SGN-24007: Advanced Audio Processing

Exercise 1

15.03.2019

This exercise contains two parts, in part 1 you will learn to implement some basic feature extraction methods - zero crossing rate (ZCR), energy spectral density (ESD) and mel-band energy (MBE) for audio signals. In part 2, you will go through the work flow of implementing a speech vs music classification system.

Part I

1 Zero crossing rate (ZCR)

Zero crossing rate is the rate at which the sign of the signal changes. It simply states the number of times the audio signal crossed zero level. For example, in the signals A = [1, 2, -2, -1] and B = [2, -2, 2] the ZCR is equal to 1 and 2 respectively.

2 Energy spectral density (ESD)

The energy spectral density is computed as $ESD = |FFT(x)|^2$, where x is the audio signal, $FFT()$ is the fast Fourier transform function and $|.|$ is the absolute function.

3 Mel-band energy (MBE)

The mel-band energy is computed by mapping the ESD to mel-scale. We can perform this by pre-computing the mapping matrix (M) which maps each of the frequencies in ESD to a mel-band in MBE. The MBE can now be computed as $MBE = dot(ESD, M)$ where $dot()$ is the matrix multiplication operator.

4 Python implementation (1 point)

The provided python script *ex1.py* is supposed to extract the above three features and visualize them for a given audio signal. Before solving the tasks read

through the script and understand the work flow. Your task in this exercise is to implement the above explained ZCR, ESD, and MBE. Currently the script has dummy-codes for these features, which specifies the exact dimensions the correct implementation should have. Replace this dummy code with your implementation.

5 Discussions (0.5 point)

After successfully completing and executing the script with $ex1.wav$ audio, the code will plot the Figure 1. In the Figure, explain the characteristics of the ZCR curve with respect to the audio signal content. Specifically, what is happening in the first 40 frames ? What is happening in between frames 40 to 70 ? What is the difference between 0-40 frames and 70-120 frames ?

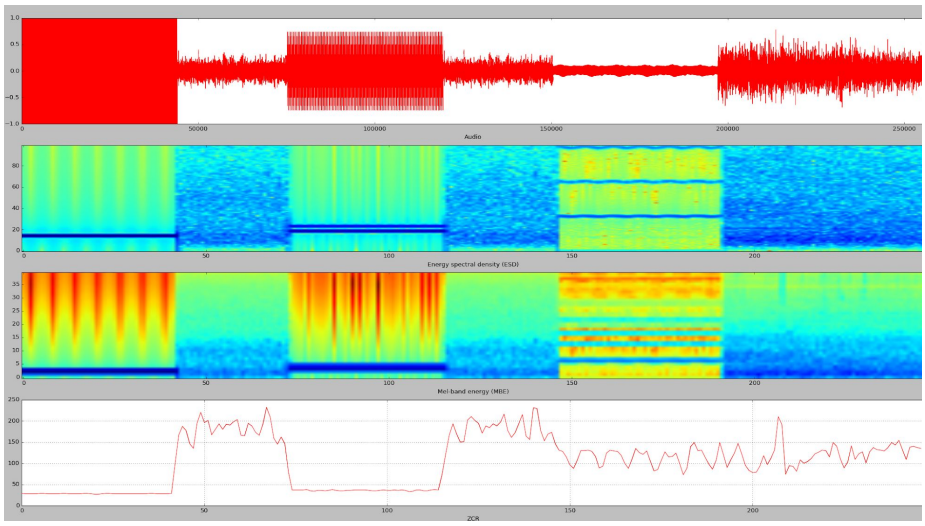

Figure 1: Input Audio, ESD, MBE, ZCR visualized.

Part II

In part 2 you will go through the work flow of implementing a speech vs music classification system. The input to the classifier is the mel-band energy (MBE) feature you implemented in the first exercise. In this class you will tune the parameters of MBE feature with respect to a fixed convolutional neural network (CNN) classifier. Specifically you will be tuning the number of mel-bands, the duration of audio and the window length to identify the best parameters for the given CNN classifier.

Two python scripts are provided in this exercise. The *extract_features.py* script creates training and testing splits from the data, extracts the MBE feature (you have to fill in the the code you developed for part 1), normalizes the features and saves them to a file. The *classify_speech_vs_music.py* script reads the normalized features from previous script and trains the network. The training is stopped when the accuracy of the test data does not change for long.

For this task we will be using the speech and music dataset from George Tzanetakis available in this link - `http://opihi.cs.uvic.ca/sound/music_speech.tar.gz`. Download and unzip the data. Provide the path of the extracted 'speech_wav' and 'music_wav' folders in the *extract_features.py* script.

Read the scripts and understand the work flow. Specifically in *extract_features.py* script, see

1. How the training and testing split is created? How the labels are generated?
2. What is the window length, number of mel-bands and the duration of audio used?
3. How the normalization is done?
4. Run the code and visualize the MBE features for speech and music, do you observe any patterns?

In *classify_speech_vs_music.py* script see

1. How the normalized data is split into sequences for feeding into the CNN?
2. What is the model specification? Can you draw a simple block diagram of this?
3. Check the model definition functions and read about them, understand what each parameter refers to.
4. How is the early stopping implemented?

Once you understand the code, start tuning the MBE feature.

6 Tune the length of audio (0.5 points)

Use the *extract_features.py* script to extract MBE features for different length of audio. Keep the *window_length* parameter fixed at 2048 and the *nb_mel_bands* parameter fixed at 32. Run the script three times for three different values of nb_frames = [40, 80, 160]. Which is approximately equal to [0.93, 1.86, 3.72] seconds of audio. Now use the same parameters in the *classify_speech_vs_music.py* script to train the CNN classifier with the three different lengths of audio and report the best duration (nb_frames).

7 Tune the number of Mel-bands (0.5 points)

Use the *extract_features.py* script to extract MBE features for different length of audio. Keep the *window_length* parameter fixed at 2048 and the *nb_frames* parameter fixed at 80. Run the script three times for three different values of *nb_mel_bands* = [32, 64, 128]. Now use the same parameters in the *classify_speech_vs_music.py* script to train the CNN classifier with the three different number of Mel-bands and report the best number of Mel-bands (*nb_mel_bands*).

8 Tune the window length (0.5 points)

Use the *extract_features.py* script to extract MBE features for different length of audio. Keep the *nb_mel_bands* parameter fixed at 128 and run the script three times with different combinations of the *nb_frames* and *window_length* parameters (80, 2048), (160, 1024), and (320, 512). Now use the same parameters in the *classify_speech_vs_music.py* script to train the CNN classifier and report the best combination of the *nb_frames* and *window_length* parameters. Why do we increase the *nb_frames* for shorter *window_length*?