AN UNSUPERVISED NOISE CLASSIFICATION SMARTPHONE APP FOR HEARING IMPROVEMENT DEVICES

N. Alamdari, F. Saki, A. Sehgal, and N. Kehtarnavaz

University of Texas at Dallas, Richardson, TX, USA
kehtar@utdallas.edu

***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.**

I. 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 II provides an overview of the previously developed unsupervised classification algorithm. Section III 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 IV. Finally, the conclusion is stated in section V.

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

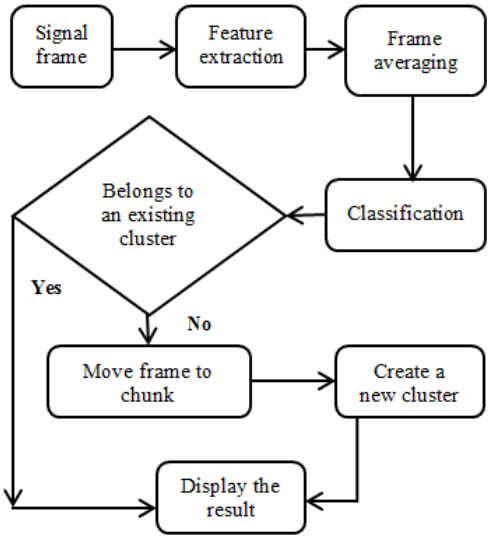


Figure 1. Block diagram of the unsupervised classification algorithm introduced in [5].

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

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

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

where $H_{b,n}$ denotes Shannon entropy of the nth 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.

III. Real-Time Implementation of Smartphone App

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

A. 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.

B. 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.

C. 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 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 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.

D. 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.

IV. Experimental Results

A. 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

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

where U denotes the total number of clusters, $\widehat{V}_l$ 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:

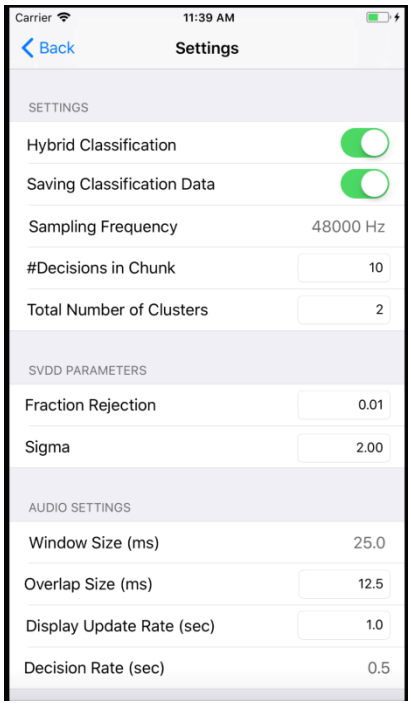


Figure 2. Settings screen of the developed smartphone app.

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

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

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

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

B. 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 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 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.

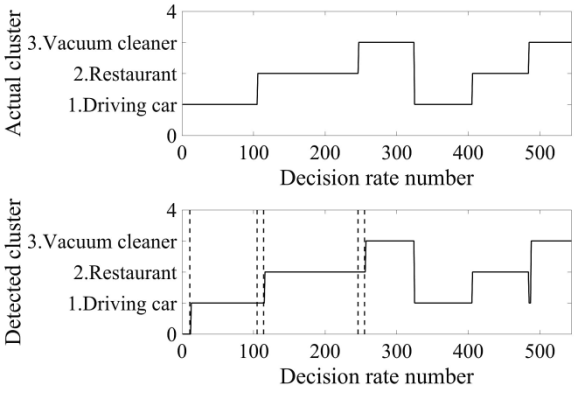
As seen from Figure 3, whenever the audio or noise environment changed to a new audio or noise environment which had not been encountered before, it took 5 seconds (illustrated by vertical lines) to detect that noise environment as a new cluster. It is worth stating that this time can be adjusted by 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. 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 1, the measures are listed with and without the new cluster creation time (denoted by I and II, respectively).

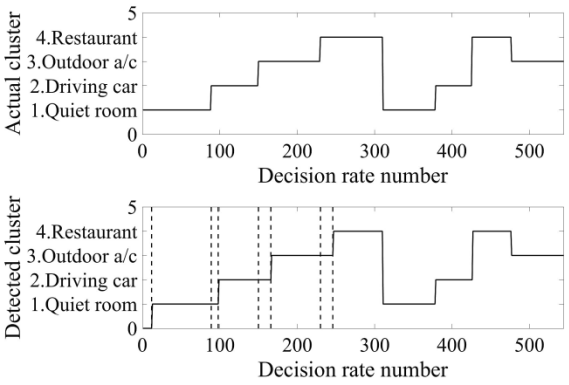
C. Real-Time Processing

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 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].

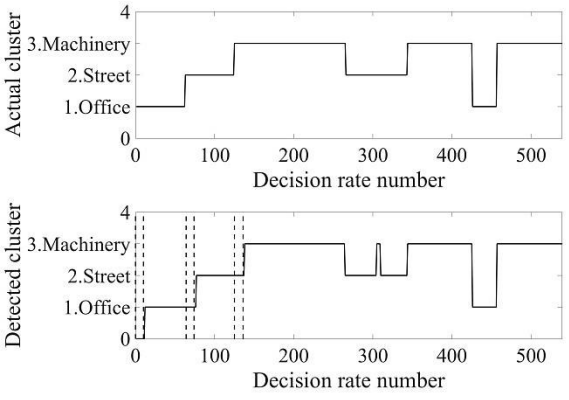
A video clip of the developed app running in real-time can be viewed at this link: http://www.utdallas.edu/~kehtar/UnsupervisedClassifierApp.mp4 .



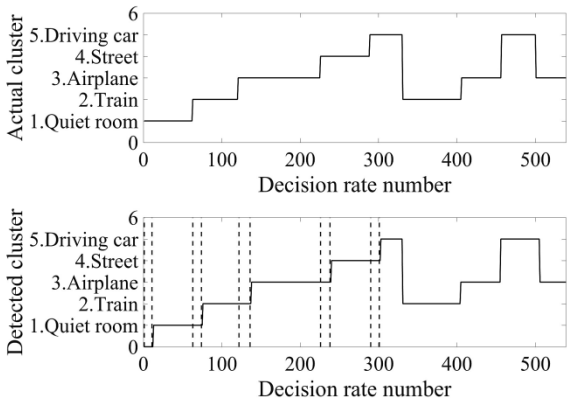
3(a)



3(b)



3(c)



3(d)

Figure 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.

	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 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.

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

Table 2. CPU consumption, memory utilization, and energy impact of the developed smartphone app as provided by Xcode IDE.

V. 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.

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.

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] https://developer.apple.com/xcode/ide/