Assignment 2

Anshul Thakur (B21CS85)

April 2, 2025

repository link: https://github.com/dr-ghost/Speech-Understanding-2.git

1 Question 1

I created functions to download VoxCeleb1, VoxCeleb2 Datasets from the Google Drive Links and created Dataset classes and other helper Functions in the `data.py` inside the `data/` directory. For Task1 and Task2 see `Task1&2.ipynb` for Task 3 see `Task3.ipynb` for Task 4 see `Task4.ipynb`

1.1 Task1 & Task2

1.1.1 Evaluation

Here I chose 'wavlm_base_plus' as a pretraind model, and use list_of_trial_pairs.txt (VoxCeleb1) for evaluation. All the helper functions and VoxCeleb1 datasets are defined in `evaluation.py`. After running evaluation here were the resulting metrics.

1. AUC: 0.9988
2. Accuracy: 0.8253
3. F1: 0.7883
4. Precision: 1.0000
5. Recall: 0.6506
6. EER: 1.51
7. TAR@1% FAR: 0.9777
8. Speaker Identification Accuracy: 0.9849

1.1.2 Finetuning

I then used `loratorch` to replace all linear layers in 'wavlm_base_plus' with `lora.Linear` layers, and froze the weights of all parameters except the lora matrices, all the implementation is present in `lora_finetune.py`. I used `pytorch-metric-learning` library for the ArcFace Loss, I made a training dataset from the first 100 identities (when sorted in ascending order) in the VoxCeleb2 Dataset and testing dataset from the remaining.
I also wrote the whole train function called `train_lora` in `lora_finetune.py`. for 10 epochs, with AdamW optimizer and the saved lora parameters are saved in 'models/wavlm_base_lora_finetune.pth'

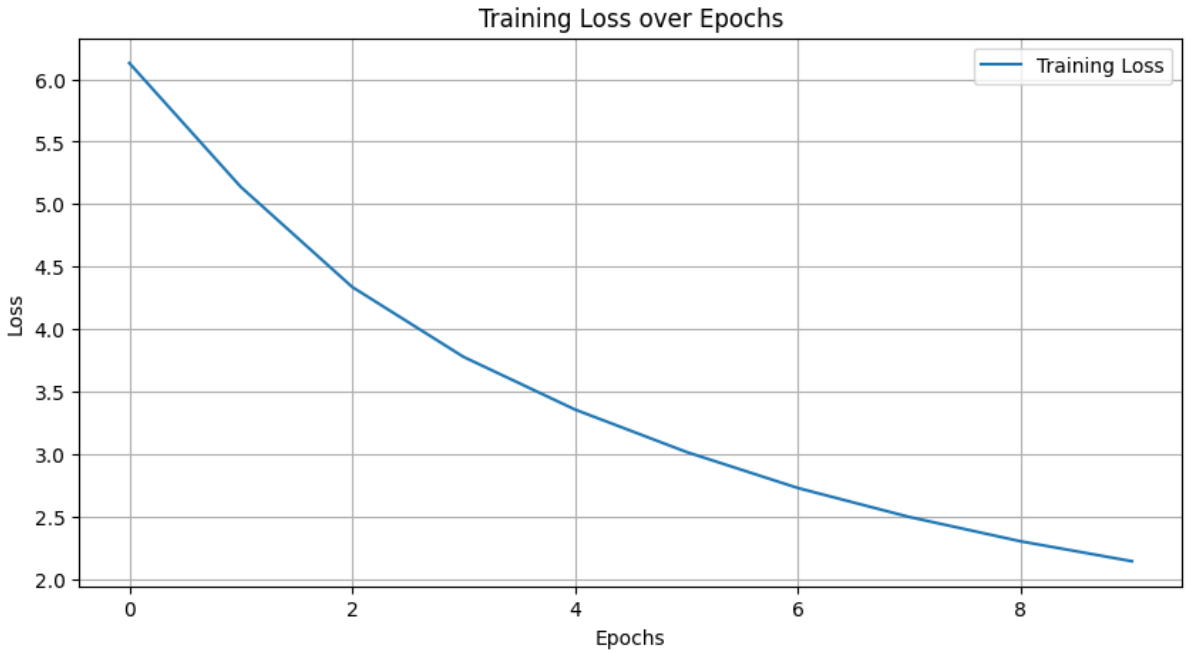

Figure 1: training loss vs epochs

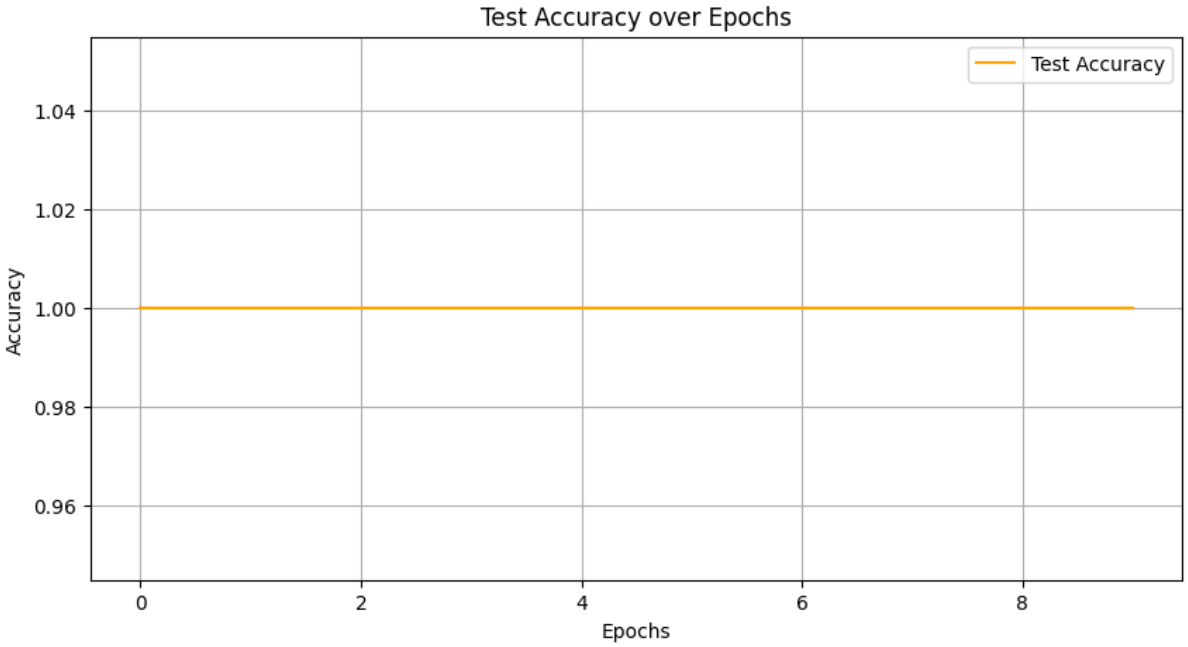

Figure 2: test accuracy vs epochs

Performance metrics on list_of_trial_pairs.txt (VoxCeleb1) of the LoRA fintuned with ArcFace Loss

1. AUC: 0.9845
2. Accuracy: 0.9231
3. F1: 0.9202
4. Precision: 0.9551

5. Recall: 0.8878
6. EER: 6.79
7. TAR@1% FAR: 0.7405
8. Speaker Identification Accuracy: 0.9321

Although metrics like Speaker Identification Accuracy and EER are worse than the pre-trained model, recall, subsequently f1 score and Accuracy are better.
The ArcFace loss is designed to enhance the discriminative power of the embeddings, which in this case improved recall and F1 score. However, this can sometimes come at the expense of metrics like EER and speaker identification accuracy, where the balance between false acceptance and false rejection rates is critical.

1.2 Task 3

1.2.1 3.A

I then created a mixture dataset called vox2_mix by running `mixture.py`. I used the first 50 identities to create training set in vox2_mix and the remaining 50 to create testing set.
The `mixture.py` script creates a multi-speaker mixture dataset from VoxCeleb2 by performing the following steps:

- **Argument Parsing:** Accepts paths for the VoxCeleb2 dataset and output directory, along with parameters such as sample rate, mix ratio, minimum utterance duration, and number of mixtures.
- **Metadata Creation:** Lists identity directories and recursively gathers all `.wav` files, mapping speaker IDs to their utterances.
- **Mixture Generation:** Randomly selects pairs of speakers, reads their audio files, and checks for minimum duration and matching sample rate. It then mixes one utterance into the other by overlapping a fraction (defined by the mix ratio) of the shorter utterance at a random offset.
- **Output Organization:** Saves the mixed audio and the original source files into designated directories for training and testing, and generates a metadata CSV file detailing the mixing process.

Now I initialize the sepformer model using helper functions in `sepformer.py` After that I use the sepformer model to calculate metrics Signal to Interference Ratio (SIR), Signal to Artefacts Ratio (SAR), Signal to Distortion Ratio (SDR) and Perceptual Evaluation of Speech Quality (PESQ) on the test set.

Key	SIR_MEAN	SIR_VARIANCE	SAR_MEAN	SAR_VARIANCE	SDR_MEAN	SDR_VARIANCE	PESQ_MEAN	PESQ_VARIANCE
id00154	10.521380	227.802958	-4.142282	18.451845	-8.231168	81.194939	1.110132	0.004038
id00866	6.617601	343.371110	-3.835985	26.908087	-10.557370	118.509879	1.121674	0.040492
id01066	8.324409	319.070619	-3.091958	26.173350	-8.877471	108.790636	1.117321	0.008584
id00419	0.870727	330.560557	-5.923366	48.326800	-14.635148	157.250794	1.071582	0.002424
id01892	-1.816690	254.610565	-8.194641	59.926393	-17.643697	88.353395	1.079743	0.008471

Figure 3: metrics statistics

1.2.2 3.B

Now I used the speaker verification model 'wavlm_base_plus' for identifying the speakers of mixed audio.
To do that for both the models, i had to first take a sample speech from each speaker from the dataset and then compute an embedding matrix.
After we separate and enhance the audios of the two speakers, I then compute the embeddings of the separated audios and then use cosine similarity between the embedding matrix and the predicted embedding to get which speaker the audio belongs to.

- Rank-1 accuracy (pretrained) : **0.7204**
- Rank-1 accuracy (finetuned) : **0.7961**

2 Question 2

All the tasks are completed in their respective .ipynb files for TaskA please see TaskA.ipynb, for TaskB please see TaskB.ipynb

2.1 Task A

2.1.1 Part 1 & 2

First, I made a script that downloaded The Audio Dataset with 10 Indian Languages from kaggle.
Then I created the dataset class IndianLanguagesDataset which automatically read audio files from the dataset and computed Mel-Frequency Cepstral Coefficients(MFCC) and also preprocessed the features (normalization).

2.1.2 Part 3

In this part I generated MFCC spectrograms for Hindi, Marathi, Bengali

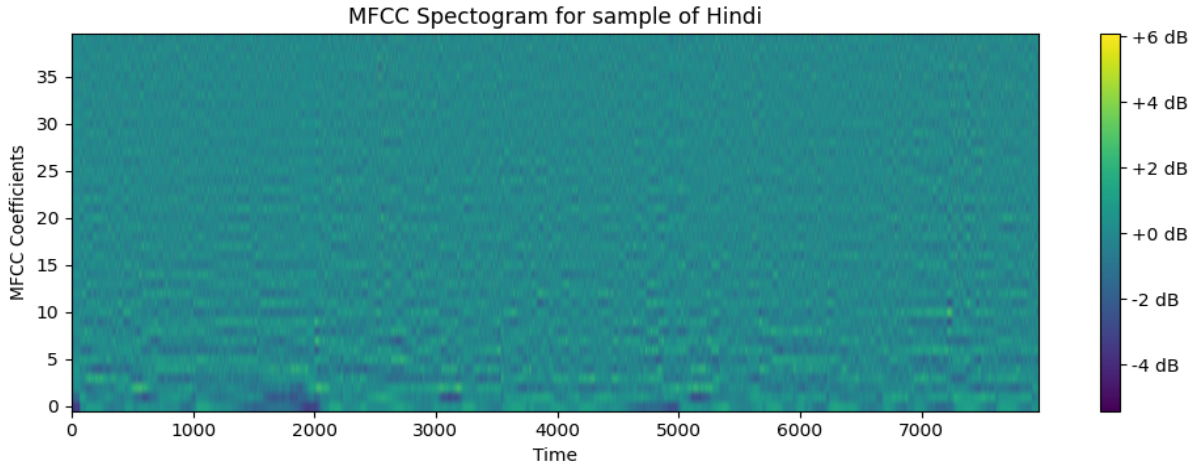

Figure 4: Hindi-1

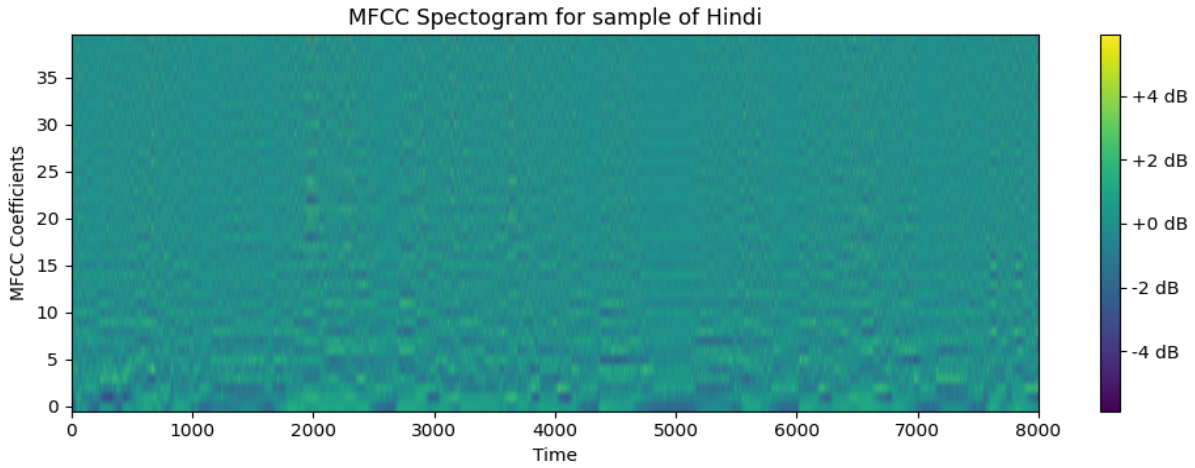

Figure 5: Hindi-2

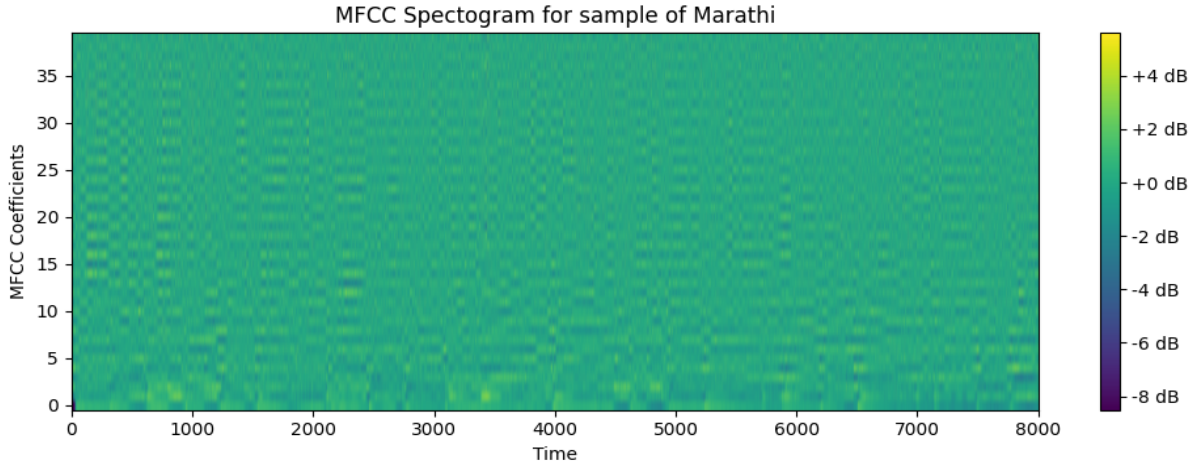

Figure 6: Marathi-1

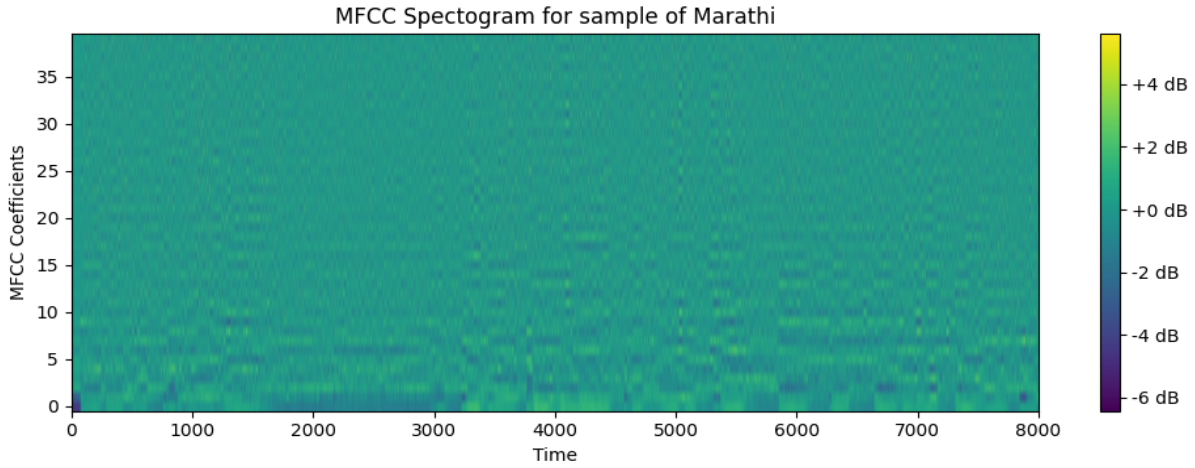

Figure 7: Marathi-2

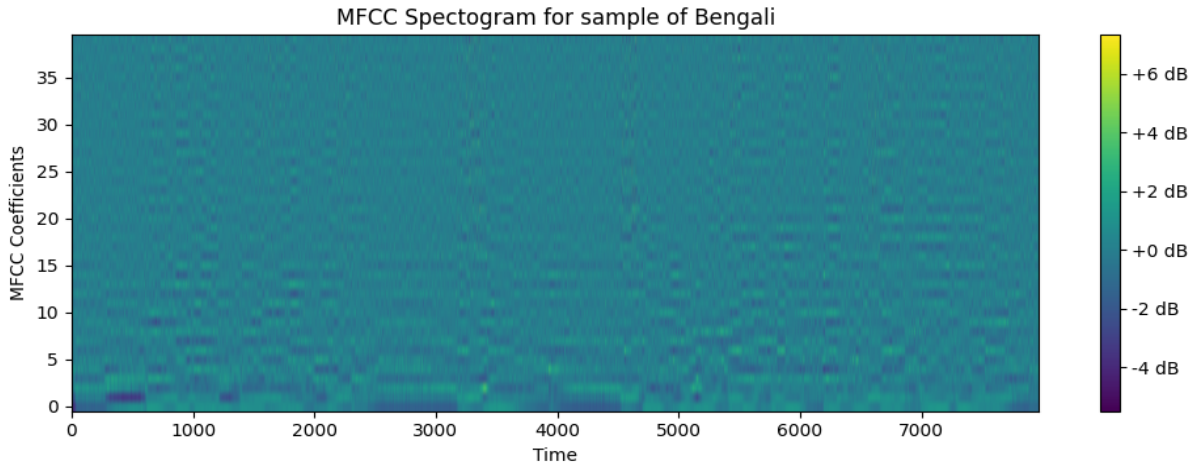

Figure 8: Bengali-1

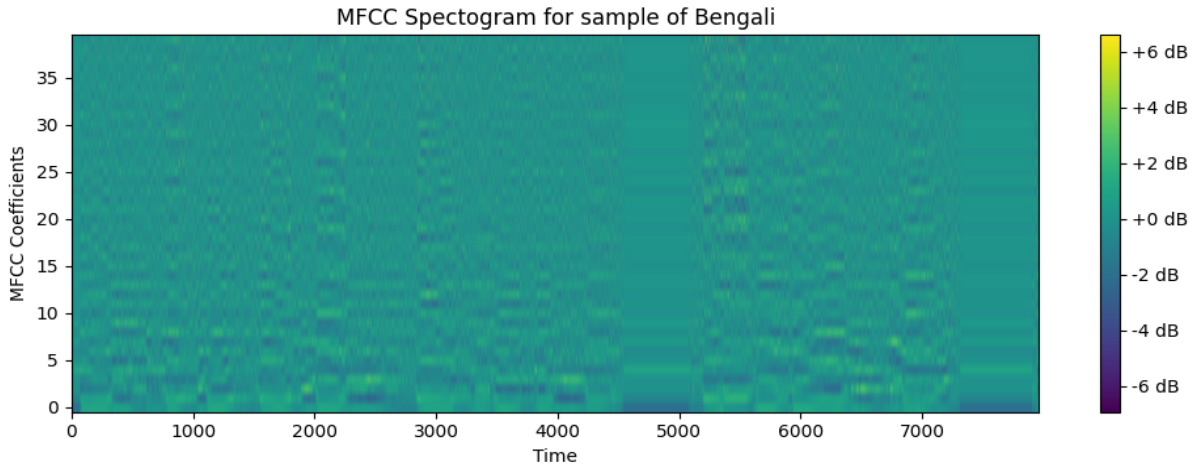

Figure 9: Bengali-2

2.1.3 Part 4

Below are all the MFCC spectrogram across different languages

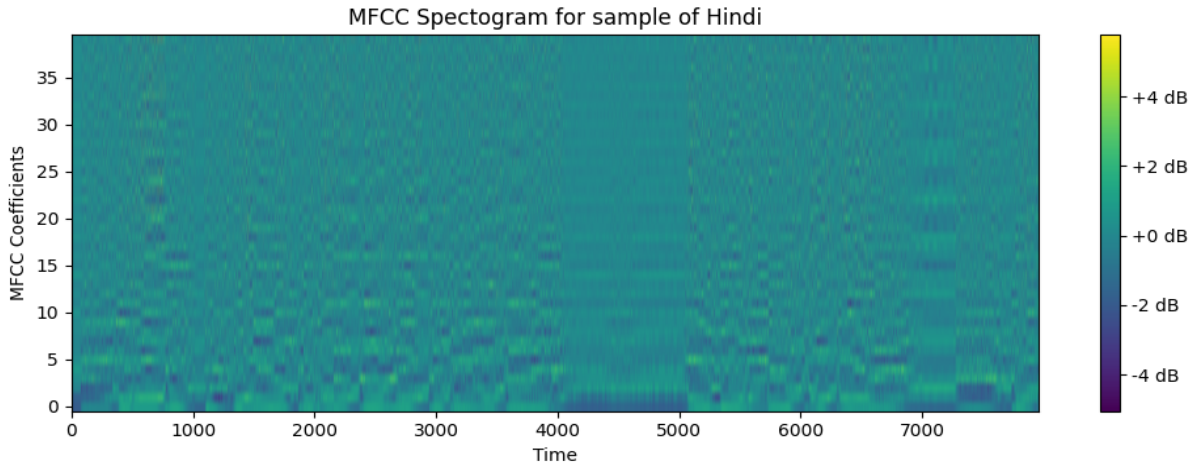

Figure 10: Hindi

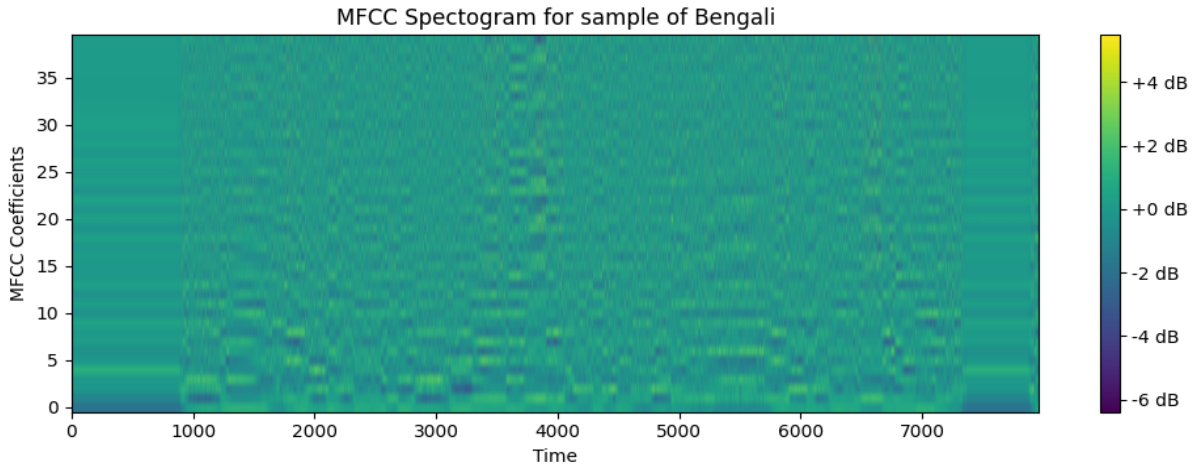

Figure 11: Bengali

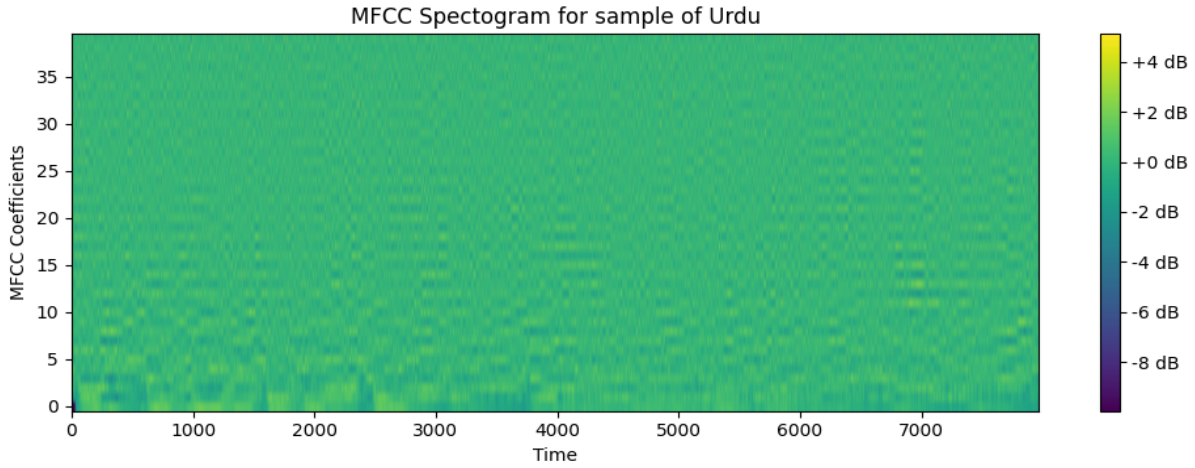

Figure 12: Urdu

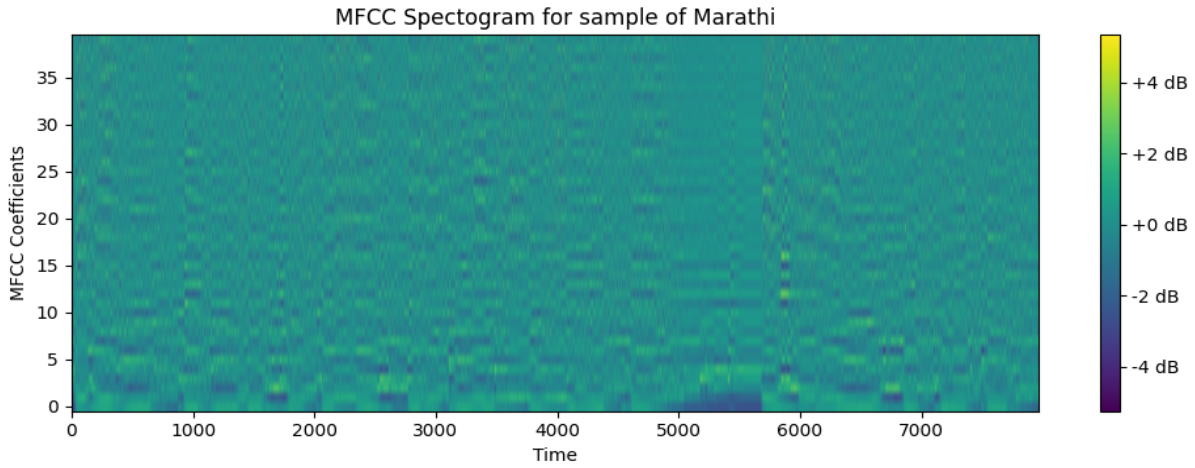

Figure 13: Marathi

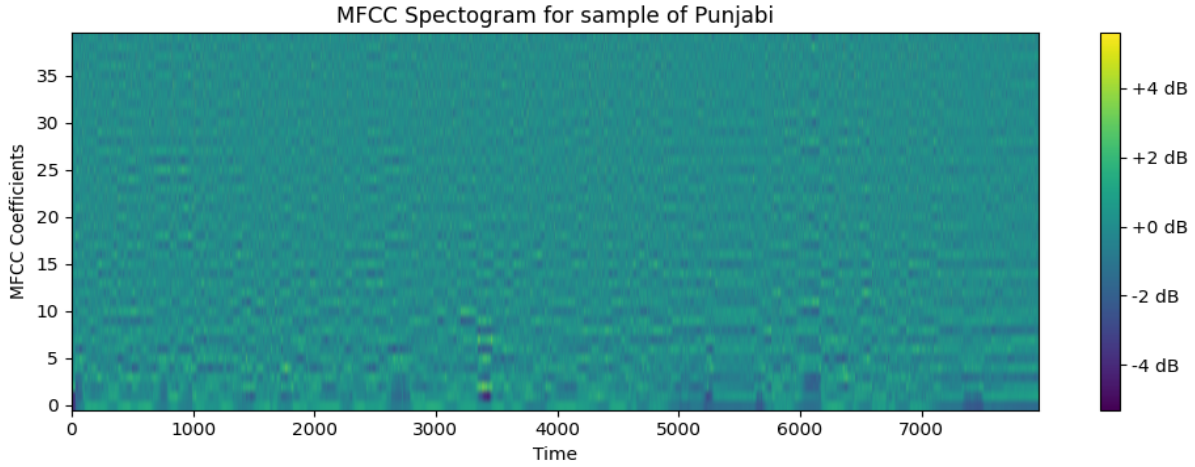

Figure 14: Punjabi

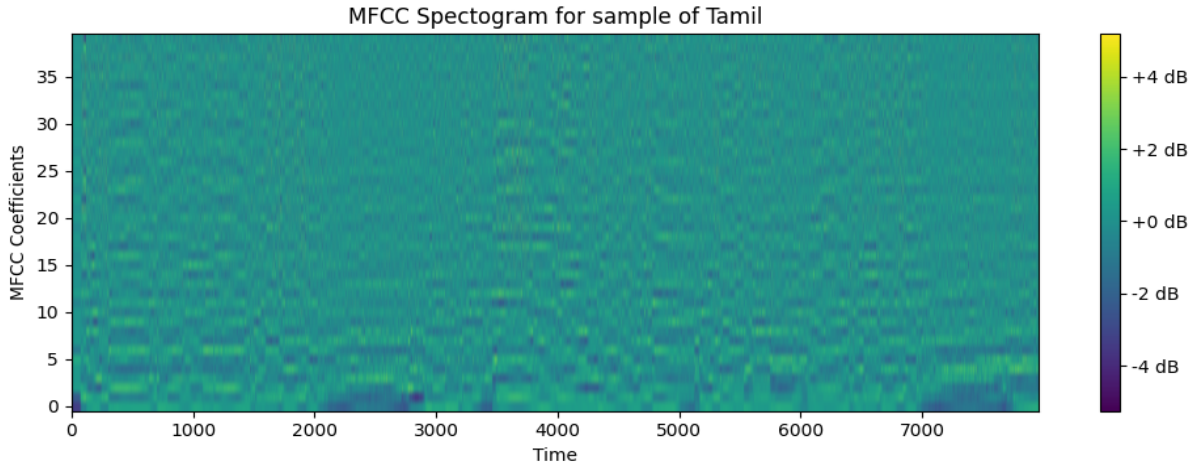

Figure 15: Tamil

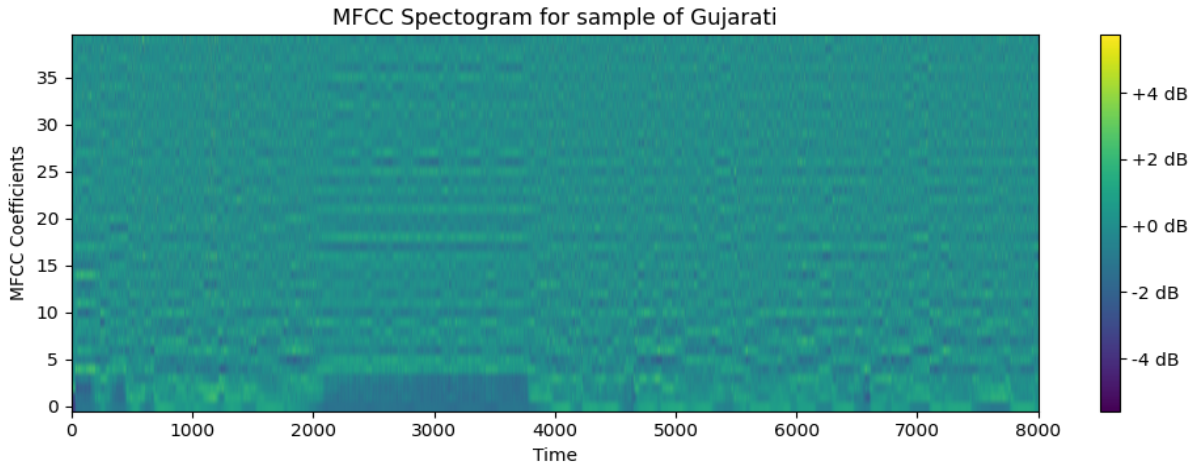

Figure 16: Gujarati

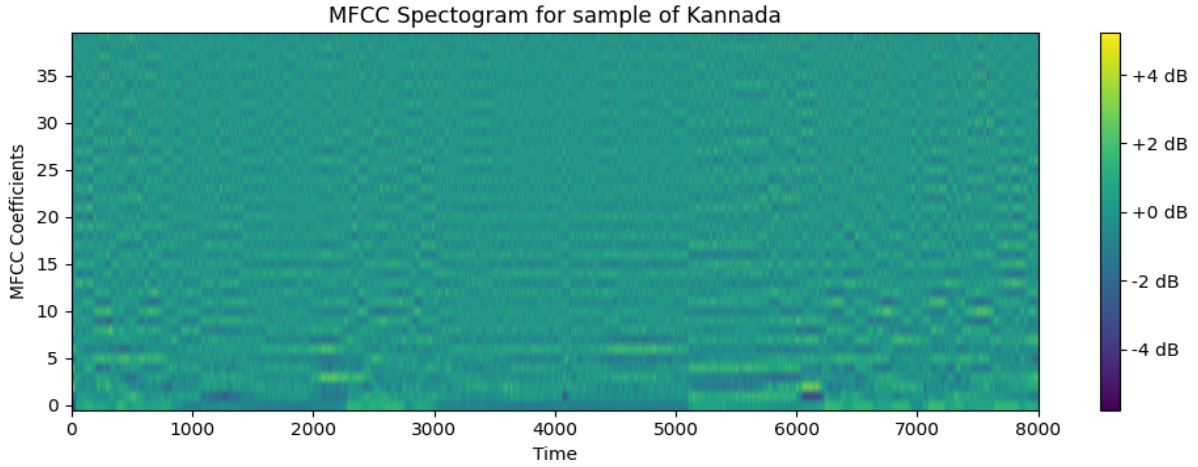

Figure 17: Kannada

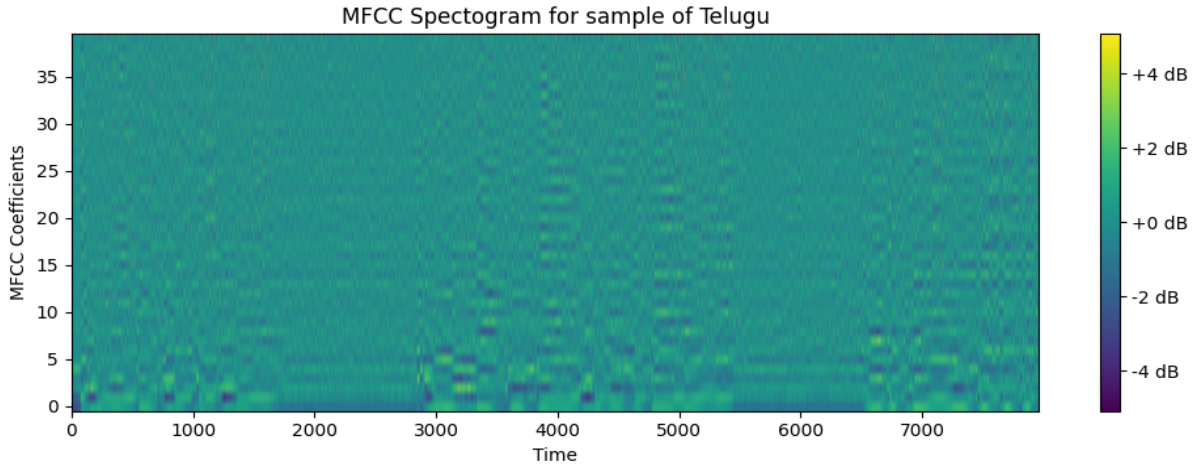

Figure 18: Telugu

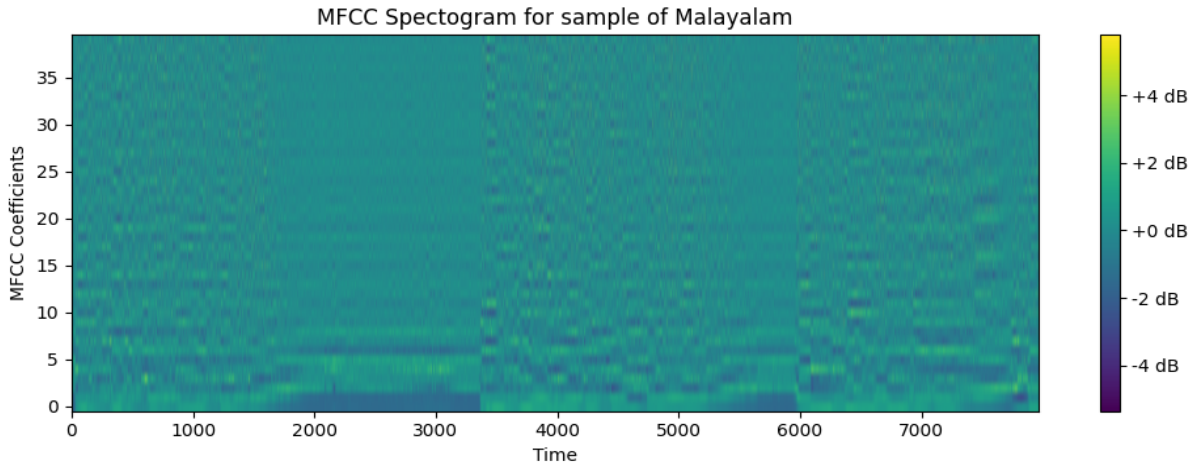

Figure 19: Malayalam

Across the set of languages, the MFCC spectrograms reveal both shared vocal characteristics and language-specific acoustic signatures. All languages show a common concentration of energy in the lower frequency bands—a reflection of the universal aspects of human speech production such as vowel resonance. However, the details differ in ways that mirror each language's phonetic and prosodic makeup. For example, the Hindi and Urdu spectrograms are quite similar overall, with both exhibiting prominent low-frequency energy; subtle differences emerge in their higher frequency bands, where Hindi shows slightly sharper transitions and more defined bursts, possibly reflecting a higher occurrence of aspirated or plosive consonants, whereas Urdu tends to display smoother, more gradual spectral transitions. In contrast, other languages in the set—such as those from different language families—present more distinct patterns. Urdu also has the most loudest features. and Malayalam has one of the softest features.

2.1.4 Part 4.a

The statistics of MFCC coeffs. are computed for each language in the .ipynb file

2.2 Task B

I chose Simple Neural Networks model to build the Language Classifier using the extracted MFCC features.
The features are already preprocessed in the dataset class (see `background.py`), in the train function I perform test-train split of the dataset.

2.2.1 Dataset and Feature Extraction

The dataset is organized into directories corresponding to different Indian languages. Each directory contains audio files in `.mp3` format. The `IndianLanguagesDataset` class (see `background.py`) handles the following:

- Loading of audio files.
- Resampling of audio to a uniform sampling rate (16 kHz).
- Preprocessing and normalization of the data.

2.2.2 MFCC Feature Extraction

MFCC features are extracted from the preprocessed audio signals using torchaudio's MFCC transformation. The configuration includes:

- 40 MFCC coefficients.
- A window length of 25 ms and a hop length of 10 ms.
- Instance normalization to ensure consistent feature scaling.

These steps convert the raw audio into a spectral representation that effectively captures the phonetic characteristics of speech.

2.2.3 Neural Network Architecture

The model structure is as follows:

1. **Convolution Block:** The input MFCC feature map is processed by a 1D convolutional layer with 64 filters, followed by a ReLU activation. A subsequent convolutional layer reduces the channel dimension to 1.
2. **Batch Normalization and Pooling:** Batch normalization is applied after the convolution block, and adaptive average pooling reduces the temporal dimension to a fixed length.
3. **Fully Connected Block:** The pooled features are flattened and fed into a series of fully connected layers which progressively reduce the dimensionality before the final classification layer outputs scores corresponding to each language.

2.2.4 Training Procedure

Within the `train` function, the dataset undergoes a train-test split using `train_test_split` from scikit-learn. A random subset of indices is selected to form the training and validation datasets. This ensures that the classifier is evaluated on unseen data to monitor its generalization capability.
The model is trained using the AdamW optimizer with a learning rate of 1×10^{-3}. The loss function used is cross-entropy loss, which is appropriate for multi-class classification tasks. The optimization process iteratively minimizes the loss over several epochs.

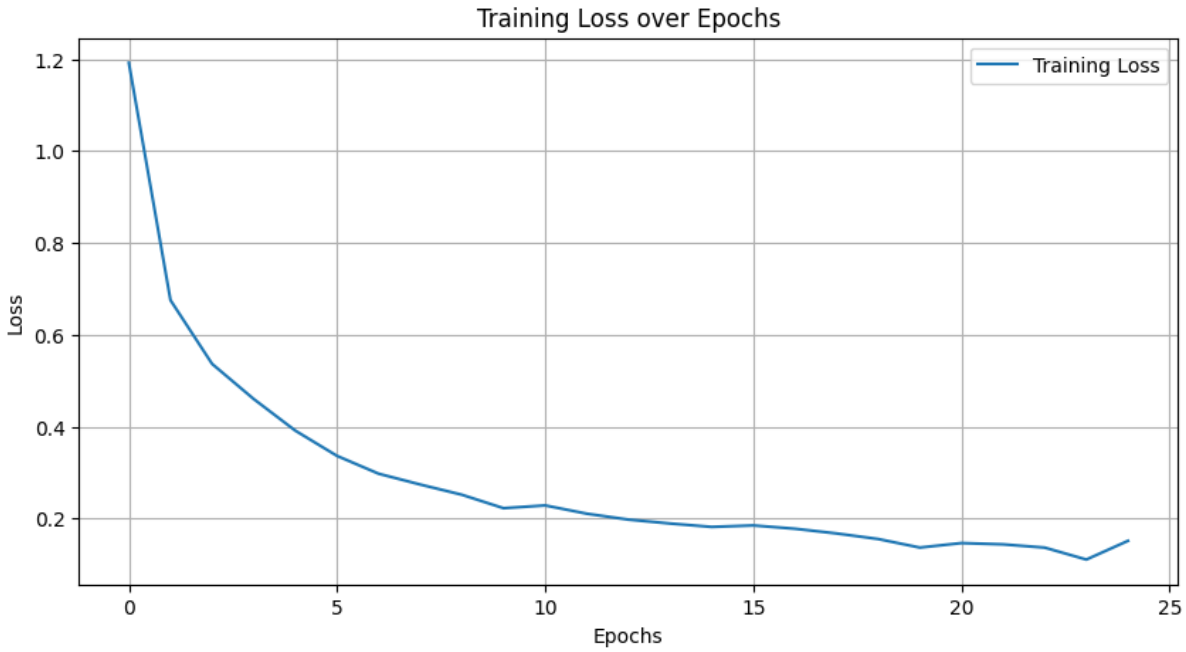

Figure 20: train-loss-plot

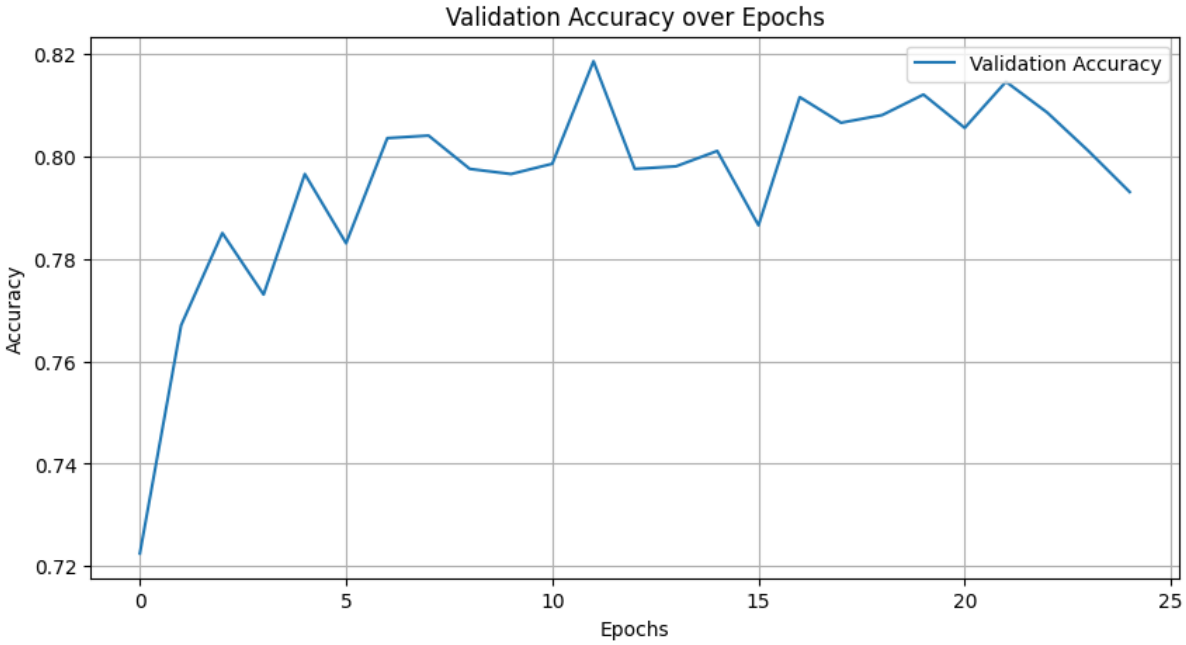

Figure 21: test-accuracy-plor

2.2.5 Evaluation Metrics

Performance of the language classifier is measured on test set using:

- **Accuracy:** The proportion of correctly classified samples.
- **Weighted F1 Score:** A balanced measure that takes into account both precision and recall, weighted by the class distribution.

2.2.6 Reflection of Acoustic Characteristics

The resulting MFCC features effectively capture various acoustic properties:

- **Vowel Resonance:** The low-frequency components in the MFCC representation predominantly capture vowel resonances, which are shaped by the vocal tract and are crucial in defining the tonal quality of speech.
- **Consonantal Dynamics:** Sharp transitions and bursts in the higher-frequency bands often indicate consonantal articulations. These features vary across languages based on phonetic inventories.
- **Prosodic Elements:** Temporal variations in MFCC coefficients can capture prosodic features such as intonation, rhythm, and stress patterns, which are vital in distinguishing the flow and cadence unique to each language.

2.2.7 Language-Specific Acoustic Signatures

Despite a shared foundation in speech production, languages can exhibit distinct MFCC patterns:

- **Energy Distribution:** While a common concentration of energy in lower frequencies is typical, differences in the distribution across frequency bands can indicate language-specific articulatory and phonetic features.
- **Temporal Dynamics:** Languages with rapid speech or distinctive rhythmic patterns may display more abrupt spectral transitions compared to languages with smoother, prolonged vowel sounds.
- **Spectral Complexity:** A richer phonemic inventory often leads to more varied spectral patterns, evident as greater variability in the MFCC coefficients.

2.2.8 Challenges in Using MFCCs for Language Differentiation

2.3 Speaker Variability

- **Physiological Differences:** Variations in vocal tract length, pitch, and overall speaking style can cause significant differences in MFCC representations, even among speakers of the same language.
- **Inter-Speaker Variability:** Differences in accent, age, and gender introduce additional variability, potentially leading to overlaps between classes.

In fact this was quite evident in my analysis as it was difficult to differentiate some laguages based on looking on the MFCC spectogram as they were similar and even if they had some differences, it was difficult to describe the differences

2.3.1 Background Noise

- **Environmental Interference:** Ambient noise can distort the spectral characteristics captured by MFCCs, making it harder to extract clean language-specific features.
- **Recording Conditions:** Variability in microphone quality and recording setups can introduce inconsistencies in the MFCC features, affecting the robustness of the classifier.

2.4 Regional Accents and Dialects

- **Accent Variability:** Regional accents can cause subtle shifts in pronunciation and intonation, leading to variations in the MFCC patterns that may not strictly correlate with the language label.
- **Dialectal Differences:** In multilingual regions, dialectal variations may blur the distinctions between languages, making it challenging for models relying solely on MFCCs to differentiate accurately.