Tan K, Wang D, A convolutional recurrent neural network for real-time speech enhancement, In: Proceedings of INTERSPEECH, 2018:3229-33.

[12] Martín-Doñas J, Gomez A, Gonzalez J, Peinado A, A deep learning loss function based on the perceptual evaluation of the speech quality, IEEE Signal Process. Lett., 2018;25(11):1680-84.

[13] Liu Y, Zhang H, Zhang X, Using shifted real spectrum mask as training target for supervised speech separation, In: Proceedings of INTERSPEECH, 2018, pp. 1151-1155.

[14] Zhao Y, Wang Z, Wang D, Two-stage deep learning for noisy-reverberant speech enhancement, IEEE/ACM Trans. Audio, Speech, Lang. Process., 2019; 27(1):53-62.

[15] Williamson D, Wang Y, Wang D, Complex ratio masking for joint enhancement of magnitude and phase, In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 2016:5220-24.

[16] Liu C L, Fu S W, Li Y J, Huang J W, Wang H M, and Tsao Y, Multichannel speech enhancement by raw waveform-mapping using fully convolutional networks, IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2020; 28:1888-1900.

[17] Tawara N, Kobayashi T, and Ogawa T, Multi-Channel Speech Enhancement Using Time-Domain Convolutional Denoising Autoencoder, In Proc. INTERSPEECH, 2019: 86-90.

[18] Wang Z Q, and Wang D, All-Neural Multi-Channel Speech Enhancement, In Proc. INTERSPEECH, 2018: 3234-3238.

[19] Stowell D, Turner R, Denoising without access to clean data using a partitioned autoencoder, arXiv preprint: 1509.05982 (2015) Available: https://arxiv.org/abs/1509.05982.

[20] Furui S, Speaker-independent isolated word recognition based on emphasized spectral dynamics, In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 1986:1991-94.

[21] Purwins H, Li B, Virtanen T, Schlüter J, Chang Sh, Sainath T, Deep Learning for Audio Signal Processing, IEEE Journal of Selected Topics in Signal Processing, 2019:13(2):206-19.

[22] Rethage D, Pons J, Serra X, A Wavenet for speech denoising, In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 2018:5069-73.

[23] Germain F, Chen Q, Koltun V, Speech denoising with deep feature losses, arXiv preprint: 1806.10522v2, (2018) Available: https://arxiv.org/abs/1806.10522.

[24] Ravanelli M, Bengio Y, Speaker recognition from raw waveform with SincNet, In: Proceedings of IEEE Spoken Lang. Tech. Workshop, 2018: 1021-28.

[25] Fu S, Wang T, Tsao Y, Lu X, Kawai H, End-to-end waveform utterance enhancement for direct evaluation metrics optimization by fully convolutional neural networks, IEEE/ACM Trans. Audio, Speech, Lang. Process., 2018;26(9):1570-84.

[26] Lu Y, and Loizou P. Estimators of the magnitude-squared spectrum and methods for incorporating SNR uncertainty. IEEE Trans. Audio, Speech, and Lang. Process., 2010;19(5):1123-37.

[27] Lehtinen J, Munkberg J, Hasselgren J, Laine S, Karras T, Aittala M, Aila T, Noise2noise: Learning image restoration without clean data, arXiv preprint: 1803.04189, (2018) Available: https://arxiv.org/abs/1803.04189, 2018.

[28] Kraft S, Zölzer U, Stereo signal separation and up mixing by mid-side decomposition in the frequency-domain, In: Proceedings of 18th International Conference on Digital Audio Effects (DAFx), 2015.

[29] Kraft S, Zölzer U, Time-domain implementation of a stereo to surround sound upmix algorithm," In: Proceedings of 19th International Conference on Digital Audio Effects (DAFx), 2016.

[30] Ioffe S, Szegedy C, Batch normalization: Accelerating deep network training by reducing internal covariate shift, In: Proceedings of 32nd Int. Conf. Mach. Learn., 2015:448–56.

[31] Kingma D, Ba J, Adam: A method for stochastic optimization, arXiv preprint: 1412.6980, (2014), Available: https://arxiv.org/abs/1412.6980.

[32] Sehgal A, Kehtarnavaz N, A convolutional neural network smartphone app for real-time voice activity detection, IEEE Access, 2018;6:9017-26.

[33] Alamdari N, Kehtarnavaz N, A real-time smartphone app for unsupervised noise classification in realistic audio environments, In: Proceedings of Int. Conf. on Consumer Elec., 2019:1-5.

[34] Xia B, and Bao C, Wiener filtering based speech enhancement with weighted denoising auto-encoder and noise classification, Speech Communication, 2014; 60:13-29.

[35] Lai Y H, Tsao Y, Lu X, Chen F, Su Y T, Chen KC, Chen Y H, Chen L C, Li L P H, and Lee C H, Deep learning-based noise reduction approach to improve speech intelligibility for cochlear implant recipients, Ear and hearing, 2018; 39 (4): 795-809.

[36] Li R, Liu Y, Shi Y, Dong L, and Cui W, ILMSAF based speech enhancement with DNN and noise classification, Speech Communication, 2016; 85:53-70.

[37] Shi W, Zhang X, Zou X, and Han W, Deep neural network and noise classification-based speech enhancement, Modern Physics Letters B, 2017; 31(19-21): 1740096.

[38] McCloy D, Souza P, Wright R, Haywood J, Gehani N, Rudolph S, The PN/NC Corpus. Version 1.0, (2013) [Online], available: https://depts.washington.edu/phonlab/resources/pnnc/pnnc1/.

[39] DARPA-ISTO, The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus (TIMIT), speech disc cd1- 1.1 edition, 1990.

[40] Veaux C, Yamagishi J, MacDonald K, English multi-speaker corpus for CSTR voice cloning toolkit, [Online] http://homepages.inf.ed.ac.uk/jyamagis/page3/page58/page58.html.

[41] Salamon J, Jacoby C, Bello J, A dataset and taxonomy for urban sound research, In: Proceedings of the 22nd ACM Int. Conf. on Multimedia, 2014:1041-44.

[42] Rix A, Beerends J, Hollier M, Hekstra A, Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs, In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 2001;2:749–52.

[43] Taal C, Hendriks R, Heusdens R, Jensen J, An algorithm for intelligibility prediction of time–frequency weighted noisy speech, IEEE/ACM Trans. Audio, Speech, Lang. Process., 2011:19(7):2125-36.

[44] Barnwell III T, Objective measures for speech quality testing J Acoust Soc Am, 1979;66(6):1658-63.

[45] Zoom iQ7, https://www.zoom-na.com/products/handy-recorder/zoom-iq7-professional-stereo-microphone-ios, 2019.

[46] Subjective Performance Assessment of Telephone-Band and Wideband Digital Codecs, document ITU-Rec.P.830, 1996.

CHAPTER 6

AN EDUCATIONAL TOOL FOR HEARING AID COMPRESSION FITTING VIA A WEB-BASED ADJUSTED SMARTPHONE APP*

Authors – Nasim Alamdari[1], Edward Lobarinas[2], and Nasser Kehtarnavaz[1]

[1]The Department of Electrical and Computer Engineering,

[2]Callier Center for Communication Disorders

The University of Texas at Dallas

800 West Campbell Road

Richardson, Texas 75080-3021

6.1 Abstract

This paper presents an educational tool to learn about how hearing aid compression fitting is prescribed from a signal processing perspective. An interactive web-based program has been developed based on the widely used desired sensation level (DSL-v5) fitting rationale. This program can be accessed and used from any internet browser to generate the parameters of compression curves that correspond to the nine frequency bands used in DSL-v5. These parameters are then transferred to a smartphone in the form of a datafile to be used by a compression app, which operates as a virtual hearing aid, running in real-time on both iOS and Android smartphones. This educational tool was found to be very easy-to-use by signal processing engineers as no programming knowledge is needed for adjusting gain across frequency bands and subsequently running the corresponding compression curves in the smartphone app to appropriately compress input sound signals.

6.2 Introduction

According to the World Health Organization (WHO), more than 450 million people around the world have hearing loss and it is estimated that by 2050 more than 900 million people will have hearing loss [1]. One way that hearing loss is managed is via the use of hearing aid devices. A hearing aid device is either worn in or behind an ear and consists of three major components: a microphone, a signal processor, and a speaker. The signal processor amplifies acoustic input from the microphone in order to compensate for a person's hearing loss. The amplified sounds are then sent to the ear through the speaker. In modern hearing aids, the signal processor runs a number of

signal processing algorithms such as compression, adaptive feedback cancellation, and noise reduction.

The main signal processing algorithm running on the signal processor of a modern hearing aid device to cope with hearing loss is compression. The human auditory system is capable of perceiving a wide dynamic range of sound intensity from soft to loud. For a person with sensorineural hearing loss, the dynamic range can be significantly reduced. Importantly, the relative intensity difference between barely audible and uncomfortably loud becomes smaller because perception for very loud sound remain relatively similar across a wider range of hearing loss and for individuals with normal hearing. Thus, in order for sound to be audible, the acoustic information for soft, moderate, and loud sounds must occupy a smaller working space. In order to address this problem, a method of squeezing information (compression) into this space is used.

The process of compression involves mapping the wide dynamic range of normal hearing range into a smaller dynamic range commensurate with the degree of hearing loss [2, 3]. Compression can be achieved via input/output compression curves or functions as illustrated in Fig. 6.1. As can be seen from this curve, more gain is provided for inputs that are less than 75 dB SPL (Sound Pressure Level) than for inputs that exceed 75 dB SPL. The gain for inputs exceeding 75 dB SPL is compressed relative to the linear gain provided for lower level inputs. Considering that hearing loss differs across frequency bands (typically 9 frequency bands are used), a series of compression curves are often needed to accommodate differences in residual dynamic range at each of these frequencies. The fitting of a hearing aid device is normally carried out by adjusting gain across different frequency bands for soft, intermediate, and loud input sound levels. Fig. 6.2 illustrates a frequency response curve with higher gain for frequencies in the 3-4 kHz range. Such a curve

could be used for an individual with noise-induced hearing loss that typically creates a notch of poorer hearing in the 4 kHz region.

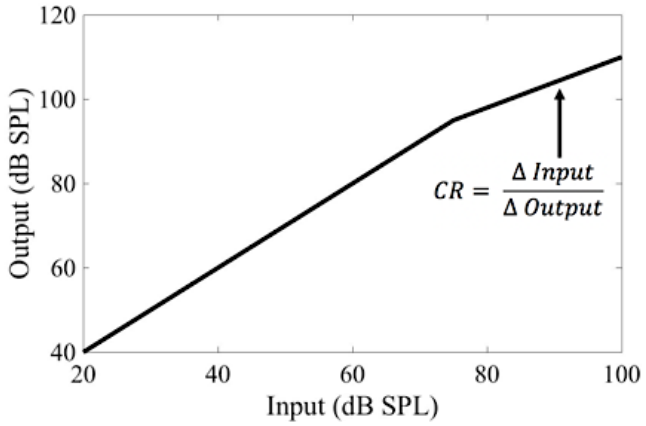


Fig. 6.1. Sample input/output compression function.

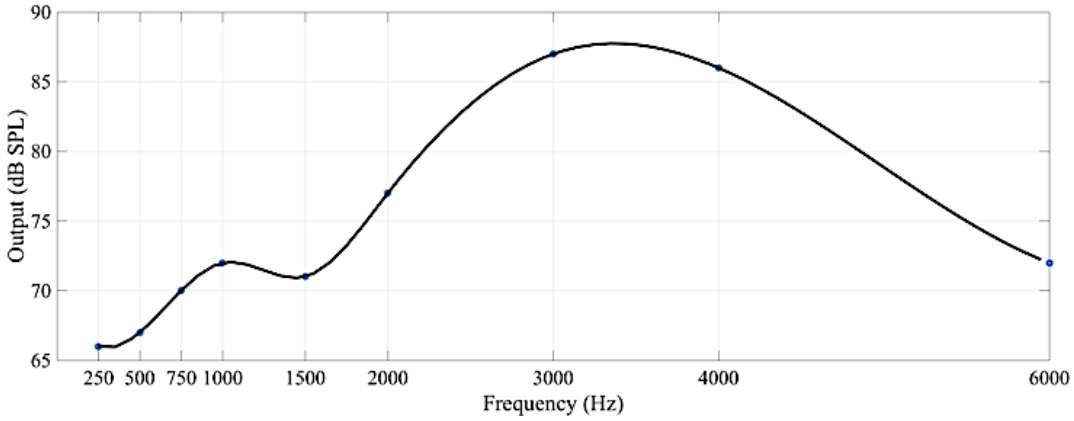


Fig. 6.2. Sample frequency response curve for an input sound level of 65 dB SPL.

The objective of this paper is to provide an easy-to-use educational tool for signal processing engineers to learn about how hearing aid amplification fitting are implemented using signal processing compression curves. The widely used prescriptive fitting of DSL-v5 (Desired Sensation Level – version 5) by Hand [4] is considered in this paper. This educational tool consists of an interactive web-based compression fitting program and a signal processing compression algorithm running on the ARM processor of smartphones (both iOS and Android) as an app. The

interaction between the web-based fitting program and the smartphone app is carried out via a datafile. This educational tool is portable, thus enabling its wide spread utilization for studying the effect of changing compression gains in realistic audio environments.

The remainder of this paper is organized as follows. Section 6.3 describes both the web-based compression and the smartphone app components of the developed educational tool. Sample compression fittings are then presented in section 6.4 followed by the conclusion in section 6.5.

6.3 Hearing Aid compression Educational Tool

Fig. 6.3 shows a block diagram of the components of the developed educational tool. The DSL-v5 compression fitting is carried out interactively on a browser. The languages used to write this interactive web-based program include HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), and JavaScript. This web-based fitting is freely accessible and can be run on any browser via this link: http://www.utdallas.edu/~kehtar/WebBasedFitting.html. No knowledge of the programming languages used to create the program is needed in order to use it.

Fig. 6.4 shows the graphical-user-interface (GUI) of the web-based compression fitting program. In this figure, the entries of the first table from the top correspond to the age of a subject (more than 3 years old, 4-5 years old, 6 years old, or more than 6 years old), the type of hearing aid (CIC (Completely in Canal), ITC (In-The-Canal), ITE (In-The-Ear, or BTE Behind-The-Ear)), and the ear coupling of hearing aid (Earmolds or Eartips). The second table from the top incorporates the subject's audiometric thresholds for whom compression fitting is done. Audiometric thresholds (called audiogram) correspond to the softest level of sound a person can hear at different frequencies. The online utility in [5] is used by the program to obtain these measurements.

The next four tables indicate gain across 9 frequency bands for speech inputs at soft, moderate, loud, and maximum levels. The entry Target SPL Output in these tables is automatically determined based on the audiogram measurements and the pre-specified values in the DSL-v5 prescriptive tables. The other pre-specified entries of Input SPL, RECD (Real-Ear to Coupler Difference) and Mic Effect in DSL-v5 are automatically subtracted to generate specific gain in the frequency bands for soft (55dB), moderate (65dB), loud (75dB), and maximum output (90dB) levels.

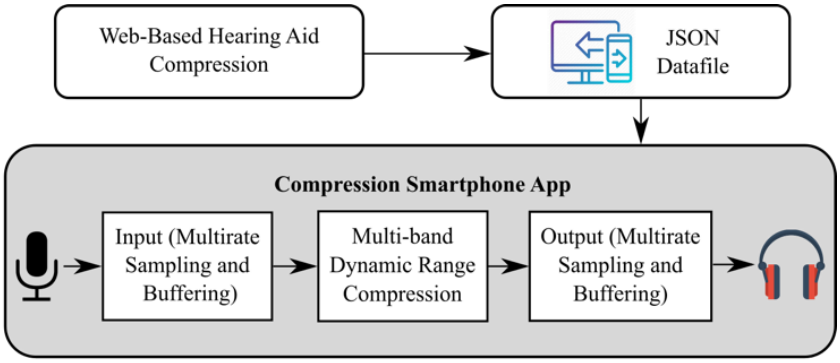
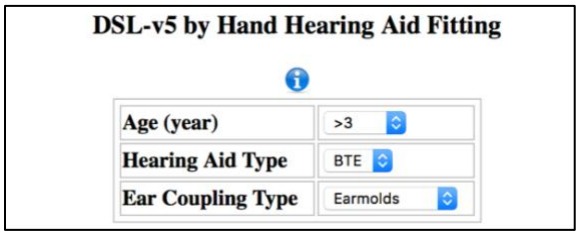
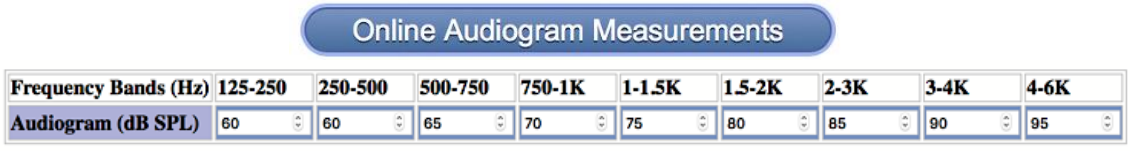


Fig. 6.3. Components of the hearing aid compression educational tool.



(a) Subject and hearing aid information.



(b) Audiogram measurements from an online test.

Fig. 6.4. Graphical-user-interface (GUI) of the web-based compression fitting program.

(c)

Soft Speech (55 dB)

Frequency Bands (Hz)	125-250	250-500	500-750	750-1K	1-1.5K	1.5-2K	2-3K	3-4K	4-6K
Target SPL Output	84	82	81	83	84	90	93	95	94
Subtract Input SPL	45	47	42	40	38	34	32	31	30
Subtract RECD	3	5	5	6	8	6	2	3	8
Subtract Mic Effect	1	1	1	1	2	3	4	2	1
Target Gain	35	29	33	36	34	43	53	58	54

Moderate Speech (65 dB)

Frequency Bands (Hz)	125-250	250-500	500-750	750-1K	1-1.5K	1.5-2K	2-3K	3-4K	4-6K
Target SPL Output	89	88	88	91	93	100	103	105	104
Subtract Input SPL	55	57	52	50	48	44	42	41	40
Subtract RECD	3	5	5	6	8	6	2	3	8
Subtract Mic Effect	1	1	1	1	2	3	4	2	1
Target Gain	30	25	30	34	35	47	55	59	55

Loud Speech (75 dB)

Frequency Bands (Hz)	125-250	250-500	500-750	750-1K	1-1.5K	1.5-2K	2-3K	3-4K	4-6K
Target SPL Output	90	92	94	97	100	106	111	113	109
Subtract Input SPL	57	65	66	64	65	61	57	56	50
Subtract RECD	3	5	5	6	8	6	2	3	8
Subtract Mic Effect	1	1	1	1	2	3	4	2	1
Target Gain	29	21	22	26	25	36	48	52	50

Max Output (90 dB)

Frequency Bands (Hz)	125-250	250-500	500-750	750-1K	1-1.5K	1.5-2K	2-3K	3-4K	4-6K
Target SPL Max Output	106	107	110	113	114	118	121	122	121
Subtract RECD	3	5	5	6	8	6	2	3	8
Target OSPL-90	103	102	105	107	106	112	119	119	113

Update Tables

(d) Gains set according to the audiogram and DSL-v5 prescriptive settings for 9 frequency bands.

Fig. 6.4, continued. Graphical-user-interface (GUI) of the web-based compression fitting program.

Gain from the DSL-v5 tables is then converted into compression curves corresponding to the 9 frequency bands. Each compression curve is expressed in terms of the following four parameters [6] (see Figs. 6.1 and 6.6): (i) Compression Threshold (CT) - This parameter indicates the point after which the compression is applied. (ii) Compression Ratio (CR) - This parameter indicates the

amount of compression. (iii) Attack Time (AT) – This parameter indicates the time it takes for the compression module to respond when the signal level changes from a low to a high value. (iv) Release Time - This parameter indicates the time it takes for the compression module to respond when the signal level changes from a high to a low value. The default values of the attack and release times are set to 5ms and 100ms, respectively. The compression ratio of each frequency band for the compression thresholds of 55 dB and 75 dB SPL is computed as follows:

$$CR = \frac{\Delta Input}{\Delta Output} = \frac{75 - 55}{(75 + Gain_{at\ 75\ dB}) - (55 + Gain_{at\ 55\ dB})}$$

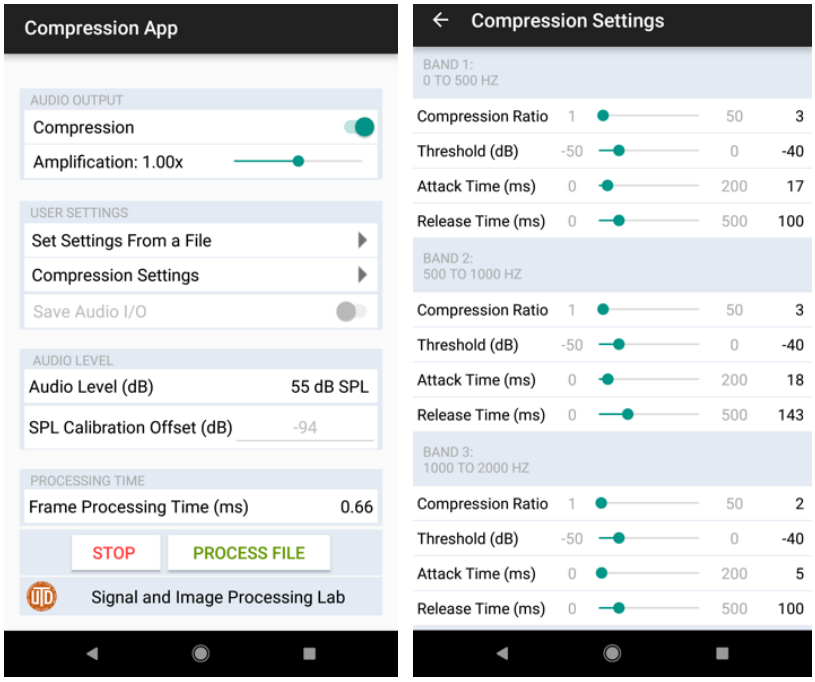
The button at the end of the web-based program generates a JSON (JavaScript Object Notation) datafile which includes the parameters of the compression curves in a readable form. JSON is a popular web programming approach for transmitting data to other platforms. The created JSON datafile is then transferred to the smartphone and placed inside the compression app folder to be used by it.

Another component of the developed educational tool is a smartphone app for both Android and iOS smartphones. This app performs multiband wide dynamic range compression (WDRC) according to the compression curves determined by the web-based compression program. The app consists of an input module, a multi-band dynamic range compression module, and an output module. The compression module in the app uses the multiband WDRC in [7] in which CTs are defined in the range of [-50, 0] dB. Positive CTs of the JSON file are redefined based on this range via this equation $CT_{app} = CT_{JSON} – 95dB$.

Multi-rating is used in the input/output modules to allow achieving the lowest audio latency on smartphones. For iOS smartphones, the audio i/o needs to run at 48kHz and the i/o buffer size needs to be kept at 64 samples or 64/48000=1.33ms in order to achieve the lowest audio latency.

For Android smartphones (for example Google Pixel), the audio i/o needs to run at 48kHz and the i/o buffer size needs to be kept at 192 samples or 192/48000 = 4ms in order to have the lowest audio latency. For the app to run in real-time, the compression module runs in frame-based manner at 16kHz with a 25ms processing frame size and with 50% overlap, (i.e. a frame gets processed every 12.5ms). To synchronize the audio i/o and the compression modules, circular buffers are utilized. An input circular buffer collects input samples from the audio i/o buffer until the overlapped frame size of 12.5ms (600 samples) is reached. It is then down-sampled, decimated by a factor of 3 and fed into the compression module. After a frame is passed through the compression module, it is up-sampled and interpolated before being placed into an output circular buffer, which then outputs the audio at the rate of 64 samples (iOS) or 192 samples (Android) at 48kHz. This process maintains the lowest audio latency available by the smartphone i/o hardware. Interested readers are referred to [8-12] for more details of the modules in the compression smartphone app. Fig. 6.5 shows the GUI of the Android version of the app. The compression settings are read from the web-based generated JSON datafile. To gain computational efficiency, the nine frequency bands are mapped into the following five frequency bands [0Hz-500Hz], [500Hz-1000Hz], [1000Hz-2000Hz], [2000Hz-4000Hz], and above 4000Hz. The compressed output can be heard via a hardwired earphone or wirelessly (via Bluetooth) on the smartphone. The web-based compression fitting program also incorporates the DSL-v5 by Hand document. A video clip demonstrating various components of this educational tool is provided at this link: http://www.utdallas.edu/~kehtar/EducationalCompressionFitting.mp4. This video clip shows the entire fitting process including obtaining audiogram measurements for the 9 frequency bands, computing gains from the DSL-v5 tables, updating the parameters of the compression curves

according to gain, and finally generating a JSON datafile containing the compression parameters. It also shows how to transfer this datafile to a smartphone platform and how to run the compression app based on this datafile.



(a) (b)

Fig. 6.5. GUI of compression smartphone app (Android version): (a) main page, (b) compression parameters.

6.4 Sample Compression Results

In this section, sample compression fitting results are provided for different degrees of hearing loss. Table 6.1 shows sample gains corresponding to three degrees of hearing loss: normal/mild, moderate, and severe. The input-output compression curve or function generated by the developed educational tool for the moderate hearing loss row of Table 6.1 is shown in Fig. 6.6. This I/O compression function has the following two CTs that can get adjusted: (i) linear gain for sound levels lower than the CT1 (soft sounds), (ii) gain between CT1 and CT2 based on the compression

ratio CR, (iii) gain for sound levels higher than CT2, and finally (v) a peak clipping control to ensure the output does not exceed the maximum power output (MPO).

Based on the scale of 1 (very difficult-to-use) through 5 (very easy-to-use), five signal processing engineers were asked to rate this educational tool. All five testers rated the tool as a 5; very easy-to-use. All the engineers commented that their rating was based on the fact that no web programming knowledge and no familiarity with smartphone software tools were needed in order to run the smartphone app using the compression curves derived from the gain settings via the DSL-v5 prescriptive fitting webpage.

Table 6.1. Sample gains for three degrees of hearing loss in 9 frequency bands for a >6-year subject, a BTE hearing aid, and an earmold ear coupling.

Degree of Hearing Loss	**Audiogram (dB SPL)**	**Gains from DSL-v5 Tables (dB SPL)**	
Normal-Mild	{10, 15, 20, 25, 25, 20, 25, 25, 15}	*Soft Speech*: *Moderate Speech*: *Loud Speech*: *MPO:*	{4, 5, 10, 13, 11, 13, 25, 27, 9} {2, 4, 9, 11, 11, 16, 25, 25, 9} {1, 2, 5, 7, 6, 12, 22, 21, 7} {88, 89, 90, 92, 90, 91, 98, 95, 86}
Moderate	{60, 60, 65, 70, 70, 75, 80, 85, 85}	*Soft Speech*: *Moderate Speech*: *Loud Speech*: *MPO:*	{35, 29, 33, 36, 34, 43, 53, 58, 54} {30, 25, 30, 34, 35, 47, 55, 59, 55} {29, 21, 22, 26, 25, 36, 48, 52, 50} {103,102,105,107,106, 112, 119, 119, 113}
Severe	{60, 60, 65, 70, 85, 85, 95, 100,110}	*Soft Speech*: *Moderate Speech*: *Loud Speech*: *MPO:*	{35, 29, 33, 36, 43, 48, 64, 69, 71} {30, 25, 30, 34, 45, 52, 66, 70, 72} {29, 21, 22, 26, 36, 43, 58, 61, 67} {103,102,105,107,114, 116, 127, 126, 127}

The processing time of the developed compression smartphone app for each frame is 0.8ms on the iPhone 8 and 0.9ms on the Google Pixel smartphones. Considering that these times are well below the overlapped frame time of 12.5ms, the app runs in real-time with no frames getting skipped.

The CPU and memory utilization for the iOS version of the app, reported by Xcode IDE [13], is 4% and 18MB, and for the Android version of the app, reported by Android Studio IDE [14], is 6% and 66MB, respectively.

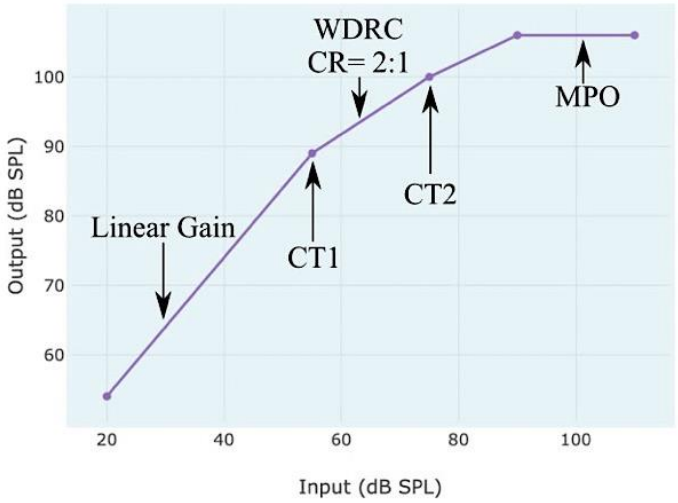


Fig. 6.6. Input-output compression curves or functions in the 5th frequency band for the moderate degree of hearing loss listed in Table 6.1.

6.5 Conclusion

In this paper, an educational tool to learn about hearing aid compression has been introduced. A web-based program has been developed that effectively mimics the DSL-v5 hearing aid fitting steps and prescriptive gain across nine frequency bands for four speech intensity levels. These gain values are converted into nine compression curves or functions whose parameters are stored into a datafile. This datafile is then used by a smartphone app (both iOS and Android) which processes input sound files in real-time based on the DSL-v5 designed compression curves. The developed web-based program can be easily duplicated for other popular hearing aid prescriptive fittings such as NAL-NL2 [15]. The approach presented in this paper provides an effective learning tool and a

framework for teaching of and research in hearing aid compression in an easy-to-use and universally accessible manner.

6.6 References

[1] World Health Organization, http://www.who.int/mediacentre/factsheets/fs300/en/, 2019.

[2] D. Eddins, *Sandlin's Textbook of Hearing Aid Amplification*, Plural Publishing, 2014.

[3] T. Venema, *Hearing Aid Amplification: Technical and Clinical Considerations,* Singular Publishing Group, 2000.

[4] Western University, DSL-v5 by Hand, https://www.dslio.com/wpcontent/uploads/2014/06/DSL-5-by-Hand.pdf, 2014.

[5] S. Pigeon, https://hearingtest.online/, 2017.

[6] Starkey, Compression Handbook, https://starkeypro.com/pdfs/The_Compression_Handbook.pdf, 2017.

[7] MathWorks, https://www.mathworks.com/help/audio/examples/multiband-dynamic-range-compression.html, 2018.

[8] N. Kehtarnavaz, S. Parris, and A. Sehgal, "Using smartphones as mobile implementation platforms for applied digital signal processing courses," *Proceedings of IEEE Signal Processing and Signal Processing Education Workshop (SP/SPE),* pp. 313-318, 2015.

[9] N. Alamdari, S. Yaraganalu, and N. Kehtarnavaz, "A real-time personalized adaptive noise reduction smartphone app for realistic audio environments," *Proceedings of IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, Philadelphia, PA, Dec 2018.

[10] T. Chowdhury, A. Sehgal, and N. Kehtarnavaz, "Integrating signal processing modules of hearing aids into a real-time smartphone app," *Proceedings of IEEE 40th International Conference on Engineering in Medicine and Biology*, Honolulu, HI, July 2018.

[11] A. Sehgal and N. Kehtarnavaz, "Utilization of two microphones for real-time low-latency audio smartphone apps," *Proceedings of IEEE International Conference on Consumer Electronics*, Las Vegas, NV, Jan 2018.

[12] N. Kehtarnavaz, A. Sehgal, and S. Parris, *Smartphone-Based Real-Time Digital Signal Processing, Second Edition*, Morgan and Claypool Publishers, 2018.

[13] Apple, https://developer.apple.com, 2018.

[14] Google, https://developer.android.com, 2018.

[15] G. Keidser, H. Dillon, L. Carter, and A. O'Brien, "NAL-NL2 empirical adjustments," *Trends in Amplification*, vol. 16, pp 211-223, 2012.

CHAPTER 7

PERSONALIZATION OF HEARING AID COMPRESSION BY HUMAN-IN-THE-LOOP DEEP REINFORCEMENT LEARNING*

Authors – Nasim Alamdari[1], Edward Lobarinas[2], and Nasser Kehtarnavaz[1]

[1]The Department of Electrical and Computer Engineering,

[2]Callier Center for Communication Disorders

The University of Texas at Dallas

800 West Campbell Road

Richardson, Texas 75080-3021

7.1 Abstract

Existing prescriptive compression strategies used in hearing aid fitting are designed based on gain averages from a group of users which may not be necessarily optimal for a specific user. Nearly half of hearing aid users prefer settings that differ from the commonly prescribed settings. This paper presents a human-in-the-loop deep reinforcement learning approach that personalizes hearing aid compression to achieve improved hearing perception. The developed approach is designed to learn a specific user's hearing preferences in order to optimize compression based on the user's feedbacks. Both simulation and subject testing results are reported. These results demonstrate the proof-of-concept of achieving personalized compression via human-in-the-loop deep reinforcement learning.

7.2 Introduction

In hearing impaired individuals, the relative intensity difference between barely audible and uncomfortably loud sound becomes smaller. Thus, in order to achieve optimal audibility, sound must be calibrated to occupy a smaller range of sound pressure levels (SPLs). This dynamic range adjustment is achieved through the process of compression [1]. Compression in a reduced dynamic range is the key function of modern hearing aids. This process involves squeezing or fitting sound into the residual audibility range of a hearing aid user. In hearing aid fitting, so-called compression curves are set up by adjusting gains across a number of frequency bands based on a user's audiometric profile. The two most widely used hearing aid prescriptions are NAL-NL2 [2] and desired sensation level (DSL-v5) [3]. These prescriptions correspond to gain tables across a number of frequency bands for three sound levels, soft, moderate, and loud.

It has been reported that up to half of individuals using fitted hearing aids preferred amplification or compression settings different than the prescription provided [4-9]. Considering that suprathreshold hearing perception varies from person to person and that acoustic environments encountered vary from person to person, several papers in the literature have examined self-adjustment or self-tuning of hearing aid fitting relative to the one-size-fits-all prescriptive fitting [10-18]. In [18], it was reported that hearing aid users favored gain settings that were different from the NAL prescription settings both in quiet and in noise. Self-adjustments carried out on a custom hardware/software such as those recently reported in [19, 20] have also demonstrated improvements in hearing perception that can be gained over prescriptive fitting. In addition, the benefits of hearing aid personalization were examined in [11, 12]. In [11], the parameter space consisted of four possible combinations of microphone mode (omnidirectional and directional) and noise reduction state (active and off). First, preferences of a user were learnt in order to create a supervised trained model. Then, the model was used to derive an optimal setting among the four choices.

There have been only a few studies reporting algorithms to learn personalize audio compression gains (or compression ratios). In [21-23], a machine learning approach for self-adjustment or self-tuning of compression was presented. In these papers, a Gaussian regression model was used to achieve personalized compression by estimating its parameters from training data. User preferences were obtained via hearing assessments by listening to music clips in [21,22]. Although the results reported show the benefits of personalization, differences in preferences between music clips and conversation in noisy environments (e.g., in babble noise) were not addressed. Understanding speech in the presence of bothersome background noise is expressed as a major

challenge by hearing aid users [24]. Furthermore, in [21-23], only twenty preference iterations were done for modeling the hearing preference of a user. In actual audio environments, this many iterations would be inadequate for modeling various non-linearities associated with hearing perception. Although these findings show the overall usefulness of personalization, preferred hearing cannot be achieved without a proper design of the personalization framework. In a recent study [25], an agent is trained using simulated contextual preferences within a controlled environment. There, the user model is created by hypothesizing correlation among users' preferences based on a number of their observable characteristics. However, the validity of the assumption made in [25], i.e. categorizing users' preferences, is not supported by clinical evidence. Hence, human feedback in the training loop is deemed vital in order to provide a personalization that can deliver preferred hearing to a specific user.

To address previous design limitations, a human-in-the-loop (HITL) interactive machine learning compression approach based on deep reinforcement learning (DRL) [26] is developed in this paper. In our approach, the user is placed in the learning loop. A DRL network is designed to receive preference feedback from the user. As a result, it becomes possible to deal with various non-linearities of human hearing perception. In general, placing human in the loop of training a machine learning model enables reducing the error made by a trained model. The use of the conventional reinforcement learning is not sufficient to perform personalization for hearing aid compression. For compression, it is vital to include data from human feedback to optimize and improve the model over time. That is why in this work, a user's feedback is placed in the learning loop for personalization of compression, considering that the user's feedback is sparse in practice. A combination of a convolutional neural network (CNN) [27] and a bidirectional long short-term

memory recurrent neural network [28] or CNN-BiLSTM is used to model a user's preferences in those audio environments that are of interest to the user.

As discussed in [29], user feedback is affected by biases. Thus, rather than absolute feedback, pairwise or relative hearing assessments are deemed more suitable [30]. Hence, in our approach, a user is subjected to a series of compressed audios to express his/her preferences towards training the model via the reward/punishment mechanism of reinforcement learning; an approach that has been successfully applied to gaming [31] and robotics [32]. The developed DRL approach provides personalized compression that can be utilized in the field for hearing aid compression studies. It should be noted that the focus of this paper is on the development of a human-in-the-loop deep reinforcement learning approach for personalizing audio compression and to show its proof-of-concept by carrying out simulated experiments and a limited clinical subject testing. However, deployment would require carrying out extensive clinical testing.

To describe our approach in detail, the remainder of this paper is organized as follows. Section 7.3 covers the developed approach to personalize hearing aid compression or fitting via human-in-the-loop DRL as well as a protocol to perform human preference assessment. The experimental results and discussion are then presented in section 7.4 followed by the conclusion in section 7.5.

7.3 Personalized Compression Approach

To set the stage for the developed personalized compression, the conventional reinforcement learning (RL) is first briefly described. In a reinforcement learning framework, an agent and an environment interact over a series of steps. At each time step *t*, the agent receives an observation or state $s_t \epsilon\ S$ from the environment and sends an action $a_t \epsilon A$ back to the environment with *S* and *A* denoting the state and action sets, respectively. In a conventional RL framework, based on a

given action, the environment generates the next state together with a reward $r_t \epsilon R$ with R denoting the reward set, and the goal is to maximize reward over time. Fig. 7.1(a) shows a block diagram of a conventional RL framework.

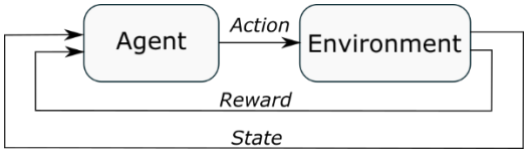
The success of RL heavily depends on setting up an effective reward function. Many real-world problems are complex, and it is often difficult to formulate an effective reward function. Inverse reinforcement learning (IRL) [33] can be used to design a reward function, which can then get deployed to train the agent using (deep) reinforcement learning. In order to build an effective reward function, human feedback can be used to evaluate the behavior of the agent [34, 35]. In our case, in order to model and learn hearing preferences via deep reinforcement learning, the listener's preferences are used.

Obtaining rewards in a direct manner, based on user feedback, is labor intensive and makes the training process impractical because thousands of iterations and user feedbacks would be needed. In order to decrease the number of user feedbacks and thus enable a practical deployment of the personalized compression, first a reward function is considered to model hearing preferences of a user in an asynchronous manner. This is achieved by carrying out comparison between instances of two different compressed audios. Then, an agent is trained to maximize reward. Fig. 7.1(b) shows a block diagram of this approach. Unlike the conventional RL in which reward is computed by the environment, here reward is computed based on user preferences.

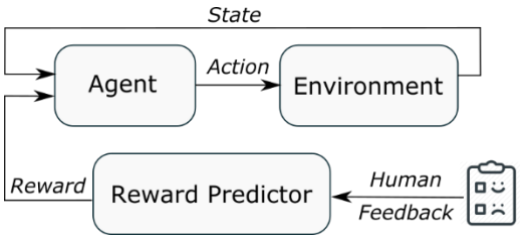
7.3.1 Personalized Fitting Protocol

Compression in hearing aid fittings is normally performed via software tools that are provided by hearing aid manufacturers. These software tools are used to automatically set gains across a

number of frequency bands using established prescriptions based on group averages or with manufacturers adding their own variations.



(a)



(b)

Fig. 7.1. (a) Block diagram of a conventional reinforcement learning framework. (b) Deep reinforcement learning with user's feedback in which reward is obtained based on user preferences.

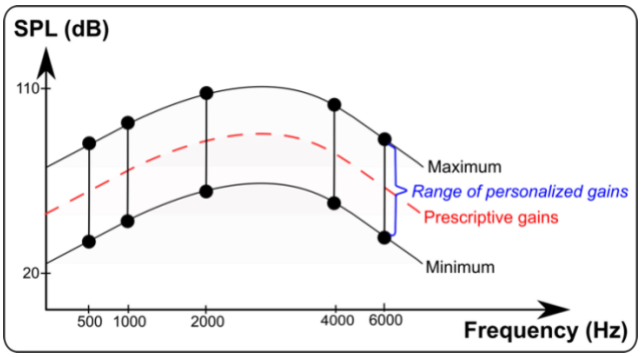


Fig. 7.2. Illustration of gain ranges across different frequency bands.

A user's audiogram and the prescription gains are used to set the target gains for that user. Across each frequency band, a different compression curve is used to generate multiband dynamic range compression (DRC). In this paper, the DSL-v5 by Hand prescription in [36] is considered to serve

as the reference compression. In other words, the gains in the DSL-v5 tables are used to compute the reference DRC parameters consisting of compression ratio (change in gain) and compression threshold (sound level at which compression is applied). The process of personalization involves modification of the gains specified by DSL-v5 or any other generic prescription based on user preferences.

In this study, the following steps are taken to achieve personalized compression for a specific user. The first step is assessing hearing sensitivity by measuring the audiogram of a user. In the second step, the compression gains of the user are set by using fitting software. In this work, the DSL-v5 prescriptive fitting software is used. The third step is initializing the human centered-DRL framework with the compression ratios obtained in the second step as the starting point. In the fourth step, the compression ratios are adjusted by going through the training process of the human centered-DRL framework (an illustration of the gain change ranges for the agent action in the DRL framework is depicted in Fig. 7.2). The final or fifth step involves comparing the performance of the personalized compression with the prescriptive or reference compression.

In the personalized framework, an agent that is interacting sequentially with the environment over a number of time steps is considered similar to the one in [34]. At each time step, the agent receives a new state (observation) from the environment and performs an action. Over an episode, the agent performs interaction for a number of time steps. In typical RL settings, a reward at each time step is fed into the agent as well.

However, here rather than a predefined reward, a reward that is modeled based on a user's preferences is considered. In other words, by putting human feedback in the learning loop, it is attempted to optimize the agent learning and thus the user hearing perception. The personalization

protocol and block diagram of the developed DRL-based personalized hearing aid compression framework is shown in Table 7.1 and Fig. 7.3, respectively. Each block in Fig. 7.3 is described in the subsections that follow.

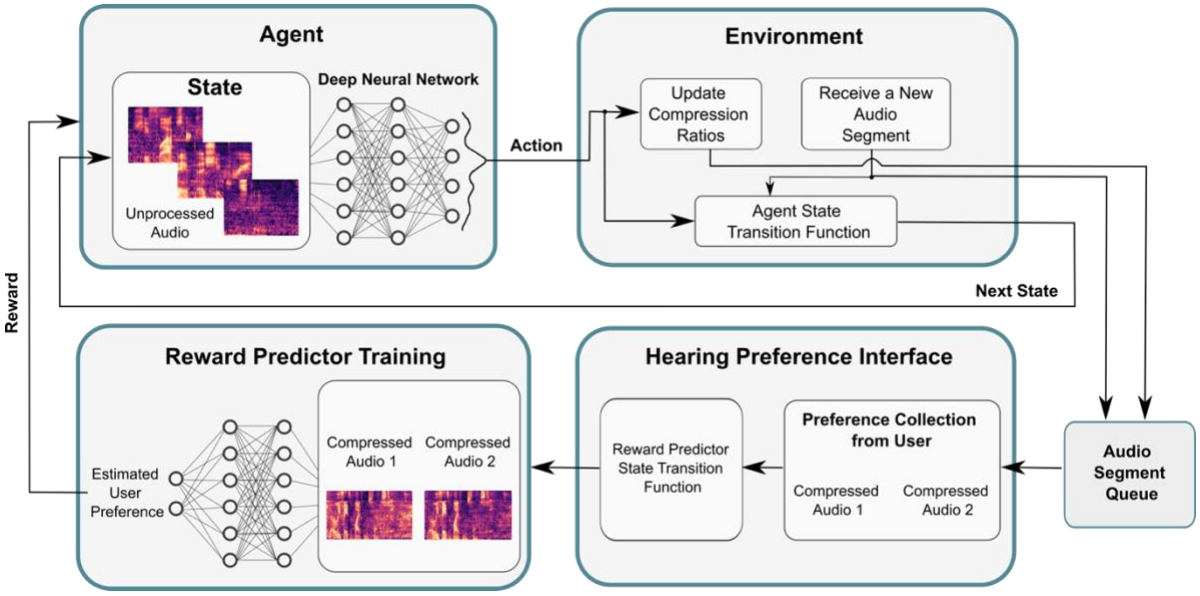
Table 7.1. Personalized Fitting Protocol

Step	Description
1.	Measuring audiogram of the user
2.	Defining the compression gains of the user using a prescriptive fitting software (e.g., DSL-v5 in [3]); computing compression ratios in a number of frequency bands
3.	Initializing the human-centered DRL framework with the above compression ratios
4.	Running the policy in the environment and storing a set of compression ratio adjustments that are resulted from randomly generated agent's actions.
5.	Generating pairs of compressed audio signals with compression ratios in step 4
6.	Asking the user to label each pair and add audio pairs and their labels to a buffer
7.	Training the preference (reward) predictor using the buffer
8.	Training the RL policy, based on the observation received from the environment with the reward from the trained reward predictor.
9.	**For** *M* iterations **do**:
10.	Training the policy in the environment for N_{steps} with the reward from the reward predictor
11.	Selecting compressed audio pairs resulted from given actions
12.	Asking the user to declare preferences and adding preferences to a buffer
13.	Fine-tuning the reward model for *k* batches from the above buffer
14.	**End for**
15.	Hearing assessment to compare personalized compression ratios with those specified by DSL-v5 prescriptive compression

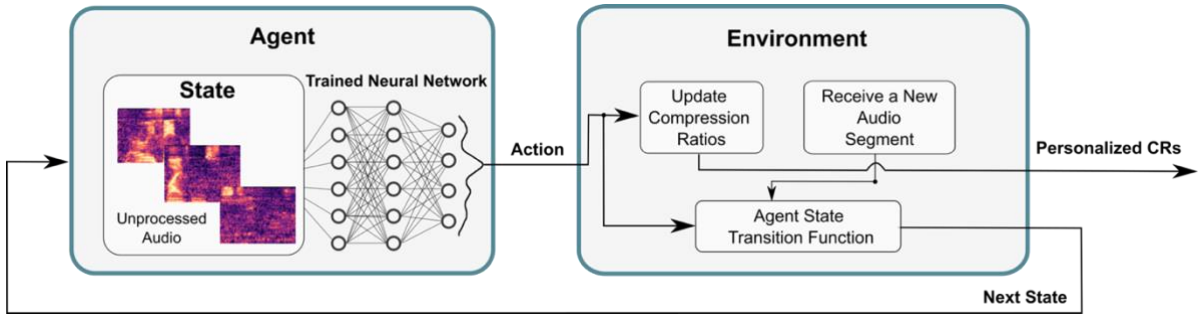
7.3.2 Environment

In the personalized framework, the environment consists of three components: audio segment creation, compression ratio update, and agent state transition function. At each policy time step, a

noisy speech audio signal is down-sampled from 48 kHz to 16 kHz to lower the computational burden.



(a) Training mode



(b) Operation mode

Fig. 7.3. Developed personalized compression DRL framework: (a) training mode and (b) operation mode.

Noisy speech audio signals are generated by distorting the widely used public domain IEEE speech dataset [37] with babble restaurant noise (provided on YouTube) and SNR of about 0 dB. The

IEEE dataset consists of 3600 speech audio files by 20 speakers (10 females and 10 males) in which each file is about 2 seconds long. The speakers are from two American English regions of the Pacific Northwest (PN) and the Northern Cities (NC) reading the IEEE "Harvard" sentences. A total of 3600 noisy speech audio signals are thus generated and at each time step, a randomly selected audio signal is used for preference training. Once a new action a_{t+1} is received from the agent, CR (compression ratio) in the frequency bands are updated based on the action a_{t+1}. Depending on the number of scales (β) specified for adjusting CR in each of the frequency bands (CR_{adj}), a set of actions is created by permutations and the action space A is given by

$$A = \prod_{i}^{nBands} \beta_i \tag{1}$$

where each β_i denotes the number of actions corresponding to *i*th frequency band. In other words, the action space *A* is the product space of the action space in all the frequency bands. Consequently, adjusting CR in each frequency band is achieved as follows:

$$CR_{new}(f) = CR_{DSL-v5}(f) \,.\, CR_{adj}(f) \tag{2}$$

where $CR_{adj}(f)$, $CR_{DSL-v5}(f)$, and $CR_{new}(f)$, respectively, stand for the compression ratio adjustment, the compression ratio computed from the DSL-v5 prescription, and the new compression ratio in the f^{th} frequency band. Permutations are defined by a dictionary in which each action is mapped to a set of compression ratio adjustments across all the frequency bands. In our experimentations, to keep the subject training time under two hours, β is set to 2 in each frequency band for an action space of 32. It should be noted that the introduced methodology is general purpose in the sense that it is applicable to higher β values. When using higher β values (β >2), more feedbacks from a subject are needed resulting in higher subject training time.

In *agent state transition function*, a new audio signal is compressed by the updated CRs from the previous iteration using the dynamic range compression whose details are described in our previous publication [36]. Due to a better match to the human hearing perception, Mel-scale frequencies are often used instead of linear scale frequencies to represent noisy speech signals in classification tasks [38]. Similarly, compressed noisy speech signals are sampled at 16kHz and framed to 20ms by using a Hanning window with 50% overlap. This translates into a frame size of 0.020×16000=320 samples. The short time Fourier transform (STFT) of frames are then computed. The conversion of frequency f in a STFT-frame to *M*th Mel-scale frequency is done as follows [39]:

$$M(f) = 2595\, log_{10}\left(1 + \frac{f}{700}\right) \tag{3}$$

Log Mel-spectrogram features are extracted from each STFT frame using a bank of 80 Mel filters. The log Mel-spectrogram features of 240 consecutive frames from an audio segment are stacked to create a 2D feature matrix (80×240). This 2D feature matrix is reshaped to a 3D feature space creating three adjacent images (80×80×3), which is considered to be one observation for agent training. For training the reward estimator, the 2D format of the observation (80×240) is considered to be the input/observation to the network.

Note that in contrast to the conventional RL, here the end-of-episode sign from the environment is not shared with the agent to make the agent training one uninterrupted episode. Moreover, reward values are received from the reward predictor rather than from the environment.

Next, unprocessed audio signals and updated CRs are added to a buffer called *audio segment queue* as depicted in Fig. 7.3(a). The audio segment queue σ shown in this figure denotes a set or

collection of audio signals *U* and compression ratios *CR* computed from a number of *k* actions, that is

$$\sigma = \big((u_0, cr_0), (u_1, cr_1), \dots, (u_{k-1}, cr_{k-1})\big)\epsilon(U, CR)^k \tag{4}$$

7.3.3 Human Preference Interface

In order to use human input in the learning loop, a hearing preference interface is created to collect the user's hearing feedbacks from a group of comparisons of audio signal pairs that are compressed with two different sets of compression ratios. The goal of this user interaction is to learn the non-linearities associated with the user's preferences or reward function.

In hearing preference interface shown in Fig. 7.3, two pairs from the queue σ are selected at each time step. Then, a corresponding pair of compressed audio signals (c^1, c^2) is computed which is used for the comparison. The user is given 4 options to indicate his/her preference: (1) $\mu = [1,0]$ if c^1 is preferable, (2) $\mu = [0,1]$ if c^2 is preferable, (3) $\mu = [0.5,0.5]$ if both compressed audio signals are equally preferred, and (4) neither compressed audio signals are desired. Hearing preferences are collected over a series of compressed audio signal pairs and are stored in a dataset D of triplets (c^1, c^2, μ), where μ denotes the feedback label, and c^1 and c^2 are the two compressed audio signals created by applying two different sets of CRs to the same noisy speech signal. Note that for option (4), the comparison is excluded from the dataset D.

In reward predictor state transition function, a batch of data from D is used to train the reward predictor model to improve the agent policy. Similar to the agent's state transition function, each

compressed audio signal is framed into 20ms frames with 50% overlap. Log Mel-spectrogram features of each frame in an audio segment are computed and considered to be one observation.

Data augmentation - The performance of the human hearing preference estimator depends on both the number of feedbacks acquired from the user and the model structure. A data augmentation is thus performed to address the limited size of training data that is available to the reward estimator in practice. For this reason, first the data size is doubled by performing data flipping. This data augmentation consists of creating realistic samples by substituting features of audio signal 1 (c^1) with features of audio signal 2 (c^2). Their corresponding preference label is switched accordingly. The goal of the data augmentation here is to enhance the generalization capability of the preference (reward) predictor.

In addition to increasing the size of the training data, it is made sure that the training data does not suffer from unbalanced labels. Unbalanced labels can cause the model not to learn the learning preferences due to: (1) the network model not getting optimized for the unbalanced label in the original dataset, and (2) the accuracy of a validation or test set drops as it is challenging to have a complete representation with few observations. To resolve imbalanced labels, an undersampling is done by reducing the number of samples in the class with more labels to match the number of samples in the class with fewer labels.

7.3.4 Preference / Reward Predictor

In the reward predictor block shown in the Fig. 7.3, the parameters reflecting the reward are obtained via a combination of a convolutional neural network and a bidirectional long short-term memory (CNN-BiLSTM) in a supervised manner. The convolutional neural network model has

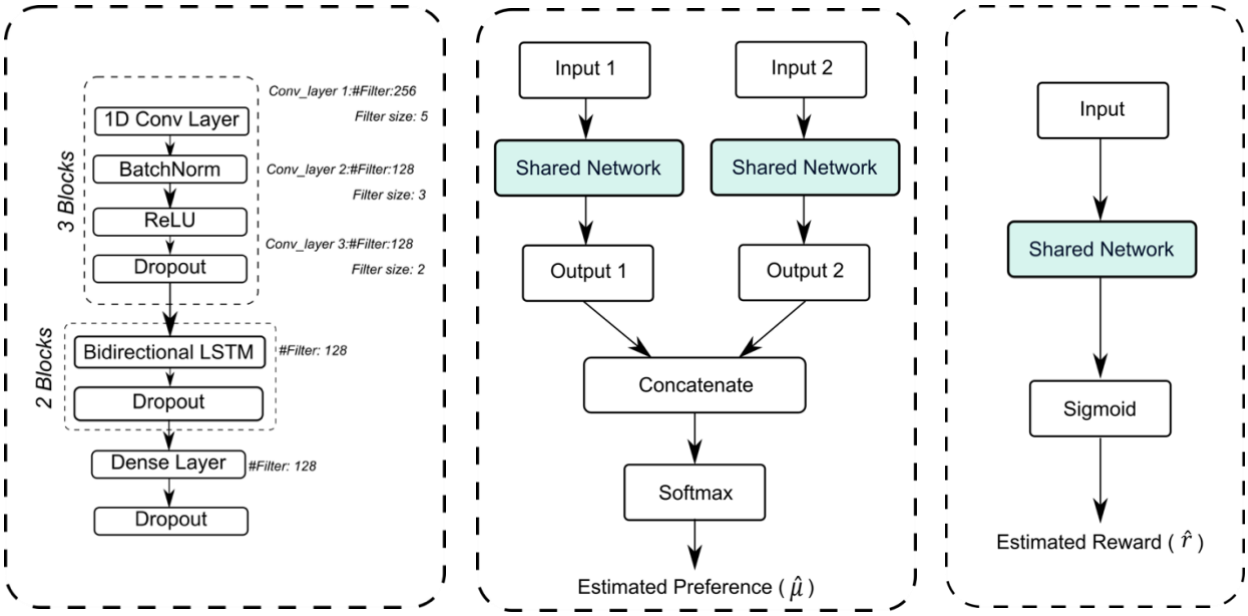
proven effective in many applications. LSTM (long short-term memory) is a recurrent neural network architecture that has been adopted for time series forecasting. Convolutional layers on top of LSTM layers are added to capture local temporal changes. Bidirectional LSTM (BLSTM) processes inputs in two ways, once from past to future and once from future to past. Hence, it preserves information from both past and future. That is why a CNN-BLSTM model is used here to learn hearing preferences of a specific user.

Log Mel-spectrogram features of compressed audio pairs constitute the two inputs of the network and user feedbacks constitute the output of the network. The reward or hearing preference predictor provides a reward prediction $\hat{r}$ and produces the probability associated with preferring a compressed audio signal c^1 over another compressed audio signal c^2. For the prediction $\hat{r}$, the following cross-entropy loss function between the predicted reward and the actual user feedback is minimized:

$$loss(\hat{r}) = -\sum_{(c^1,c^2,\mu)\epsilon D}\big(\mu(1)\log\hat{P}[c^1 > c^2] + \ \mu(2)\log\hat{P}[c^1 < c^2]\ \big) \qquad (5)$$

Learning preferences and predicting reward from comparison pairs poses an implementation difficulty as a comparison pair does not provide a numeric feedback. To estimate the agent reward, one needs to estimate it from an intermediate model or network. The network structure of the developed hearing preference predictor is depicted in Fig. 7.4. Fig. 7.4(b) shows the overall structure of the network used for training the reward predictor based on the dataset of pair comparisons. Batch normalization [40] is applied to the convolutional layers using a decay rate of 0.90 together with a dropout with alpha = 0.5. The main purpose of the dropout is to prevent the network from overfitting. The dropout decorrelates the weights of the hidden layers by randomly setting some hidden units to zeros at each training update step.

During the training phase, the model is trained on a batch size of 64, and optimized using the Adam algorithm [41]. Furthermore, during the training phase, early stopping and adaptive learning rate are applied to further avoid overfitting. Once the reward predictor is trained, the intermediate model or shared network as depicted in Fig. 7.4 (c) is used for agent training. Due to the fact that DRL is sensitive to the reward scale, a sigmoid layer is added at the end of the shared reward predictor model to bring the predicted reward between 0 and 1, see Fig. 7.4 (c).



(a) Structure of shared network (b) Training mode and its network structure (c) Operation mode

Fig. 7.4. Network structure of the reward predictor.

7.3.5 RL Agent

The training for a RL policy π is carried out based on the Bellman equation [42] in which at each time step *t*, the RL policy provides an action a_t for a given state s_t as expressed below

$$\pi: S \rightarrow A \tag{6}$$

$$a_t = \pi(s_t)$$

Action a_t influences the future state of the agent. The success of RL in learning the policy is reflected in the reward and the goal of RL is to maximize the overall reward. The parameters of the policy can get updated based on deep Q-learning [42], which is shown to be an effective RL training for personalization purposes [43], to maximize the overall estimated reward $\hat{r}$. Here, Q-value in Q-learning is optimized by a convolutional neural network. Q-value at a time step *j* is computed as follows:

$$y_j \leftarrow r(s_j, a_j) + \gamma max_{a'} Q_\phi(s'_j, a'_j) \tag{7}$$

where $r(s, a)$ denotes the reward of a state and an action, and γ is a discount factor. A Q-value is basically a prediction of the future reward which allows selecting a next action for a given state. To convert the output values of the CNN into action probabilities, the so-called one-hot representation is utilized. As noted below, the action with the highest probability is then selected

$$a_j = \arg max_a Q_\phi(s_j, a_j) \tag{8}$$

The loss function in this CNN-based Q-learning is the mean-square-error (MSE) and the optimization is done by using the Adam algorithm, resulting in the CNN weights to get updated as follows:

$$\phi \leftarrow \arg min_\phi \left\| Q_\phi(s_j, a_j) - y_j \right\|^2 \tag{9}$$

It is important to note that in contrast to supervised learning in which targets are fixed before training, here targets of the CNN-based agent depend on the network's weights that get updated gradually.

Before starting to train the RL agent and in order to reduce the chance of training a bad policy based on an untrained reward predictor, the reward predictor is trained with the dataset *D* of user

preferences (mentioned earlier in subsection D). This means that the training of the reward predictor is performed asynchronously with respect to the DRL agent. For example, 200 comparison pairs can be conducted by the user at the beginning of the DRL training. Then, querying of the user feedback can be done every *M* time steps (see Table 7.1).

For the CNN $Q_\phi(s_j, a_j)$ model, a similar configuration used in the Atari experiment in [35] is utilized here. In this work, 80×80× 3 stacked log Mel-spectrogram images of an unprocessed audio segment are used as the input to the policy. The policy model consists of 3 convolutional layers having 32, 64, and 128 filters, respectively, with rectified linear unit (ReLU) activation and alpha = 0.01. Then, the flattened output of the last convolution layer is concatenated with the compression ratio adjustments (CR_{adj}) of the previous time step.

Table 7.2. Parameters associated with agent training

Parameter	Value	Description
NEpisode	300	Number of episodes for training
NSteps	20	Number of steps in an episode
Training frequency	20	Number of steps to train agent
Batch size	50	Number of training observations used in one iteration
γ	0.99	Discount factor in updating Q-learning
No-op	30	Number of time steps before starting to train the agent

This is followed by two fully-connected layers of size 256 with ReLU activation, and a fully-connected layer with a size equal to the action space size. A fraction of the dataset is used as validation data to avoid overfitting. The agent is trained for 300 episodes, each containing 20 agent

time steps. The trained reward model is fixed during the agent training. The value and description of parameters associated with the agent training are summarized in Table 7.2.

In this approach, non-numerical feedback rather than absolute feedback is obtained from a user. The goal is to learn a policy that is most consistent with the user's preferences. As a result, personalization emulates the intention of the user and finds a policy that is ideally consistent with it.

The above learning can be viewed as an active learning approach for achieving personalized compression. This approach has the advantage of being able to get trained in an online manner, thus allowing its utilization in the field or in real-world audio environments. The described approach constitutes the first attempt at personalizing compression via DRL by placing a user in the process of learning hearing preferences as reward. The key attribute of the developed personalization compression is that it is capable of modeling the non-linearities of a user's hearing preferences and dealing with noisy preference feedbacks, and thus improving over time by fine-tuning the model based on more preferences and new encountered audio environments.

7.4 Experimental Results

Two sets of experiments were conducted to examine the performance of the developed personalized DRL compression. The first set included simulations of the HITL deep reinforcement learning. In the second set of experiments, five adult human participants with bilateral, mild to moderate hearing loss were tested. All human subject testing was performed under an approved IRB (Institutional Review Board) protocol at the University of Texas at Dallas. In the two subsections that follow, these two sets of experiments are described. The purpose of the simulation experiments was to show the capability of the developed personalized approach to learn hearing

preferences towards generating compression ratios that best matched a specific setting or user. In addition, the results of the personalized compression for five participants with hearing loss are reported.

7.4.1 Simulation Experiments

In the first set of experiments, hearing preference scenarios were simulated, and the outcomes were examined to see the learning capability of the developed personalized DRL compression. The simulated experiments refer to simulating users with different hearing preferences to evaluate the effect of β value, user's feedback, and error in the training of the reward predictor and agent. In order to analyze the training performance of the DRL compression, simulations allow more control over preferences. Hence before testing the framework on subjects, five different hearing preference scenarios were simulated by using sets of if-then conditions. As described earlier, when the size of the action space is grown by increasing the number of frequency bands or the number of scales β, the personalization consequently demands more iterations, which poses difficulties for its real-world deployment. For simulated hearing aid users 1, 2, and 3, adjustments in all five frequency bands were considered and for simulated hearing aid users 4 and 5 only in two and three frequency bands (first, third, and fifth bands), were considered, respectively. To visualize the permutation and possible compression ratio (CR) adjustments for simulated users, a mosaic representation also known as Marimekko diagram is displayed in Fig.7.5. Each column represents one possible compression ratio adjustment in the action space A.

For all simulated hearing aid users, the same audiogram or the same prescription gains from [36] were used. Then, the DSL-v5 prescriptive gains expressed in nine frequency bands were mapped

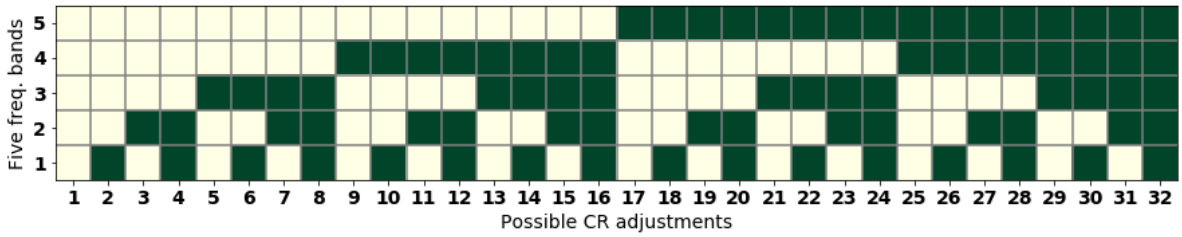
to five bands to reduce the computational complexity. The developed DRL methodology is general purpose in the sense that it can be applied to any number of bands. The number of bands that are commonly used are five, seven, and nine. Naturally, more training time with a subject in the loop would be needed as the number of bands is increased since the action space becomes larger.

Five frequency bands used were: [0-500] Hz, [500-1000] Hz, [1.0-2.0] kHz, [2.0-4.0] kHz, and [4.0-6.0] kHz. The compression ratios (gain changes) were computed from the gains. The attack time (time it takes to respond to higher sound levels) and the release time (time it takes to respond to lower sound levels) were set to the typical values of 0.01s and 1.0s, respectively. Basically, the attack and release time regulate the reaction pace of compression. Moderate and loud compression thresholds were also set to 60dB and 80dB to be consistent with the prescriptive hearing aid compression fittings.

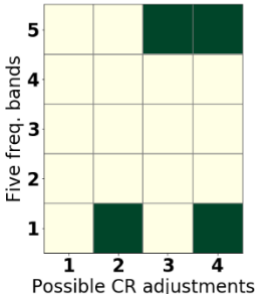
At each training time step, based on the agent's input or state, the agent outputs one of the CR adjustments as an action in A to the environment. Each action corresponds to compression setting adjustments in the five frequency bands. Action space A of simulated users is shown in Fig. 7.5 via a mosaic representation for two possible CR adjustments, $\beta = 2$ ($CR_{adj} = 1 \; or \; 4$). The column in this figure shows all possible combinations of CR adjustments across the five frequency bands. As a result, the action space of the first three simulated users became as depicted in Fig. 7.5(a), exhibiting 32 possible compression ratio adjustments across all five frequency bands ($A = \prod_i^5 2 = 32$). Fig. 7.5(b) depicts the action space of a simpler case by changing the compression ratios in only the first and the fifth frequency bands, exhibiting 4 possible CR adjustments. Likewise, the action space for the case of adjustments in only three frequency bands (first, third, and fifth) is depicted in Fig. 7.5(c). In Fig. 7.5, light color indicates $CR_{adj} = 1$ (no adjustment) and dark color

indicates $CR_{adj} = 4$ (users often prefer higher compression ratio in a noisy environment). As an example, in Fig. 7.5(a), the second column in Fig. 7.5(a) mosaic plot exhibits the CR adjustment settings as [4,1,1,1,1], and for the 20th column, the CR adjustment settings as [4,4,1,1,4]. In other words, based on the agent's input, the agent can give one of the 32 actions in A, each one corresponding to a compression setting adjustments in five frequency bands. As can be seen from Fig. 7.5, the action space of the agent is influenced by two factors: number of frequency bands and number of adjustments in each frequency band.

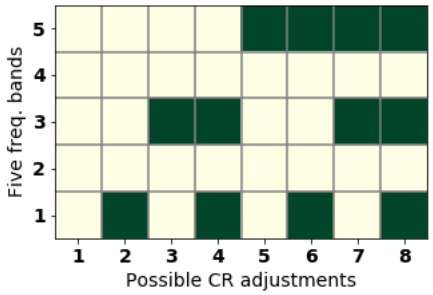
Overall, 200 hearing preferences over audio pairs were considered for each simulated user. As illustrated in Fig. 7.6, when the action space became larger, more user feedbacks were required to model user's hearing preferences. Preference space is displayed in Fig. 7.6 not only show the complexity of hearing feedbacks in larger action spaces, but also differences in preferences between the users 1, 2, and 3. To make the simulation close to reality, the following conditions for the simulated users were considered (users 1 to 3 are shown in Figs. 6a to 6c, respectively): (1) receiving noisy feedback from user, (2) receiving inconsistent and highly noisy feedback from user, (3) receiving neutral preferences across various compression settings from user. The simulated user 2 is an example of the situation when an actual user does not give proper feedback preferences when listening to audio pairs of two sets of compression ratios. This led to failure in learning preferences during the reward predictor training which consequently led to failure in the agent learning or learning the best settings for that user. The simulated user 3 is an example of the situation when an actual user is very strict about some settings and has neutral preferences over the other settings. This led to having more neutral preferences and therefore the training dataset became highly unbalanced.



(a) Adjustments in all five frequency bands that is mapped to 32 actions



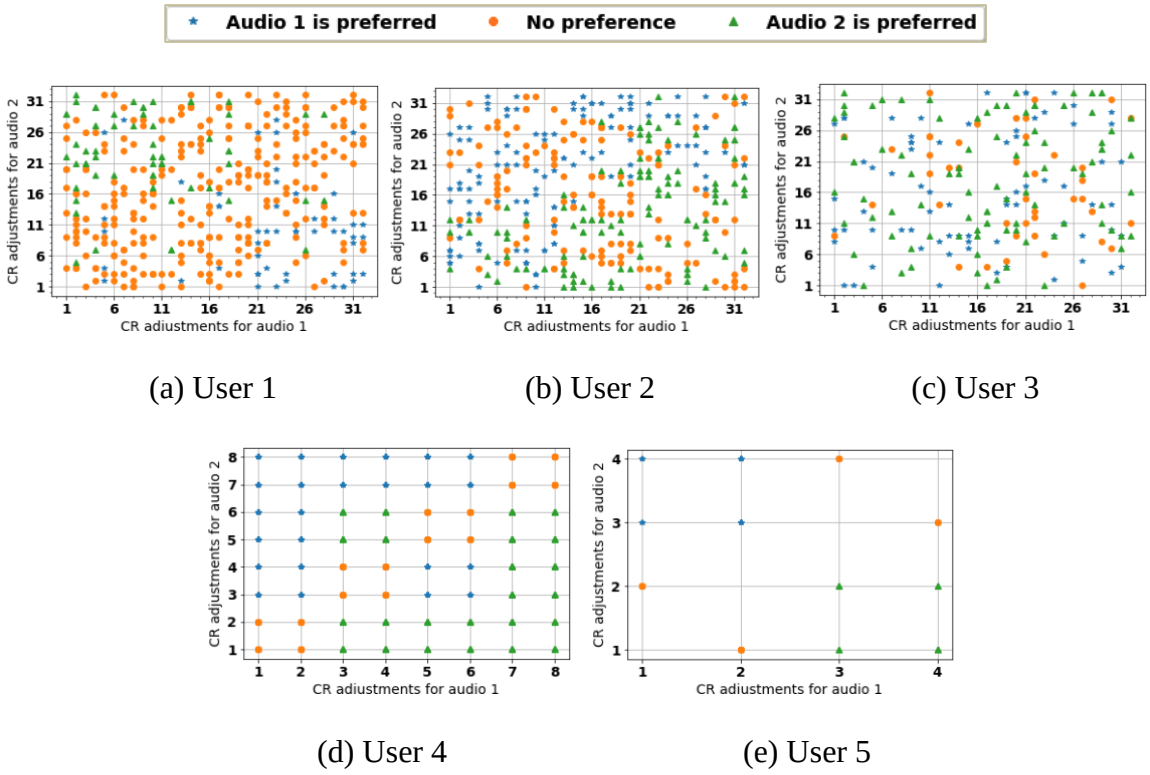
(a) Adjustments only in the first and the fifth frequency bands



(c) Adjustments only in the first, the third, and the fifth frequency bands

Fig. 7.5. Mosaic representation of action space corresponding to simulated (a) users 1, 2, and 3, (b) simulated user 4, and (c) simulated user 5, with β = 2 ($CR_{adj} = 1\ or\ 4$) and compression in five frequency bands. Light color indicates $CR_{adj} = 1$ and dark color indicates $CR_{adj} = 4$.

The learning loss value of hearing preferences with respect to training epochs in different scenarios is displayed in Fig 7.7. By comparing Fig. 7(a) with Fig. 7.7(d), it can be seen that the learning process in the reward predictor training became more challenging as the action space became larger. As can be seen from Fig. 7.7, simulated users 2 and 3 have worse validation loss in training the reward predictor. Failure in preference learning for simulated user 2 is due to inconsistent feedbacks from the user. For simulated user 3, due to having highly imbalanced data (more neutral preferences), failure occurs in the training of the reward predictor.



(a) User 1 (b) User 2 (c) User 3

(d) User 4 (e) User 5

Fig. 7.6. Preference space collected from simulated (a) user 1, (b) user 2, (c) user 3, (d) user 4, and (e) from simulated user 5.

The mean of the normalized reward and the mean of the Q-values (target outputs) across the agent training episodes for each simulated user are displayed in Figs. 7.8 and 7.9, respectively. From these figures, it can be seen that both the mean reward and the mean Q-value exhibited an increasing trend, indicating that the personalized compression was gradually learning the policy that was ideally consistent with the users' hearing preferences. As mentioned earlier, the success of RL is heavily dependent on the performance of the reward predictor. That is why, although an increasing trend for user 2 is exhibited in Fig. 7.8(b) and Fig. 7.9(b), the personalization was not effective due to the poor training of the reward predictor.

7.4.2 Subject Testing Experiments

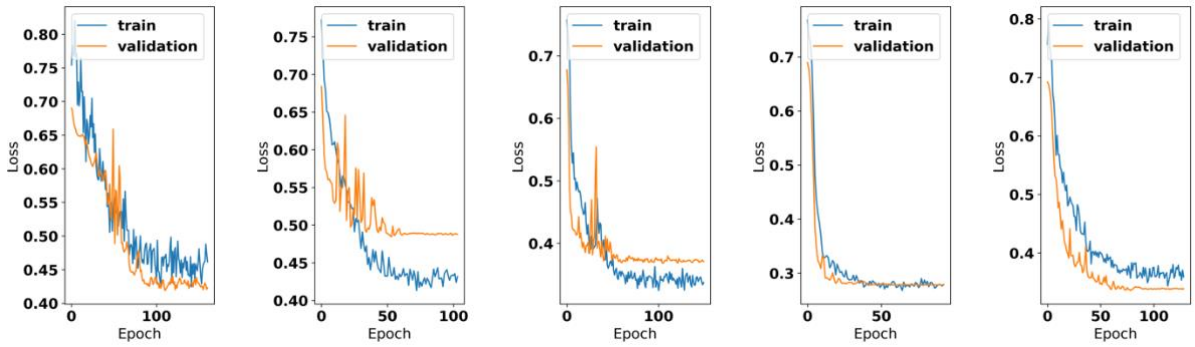
In addition to the above simulations, actual human subject testing was performed according to the IRB protocol described earlier with one modification. Due to the Covid-19 pandemic, the original IRB was modified to allow participant testing to be conducted online instead of in a soundbooth. For the subject testing experiments, "virtual" visits were conducted using a video conference utility. Secure links were emailed to the participants to access online experimental sessions.

For subject testing, a crowdsourcing approach similar to the P.808 standard [44] was considered. It was ensured that the subjects listened to different audio pairs many times in a random order for a trustworthy comparison between the DSL-v5 settings and the personalized DRL settings. Eligibility conditions of the participants in our approved IRB included: (i) range of hearing loss of participants being mild to moderate, (ii) participants being native English speakers or presenting a native-level fluency of English, (iii) participants having symmetric hearing loss, and (iv) age range of participants being in the range 18-80 years old.

Initially, the participants obtained their audiograms using the web-based hearing test at https://hearingtest.online/. As per Table 7.1, for training the developed deep reinforcement learning personalized compression, 210 (7 sessions of 30, with breaks in between) pairs of sound files consisting of the spoken sentences, discussed earlier, in noisy (babble) background were played at SNR of 0 dB.

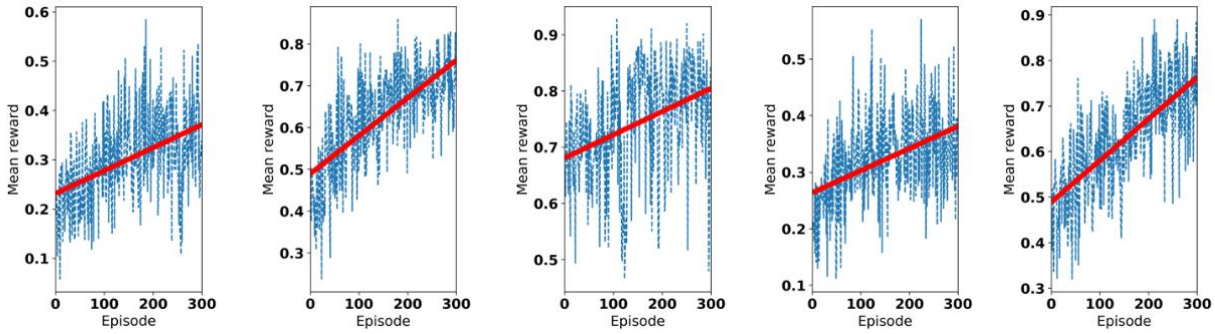
The participants were asked to indicate which sound file or clip they preferred or whether both sound clips sounded the same to them. It is worth mentioning here that increasing the number of sound files naturally improves the training and 210 audio sound files may not be adequate to cover all possible combinations of compression settings. For example, for $\beta = 2$, the number of possible

actions is $2^5 = 32$, demanding $\binom{32}{2} = 496$ pairs of sound files. However, to avoid human fatigue, the test sessions were limited to 2 hours. This time frame constrained the number of audio sound files to 210 over 7 sessions. The data augmentation mentioned earlier was then applied to the collected dataset D of triplets (c^1, c^2, μ). The reward predictor was trained based on the augmented data to learn a participant's hearing preferences.



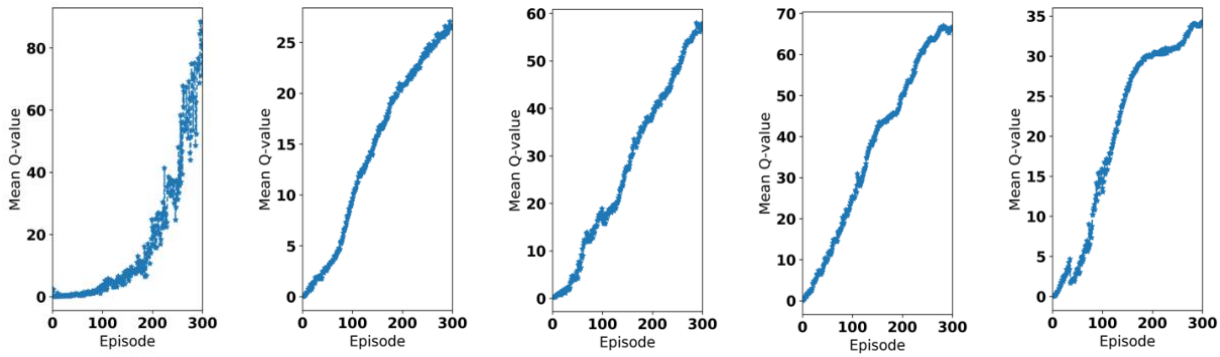
(a)User 1 (b) User 2 (c) User 3 (d) User 4 (e) User 5

Fig. 7.7. Cross-entropy loss value in training reward predictor for simulated (a) user 1, (b) user 2, (c) user 3, (d) user 4, and for (e) user 5



(a)User 1 (b) User 2 (c) User 3 (d) User 4 (e) User 5

Fig. 7.8. Mean of normalized reward per episode and its trend (in solid red line) in the agent training for simulated (a) user 1, (b) user 2, (c) user 3, (d) user 4, and for (e) simulated user 5.



(a)User 1 (b) User 2 (c) User 3 (d) User 4 (e) User 5

Fig. 7.9. Mean Q-value per episode in the agent training for simulated (a) user 1, (b) user 2, (c) user 3, (d) user 4, and for (e) simulated user 5.

After training the policy, a comparison test was conducted between the personalized compression and the DSL-v5 reference prescriptive compression by playing 60 randomly selected sentences across different talkers in a noisy (babble) background at the same SNR level of 0dB. To remove any bias associated with the timeline of the training and testing phases, a gap of at least one-week was placed between these phases. Note that the training is carried out offline and once the offline training is completed, the actual operation/testing of the trained DRL compression only takes 71ms for a 2.5 seconds noisy speech sentence using a 2.9 GHz dual-core i5 processor computer. Thus, for all practical purposes, the developed personalized DRL compression runs in real-time during operation or testing. The "preference metric" used here is an integrated metric to enable personalization as it incorporates speech quality, speech intelligibility or word error rate, and audio comfort at the same time in a collective manner. In other words, when users judge audio pairs, they consider all these metrics together. For example, one user may prefer or prioritize speech intelligibility over speech quality and one user may prioritize speech quality over speech intelligibility.

Table 7.3 provides the compression ratios of DSL-v5 versus the compression ratios of the developed personalized approach for five participants with mild to moderate bilateral hearing loss who took part in this study. The outcomes of the participant testing experiments in terms of preference percentages are shown as a bar chart in Fig. 7.10. In this figure, the "personalized settings preferred" implies that in the audio pair assessment indicated in step 15 of Table 7.1, the subject preferred the audio that was compressed by personalized DRL compression settings. Likewise, the "DSL-v5 settings preferred" refers to when the subject preferred the audio compressed by the baseline or reference prescriptive DSL-v5. Similar preference denotes that the subject had equal preference over the audio compressed with the personalized DRL compression settings, and the audio compressed by the reference prescriptive DSL-v5 compression settings.

As can be seen from this figure, on average, personalized settings were clearly preferred by the participants over the DSL-v5 settings across different talkers and sentences heard. In other words, the number of times the personalized settings were preferred by the participants were nearly 7 times greater than the number of times the DSL-v5 settings were preferred. These results indicate that the developed personalized or individualized compression indeed is more effective than a one-size-fits-all DSL-v5 prescriptive compression approach. Audio samples of the subject testing experiments can be heard at this link: www.utdallas.edu/~kehtar/DRLcompression.html.

7.5 Conclusion and Future Work

In this paper, an active human-in-the-loop DRL-based personalized hearing aid fitting approach is developed to improve the currently practiced one-size-fits-all hearing aid fitting. The current fitting practice involves setting compression gains based on gain averages of a group of users which are not necessarily optimal for a specific user. The developed approach personalizes compression

settings via a deep reinforcement learning framework. This is the first time human-in-the-loop DRL has been used to achieve improved hearing aid compression. Both simulation and experimental results show the effectiveness of the developed personalization approach in achieving preferred hearing outcomes.

Table 7.3. subject testing experiments: DSL-v5 vs. personalized compression ratios.

Subject	**Level of hearing loss**	**Audiogram in freq. bands [0.5, 1.0, 2.0, 4.0, 6.0] kHz**	**DSL-v5 gains for soft speech**	**DSL-v5 compression ratios**	**Personalized compression ratios**
1	Mild	[15,20,20,30,30]	[7,8,14,17,15]	[1.1,1.2,1.3,1.2,1.3]	[1.1,1.2,1.3,**4.8**,**5.2**]
2	Mild	[15,15,20,20,30]	[5,6,14,15,15]	[1.1,1.2,1.3,1.2,1.2]	[**4.4**,1.2,**5.2**,1.2,**4.8**]
3	Moderate	[20,20,40,50,60]	[11,12,24,29,34]	[1.1,1.2,1.3,1.2,1.4]	[**4.4**,1.2,1.3,**4.8**,**5.6**]
4	Moderate	[25,20,20,40,30]	[13,11,14,22,15]	[1.1,1.3,1.3,1.3,1.3]	[**4.4**,1.3,1.3,**5.2**,1.3]
5	Moderate	[20,20,30,40,40]	[6,11,20,23,20]	[1.1,1.2,1.3,1.2,1.4]	[1.1,1.2,1.3,**4.8**,**5.6**]

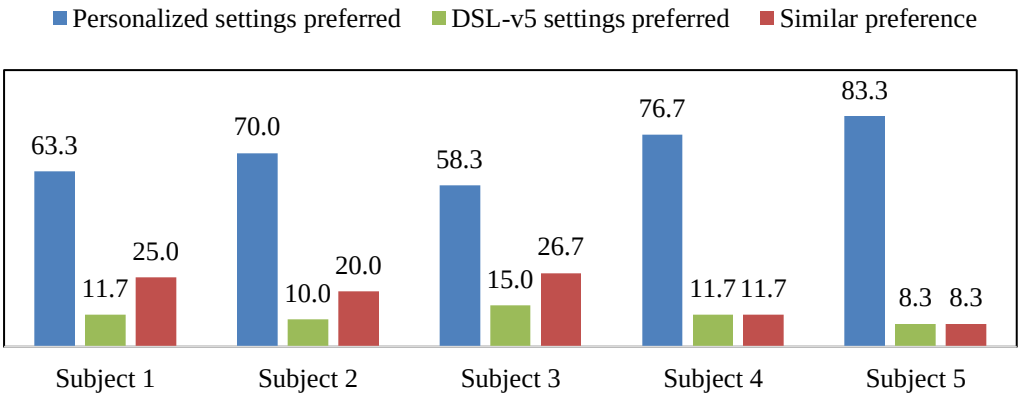


Fig. 7.10. Outcome of subject testing experiments in percentages: comparison of hearing preference between personalized compression and DSL-v5 compression.

The overall goal of this paper was to leverage simulation with limited clinical subject testing, to show the proof-of-concept of our novel personalized compression using HITL DRL approach. In future studies, the deployment and efficacy of our approach can be further assessed by carrying out extensive clinical testing. This would require examining a large number of subjects in

controlled audio environments and in the field. It would also be useful to examine several noise types, SNRs, numbers of frequency bands, and alternative metrics other than the preference metric used in this work. One key advantage of the introduced personalization approach is that it is general purpose in the sense that all of these parameters are already built into its training/testing and additional parameters can be added to explore an even larger variable space.

7.6 Acknowledgment

We wish to express our appreciation to Ms. Tina Campbell and Dr. Celia Escabi for their help in putting together the IRB materials and their assistance with the subject testing. We also thank Mr. Ali Salman for his help with the codes written for this project.

7.7 References

[1] D. Giannoulis, M. Massberg, and J. D. Reiss. "Digital dynamic range compressor design—A tutorial and analysis," Journal of the Audio Engineering Society, vol. 60, no. 6, pp. 399-408, 2012.

[2] "NAL-NL2," Jul. 28, 2014. [Online]. Available: http://www.nal. gov.au/nal-software_tab_nal-nl-2.shtml, National Acoustic Laboratories.

[3] Western University, DSL-v5 by Hand, 2014. [Online]. Available: https://www.dslio.com/wpcontent/uploads/2014/06/DSL-5-by-Hand.pdf.

[4] S. Kochkin, D. L. Beck, L. A. Christensen, C. Compton-Conley, B. J. Fligor, P. B. Kricos, J. B. Mcspaden, H. G. Mueller, M. J. Nilsson, J. L. Northern, T. A. Powers, R. W. Sweetow, B. Taylor, and R. G. Turner. MarkeTrak VIII, "The Impact of the Hearing Healthcare Professional on the Hearing Aid User Success," The Hearing Review, vol. 17, no. 4, pp. 12-34, 2010.

[5] G. Keidser, and H. Dillon. "What's new in prescriptive fittings down under," Hearing care for adults, pp. 133-142, 2006.

[6] G. Keidser, and K. Alamudi, "Real-Life Efficacy and Reliability of Training a Hearing Aid", Ear and Hearing, vol. 34, no. 5, pp. 619-629, 2013.

[7] K. Smeds, “Is Normal or Less Than Normal Overall Loudness Preferred by First-Time Hearing Aid Users?”, Ear and Hearing, vol. 25, no. 2, pp.159-172, 2004.

[8] L. LN Wong, “Evidence on Self-Fitting Hearing Aids”, Trends in Amplification, vol. 15, no. 4, pp. 215-225, 2011.

[9] B. Johansen, M. Petersen, M. Korzepa, J. Larsen, N. Pontoppidan, and J. Larsen, "Personalizing the fitting of hearing aids by learning contextual preferences from internet of things data," Computers, vol. 7, no. 1, 2018.

[10] H. Dillon, J.A. Zakis, H McDermott, G. Keidser, W. Dreschler, and E. Convery, “The trainable hearing aid: What will it do for clients and clinicians?,” The Hearing Journal, vol. 59, no. 4, 2006.

[11] G. Aldaz, S. Puria, and L. J. Leifer. "Smartphone-based system for learning and inferring hearing aid settings." Journal of the American Academy of Audiology, vol. 27, no. 9, pp. 732-749, 2016.

[12] A. Pasta, M. Petersen, K. Jensen, and J. Larsen, "Rethinking hearing aids as recommender systems," Proceedings of HealthRecSys, pp. 11-17, 2019.

[13] D. Cuda, A. Murri, A. Mainardi, and J. Chalupper, "Effectiveness and efficiency of a dedicated bimodal fitting formula." Audiology research, vol. 9, no. 1, 2019.

[14] L. Brody, Y. Wu, and E. Stangl, "A Comparison of Personal Sound Amplification Products and Hearing Aids in Ecologically Relevant Test Environments," American journal of audiology, vol. 27, no. 4, 2018.

[15] P. B. Nelson, T. T. Perry, M. Gregan, and D. VanTasell, "Self-adjusted amplification parameters produce large between-subject variability and preserve speech intelligibility," Trends in hearing, vol. 22, 2018.

[16] A. Boothroyd, and C. Mackersie, "A “Goldilocks” approach to hearing-aid self-fitting: User interactions," American journal of audiology, vol. 26, no. 3S, pp 430-435, 2017.

[17] G. Keidser, and E. Convery, "Outcomes with a self-fitting hearing aid," Trends in hearing, vol. 22, 2018.

[18] E. Convery, G. Keidser, L. Hickson, and C. Meyer, "Factors associated with successful setup of a self-fitting hearing aid and the need for personalized support," Ear and Hearing, vol. 40, no. 4, 794-804, 2019.

[19] C. L. Mackersie, A. Boothroyd, and H. Garudadri. "Hearing Aid Self-Adjustment: Effects of Formal Speech-Perception Test and Noise," Trends in Hearing, vol. 24, pp. 1-16, 2020.

[20] A. T. Sabin, D. Van Tasell, B. Rabinowitz, S. Dhar, "Validation of a Self-Fitting Method for Over-the-Counter Hearing Aids," Trends in Hearing, vol 24, pp. 1-19, 2020.

[21] J. B. Nielsen, J. Nielsen, B. S. Jensen, and J. Larsen, "Hearing aid personalization," In 3rd NIPS Workshop on Machine Learning and Interpretation in Neuroimaging, Lake Tahoe, NV, USA, 5-10 Dec 2013.

[22] J. B. B. Nielsen, J. Nielsen, and J. Larsen, "Perception-based personalization of hearing aids using Gaussian processes and active learning." IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 23, no. 1, pp 162-173, 2014.

[23] N. S. Jensen, O. Hau, J. B. B. Nielsen, T. B. Nielsen, and S. V. Legarth, "Perceptual Effects of Adjusting Hearing-Aid Gain by Means of a Machine-Learning Approach Based on Individual User Preference", Trends in hearing, vol. 23, p.2331216519847413, 2019.

[24] J. R. Dubno, D. D. Dirks, and D. E. Morgan. "Effects of age and mild hearing loss on speech recognition in noise." The Journal of the Acoustical Society of America, vol. 76, no. 1, pp. 87-96, 1984.

[25] M. Korzepa, M. Petersen, J. Larsen, and M. Mørup, "Simulation environment for guiding the design of contextual personalization systems in the context of hearing aids," Proceedings of 28th ACM Conference on User Modeling, Adaptation and Personalization, pp. 293-298. 2020.

[26] C. Szepesvári, "Algorithms for reinforcement learning." Synthesis lectures on artificial intelligence and machine learning, vol. 4, no. 1, pp. 1-103, 2010.

[27] S. Albawi, M. T. Tareq Abed, and S. Al-Zawi. "Understanding of a convolutional neural network," In 2017 IEEE International Conference on Engineering and Technology (ICET), 2017, pp. 1-6.

[28] M. Schuster, and K. K. Paliwal, "Bidirectional recurrent neural networks," IEEE transactions on Signal Processing, vol. 45, no. 11, pp. 2673-2681, 1997.

[29] S. Bech, and N. Zacharov, "Perceptual Audio Evaluation Theory," Method and Application. John Wiley n& Sons, Ltd, 2007. ISBN 0470869240.

[30] G. R. Lockhead, "Absolute Judgments Are Relative: A Reinterpretation of Some Psychophysical Ideas," Review of General Psychology, vol. 8, no. 4, pp. 265-272, 2004.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.

[32] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," In 2017 IEEE international conference on robotics and automation (ICRA), 2017, pp. 3389-3396.

[33] A. Y. Ng, and S. J. Russell. "Algorithms for inverse reinforcement learning," In ICML, vol. 1, 2000, pp. 663-670.

[34] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," In Advances in Neural Information Processing Systems, 2017, pp. 4299-4307.

[35] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei. "Reward learning from human preferences and demonstrations in Atari," In Advances in neural information processing systems, 2018, pp. 8011-8023.

[36] N. Alamdari, E. Lobarinas, and N. Kehtarnavaz, "An educational tool for hearing aid compression fitting via a web-based adjusted smartphone app," in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), 2019, pp. 7650-7654.

[37] D. McCloy, P. Souza, R. Wright, J. Haywood, N. Gehani, S. Rudolph, The PN/NC Corpus. Version 1.0, 2013. [Online]. Available: https://depts.washington.edu/phonlab/resources/pnnc/pnnc1/.

[38] J. Abeßer, “Review of deep learning based methods for acoustic scene classification.” Appl. Sci., 10, 2020.

[39] S. Stevens, J. Volkmann, and E. Newman, “A Scale for the measurement of the psychological magnitude pitch,” J. Acoust. Soc. Amer., vol. 8, no. 3, pp. 185–190, Jan. 1937.

[40] S. Ioffe, C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift”, In Proceedings of 32nd Int. Conf. Mach. Learn, 2015, pp 448–456.

[41] D. Kingma, J. Ba, “Adam: A method for stochastic optimization”, arXiv preprint: 1412.6980, (2014), Available: https://arxiv.org/abs/1412.6980.

[42] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529-533, 2015.

[43] S. Liu, K. See, K. Ngiam, L. Celi, X. Sun, and M. Feng, "Reinforcement Learning for Clinical Decision Support in Critical Care: Comprehensive Review," Journal of Medical Internet Research, vol. 22, no. 7, 2020.

[44] ITU-T Recommendation P.808, “Subjective evaluation of speech quality with a crowdsourcing approach,” Geneva: International Telecommunication Union, 2018.

CHAPTER 8

CONCLUSION AND POSSIBLE EXTENSIONS

The thrust of this dissertation has been on the development of personalization frameworks for the two main signal processing modules of hearing aid devices, namely noise reduction and dynamic range compression. The main contributions of this dissertation are summarized below:

1. A personalized audio noise classification has been developed as part of a personalization framework for noise reduction by conducting the classification training in an online manner without the need to carry out any offline training. This personalized approach is implemented as a smartphone app for real-time field deployment.

2. An open-source educational tool has been developed to study hearing aid compression in actual audio environments. This tool is a web-based program mimicking the prescriptive DSL-v5 hearing aid fitting gains across its frequency bands. The gains are then programmed into a smartphone app to perform real-time field testing.

3. A personalized deep learning-based noise reduction has been developed which is more suited for field deployment and adaptive to unseen audio conditions that occur in real-world environments. The novelty of this approach is that it eases the requirement of commonly used approaches that require the availability of clean ground-truth speech signals. In addition, a personalized noise reduction smartphone app running in real-time with low-latency has been developed.

4. More prominently, a personalized human-in-the-loop deep reinforcement learning-based hearing aid fitting approach has been developed to enable improved hearing as compared

to the currently practiced one-size-fits-all hearing aid fitting strategies. This is the first time a human-in-the-loop DRL has been considered to achieve hearing aid compression.

Possible extensions of this dissertation work include, but not limited to, the following:

1. Integration of personalized noise reduction and personalized compression into one unified speech processing pipeline.
2. Carrying out extensive clinical testing to further assess the efficacy of the personalization of noise classification and reduction, i.e. examining a large number of subjects in the field across different non-stationary noise environments.
3. Carrying out extensive clinical testing to further assess the efficacy of the personalization of compression, i.e. examining a large number of subjects in both controlled audio environments and in the field across different noise types, signal-to-noise ratios, numbers of frequency bands, etc.
4. Based on (2) and (3) above, performing data mining to establish more effective noise reduction and compression to what is currently being used.

BIOGRAPHICAL SKETCH

Nasim Taghizadeh Alamdari received her BS degree in Biomedical Engineering-Bioelectric from Azad University in Tehran, Iran, in 2013. She received her MS degree in Electrical Engineering from University of North Dakota, ND, in 2016. Then, she joined The University of Texas at Dallas for her doctoral studies in Electrical Engineering in 2017, working as research assistant with Dr. Nasser Kehtarnavaz. The focus of her doctoral research has been on the personalization of speech reduction and dynamic range compression for hearing devices. During 2020, she has been an intern at Google and earlier at Tencent. She has experience in developing machine learning solutions for real-time audio processing, and signal/image classification for biomedical applications.

CURRICULUM VITAE

Nasim Taghizadeh Alamdari

EDUCATION

PhD in Electrical Engineering, The University of Texas at Dallas, TX. 2017 - March 2021

MS in Electrical Engineering, University of North Dakota, ND. 2014 - 2016

BS in Biomedical Engineering-Bioelectric, Azad University, Science and Research Branch, Tehran, Iran. 2008 - 2013

PROFESSIONAL EXPERIENCES

Technical Intern, *Google* Aug. 2020 - Dec. 2020

Research Intern, *Speech at Tencent AI Lab, Tencent* May 2020 - Aug. 2020

Graduate Teaching Assistant, *The University of Texas at Dallas* Aug. 2019- May 2020
Courses: (1) machine learning, (2) signals and systems laboratory

Graduate Research Assistant, *The University of Texas at Dallas* May 2018 - Aug. 2019
Developed open-source real-time audio processing applications on smartphone for hearing enhancement (NIH-NIDCD funded).

Graduate Student Researcher, *University of North Dakota* Sep. 2015 - Dec. 2016
Implemented image processing and machine learning solutions for skin condition monitoring smartphone apps. (in collaboration with eTreat Medical Diagnostics Inc.)

Graduate Teaching Assistant, *University of North Dakota* Aug. 2014 – Aug. 2016

RESEARCH INTERESTS

Audio/Speech Processing, Speech Enhancement, Embedded Machine/Deep Learning, Real-Time Signal Processing

TECHNICAL SKILLS

Programming Languages: Most experienced with Python, MATLAB, C/C++

Familiar with Java, Objective-C, Swift

Machine Learning Software Library: Keras, TensorFlow, PyTorch

Development: Familiar with Android, iOS, HTML/JavaScript/CSS.

AWARDS & HONORS

PhD Research Small Grants Award Jan. 2019
The University of Texas at Dallas
The Office of Vice President for Research

Outstanding Contribution in Reviewing Jan. 2018
Journal of Biomedical Signal Processing and Control, Elsevier, Amsterdam, The Netherlands

Jonsson School Graduate Study Scholarship Jan. 2017
The University of Texas at Dallas

Graduate Teaching Assistant Scholarship Aug. 2014
University of North Dakota

PUBLICATIONS

1. **N.Alamdari**, A.Azarang, N.Kehtarnavaz, Improving Deep Speech Denoising by Noisy2Noisy Signal Mapping, Applied Acoustics, vol. 172, 107631, 15 Jan. 2021.
2. **N.Alamdari**, E.Lobarinas, N.Kehtarnavaz, Personalization of Hearing Aid Compression by Human-in-the-Loop Deep Reinforcement Learning, IEEE Access, vol. 8, pp. 203503-203515, 2020.
3. S.Akbarzadeh, **N.Alamdari**, T.Campbell, E.Lobarinas, N.Kehtarnavaz, Word Recognition Clinical Testing of Personalized Deep Rein- forcement Learning Compression, 14th IEEE Dallas Circuits and Systems Conference, Nov. 2020.
4. **N.Alamdari**, A.Azarang, N.Kehtarnavaz, Self-Supervised Deep Learning-Based Speech Denoising, arXiv preprint arXiv:1904, 2019.
5. **N.Alamdari**, E.Lobarinas, N.Kehtarnavaz, An Educational Tool for Compression Fitting via a Web-based Adjusted Smartphone App, In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 7650-7654. IEEE, 2019.

6. **N.Alamdari**, N. Kehtarnavaz, A Real-Time Smartphone App for Unsupervised Noise Classification in Realistic Audio Environments, In IEEE International Conference on Consumer Electronics (ICCE), pp. 1-5. IEEE, 2019.
7. **N.Alamdari**, S. Yaraganalu, N. Kehtarnavaz, A Real-Time Personalized Adaptive Noise Reduction Smartphone App for Realistic Audio Environments, In IEEE Signal Processing in Medicine and Biology Symposium (SPMB), pp. 1-5. IEEE, 2018.
8. **N.Alamdari**, F. Saki, A. Sehgal, N. Kehtarnavaz, An Unsupervised Noise Classification Smartphone App for Hearing Improvement Devices, In IEEE Signal Processing in Medicine and Biology Symposium (SPMB), pp. 1-5. IEEE, 2017.
9. V. Zakeri, A. Akhbardeh, **N.Alamdari**, R. Fazel-Rezai, M. Paukkunen, and K. Tavakolian, Analyzing Seismocardiogram Cycles to Identify the Respiratory Phases, IEEE Transactions on Biomedical Engineering 64, no. 8, 1786-1792., 2017.
10. **N.Alamdari**, N. MacKinnon, F. Vasefi, R. Fazel-Rezai, M. Alhashim, A. Akhbardeh, D. L. Farkas, and K. Tavakolian, Effect of Lesion Segmentation in Melanoma Diagnosis for a Mobile Health Application, In Frontiers in Biomedical Devices, vol. 40672, American Society of Mechanical Engineers, 2017.
11. **N.Alamdari**, K. Tavakolian, V. Zakeri, R. Fazel-Rezai, and A. Akhbardeh, A morphological approach to detect respiratory phases of seismocardiogram,In38thAnnualInternationalConferenceoftheIEEEEngineeringinMedicineandBiologySociety(EMBC),pp. 4272-4275. IEEE, 2016.
12. **N.Alamdari**, K. Tavakolian, V.Zakeri, R. Fazel-Rezai, A. Akhbardeh, Fusion of Electrocardiogram and Accelerocardiogram Derived Res- piration Methods for Estimation of Respiratory Phases, Journal of Medical Devices, vol. 10, no. 2, 2016.
13. **N.Alamdari**, A. Haider, R. Arefin, A. K. Verma, K. Tavakolian, and R. Fazel-Rezai, A review of methods and applications of brain computer interface systems, In IEEE International Conference on Electro Information Technology (EIT), pp. 0345-0350. IEEE, 2016.
14. **N.Alamdari**, K. Tavakolian, M. Alhashim, and R. Fazel-Rezai, Detection and classification of acne lesions in acne patients: A mobile application, In IEEE International Conference on Electro Information Technology (EIT), pp. 0739-0743. IEEE, 2016.
15. **N.Alamdari**, K. Tavakolian, V. Zakeri, R. Fazel-Rezai, M. Paukkunen, R. Sepponen, and A. Akhbardeh, Using electromechanical signals recorded from the body for respiratory phase detection and respiratory time estimation: A comparative study, In Computing in Cardiology Conference (CinC), pp. 65-68. IEEE, 2015.
16. **N.Alamdari**, S Akbari, E Fatemizadeh, Unsupervised Versus Supervised Methods for Categorizing Mental States from fMRI Data, Journal of Medical Devices, vol. 9, no. 2, 2015.
17. **N.Alamdari**, and E Fatemizadeh, Comparison of classification and dimensionality reduction methods used in fMRI decoding, 8th Iranian Conference on Machine Vision and Image Processing (MVIP), 2013.