User's Guide:
A Smartphone App Integrating VAD and Supervised Noise Classifier for Hearing Improvement Studies

T. CHOWDHURY, A. SEHGAL, AND N. KEHTARNAVAZ

UNIVERSITY OF TEXAS AT DALLAS

DECEMBER 2017

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 an integrated smartphone app consisting of two previously developed apps of Voice Activity Detector (VAD) covered in [1, 2] and the supervised noise classifier covered in [3, 4].

The first part of this guide covers the iOS version of this integrated app and the second part covers its Android version. Each part consists of four sections. The first section discusses the GUI of the app. The second section covers the code flow of the app. In the third section, the process of data collection for training is mentioned. Finally, how to run the MATLAB training code to generate the parameters associated with the classifiers, is covered in the fourth section.

Devices used for running this integrated app

- iPhone7 as iOS platform
- Google Pixel as Android platform

Folder Description

The codes for the Android and iOS versions of the app are arranged as described below. Fig. 1 lists the contents of the folder containing the app codes. These contents include:

- "VAD_NC_iOS" includes the iOS Xcode project. To open the project, double click on the "VAD_NC iOS.xcodeproj" inside the folder.
- "VAD_NC_Android" 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_NC_Android".
- "VAD_RandomForest_Training" contains the MATLAB code to train the random forest classifier for VAD, based on data collected by the smartphone on which the app is to be run.
- "NoiseClassifier_RandomForest_Training" contains the MATLAB code to train the random forest classifier for NoiseClassifier, based on data collected by the smartphone on which the app is to be run.

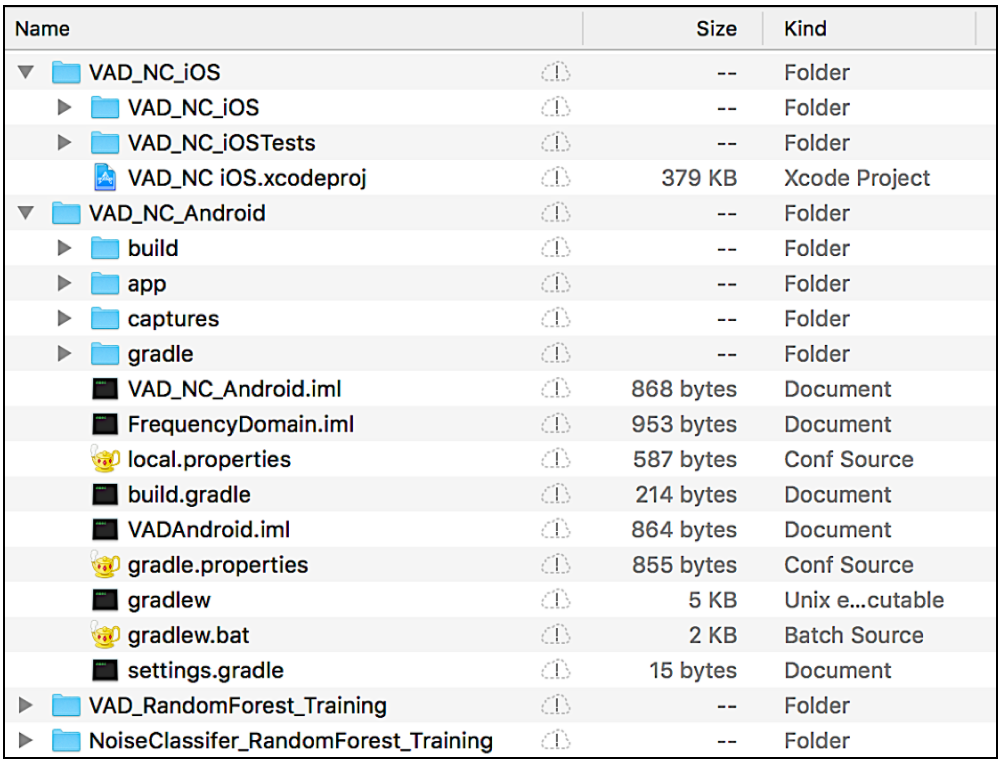

Fig. 1: Integrated app: Contents of VAD and NoiseClassifier folders

Part 1

iOS

Section 1: iOS GUI

This section covers the iOS GUI of the integrated app consisting of VAD and Noise Classifier and its entries. The GUI consists of 4 views, see Fig. 2:

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

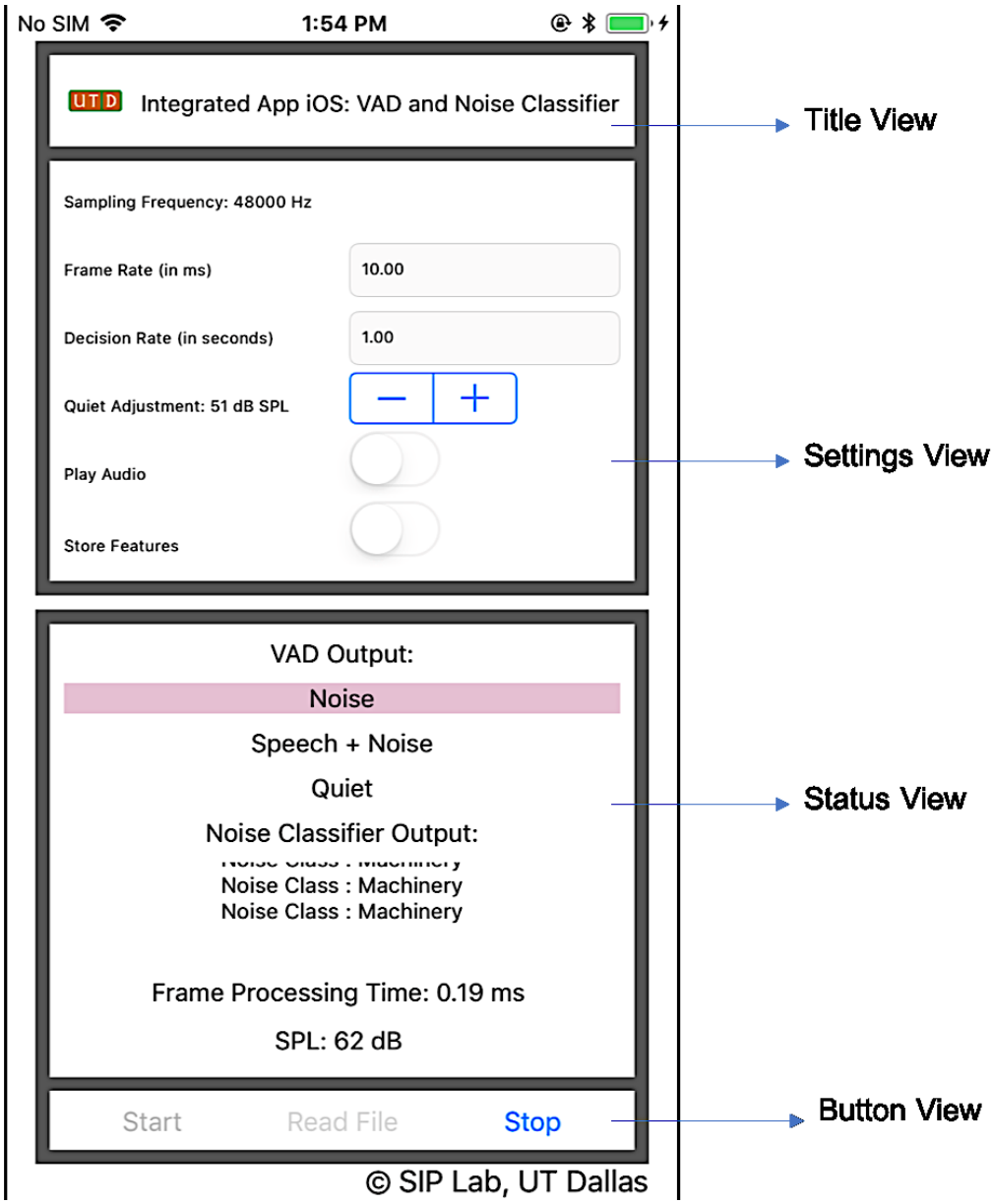

Fig. 2: Integrated app GUI – iOS version

1.1 Title View

The title view displays the title of the app.

1.2 Settings View

This view controls the app settings as follows:

- **Sampling Frequency** in Hertz (Hz),
- **Frame Rate** in milliseconds (ms),
- **Decision Rate** in seconds,
- **Quite Adjustment** in dB SPL (sound pressure level),
- **Play Audio**, a switch to play audio signal to the smartphone speaker,
- **Store Features**, a switch to enable feature collection from audio environment.

Details of these settings are described in the user's guides for the individual VAD and Noise Classifier apps in [2] and [4]. For integrating these two modules, the sampling frequency is kept fixed at 48kHz in order to have the lowest audio latency imposed by the i/o hardware of smartphones. Other settings can be changed by the user before the app is run.

1.3 Status View

The status view provides real-time feedback on:

- Whether captured audio signal frames are noise or speech+noise;
- If noise is detected, it shows the noise type (babble, machinery or traffic) for which the app has been trained;
- How much processing time is taken per frame in milliseconds; and
- SPL in dB.

1.4 Button View

This view allows the user to start and stop the app. The app can also operate on audio files of ".wav" format. The user's guides in [2] and [4] describe how to add an audio file to the app project.

Section 2: Code Flow

This section states the app code flow. The user can view the code by running "VAD_NC iOS.xcodeproj" as shown in Fig. 1. The code is divided into 3 parts, see Fig. 3:

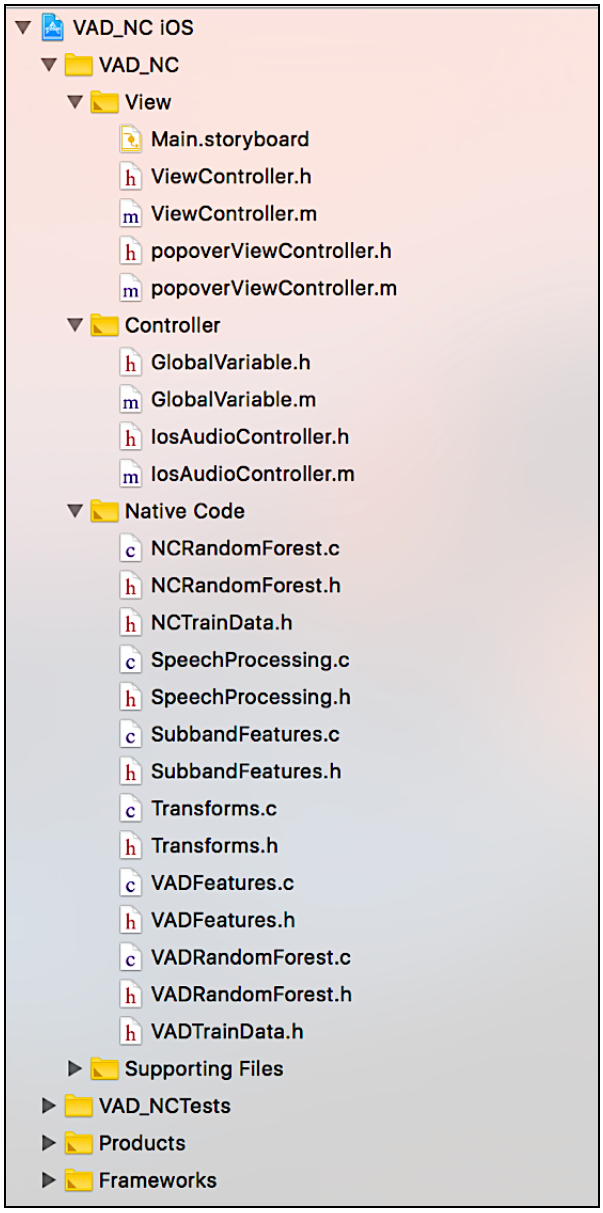

Fig. 3: Integrated app code flow – iOS version

- View
- Controller
- Native Code

2.1 View

This part provides the setup for the GUI of the app and has the following components:

- **Main.storyboard:** This component provides the layout of the app GUI.
- **ViewController:** This component provides the GUI elements and actions of the GUI View.
- **popoverViewController:** This component enables popping over the app GUI when the“Read File” button is clicked, populating a table view of audio files.

2.2 Controller

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

- **GlobalVariable:** This component acts as a bridge between the view and the native code.
- **IosAudioController:** This component controls the audio i/o setup for processing incoming audio frames by calling the native code.

3.3 Native Code

This part comprises the integrated app modules written in C. It is divided into the following components:

- **Speech Processing:** This component is the entry point of the native codes of the app. It initializes all the settings for the two modules and then processes incoming audio signal frames to detect noise/speech+noise and noise type. It consists of:
  - **Transform:** This component computes the FFT of the incoming audio frames.
  - **SubbandFeatures:** This component uses the FFT to extract subband features used by both the VAD and NoiseClassifier modules of the app.
  - **VADFeatures:** Besides the subband features, this component is used to extract the additional features required by the VAD module of the app.
  - **VADRandomForest:** This component classifies the incoming feature vector as noise or speech+noise.

- **NCRandomForest:** This component classifies the noise type if the VAD detects the incoming feature vector as noise.
- **VADTrainData:** This component is the random forest classifier designed for the VAD module. Its parameters are obtained by performing training in MATLAB on the HINT sentences audio files [5].
- **NCTrainData:** This component is the random forest classifier designed for the NoiseClassifier module. Its parameters are obtained by performing training in MATLAB on the HINT sentences audio files [5].

Readers are referred to the user's guides of the individual apps in [2] and [4] for more details. The above components appear in a modular manner to allow their modification or replacement with ease.

Section 3: Data Collection for Training

Considering that both the classifiers in the VAD and NoiseClassifier modules are supervised classifiers, data are needed to train them. Data can be collected using the "Store Features" switch button in the app. When this button is turned on and the app is run, the app stores the data needed for training the classifiers. The data get stored in the form of a ".txt" file with comma separated values and each data frame appears on a new line. The data corresponds to the following features extracted from audio signal frames:

- 4 band periodicities (BP1, BP2, BP3, BP4),
- 4 band entropies (BE1, BE2, BE3, BE4),
- Short-Time Energy Deviation (STED),
- Subband Power Spectral Deviation (SPSD), and
- Spectral Flux (SF).

All of the above 11 features are used by the VAD module. From these features, the first 8 modules (BPs and BEs) are also used by the NoiseClassifier module. The user's guides in [2] and [4] provide more details on data collection.

Section 4: Training Classifiers

After data get collected, the random forest classifiers need to be trained in MATLAB for obtaining their parameters. The obtained parameters for both of the classifiers are then assigned to the native code for actual operation.

The classification operation consists of two parts:

- **VAD operation:** This module classifies whether audio signal frames are noise or speech+noise. A random forest training script in MATLAB appears inside the folder "VAD_RandomForest_Training", see Fig. 1. This generates a header file containing the model which is renamed as "VADTrainData.h" and used in the project.

- **NoiseClassifier operation:** This module classifies the noise type detected by the VAD among three previously trained classes of babble, machinery, and traffic. A random forest training script in MATLAB appears inside the folder "NoiseClassifier_RandomForest_Training", see Fig. 1. This generates a header file containing the model which is renamed as "NCTrainData.h" and used in the project.

Part 2

Android

Section 1: Android GUI

This section covers the Android GUI of the integrated app consisting of VAD and Noise Classifier and its entries. The GUI consists of 4 views, see Fig. 4:

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

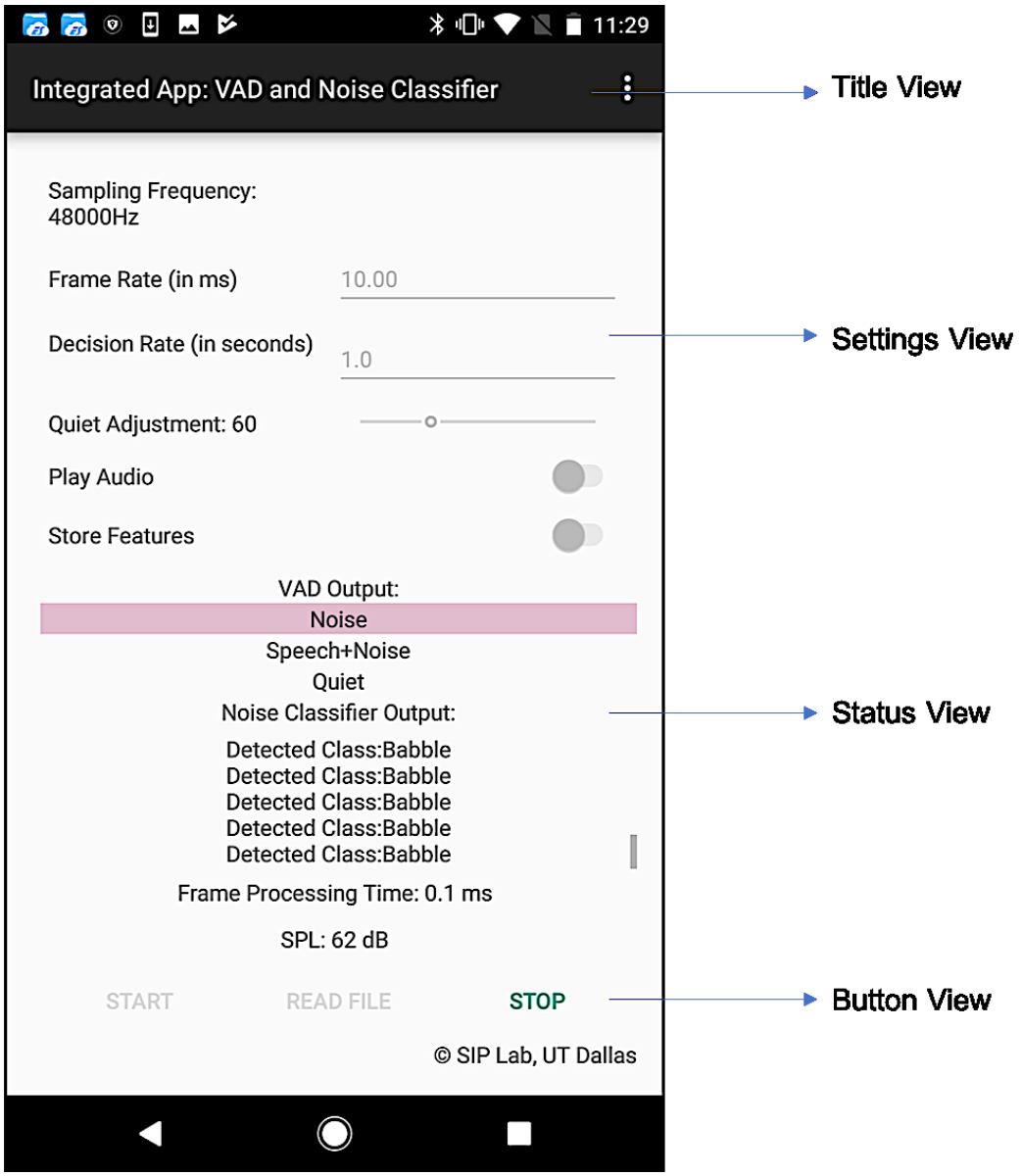

Fig. 4: Integrated app GUI – Android version

1.1 Title View

The title view displays the title of the app.

1.2 Settings View

This view controls the app settings as follows:

- **Sampling Frequency** in Hertz (Hz),
- **Frame Rate** in milliseconds (ms),
- **Decision Rate** in seconds,
- **Quite Adjustment** in dB SPL (sound pressure level),
- **Play Audio**, a switch to play audio signal to the smartphone speaker,
- **Store Features**, a switch to enable feature collection from audio environment.

Details of these settings are described in the user's guides for the individual VAD and Noise Classifier apps in [2] and [4], respectively. For integrating these two modules, the sampling frequency is kept fixed at 48kHz in order to have the lowest audio latency imposed by the i/o hardware of smartphones. Other settings can be changed by the user before the app is run.

1.3 Status View

The status view provides real-time feedback on:

- Whether captured audio signal frames are noise or speech+noise;
- If noise is detected, it shows the noise type (babble, machinery or traffic) for which the app has been trained;
- How much processing time is taken per frame in milliseconds; and
- SPL in dB.

1.4 Button View

This view allows the user to start and stop the app. The app can also operate on audio files of ".wav" format. The user's guides in [2] and [4] describe how to add an audio file to the app project.

Section 2: Code Flow

To be able to run the Android version of the app, it is required to link the Superpowered SDK library [6] to it. The steps described in the user's guides in [2] and [4] need to be followed to open the "VAD_NC_Android" project and run the app.

2.1 Project Organization

After the project is opened in Android Studio, the project organization appears as shown in Fig. 5.

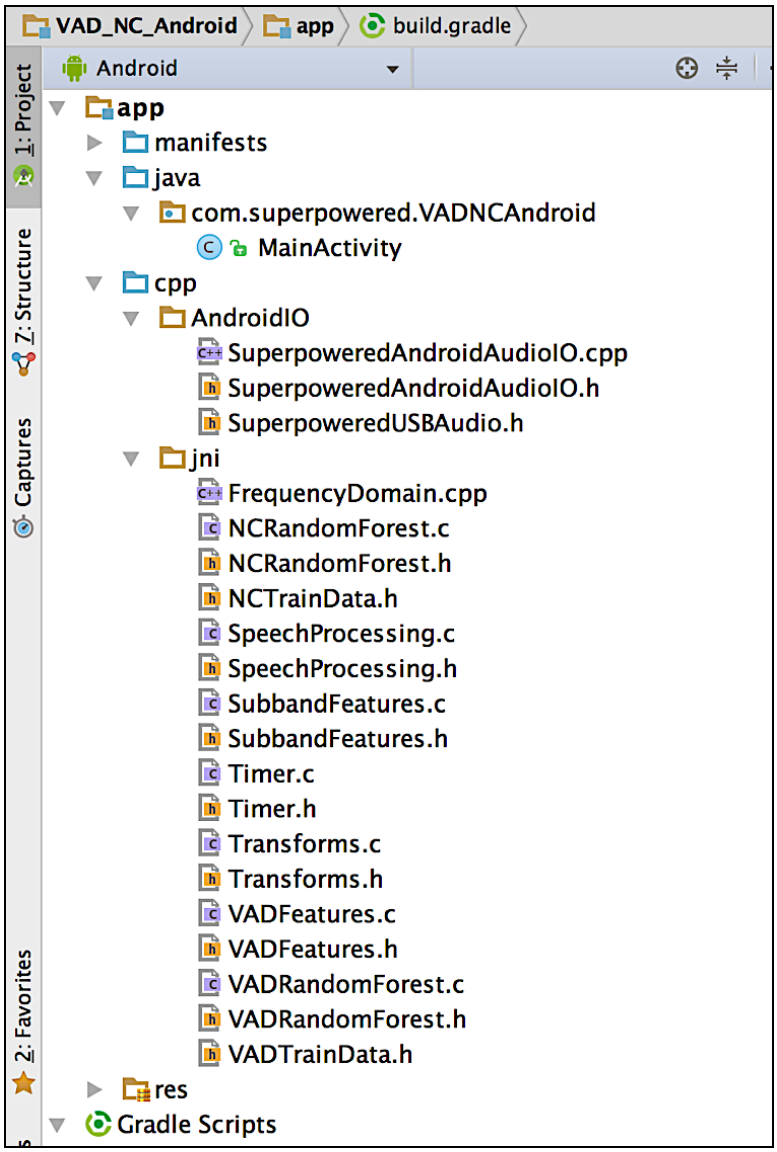

Fig. 5: Integrated app code flow – Android version

- **java:** This java 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.
- **cpp:** This folder contains the native code divided into two subfolders:
    - **AndroidIO:** This subfolder contains the Superpowered sources root files which control the audio interface to the app.
    - **jni:** This subfolder includes the native codes to start audio i/o and also the audio processing files of the VAD and NoiseClassifier modules.

2.2 Native Code

The native code part comprises the integrated app modules written in C and a bridging file written in C++. It is divided into the following components:

- **Frequency Domain:** This component is the C++ file that acts as a bridge between the native code and the java GUI. This file creates an audio i/o interface with the GUI settings.
- **Speech Processing:** This component is the entry point of the native codes of the app. It initializes all the settings for the two modules and then processes incoming audio signal frames to detect noise/speech+noise and noise type. It consists of:
    - **Transform:** This component computes the FFT of the incoming audio frames.
    - **SubbandFeatures:** This component uses the FFT to extract subband features used by both the VAD and NoiseClassifier modules of the app.
    - **VADFeatures:** Besides the subband features, this component is used to extract the additional features required by the VAD module of the app.
    - **VADRandomForest:** This component classifies the incoming feature vector as noise or speech+noise.
    - **NCRandomForest:** This component classifies the noise type if the VAD detects the incoming feature vector as noise.
    - **VADTrainData:** This component is the random forest classifier designed for the VAD module. Its parameters are obtained by performing training in MATLAB on the HINT sentences audio files [5].
    - **NCTrainData:** This component is the random forest classifier designed for the NoiseClassifier module. Its parameters are obtained by performing training in MATLAB on the HINT sentences audio files [5].

Readers are referred to the individual user's guides in [2] and [4] for more details. The above components appear in a modular manner to allow their modification or replacement with ease.

Section 3: Data Collection for Training

Considering that both the classifiers in the VAD and NoiseClassifier modules are supervised classifiers, data are needed to train them. Data can be collected using the “Store Features” switch button in the app. When this button is turned on and the app is run, the app stores the data needed for training the classifiers. The data get stored in the form of a “.txt” file in the device storage under the folder labelled “VAD_NCAndroid”, with comma separated values and each data frame appears on a new line. The data corresponds to the following features extracted from audio signals:

- 4 band periodicities (BP1, BP2, BP3, BP4),
- 4 band entropies (BE1, BE2, BE3, BE4),
- Short-Time Energy Deviation (STED),
- Subband Power Spectral Deviation (SPSD), and
- Spectral Flux (SF).

All of the above 11 features are used by the VAD module. From these features, the first 8 modules (BPs and BEs) are also used by the NoiseClassifier module. The user’s guides in [2] and [4] provide more details on data collection.

Section 4: Training Classifiers

After data get collected, the random forest classifiers need to be trained in MATLAB for obtaining their parameters. The obtained parameters for both of the classifiers are then assigned to the native code for actual operation.

The classification operation consists of two parts:

- VAD operation: This module classifies whether audio signal frames are noise or speech+noise. A random forest training script in MATLAB appears inside the folder "VAD_RandomForest_Training", see Fig. 1. This generates a header file containing the model which is renamed as "VADTrainData.h" and used in the project.

- NoiseClassifier operation: This module classifies the noise type detected by the VAD among three previously trained classes of babble, machinery, and traffic. A random forest training script in MATLAB appears inside the folder "NoiseClassifier_RandomForest_Training", see Fig. 1. This generates a header file containing the model which is renamed as "NCTrainData.h" and used in the project.

References

[1] A. Sehgal, F. Saki, and N. Kehtarnavaz, "Real-Time Implementation of Voice Activity Detector on ARM Embedded Processor of Smartphones," *Proceedings of IEEE 26th International Symposium on Industrial Electronics (ISIE),* Edinburgh, UK, pp. 1285-1290, 2017.

[2] http://www.utdallas.edu/ssprl/files/Users-Guide-VAD.pdf

[3] 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 26th International Conference on Acoustics, Speech and Signal Processing (ICASSP),* Shanghai, China, pp. 2204 – 2208, 2016.

[4] http://www.utdallas.edu/ssprl/files/UsersGuide-NoiseClassifier-App1.pdf

[5] http://www.californiaearinstitute.com/audiology-services-hint-bay-area-ca.php

[6] http://superpowered.com