EE6307 Project: Language Diarization

Ankit Saha - AI21BTECH11004
Pranav Balasubramanian - AI21BTECH11023

Problem Statement

- Language diarization is the process of automatically labeling and segmenting the languages in a speech signal that contains multiple languages
- Useful in multilingual settings, where speakers frequently code-switch between languages within a single conversation

Dataset Description

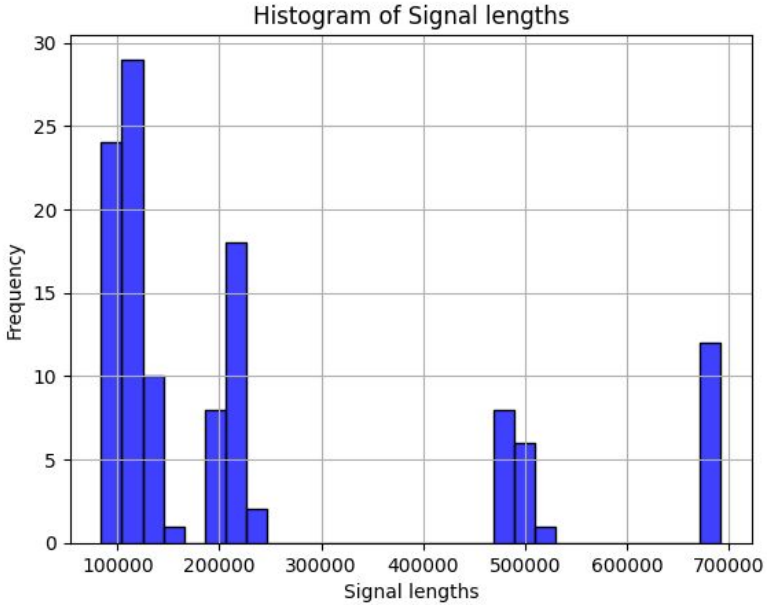
- It is a synthetically generated dataset built upon Single speaker same environment; L1L2 database, which is going through a PAN IIT project
- Speakers were asked to speak for 4-5 minutes in English, Hindi and their native language (Assamese, Bengali were chosen) by keeping the extempore topic as same as possible
- The utterances of each speaker is divided in different block sizes using the vowel end point detection algorithm, whose permutations are used to generate the speech signal
- Link to dataset

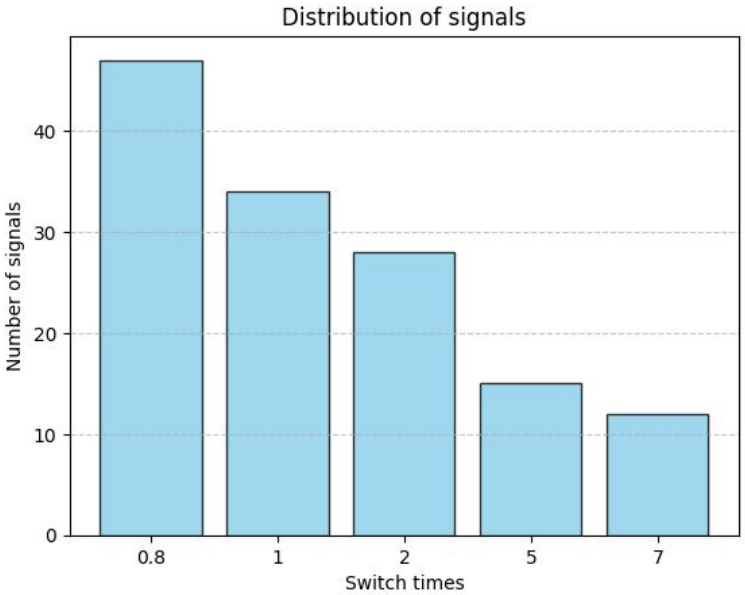
Block duration (sec)	7	5	2	1	0.8
No. of utterances	38	54	133	231	302

Dataset Description (contd.)

- We only consider files with Hindi-English switching by a single speaker for this problem statement
- There exist:
  - 47 sound files with 0.8 s switch time; we use 32 of them
  - 34 sound files with 1 s switch time; we use 32 of them
  - 28 sound files with 2 s switch time
  - 15 sound files with 5 s switch time
  - 12 sound files with 7 s switch time
- Switch time here represents the duration after which the language changes.
- Had to manually determine the starting language for each file, files which were ambiguous to determine were discarded
- After we know the starting language, we can determine language labels for each signal
- Since we have language labels, we have chosen to use supervised approaches

Exploratory Data Analysis





Data pre-processing

- Read the audio file
- Split the audio file into frames
- Determine labels for the frames using duration of switch points and starting language
  - ‘0’ label for Hindi
  - ‘1’ label for English
- Compute frame energies and determine the silent frames based on threshold. A dynamic threshold is used in our case, for each audio file :
  - **(max(frame_energy) - min(frame_energy)) * 0.02**
- Now change labels of frames with frame energies less than threshold to ‘2’ which represents silent frames.

Approaches Used

- **Statistical**
  - GMM
- **Neural**
  - MFCC features + CNN classifier
  - HuBERT features + CNN classifier

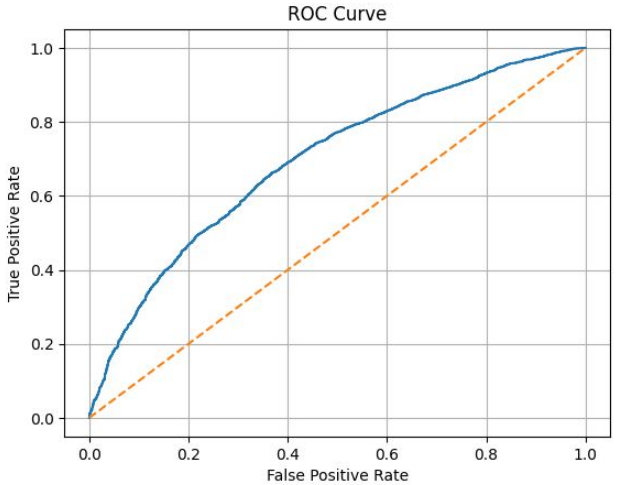
Statistical Approaches

GMM

- Extract MFCCs , velocity and acceleration coefficients as a representative of frame
- Split (feature_vector, label) pairs into train and test set (0.8 split ratio used)
- Fit a Hindi GMM on vectors which have label ‘0’ in train set and English GMM on vectors which have label ‘1’
- While testing compute score using both GMMs and assign the frame to the GMM with higher score

GMM

- The GMMs had 128 components with full covariances
- frame size = 0.25 s , hop length = 0.05 s and 30 MFCCs were extracted using 40 Mel filters
- Train set : 28176 frames of 90 dimensions
    - Hindi: 14090 frames, 90 dimensions
    - English: 14086 frames, 90 dimensions
- Test set : 7044 frames of 90 dimensions
- Accuracy: **64.59%**
- EER: **0.35356**



Reasons for poor accuracies?

- These statistical models are used for frame-by-frame classification ignoring contextual information
- A language is often characterized by particular phoneme and syllable transitions longer than a frame, which cannot be captured by the above GMM model

Neural Approaches

Initial Approach

- Initially, we tried a simple dense frame classification network on the Mel energy coefficients
- It takes one speech frame as input and outputs one of 2 classes: Hindi or English
- The accuracy of the trained model was only around **0.58**
- Thus, it did not perform much better than a random binary classifier
- This poor performance is probably due to the fact that the network does not take into account contextual information, just like our statistical approaches
- Thus, we need to look at architectures that consider a temporal context while classifying

CNN Classification

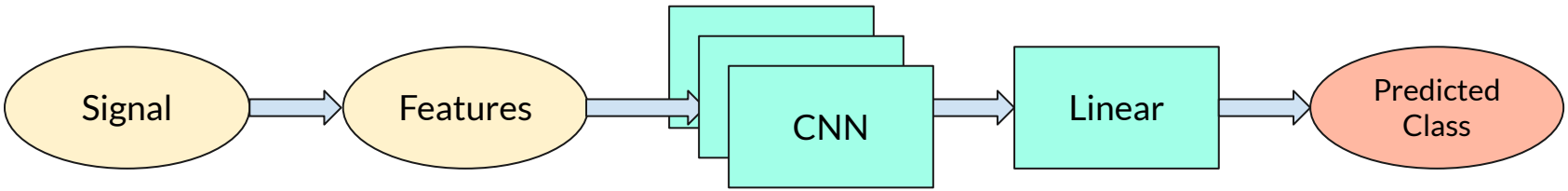
- The idea here is that instead of classifying a single frame, we take the whole signal as input and output a sequence of labels corresponding to the language
- Thus, this is a **Seq2Seq** model
- We are using CNNs to capture contextual information to make predictions
- The context window is determined by the kernel size
- We extract features (of two types) and label for each frame in the signal
    - MFCC features
    - HuBERT features

CNN Classification (contd.)

- For such an architecture, we require all input signals to be of the same length
- To achieve this, we zero-pad all the signals upto the length of the longest signal
- This leads to a third label (**Silent**) in the dataset apart from Hindi and English
- We also perform a VAD on the signal to assign the Silent label to any frames in between the signal below the energy threshold as well
- Thus, this becomes a three-class classification problem

CNN Architecture

- The input of the CNN is of the shape *(batch_size, num_features, num_frames)*
- The CNN consists of two size-preserving 1D convolutional layers with 128 and 256 channels respectively and kernel size 21 with ReLU activation to give an output of shape *(batch_size, 256, num_frames)*
- This is followed by a linear layer with 3 output neurons for each frame
- Thus, the final shape is *(batch_size, num_frames, 3)* after a transpose
- The CNN's weights are trained using the **cross-entropy loss** over 3 classes (Hindi, English, Silent)



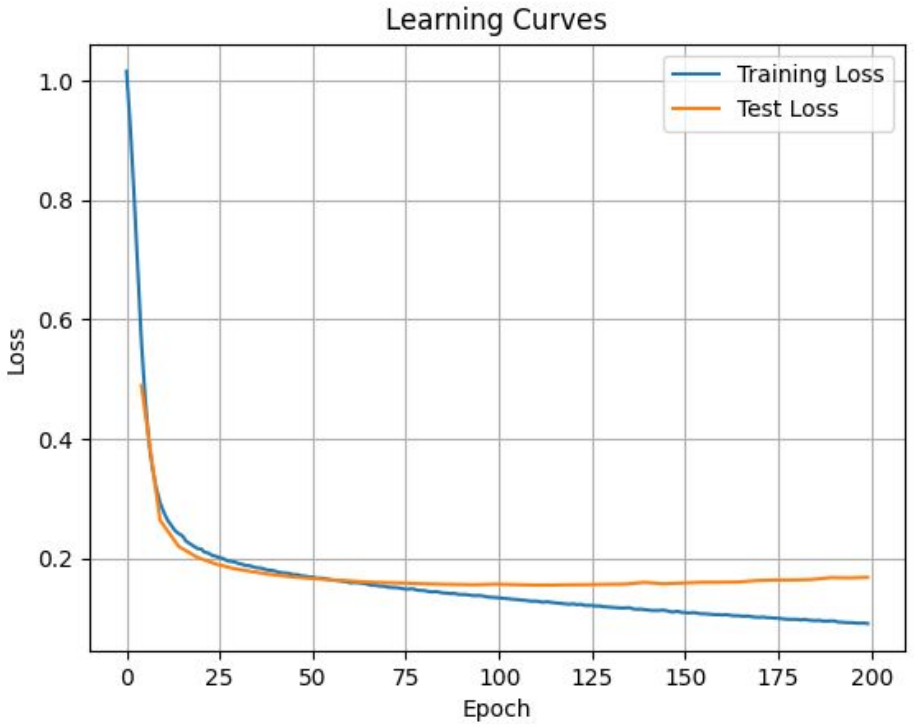
Training Hyperparameters

- Number of epochs: 200 for MFCC, 100 for HuBERT
- Learning rate: 1e-5
- Batch size: 4
- Number of CNN filters: [128, 256]
- Kernel size: 21

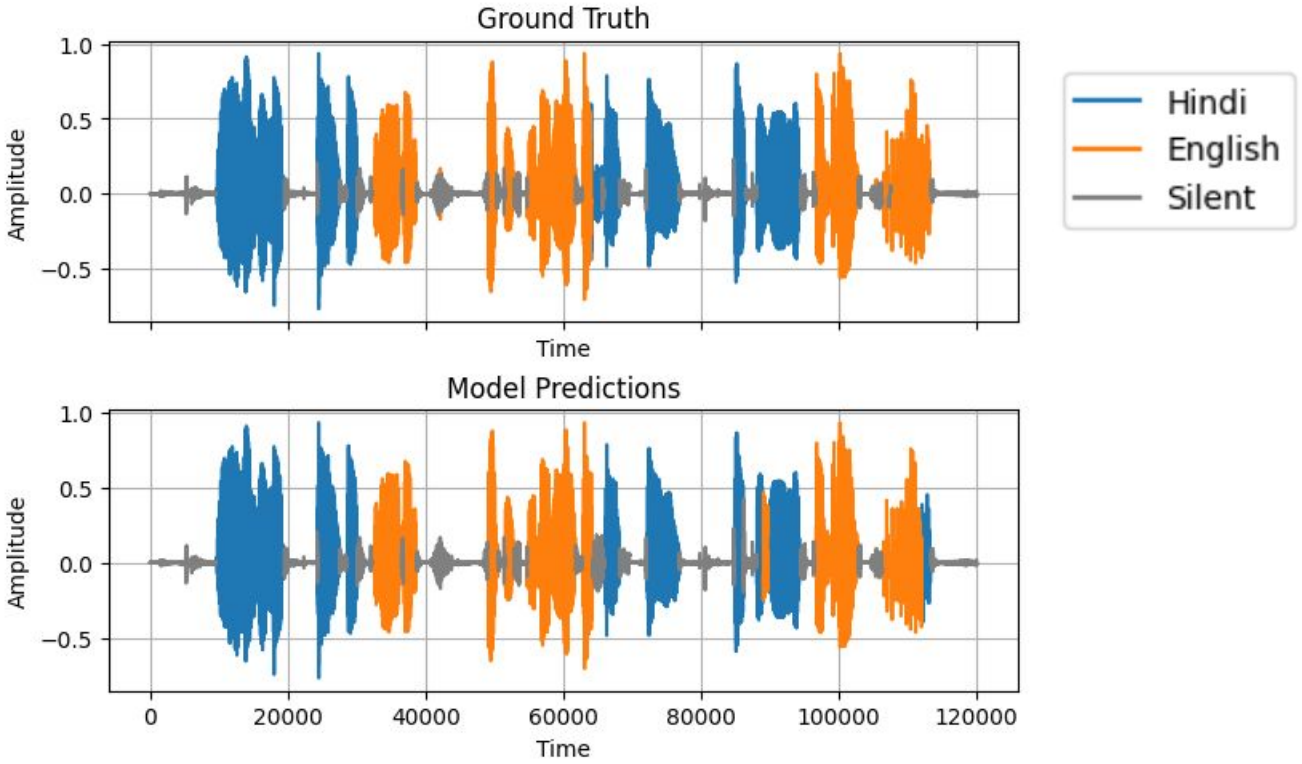
MFCC Features

- The first way of extracting features we chose was MFCC
- We chose a window length of 25 ms and a hop length of 20 ms
- We used 13-dimensional MFCCs along with its velocity and acceleration coefficients
- CMVN was also performed to remove channel and noise effects
- Thus, we got **39-dimensional features** for each frame in the signal
- These features along with the labels computed earlier are used to train the CNN classifier
- We obtained an overall classification accuracy of **92.90%** including padded frames
- If we ignore padded frames for accuracy computation, we get **78.02%**

Learning Curves with MFCC Features



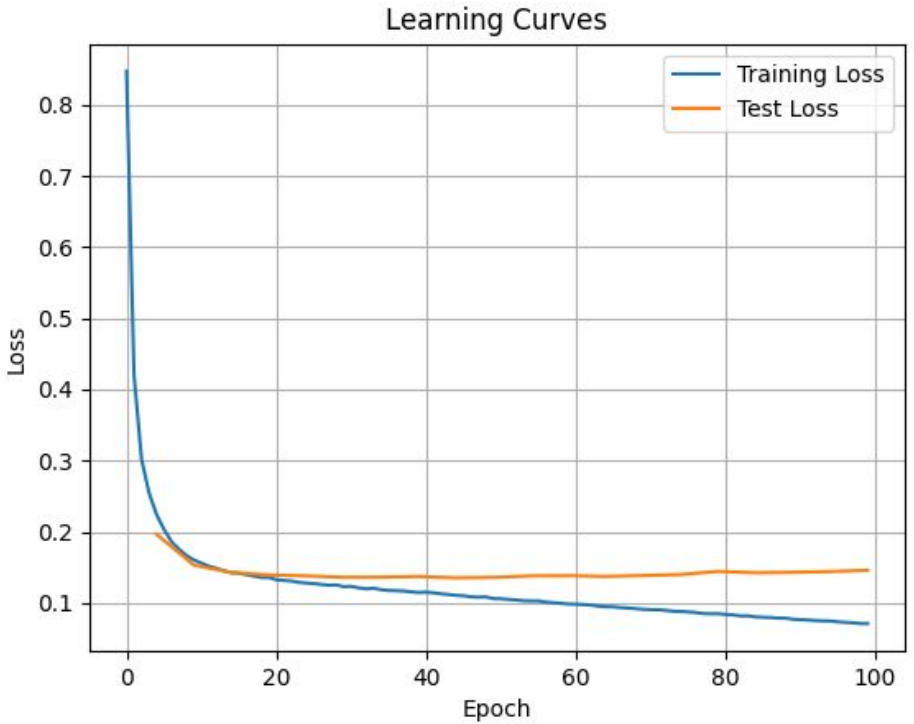
Example of MFCC + CNN Predictions



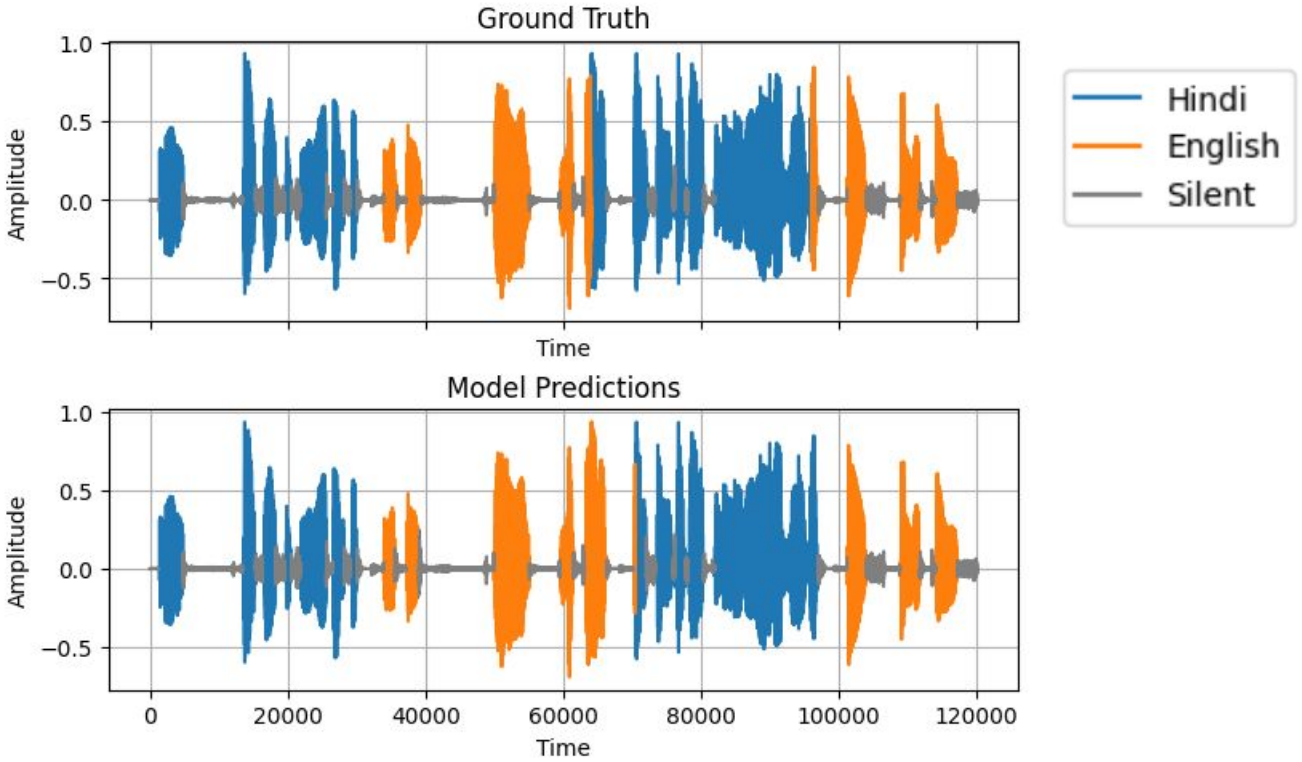
HuBERT Features

- For another way of extracting features, we used the **facebook/hubert-base-ls960** model from the HuggingFace Transformers library
- Internally, HuBERT uses a window length of 25 ms and a hop length of 20 ms for framing the signal and outputs **768-dimensional features** for each frame in the signal
- Since HuBERT's architecture consists of attention layers, the embeddings themselves are capturing contextual information too
- After training on these features, we obtained an overall classification accuracy of **93.77%** including padded frames
- If we ignore padded frames for accuracy computation, we get **80.71%**
- Note that HuBERT's weights are freezed and are not being touched

Learning Curves with HuBERT Features

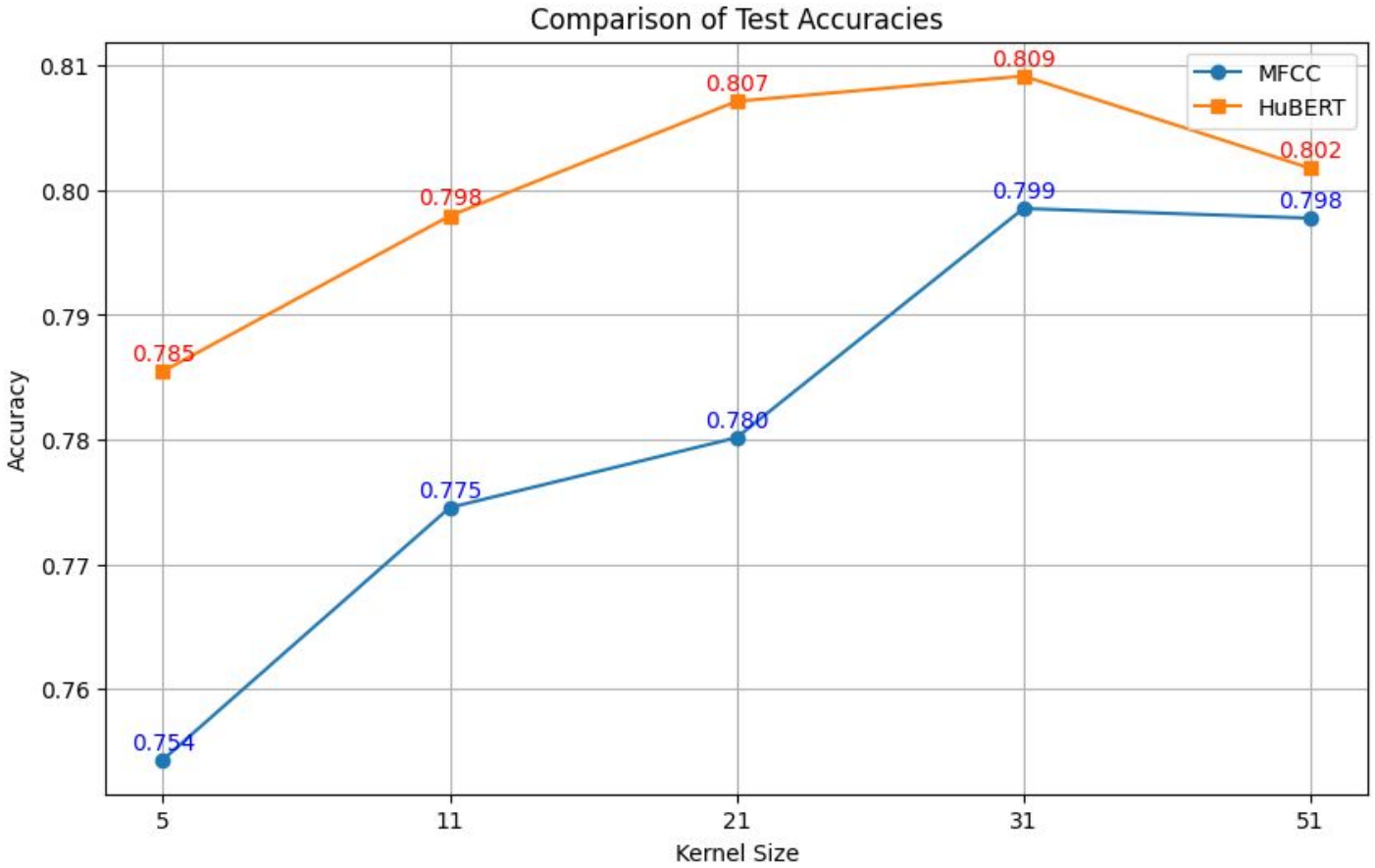


Example of HuBERT + CNN Predictions



Effect of Kernel Size

- We also investigated the effect of kernel size on the performance of these models
- The kernel size determines the context window (the number of neighbouring frames taken into consideration)
- We observed that increasing the kernel size improves the performance upto a point, beyond which it deteriorates
- This is expected because a kernel size too small does not have enough context
- If the kernel size is too large, the context ends up including frames of different languages
- We can also observe that HuBERT features consistently outperform MFCC features as expected


Comparison of Test Accuracies
MFCC
HuBERT
Accuracy
Kernel Size
0.81
0.80
0.79
0.78
0.77
0.76
5
11
21
31
51
0.785
0.798
0.807
0.809
0.802
0.754
0.775
0.780
0.799
0.798

References

- Spoken Language Diarization Using an Attention based Neural Network
- Link to dataset
- GMM (scikit-learn)
- MFCC (Librosa)
- HuBERT (HuggingFace Transformers)
- HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units