COSC 522 Assignment 2 Report

Team 15: Fujiao Ji, Ark Ifeanyi, Sobhan Bahrami Zadegan

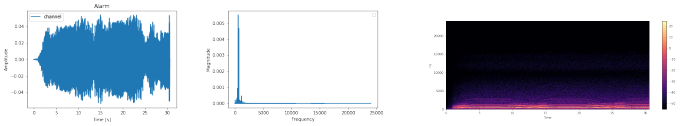
***Abstract*—In this assignment, we collected and analyzed activities' or events' (sounds) to see if we could build machine learning pipelines that correctly identified them. Specifically, we first collected data through mobile application, then applied a median filter preprocessing. After that, we designed data analysis pipelines and finally got the result. We further made real-time prediction to test the trained models and saved the video to show that. To run the code, you should put folder "Dataset" and executavle python file in the folder "Assignment2". All data should be in the folder "Dataset". Then the training models were saved in folder "Model". Some confusion matrix pictures were saved in the folder "Pictures". All the datasets and python file can be accessed through this link.**

I. Data Collection

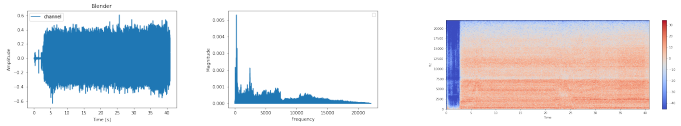
We focus on 6 classes, which are Microwave, Blender, Alarm, Vacuum Cleaner, Music and Silence. For each class, we collect 20 samples and each sample is longer than 30 seconds. For recording the sounds, we use AVRX application with frequency higher or equals to 44.1 KHz on Android mobile phone to record sounds, and save them to uncompressed WAV files. The collected dataset can be accessed through this link.

II. Pre-Processing

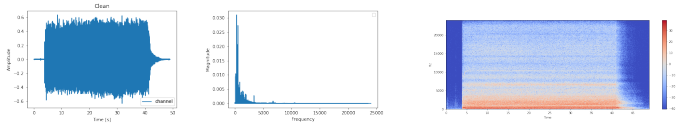
As shown in the Fig.1, we take one sample for each six classes as examples. We first perform a median filter on the time-domain data, which is a non-linear digital filtering technique used to remove noise from the input signal data using a local window-size given by kernel size. In this assignment, we set kernel size as 3. The time-domain figures are shown at the first column of Fig.1. In order to further dig out the data attributes, we apply the Fast Fourier Transform to convert the raw time-domain signal into frequency-domain. We can see that these six classes signals are composed of thousands of different frequencies. Most of them concentrate on low frequency while blender also has high frequency. We plot frequencies ranging from 0 to 25000Hz because our signals are sampled at 48000 and 44100 sampling rate. According to the Nyquist sampling theorem, it should only posses frequencies below that 24000Hz.



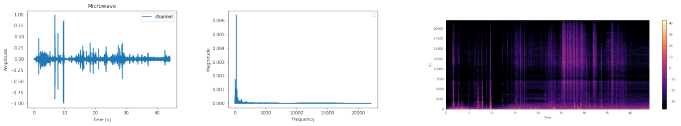
(a) Alarm Class: time domain, magnitude of frequency, and spectrum



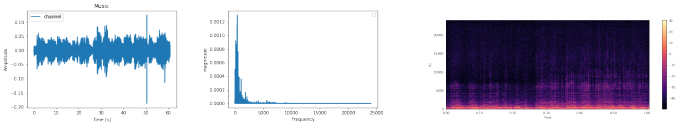
(b) Blender Class: time domain, magnitude of frequency, and spectrum



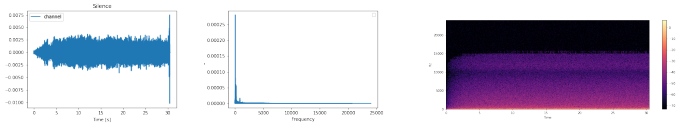
(c) Clean Class: time domain, magnitude of frequency, and spectrum



(d) Microwave Class: time domain, magnitude of frequency, and spectrum



(e) Music Class: time domain, magnitude of frequency, and spectrum



(f) Silence Class: time domain, magnitude of frequency, and spectrum

Fig. 1. Six classes' time domain, magnitude of frequency, and spectrum pictures.

III. Data Analysis Pipeline

A. Reasoning behind Feature Selection Approach

Upon exploration of the dataset, it was discovered that frequency distribution of each class differed significantly and to best capture this difference, the entire bins were selected as windows' or samples' feature in the binned approach. The bin highlights the difference in magnitudes of frequencies over the segmented time frames.

Mel Frequency Cepstral Coefficient (MFCC) is often used in speech recognition for identifying the different speech sources represented by their spectral envelope. Our instruments can essentially serve as different sound sources with theoretically different tracts. Furthermore, MFCC has been applied in the identification of phone brands and models through speech. As a result, it was suspected that MFCC could potentially capture the differences in sounds coming from different electronic devices [2].

Detailed feature selection results can refer to section V-B.

B. Feature Engineering

For feature engineering, we would like to take samples' feature as machine learning input. Thus, we try two approaches and further divide them into two types, as shown in Fig.2:

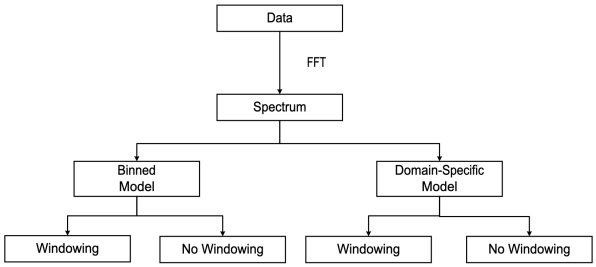


Fig. 2. The whole process to get different features and its model

- Binning the spectrogram data from the recordings and using each bin as a feature
  - Using windows
  - No windowing
- Extracting domain-specific features
  - Using windows
  - No windowing

1) Binned Features With Windows: To get binning features of data, the whole process is shown as below:

- We first cut each sample into windows on the time-domain. Specifically, we set window length as $frequency * 4$, it means every 4 seconds as a window. The percentage of windows' overlap is set as 50%. For each sample, we only use 12 windows.
- For each window, we apply the FFT to get windows' spectrum information. To be specific, the window size for spectrum is set to FFT_SIZE, the overlap of window is set to 25%.
- After obtaining the sepctrum information, we apply binning to every window with shape $(num\_freq\_bins, num\_time\_bins)$. Then reshaping the initial shape to shape$(num_freq_bins * num_time_bins,)$ and taking this vector as the feature of one window.
- When we get windows' features, we concatenate them as the sample's feature.

2) Binned Features Without Windows:

- We directly apply FFT to the whole data to get the spectrum information. The window size and overlap is similar to the former one.
- Then, we apply binning method to the spectrum with another shape $(num\_freq\_bins, num\_time\_bins)$. Then reshaping the initial shape to shape$(num_freq_bins * num_time_bins,)$ and taking this vector as the feature of samples.

3) Domain-Specific Features With Windows: To get domain-specific features through windows, the process is shown as follows:

- The first two processes are similar to the first type. We also set window length as FFT_SIZE, the overlap of window is set to 50% to cut data into several windows. Then applying the FFT to get spectrum information.
- Instead of manually selecting features, we utilize the mean of MFCC (Mel-frequency Cepstral Coefficient) as the window feature. It is commonly derived as follows [1]:
  - Take the Fourier transform of (a windowed excerpt of) a signal.
  - Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows or alternatively, cosine overlapping windows.
  - Take the logs of the powers at each of the mel frequencies.
  - Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
  - The MFCCs are the amplitudes of the resulting spectrum
- Then reshaping the initial shape and taking this vector as the feature of samples.

4) Domain-Specific Features Without Windows: For domain-specific features without windows, we directly apply MFCC to the spectrum data and use mean as data features.

C. Feature normalization

After obtaining the features of data, we find the scale between features or different classes is very large, thus we utilize StandardScaler to normalize the data.

D. Machine Learning Models for Classification

We use Random Forest Classifier as the machine learning model. It is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. We also tried SVC with linear with rgb kernel but find the performance is not good. Compared to the SVC, the random forest classifier combines multiple classifiers and average them. Thus improve the performance. Specifically, we use 10 folder cross-validation to train and show its performance on the test data.

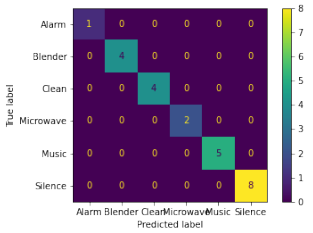
IV. Real-Time Predictions

After we getting the features and training the models, we further apply the trained models and leverage pyaudio packages to predict real-time audios. You can see detailed videos from the YouTube .

V. Experimental Results

A. Classification Results

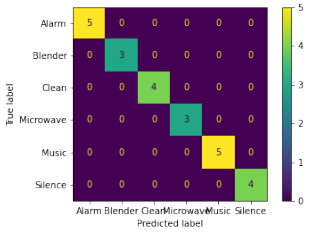
We split our dataset into training data (100 samples) and testing data (20 samples). After training four type of models for six classes, we test the different feature-models' performance on the testing data. Fig.3 shows the results of different classifiers with different feature sets. The left column is confusion matrix and the right column is the classification result in terms of precision, recall and F1 score. From Fig.3 we can observe that the performance of all four models reaches



Test Statistics:

	precision	recall	f1-score	support
Alarm	1.00	1.00	1.00	1
Blender	1.00	1.00	1.00	4
Clean	1.00	1.00	1.00	4
Microwave	1.00	1.00	1.00	2
Music	1.00	1.00	1.00	5
Silence	1.00	1.00	1.00	8
accuracy			1.00	24
macro avg	1.00	1.00	1.00	24
weighted avg	1.00	1.00	1.00	24

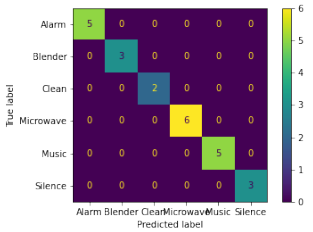
(a) Binning features with windows



Test Statistics:

	precision	recall	f1-score	support
Alarm	1.00	1.00	1.00	5
Blender	1.00	1.00	1.00	3
Clean	1.00	1.00	1.00	4
Microwave	1.00	1.00	1.00	3
Music	1.00	1.00	1.00	5
Silence	1.00	1.00	1.00	4
accuracy			1.00	24
macro avg	1.00	1.00	1.00	24
weighted avg	1.00	1.00	1.00	24

(b) Binning features without windows



Test Statistics:

	precision	recall	f1-score	support
Alarm	1.00	1.00	1.00	5
Blender	1.00	1.00	1.00	3
Clean	1.00	1.00	1.00	2
Microwave	1.00	1.00	1.00	6
Music	1.00	1.00	1.00	5
Silence	1.00	1.00	1.00	3
accuracy			1.00	24
macro avg	1.00	1.00	1.00	24
weighted avg	1.00	1.00	1.00	24

(c) Domain features with windows

Test Statistics:

	precision	recall	f1-score	support
Alarm	1.00	1.00	1.00	6
Blender	1.00	1.00	1.00	5
Clean	1.00	1.00	1.00	3
Microwave	1.00	1.00	1.00	3
Music	1.00	1.00	1.00	4
Silence	1.00	1.00	1.00	3
accuracy			1.00	24
macro avg	1.00	1.00	1.00	24
weighted avg	1.00	1.00	1.00	24

(d) Domain features without windows

Fig. 3. Classification confusion matrix and results of six classes.

to 100% (F1) on six classes. It is because these six type of data have distinct difference, you can see from Fig 1. Besides, the feature we extract and the selected model also help us get the good performance. Compared to single classifier, the random forest classifier is an ensemble method.

B. Feature Selection

In this section, we take domain-specific model with windows as an example to illustrate the process. You can see the code at this link. The python file is named Feature_Selection. And the result for different features can be shown as Fig 4. Thus, from the picture we can see that the mfcc feature is the best. Therefore, we apply it to train the domain-specific model. Furthermore, we can also use the combination of median, $1/4$ and $3/4$ quantile.

Feature Type								
max_index min_index	+							
max_per_window min_per_window		+						+
sum_per_window std_per_window			+				+	
mean_per_window				+			+	
median_per_window ¼ quan_per_window ¾ quan_per_window					+			+
mfcc						+		
Training Accuracy	91.67	96.89	93.89	82.56	98.89	1.0	91.89	98.89
Testing Accuracy	95.83	95.83	95.83	91.67	1.0	1.0	95.83	1

Fig. 4. Feature Selection Results.

REFERENCES

[1] Sahidullah, Md.; Saha, Goutam (May 2012). "Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition". Speech Communication. 54 (4): 543–565. doi:10.1016/j.specom.2011.11.004.

[2] Tauhidur Rahman, Alexander T. Adams, Mi Zhang, Erin Cherry, Bobby Zhou, Huaishu Peng, and Tanzeem Choudhury. 2014. BodyBeat: a mobile system for sensing non-speech body sounds. In Proceedings of the 12th annual international conference on Mobile systems, applications, and services (MobiSys '14). Association for Computing Machinery, New York, NY, USA, 2–13. https://doi.org/10.1145/2594368.2594386