A Real-Time Smartphone App for Unsupervised Noise Classification in Realistic Audio Environments

N. Alamdari, *Student Member, IEEE*, and N. Kehtarnavaz, *Fellow, IEEE*
Department of Electrical and Computer Engineering, University of Texas at Dallas, USA
Email: {Nasim.TaghizadehAlamdari, kehtar}@utdallas.edu

***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.**

I. Introduction

As stated in [1], smartphones can be used as open-source research platforms to conduct hearing improvement studies. In our previous works [2-5], a number of smartphone apps for various components of the signal processing pipeline of digital hearing aids were developed. One way to improve this pipeline is to automatically adjust its hearing improvement parameters depending on the noise environments encountered. This in turn requires having a noise classifier module as part of the pipeline.

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 [2], 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 to generate a new noise class in an on-the-fly manner 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 operating 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 II, the components of the developed unsupervised noise classifier app are described. Then, in section III, the real-time implementation aspects of the app are discussed. The experimental results are reported in section IV followed by the conclusion in section V.

II. Components of Developed Unsupervised Noise Classifier

Fig. 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].

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.

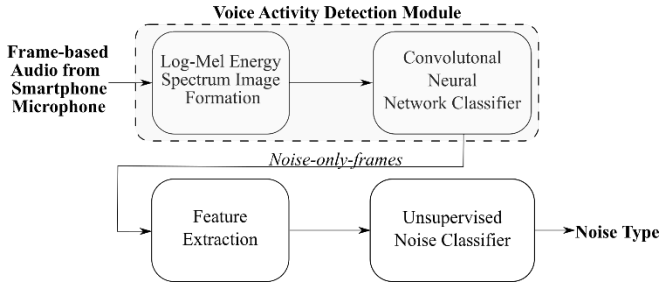

Fig. 1. Block diagram of integrating voice activity detection with the unsupervised noise classifier.

A. 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 [2]. 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 nth frame and its adjacent frames in band b and N denotes the number of frames in the time duration $[t\text{-}T, t]$. 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 nth 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 mth 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 $\hat{m}$:

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

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

$$H_n(k) = \begin{cases} 0 & ,\ k < \hat{f}(n-1) \\ \frac{k - \hat{f}(n-1)}{\hat{f}(n) - \hat{f}(n-1)} & ,\ \hat{f}(n-1) \le k \le \hat{f}(n) \\ \frac{\hat{f}(n+1) - k}{\hat{f}(n+1) - \hat{f}(n)} & ,\ \hat{f}(n) < k \le \hat{f}(n+1) \\ 0 & ,\ k > \hat{f}(n+1) \end{cases}, \tag{6}$$
$$k = 1, \dots, \frac{K}{2} \text{ and } n = 1, \dots, N.$$

Then, the MFSC coefficients are computed as follows:

$$MFSC(n) = \log\left(\sum_{k=0}^{K} H_n(k) * |F(k)|^2\right), \quad n = 1, \dots, N \tag{7}$$

where $|F(k)|^2$ denotes the FFT power spectrum. Log-mel energy spectrum images are then formed by concatenating the coefficients over time. Fig. 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.

B. 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].

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 $w_k = x$.

C. 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 Fig. 3 was implemented as a real-time smartphone app.

III. 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 Fig. 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].

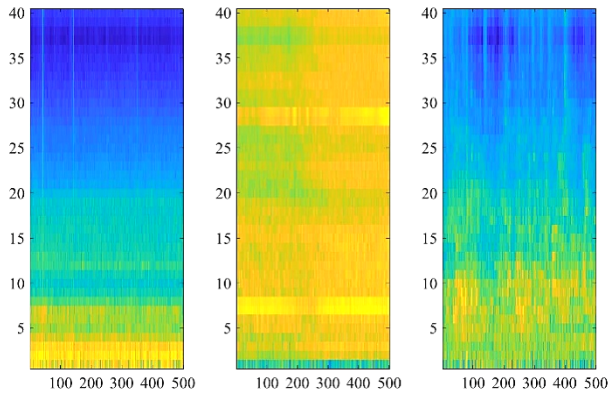

Fig. 2. Sample log-mel energy spectra of different noise types; x-axis represents frames in time, and y-axis represents mel-frequency spectral coefficients: (a) driving car noise, (b) machinery noise (vacuum cleaner), and (c) babble noise.

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 up to 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. Fig. 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.

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.

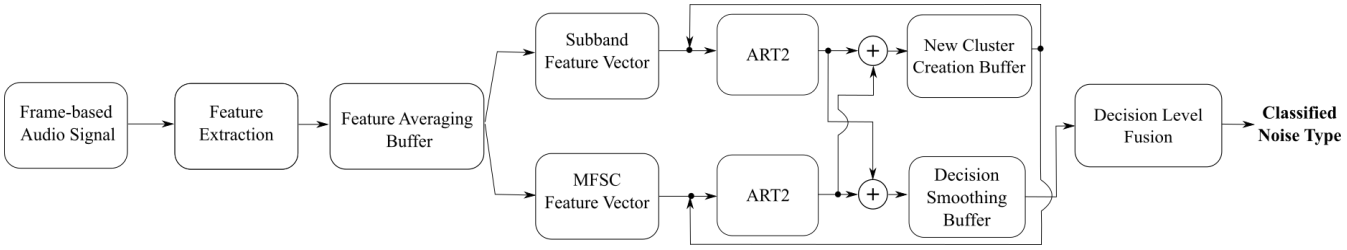

Fig. 3. Block diagram of the developed smartphone app implementing decision-level fusion of two ART2 unsupervised classifiers.

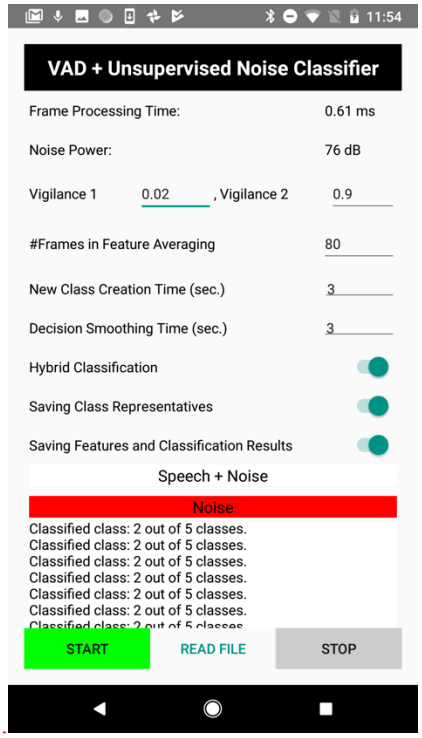

Fig. 4. Graphical user interface (GUI) of the unsupervised noise classifier smartphone app (Android version).

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 next time. These switches allow the app to be personalized for the noise environments of interest to a specific user.

IV. 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 I, 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 II 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.

Next, field testing was conducted in realistic noise environments. Figs. 5 and 6 show the outcome for two sample field testing runs corresponding to a three and a 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 Fig. 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%.

A. 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).

TABLE I 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 II 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

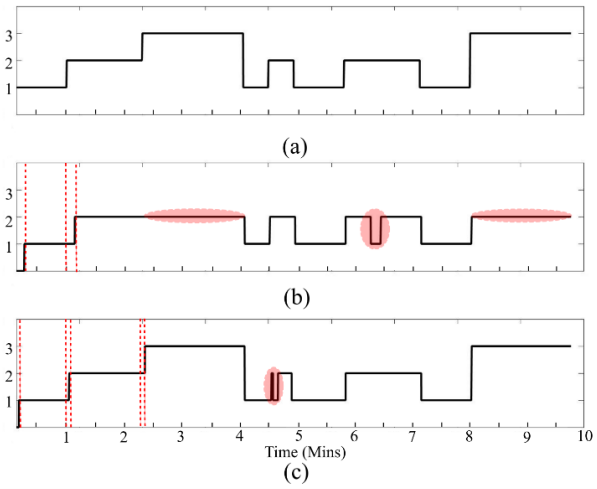

Fig. 5. Actual field testing results in three noise environments of street (label 1), dust blower (label 2), babble in activity center (label 3) (x-axis denotes time in minutes): (a) actual clusters or ground truth specified by the user, (b) unsupervised noise classification outcome by the app in [3], 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.

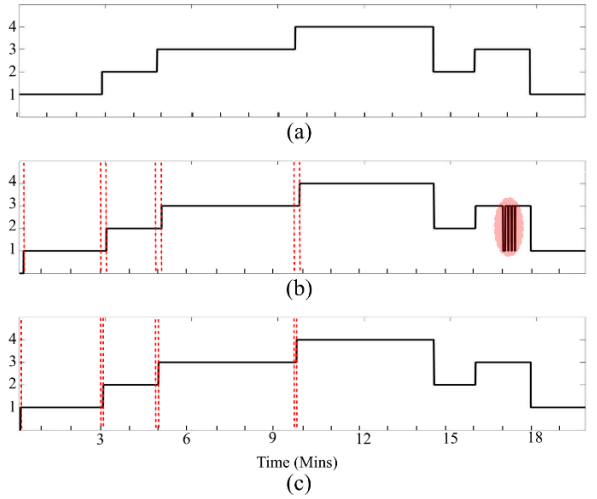

Fig. 6. Actual field testing results in four noise environments of babble in dining hall (label 1), driving car (label 2), machinery (label 3), and traffic (label 4) (x-axis denotes time in minutes): (a) actual clusters or ground truth specified by the user, (b) unsupervised noise classification outcome 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.

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.

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 .

V. 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.

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.

References

[1] N. Kehtarnavaz and I. Panahi, “Smartphones as research platform for hearing improvement studies,” *Journal of Acoustical Society of America*, vol. 141, p. 3495, 2017.

[2] 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, March 2016.

[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, Dec 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] 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, June 2017.

[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*, Las Vegas, NV, Jan 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