User's Guide:

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

N. ALAMDARI AND N. KEHTARNAVAZ

UNIVERSITY OF TEXAS AT DALLAS

APRIL 2019

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.

Table of Contents

Introduction

This user's guide describes how to run and use the code of an unsupervised noise classifier app discussed in [1]. The algorithm implemented in this app is discussed in detail in [1]. This app does not require any offline training to classify noise environments that are of interest to a particular user. It addresses the two limitations of a previously developed smartphone app in [2] for unsupervised noise classification. A voice activity detection [3] 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 (ART2) 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 operated in realistic audio environments.

This user's guide consists of two parts. The first part covers the iOS version of the app and the second part its Android version. Each part has two sections. The first section describes the app GUI and the second section the app code flow.

Devices used for running the app

- iPhone7 and iPhone8 as iOS platforms
- Google Pixel and Pixel 2 as Android platforms

Folder Description

The codes for the iOS and Android versions of the app are arranged as noted below. The folders containing the app codes include (see Fig. 1):

- "VAD_UNC_iOS" – This folder includes the iOS Xcode project. To open the project, double click on the "VAD_UNC.xcodeproj" inside the folder.
- "VAD_UNC_Android" – This folder includes the Android Studio project. To open the project, open Android Studio, click on "Open an existing Android Studio Project" and navigate to the project "VAD_UNC_Android".

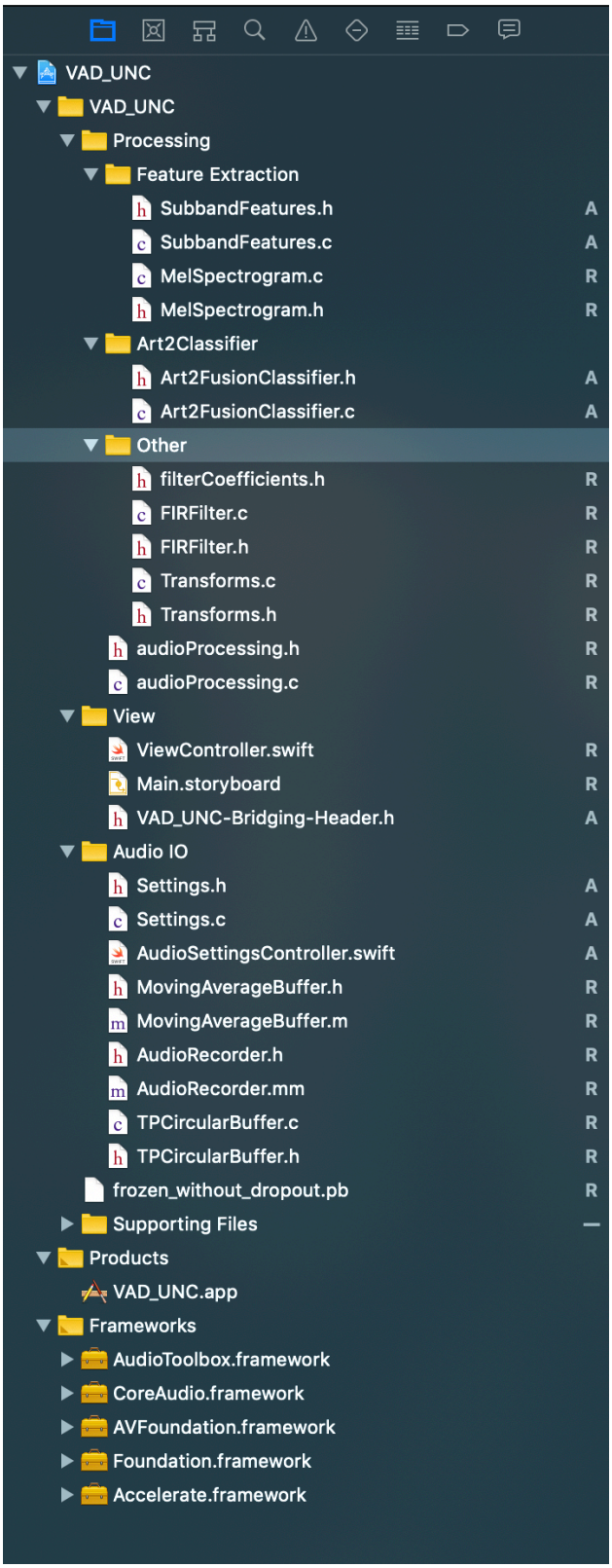

Fig. 1: Content of voice activity detection and unsupervised noise classifier (VAD_UNC) folders.

Part 1

iOS

Section 1: iOS GUI

This section covers the iOS GUI of the app consisting of the main page and also the settings page. The GUI layer, written in Swift [4], is used to handle the entire user interface for the app by implementing several view controllers with navigation between them.

The main page is the root view for navigation, and is the view that opens when the app is launched and it consists of the following four views (see Fig. 2):

- Title View
- Settings View
- VAD View
- Status View
- Button View

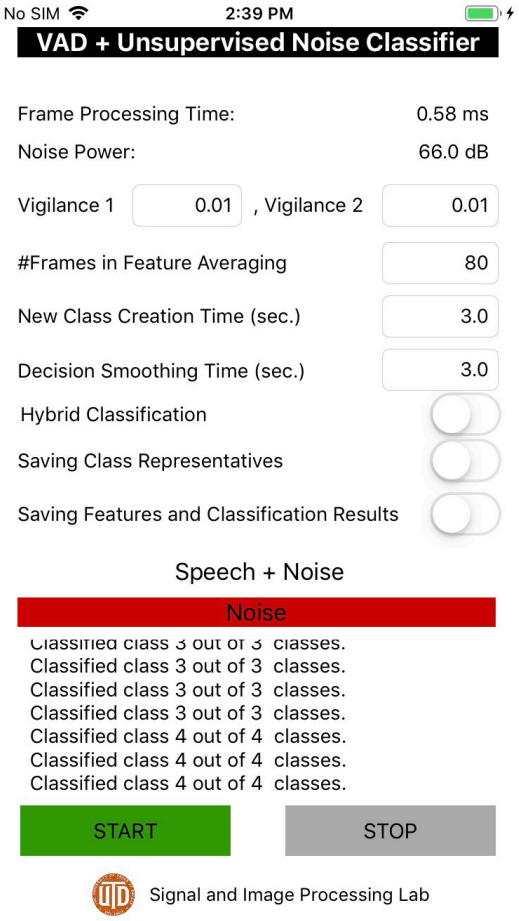

Fig. 2: App GUI – iOS version

1.1 Title View

This view displays the app title.

1.2 Settings View

This view controls the noise classifier settings as listed below:

- **Frame Processing time –** This field shows frame processing time in milliseconds.
- **Noise Power –** This field shows noise power or noise pressure level in decibel (dB).
- **Vigilance 1 & Vigilance 2 -** Two vigilance parameters associated with the two ART2 classifiers (vigilance 1 and vigilance 2).
- **#Frames in Feature Averaging** – This field sets how many feature frames to have in a chunk before performing noise classification.
- **New Class Creation time (sec.) –** This fields sets new class creation in seconds. 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.
- **Decision Smoothing Time (sec.) –** This fields sets noise class decision smoothing buffer size in seconds. 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.
- **Hybrid classification -** By enabling this switch**,** the app uses previously detected and saved clusters for classification of future noise frames.
- **Saving Class Representatives -** By enabling this switch**,** all the information regarding current detected clusters are saved for future use.
- **Saving Features and Classification Results -** By enabling this switch, the noise signal features and classification results are saved**.**

1.3 VAD View

This view shows the results of voice activity detection (VAD) which either is pure *Noise* or *Speech+Noise*.

1.4 Status View

This view provides real-time feedback on:

- Currently detected cluster
- How many clusters detected so far

1.5 Button View

This view allows the user to start and stop the app.

Section 2: Code Flow

This section states the app code flow. The app "UNC_iOS.xcodeproj" can be run from the app codes listed in Fig. 1. The connections between Swift, Objective-C and native C/C++ codes are illustrated in Fig. 3.

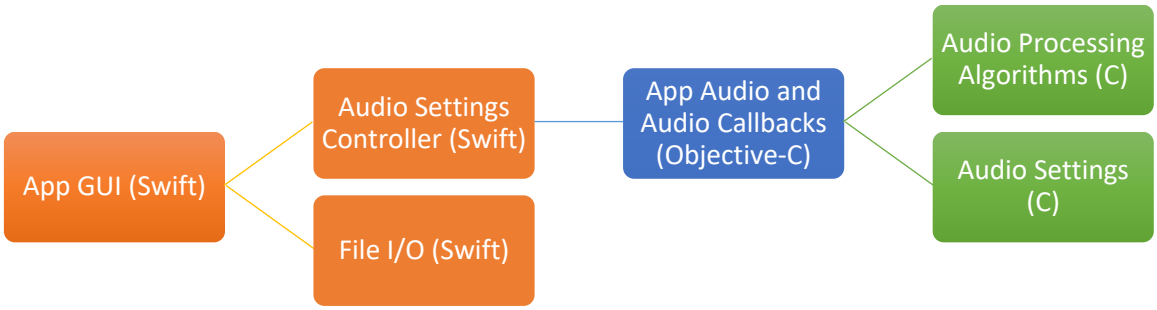

Fig. 3: App code flow – iOS version

The code includes four parts:

- Processing (including feature extraction and unsupervised noise classifier)
- GUI (View)
- Audio IO (Controller)
- Supporting Files

2.1 Processing

This part includes the signal processing module written in C. It includes feature extraction and unsupervised classification codes:

- **audioProcessing -** This is the main native codes called by the Objective-C code AudioRecorder. This code initializes all the settings and then computes the FFT of incoming audio frames, extracts features using either subband features or mel spectrum images, then feeds the extracted features into the classifier module. This main native code consists of:
- **Transforms.h -** This component computes the FFT of the incoming audio frames.
- **filterCoefficients.h** -> This component includes filter coefficients for FIR filter.

- **FIRFilter.h** – This component computes FIT filter.
- **SubbandFeatures.h** - This component computes the subband features for each frame.
- **MelSpectrogram.h** - This component computes the Log Mel Spectrogram features for each frame.
- **Art2FusionClassifier.h –** This component performs unsupervised noise classification.

2.2 View (GUI)

This part involves the app GUI containing the following components:

- **Main.storyboard -** This component provides the layout of the app GUI.
- **ViewController -** This component provides the main view GUI elements and actions of the GUI View.
- **VAD_UNC-Bridging-Header.h –** This is the bridge header function between swift and C/C++ functions.

2.3 Audio IO (Controller)

This part provides the controllers to pass data from the GUI to the native C/C++ code and to update the status from the native code to appear in the GUI. The components are:

- **AudioSettingsController -** AudioSettingsController provides audio settings adjustments at the Swift level or layer. This component acts as a bridge between the view and the native code.
- **AudioRecorder.h -** This component controls the audio i/o setup for processing incoming audio frames by calling the native code. In other words, the app's audio settings are handled by this component.
- **Settings.h –** This component includes initialization for all the settings parameters.
- **MovingAverageBuffer.h –** This provides a separate class written in Objective-C to compute the frame processing time.
- **TPCircularBuffer.h –** This component is for utilizing circular buffer to capture audio from microphone and play it back to the speaker. In order to capture and playback audio frames with the lowest latency offered by the i/o hardware of smartphones, similar to [5], 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.

Part 2

Android

Section 1: Android GUI

This section covers the GUI of the Android version of the app. It consists of the following four views (see Fig. 4):

- Title View
- Settings View
- VAD View
- Status View
- Button View

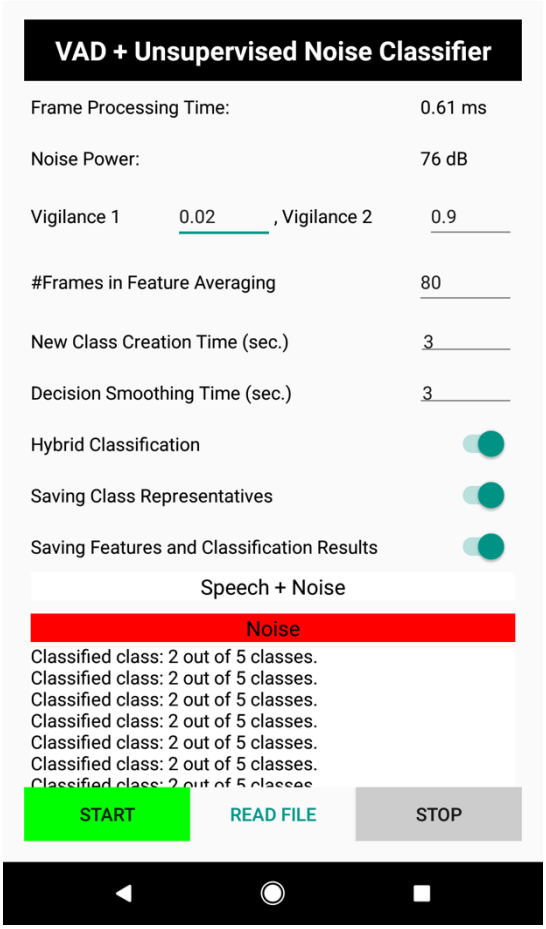

Fig. 4: App GUI – Android version

1.1 Title View

This view displays the app's title.

1.2 Settings View

This view controls the noise classifier settings as listed below:

- **Frame Processing time –** This field shows frame processing time in milliseconds.
- **Noise Power –** This field shows noise power or noise pressure level in decibel (dB).
- **Vigilance 1 & Vigilance 2 -** Two vigilance parameters associated with the two ART2 classifiers (vigilance 1 and vigilance 2).
- **#Frames in Feature Averaging** – This field sets how many feature frames to have in a chunk before performing noise classification.
- **New Class Creation time (sec.) –** This fields sets new class creation in seconds. 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.
- **Decision Smoothing Time (sec.) –** This fields sets noise class decision smoothing buffer size in seconds. 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.
- **Hybrid classification -** By enabling this switch**,** the app uses previously detected and saved clusters for classification of future noise frames.
- **Saving Class Representatives -** By enabling this switch**,** all the information regarding current detected clusters are saved for future use.
- **Saving Features and Classification Results -** By enabling this switch, the noise signal features and classification results are saved**.**

1.3 VAD View

This view shows the results of voice activity detection (VAD) which either is pure *Noise* or *Speech+Noise*.

1.4 Status View

This view provides real-time feedback on:

- Currently detected cluster

- How many clusters detected so far

1.5 Button View

This view allows the user to start and stop the app.

Section 2: Code Flow

To be able to run the Android version of the app, the Superpowered SDK library [6] needs to be linked to it.

2.1 Project Organization

After the project is opened in Android Studio, the project organization appears as shown in Fig. 5. The app/src/main consists of the folders as shown in Fig. 5.

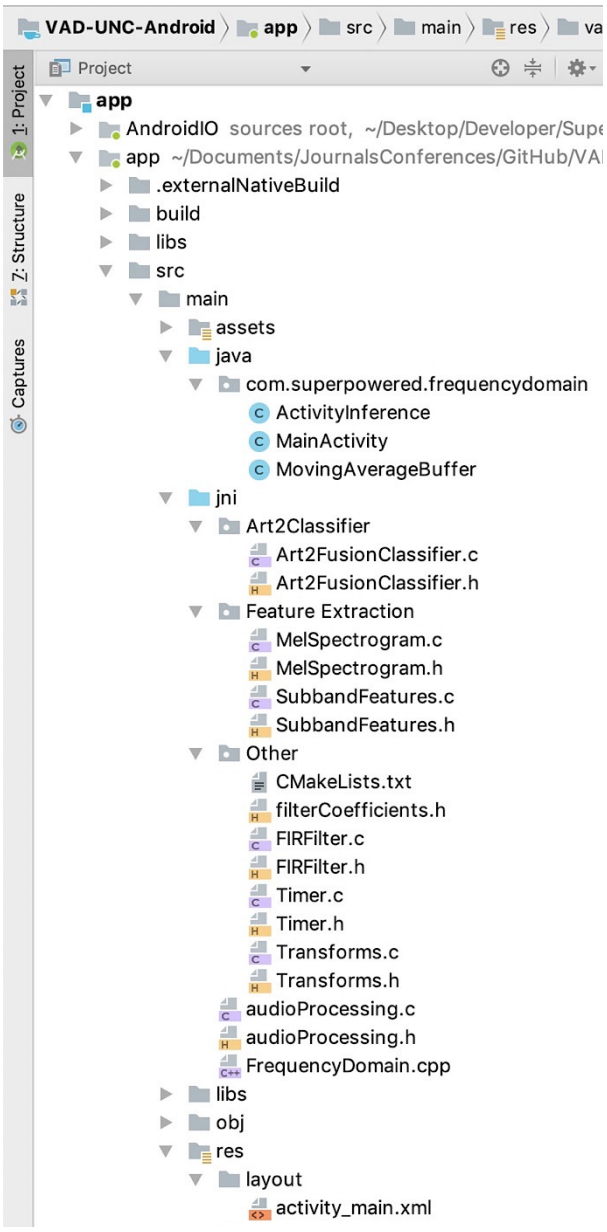

Fig. 5: App codes listing – Android version

- **Java -** This folder contains the “MainActivity.java” file which handles all the operations of the app and allows the user to link the GUI to the native code.
- **JNI -** This folder includes the native codes to start the audio i/o and also the audio processing files to perform unsupervised noise classification.

2.2 Native Code

The JNI folder contains the code of the audio processing for the unsupervised noise classification which is divided into the following subfolders and files:

- **Frequency Domain -** This component is the C++ file that acts as a bridge between the native code and the java GUI. It creates an audio i/o interface with the GUI settings.
- **audioProcessing -** This component is the entry point of the native codes of the app. It initializes all the settings for the modules and then processes incoming audio signal frames to perform feature extraction and recognize the noise or cluster label of a current frame. It involves the following subfolders:
    - **Art2Classifier –** This subfolder includes Art2FucsionClassifier.h to perform unsupervised noise classification.
    - **Feature Extraction –** This subfolder includes MelSpectrogram.h and SubbandFeatures.h to extract features from each audio frame.
    - **Other –** This subfolder includes the following functions:
        - **Timer.h –** This component is for computing processing time of each frame, including time for extracting features and classification.
        - **Transforms.h -** This component computes the FFT of the incoming audio frames.
        - **filterCoefficients.h** -> This component includes filter coefficients for the FIR filter.
        - **FIRFilter.h** – This component computes the FIR filter.

References

[1] N. Alamdari, and N. Kehtarnavaz, “A real-time smartphone app for unsupervised noise classification in realistic audio environments” *Proceedings of IEEE Signal Processing in Medicine and Biology Symposium*, Philadelphia, PA, Dec 2018.

[2] 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.

[3] 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.

[4] Swift, https://developer.apple.com/swift/, 2019.

[5] 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.

[6] Superpowered, https://superpowered.com, 2019.