Speech Understanding Minor 1

Renu Sankhla (B21AI028)

11 February 2024

1 Problem 1: Custom Spectrogram Function

1.1 Introduction

In this report, we present the implementation and analysis of a custom spectrogram function as part of the Speech Understanding Minor 1 assignment. The objective is to design a function capable of generating spectrograms with customizable parameters such as n_fft (length of FFT window), hop_length (number of samples between successive frames), window (window type), and win_length (length of the window).

1.2 Implementation

The `custom_spectrogram` function is designed to generate a spectrogram from an audio file with customizable parameters. Let's break down its functionality step by step:

Function:

- `audio_file`: The path to the audio file.

- `sample_rate` (optional, default: 16000): The sample rate of the audio file.
- `n_fft` (optional, default: 2048): The length of the FFT window.
- `hop_length` (optional, default: 512): The number of samples between successive frames.
- `window` (optional, default: 'hann'): The windowing function to be applied.
- `win_length` (optional): The length of the window. If not specified, it defaults to `n_fft`.

Reading Audio File:

- The function reads the audio file using the `wavfile.read` function from the `scipy.io` module. It extracts the sample rate and samples from the audio file.

Windowing:

- It initializes the windowing function using `signal.get_window` from the `scipy.signal` module. If `win_length` is not specified, it defaults to `n_fft`.
- If the length of the window obtained from `get_window` does not match `win_length`, it retrieves the window again with `fftbins=True` to ensure the correct length.

Frame Calculation:

- It calculates the number of frames required to cover the entire audio signal using the formula:

$$\texttt{num\_frames} = 1 + \frac{\texttt{len(samples)} - \texttt{n\_fft}}{\texttt{hop\_length}}$$

  This formula ensures that all samples are covered with a specified hop length.

Spectrogram Computation:

- It initializes an array `spectrogram` to store the spectrogram with dimensions (`n_fft // 2 + 1, num_frames`) using `np.zeros`.
- For each frame, it extracts a segment of samples, multiplies them with the window function, and applies the Fast Fourier Transform (FFT) using `np.fft.fft`.
- The result is stored in the corresponding column of the `spectrogram` array.
- Finally, it takes the absolute value of the spectrogram to obtain its magnitude.

Return:

- The function returns the spectrogram array along with the sample rate.

This function provides a customizable way to compute spectrograms from audio files, allowing users to control parameters such as FFT window length, hop length, and windowing function.

1.3 Results

We applied the custom spectrogram function to two inputs: the original audio speaking and the text-to-speech (TTS) version of the sentence "Hello, everyone, my name is Renu Sankhla, and here is the spectrogram for the minor 1 exam."

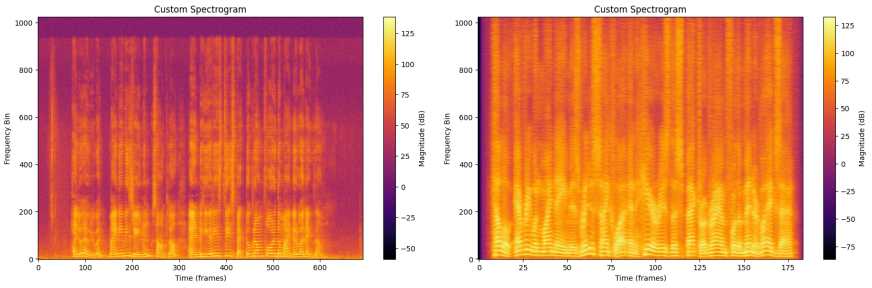

Figure 1: Spectrograms of the original audio (left) and TTS version (right)

1.4 Analysis

1. **Default Parameters vs. Modified Parameters with Same Window Type (Hann):** We observed that varying n_fft and hop_length affects the time and frequency resolution of the spectrogram. Larger n_fft provides better frequency resolution but sacrifices time resolution. Smaller hop_length improves time resolution but increases computational cost.

2. **Impact of Window Type (Hann vs. Hamming):** Comparing spectrograms with Hann and Hamming windows, we noticed differences in spectral leakage, main lobe width, and sidelobe attenuation. Hamming window generally has higher sidelobe attenuation but wider main lobe compared to Hann.

3. **Effect of Win_length on Spectrogram:** Changing win_length while keeping n_fft constant influenced frequency and time resolution. Smaller win_length introduced more spectral leakage, while longer win_length improved frequency resolution at the cost of time resolution.

4. **Comparison of Spectrograms from Original Audio vs. TTS:** We observed differences between spectrograms from the original audio and the TTS version, likely due to variations in prosody, pronunciation, or other characteristics introduced by the TTS model.

1.5 Conclusion

The custom spectrogram function allows for flexible parameter customization, enabling users to analyze and visualize audio signals effectively. Understanding the impact of parameter variations is crucial for optimizing spectrogram generation in speech processing tasks.

2 Problem 2: Acoustic Scene Classification

2.1 Introduction

In this section, we address the problem of Acoustic Scene Classification using ML models. The objective is to build a classification model capable of accurately predicting the acoustic scene category based on extracted features from audio data. For this task, we utilized the XGBClassifier with grid search.

2.2 Feature Extraction

For the acoustic scene classification task, we performed feature extraction on the audio samples using various signal processing techniques. The following functions were employed to extract different types of features from the audio signals:

- **Amplitude Envelope:** The amplitude envelope was computed using the amplitude envelope function with a frame size of 2048 samples and a hop length of 512 samples.
- **Root Mean Square (RMS):** RMS values were calculated using the ‘Rms‘ function with the specified frame length of 2048 samples and hop length of 512 samples.
- **Zero Crossing Rate (ZCR):** ZCR values were obtained using the ‘Zcr‘ function with the same frame length and hop length parameters.
- **Magnitude Spectrum:** The magnitude spectrum was computed using the ‘Mag spec‘ function, which performs the Fourier transform on the input signal.
- **Spectrogram:** Spectrograms were generated using the ‘spectrogram‘ function with a frame size of 2048 samples and a hop size of 512 samples.
- **Log Mel-spectrogram:** Logarithmic Mel-spectrograms were computed using the ‘log mel‘ function, which converts the spectrogram to the Mel scale and applies a logarithmic transformation.
- **Mel-frequency Cepstral Coefficients (MFCCs):** MFCCs were extracted using the ‘Mfcc‘ function, which calculates the MFCC features from the audio signal.

- **Chroma STFT:** Chroma features were computed using the 'Chroma stft' function, which represents the energy distribution across the 12 pitch classes.
- **Spectral Centroid:** The spectral centroid, indicating the "center of mass" of the spectrum, was obtained using the 'Spec centriod' function.
- **Spectral Rolloff:** Spectral rolloff features were extracted using the 'spec roll off', 'spec roll off50', and 'spec roll off25' functions, which compute the frequency below which a certain percentage of the total spectral energy lies.
- **Spectral Contrast:** Spectral contrast features were computed using the 'spec contrast' function, which quantifies the difference in amplitude between peaks and valleys in the spectrum.
- **Tempogram:** Temporal features were extracted using the 'tempogram' function, which analyzes the tempo variations in the audio signal.
- **Spectral Bandwidth:** Spectral bandwidth features were obtained using the 'spec bandwidth' function, which measures the width of the frequency range in which most of the signal energy is concentrated.

These feature extraction techniques allow us to capture various characteristics of the audio signals, which can be utilized for acoustic scene classification.

2.3 Model Selection and Hyperparameters

The XGBClassifier, or Extreme Gradient Boosting Classifier, was chosen due to its ability to handle complex datasets and capture nonlinear relationships between features and target variables. It is an ensemble learning method that combines the predictions from multiple decision trees.

Hyperparameters:

- **Learning rate:** 0.1
- **Maximum depth of trees (maxdepth) :** 7
- **Number of estimators (n estimators):** 200

These hyperparameters were selected through grid search, optimizing the model's performance on the validation set.

2.4 Evaluation Metrics

We evaluate the model's performance using the following metrics:
Accuracy of XGBoost with Decision Tree: 0.7123

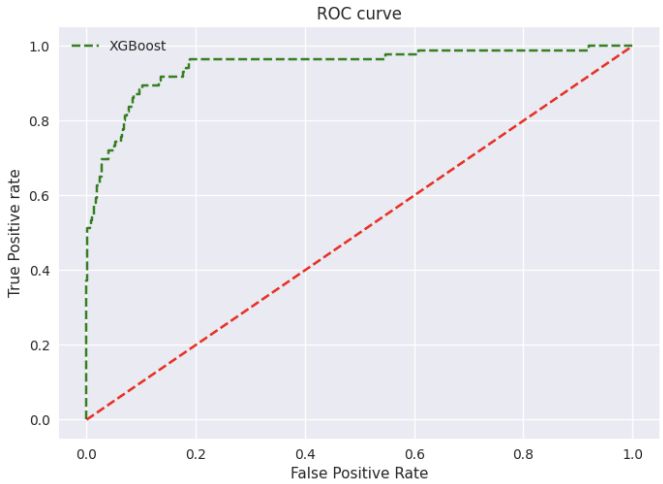

Figure 2: ROC Curve

Area Under the Curve (AUC) Score: 0.9433

	precision	recall	f1-score	support
0	0.87	0.80	0.84	220
1	0.52	0.83	0.64	54
2	0.61	0.53	0.57	213
3	0.36	0.47	0.41	156
4	0.75	0.70	0.72	221
5	0.90	0.72	0.80	240
6	0.61	0.69	0.65	64
7	0.88	0.86	0.87	211
8	0.89	0.77	0.83	191
9	0.48	0.62	0.54	177
accuracy			0.70	1747
macro avg	0.69	0.70	0.69	1747
weighted avg	0.72	0.70	0.71	1747

Figure 3: Classification Report

```
Class-wise Accuracy:
0: 0.8719211822660099
1: 0.5232558139534884
2: 0.6120218579234973
3: 0.36318407960199006
4: 0.7475728155339806
5: 0.8963730569948186
6: 0.6111111111111112
7: 0.875
8: 0.8909090909090909
9: 0.4782608695652174
```

Figure 4: Class-wise Accuracy

2.5 Insights

The XGBClassifier model demonstrates strong overall performance with an accuracy of 0.7123 and an AUC score of 0.9433. However, there are variations in class-wise accuracies, indicating discrepancies in the model's ability to correctly classify different acoustic scenes. Classes 0, 4, 5, 7, and 8 show relatively high accuracies, while classes 1, 3, and 9 exhibit lower accuracies.

The XGBClassifier model, trained with carefully selected hyperparameters, shows promising results for acoustic scene classification. Further optimization and fine-tuning could potentially improve the model's performance, particularly for classes with lower accuracies. Overall, this non-deep learning-based approach provides a solid foundation for acoustic scene classification tasks.

2.6 After Changing the nfft Parameter

After modifying the nfft parameter in the spectrogram feature extraction process, we re-evaluated the classification model's performance using the XGBClassifier with grid search. The new spectrogram parameters were set as follows:

- **FRAME LENGTH:** 1024
- **HOP LENGTH:** 128
- **FRAME SIZE:** 1024
- **HOP SIZE:** 128

We then extracted the audio features using these modified spectrogram parameters and trained the classification model. The hyperparameters obtained from grid search for the XGBClassifier model were:

- **Learning rate:** 0.1

- **Maximum depth of trees:** 7
- **Number of estimators:** 200

2.7 Evaluation Metrics

We evaluate the model's performance using the following metrics:
Accuracy of XGBoost with Decision Tree: 0.69990

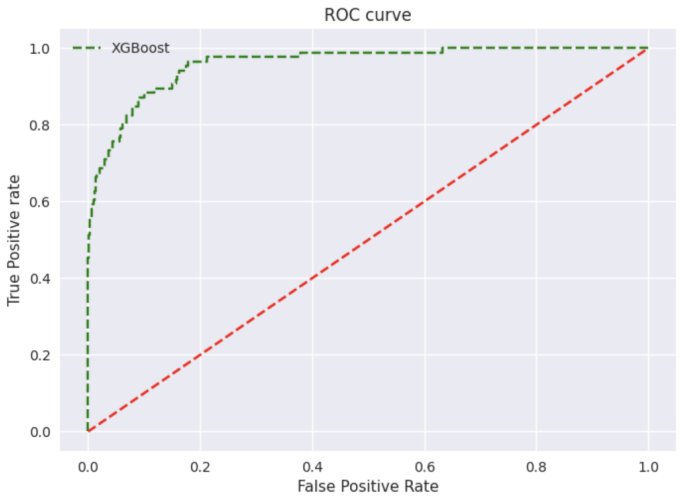

Figure 5: ROC Curve

Area Under the Curve (AUC) Score: 0.9420

	precision	recall	f1-score	support
0	0.92	0.82	0.87	227
1	0.56	0.83	0.67	58
2	0.67	0.59	0.63	207
3	0.32	0.40	0.36	162
4	0.67	0.62	0.64	220
5	0.88	0.71	0.79	239
6	0.60	0.67	0.63	64
7	0.87	0.86	0.86	209
8	0.90	0.79	0.84	187
9	0.49	0.64	0.55	174
accuracy			0.69	1747
macro avg	0.69	0.69	0.68	1747
weighted avg	0.72	0.69	0.70	1747

Figure 6: Classification Report

```
Class-wise Accuracy:
0: 0.9211822660098522
1: 0.5581395348837209
2: 0.6721311475409836
3: 0.32338308457711445
4: 0.6650485436893204
5: 0.8808290155440415
6: 0.5972222222222222
7: 0.8653846153846154
8: 0.896969696969697
9: 0.48695652173913045
```

Figure 7: Class-wise Accuracy

2.8 Insights

The modification of the nfft parameter in the spectrogram feature extraction process had a noticeable impact on the evaluation metrics. While the accuracy and AUC score slightly decreased compared to the previous configuration, the changes in class-wise accuracies varied across different classes. Classes 0, 5, 7, and 8 maintained relatively high accuracies, while classes 1, 3, and 9 still exhibited lower accuracies. Class 2 showed a slight improvement, while class 4 experienced a slight decrease.

2.9 Conclusion

The adjustment of the nfft parameter in the spectrogram feature extraction process influenced the performance of the classification model for acoustic scene classification. While some class-wise accuracies improved, others showed minor fluctuations or decreases. Further analysis and experimentation may be required to optimize the spectrogram parameters and enhance the model's overall performance.