A Real-Time Personalized Noise Reduction Smartphone App for Hearing Enhancement

Nasim Alamdari, Shashank Yaraganalu, Nasser Kehtarnavaz

Department of Electrical & Computer Engineering, University of Texas at Dallas, Richardson, TX, USA
{alamdari, sxy163530, kehtar}@utdallas.edu

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.

I. 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 II provides an overview of the modules that are integrated to enable a personalized noise reduction speech processing pipeline. In Section III, 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 IV, followed by the conclusion in Section V.

II. Personalized Noise Reduction Pipeline

Figure 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.

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 upsampling 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.

A. 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.

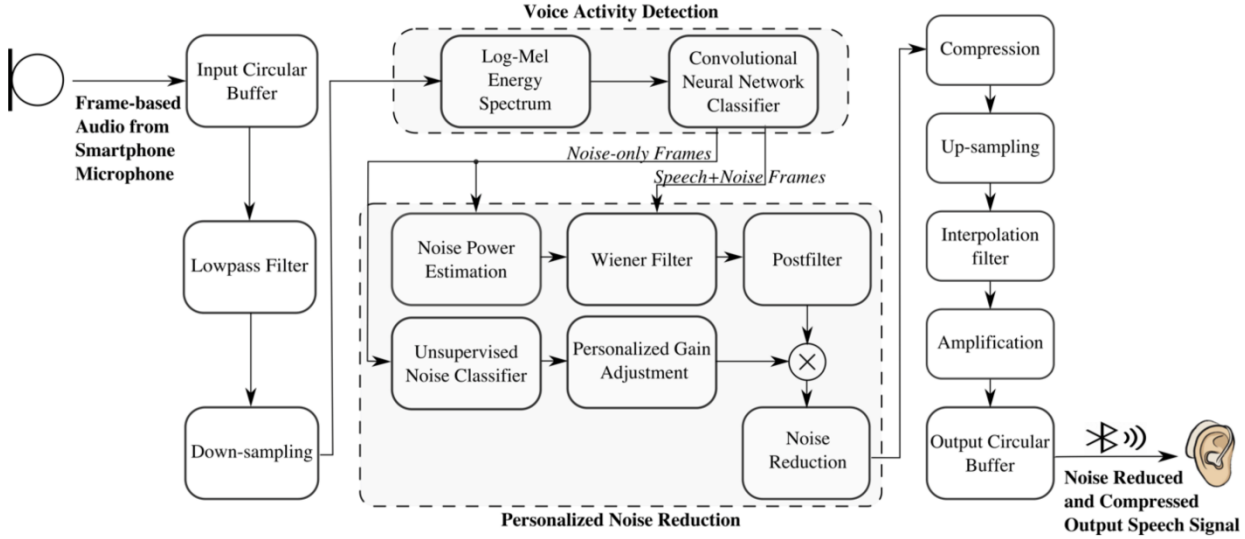


Figure 1. Block diagram of the real-time low-latency personalized noise reduction (PNR) app.

B. 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.

III. 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 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.

Figure 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.

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.

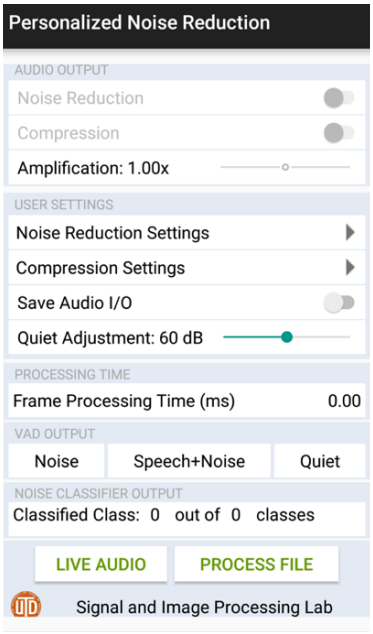


Figure 2. Main GUI page of the personalized noise reduction app.

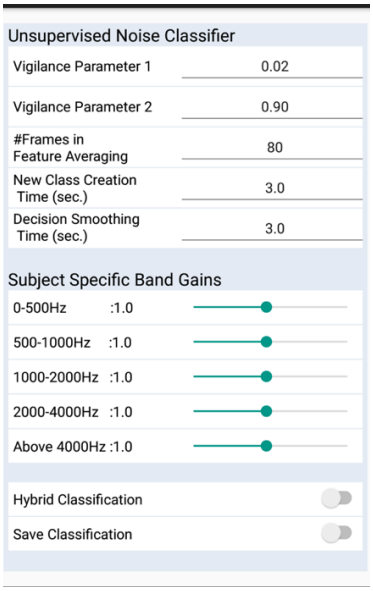


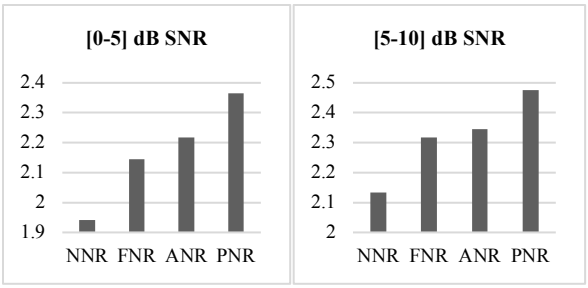
Figure 3. Noise classification settings and personalized gains adjustment GUI page.

IV. Results and Discussion

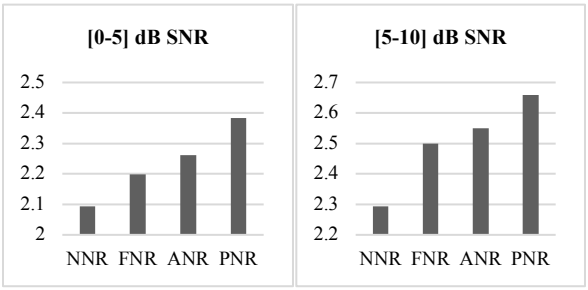
Two public domain datasets, one dataset consisting of clean speech files and the other dataset consisting of environmental noise files, were used to evaluate the performance of the developed personalized noise reduction app. The PN/NC speech dataset [15] was used here which consists of 3600 speech files 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 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. As can be seen from Figure 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.

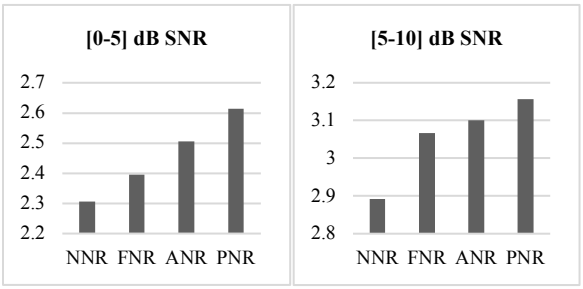
The processing time of the developed PNR app per frame is about 1.4ms on Google Pixel, which is less than 4ms frame time, and 0.75ms on iPhone 8, which is less than 1.3ms frame time. These processing times are comparable to the processing times of the ANR app, which are 0.73ms on iPhone 8 and 1.25ms 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 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.



(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)

Figure 4. PESQ measure for no noise reduction (NNR), fixed noise reduction (FNR), adaptive noise reduction (ANR), and personalized noise reduction (PNR) apps.

Table 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%	23 MB

V. 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.

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