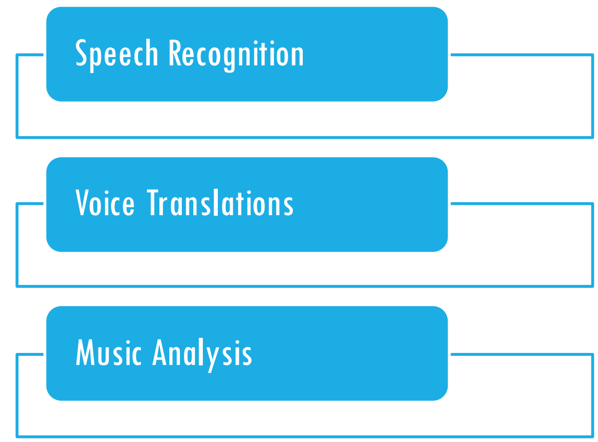
ACCENT CLASSIFICATION AND CONVERSION

Arvind Raghavendran (MDS202214)
Swastika Mohapatra (MDS202245)

INTRODUCTION TO AUDIO DATA

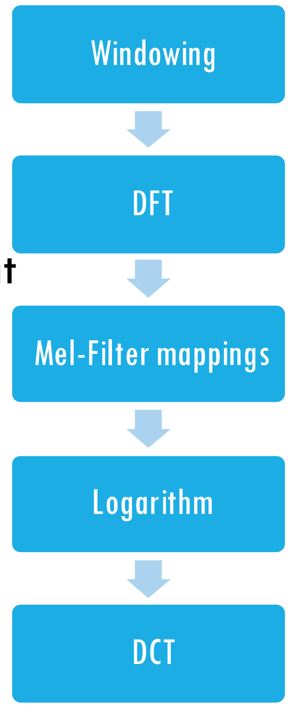
- Versatile medium utilized across various domains
- Multiple file formats like .mp3, .wav, etc.
- Challenges:
  - Raw data
  - Device/environmental/random noise additions
  - Variable durations



- Aim: explore the application of machine learning techniques to audio data

MEL-FILTER CEPSTRUM COEFFICIENTS (MFCC)

- Based on human auditory system's response to audio
- Effective for capturing spectral features, such as the shape of the vocal tract and mimic human voice system
- We can generate as many coefficients as we want but empirically is observed that the 13 coefficients are enough for ***most*** problems
- Advantages:
  - Dimensionality Reduction
  - Robust to Noise
  - Computationally Efficient
- Drawbacks:
  - Cannot capture finer temporal differences like pitch, etc. effectively



SPECTROGRAMS

Windowing → STFT → Squared Frequency → Visualization

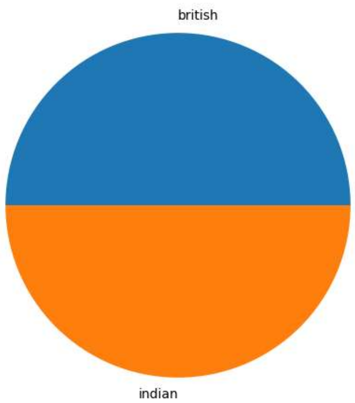
- Visual representation of the frequency content of an audio signal over time
- The resulting magnitude values are represented using a color scale, where brighter colors indicate higher magnitudes, capturing both temporal and spectral information.
- Suitable for tasks requiring detailed analysis of audio features, such as accent classification and music genre classification.
- Advantages:
  - Captures subtle changes in frequency content
- Drawbacks:
  - Computationally expensive
  - Require careful hyperparameter tuning

ACCENT CLASSIFICATION VIA MFCCS

Task 1

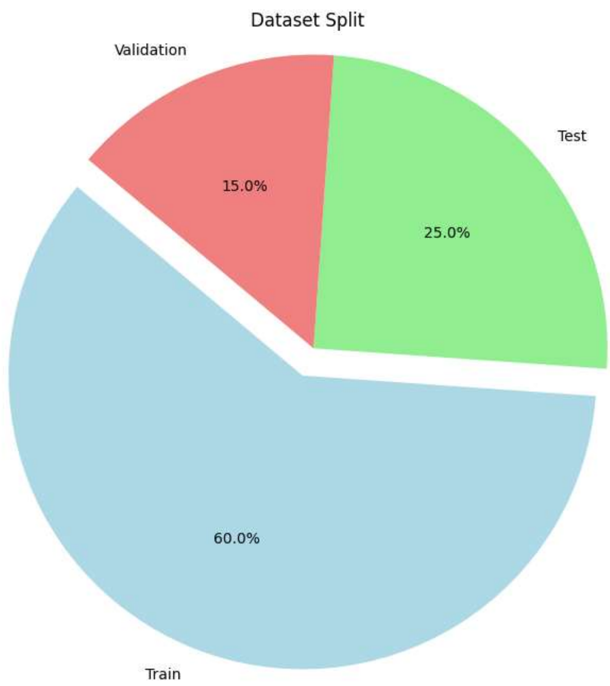
DATA DESCRIPTION

- Consists of audio recordings of speakers from different countries having diverse range of accents like UK, India, US, Australia, etc. Our focus is on Indian and British.
- 742 recordings , speaking the same passage
- Extracted 13 MFCCs across windowed timestamps to generate (13,*n*) feature matrices, where *n* depends on length of audio
- Compressed information by taking row-wise average to get feature vector of length 13



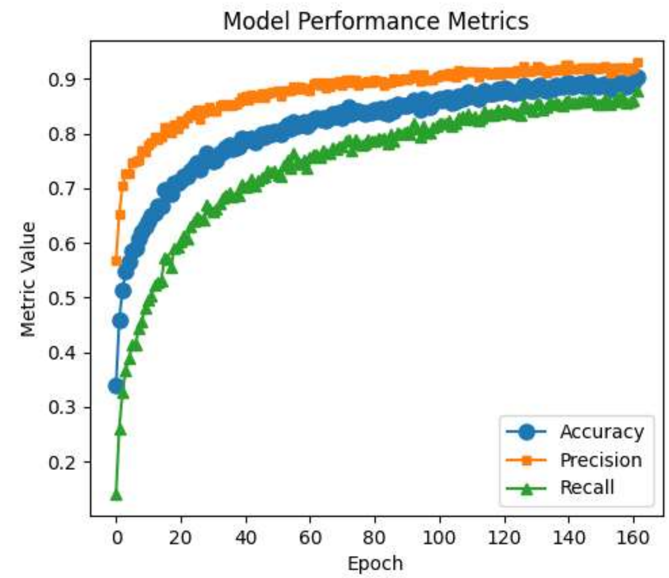
MODEL CONSTRUCTIONS

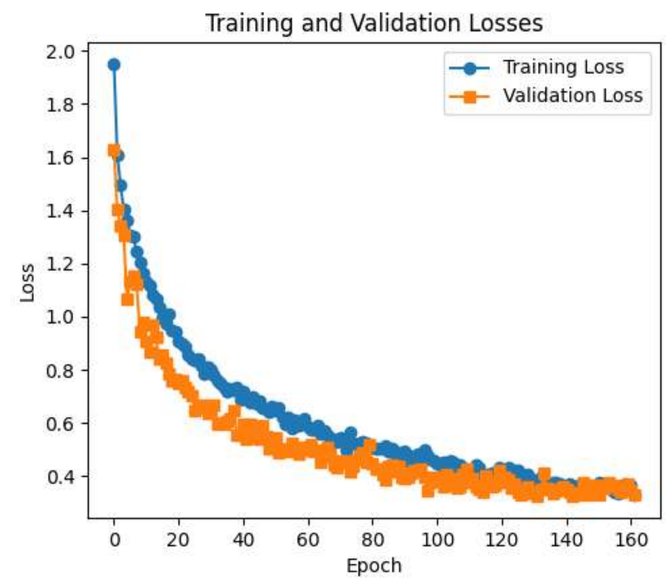
- Experimented with simple linear models and 1-D Convolutional layers to capture contextual information
- Batch Normalizations to ensure steady training and convergence
- Dropout to prevent overfitting
- Added padding for deeper networks with ReLU activations
- Compiled model with Cross entropy loss and Adam optimizer
- Prevented overfitting via early stopping and adding model checkpoints



TRAINING AND EVALUATION

- Best model had 4 convolutional layers of sizes 1024, 512, 512, 256 followed by GlobalAveragePooling and then 3 Dense layers of size 2048, 512, 512
- Trained in batches of 32 for 1000 epochs, with patience of 15 epochs
- Out-sample metrics:
  - Precision: 91%
  - Recall: 86%
  - Accuracy: 88%
- Diagnosis:
  - Model rarely misclassifies true-negatives, but does so for some true-positives
  - Model unable to capture finer nuances in accents



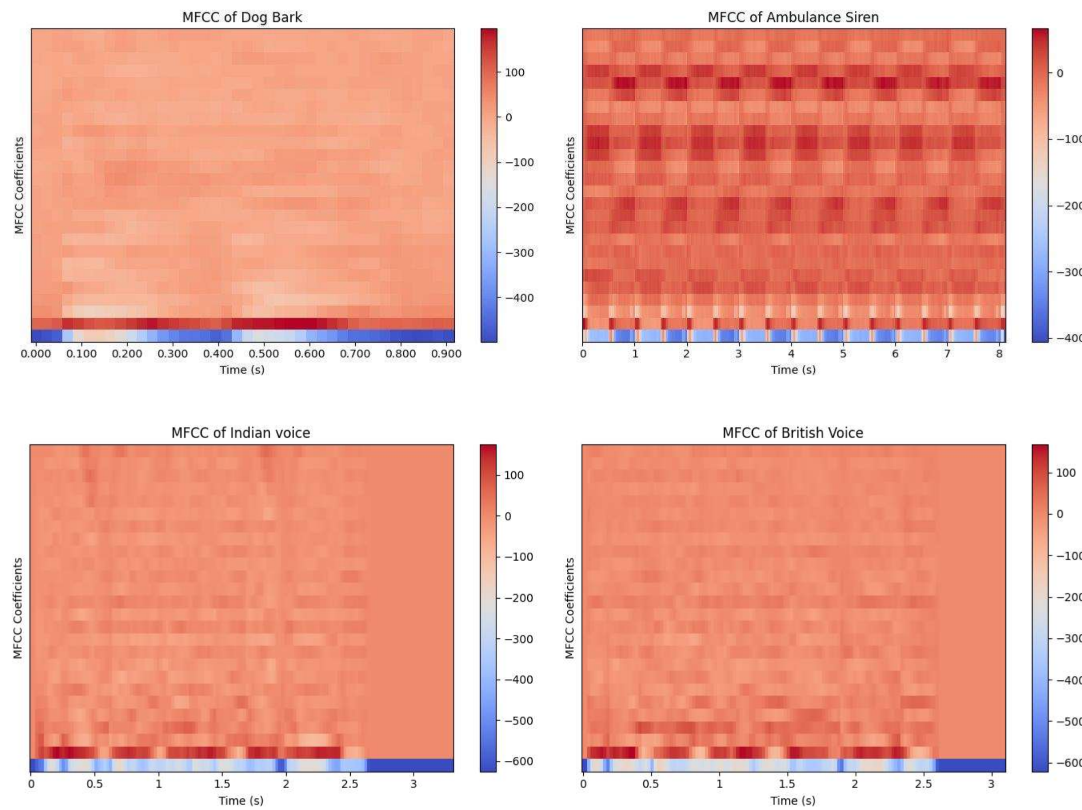


ACCENT CLASSIFICATION VIA SPECTROGRAMS

Task 2

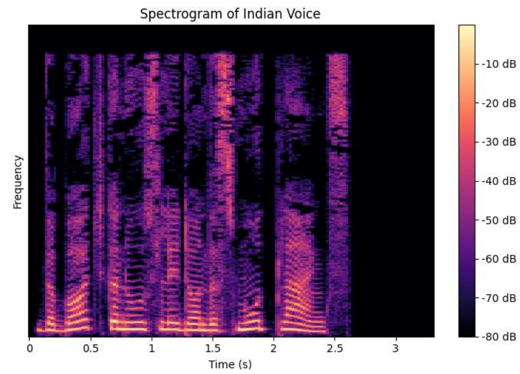
ISSUES WITH USING MFFCs

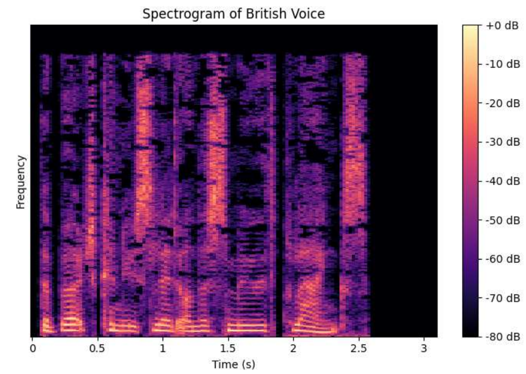
- MFCCs capture high-level, vocal tract information that might not capture subtle pitch and frequency changes crucial for accent classification
- Issue persisted even without averaging the features
- MFCCs are more useful if audios are very distinct, for example a dog barking vs an ambulance siren
- Accents are copies of the same waveform with slight nuances in pitch, temporal frequencies, etc and hence MFCCs fail to be distinctive



SPECTROGRAMS AS A SOLUTION

- They represent audio signals in the time-frequency domain, capturing both temporal and spectral features simultaneously.
- High dimensionality is advantageous here since that is what the task demands
- Offer precise time-frequency localization and identify rapid changes in speech characteristics



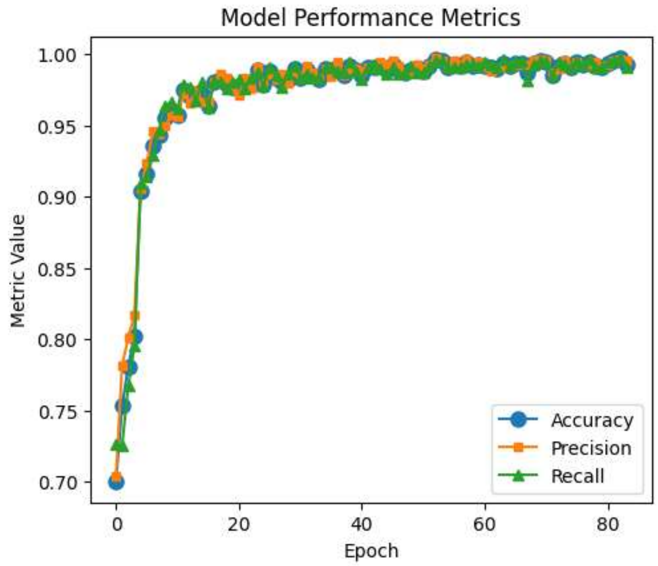


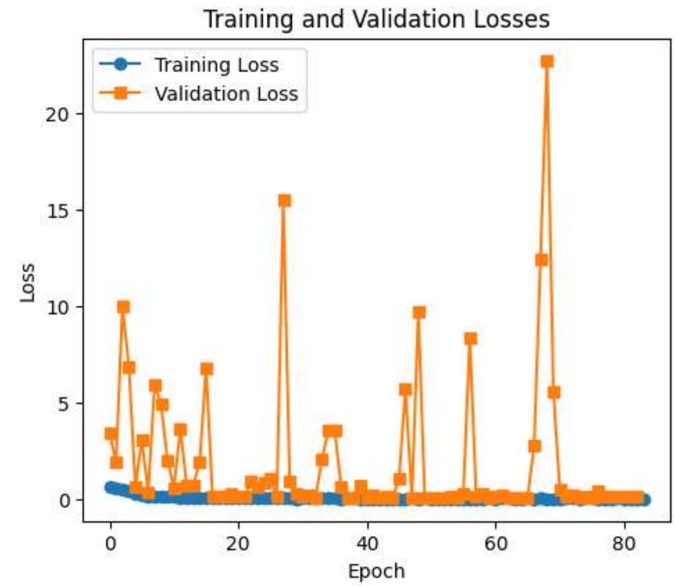
DATA CONSTRUCTION

- Experimented with various hyperparameters like window length (512), hop length(1024), sampling rate(22.05 KHz)
- Hanning algorithm to obtain the windows for applying STFT
- The result was 513 spectral frequencies distributed across *n* timesteps, that vary depending upon audio length.
- We padded the matrices with zeros to have all inputs of the same size (513, 229)

TRAINING AND EVALUATION

- We used the same earlier model with the same training steps except that the 1D layers are replaced with 2D layers
- We also introduced MaxPooling at each layer to reduce size of feature maps
- Out-sample metrics:
  - Precision: 97.68%
  - Recall: 98.13%
  - Accuracy: 97.97%
- Diagnosis:
  - Model able to identify subtle changes in accents via spectrograms
  - Does a much better job generalizing to test set





ACCENT CONVERSION

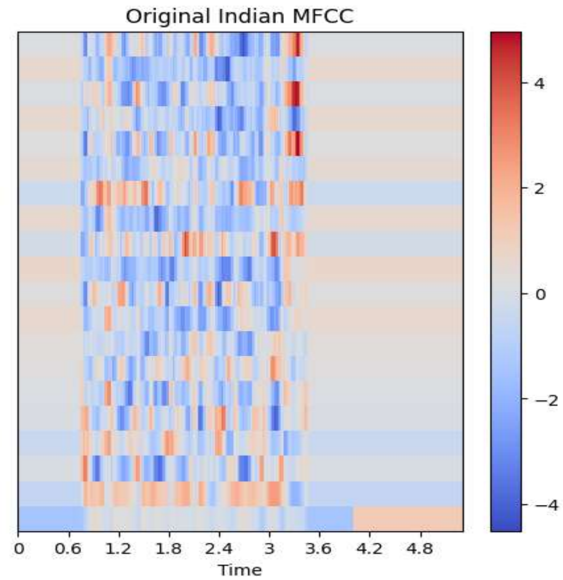
Task 3

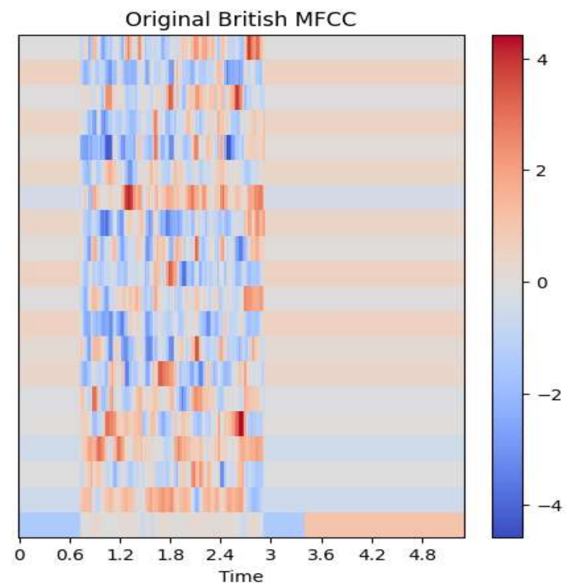
MODEL CONSIDERATIONS

- Involves compressing input audio into latent space and reconstructing back to target domain
- CNN-Dense models would not work well since reconstruction is poor
- Autoencoder frameworks or CNN+LSTMs would be a better match because they capture spatial context while reconstructing
- Best possible model on a spectrogram was quite shallow having encoder and decoders as 3 layer CNNs with 256 filters at most. Model performed poorly, highlighting need for deeper architecture

DATA DESCRIPTION

- Although spectrograms captured accent nuances more than MFCCs, high dimensionality combined with deep architectures make training infeasible
- Low dimensionality of MFCC makes it a good choice, so we proceed to extract 26 MFCCs for higher representation
- We don't want to compress the matrices since we cannot reconstruct audio back from averaged features.
- So we pad using zeros to obtain (26, 229) feature matrices for all audios.
- We also normalized the data to reduce overhead and stored mean, variances



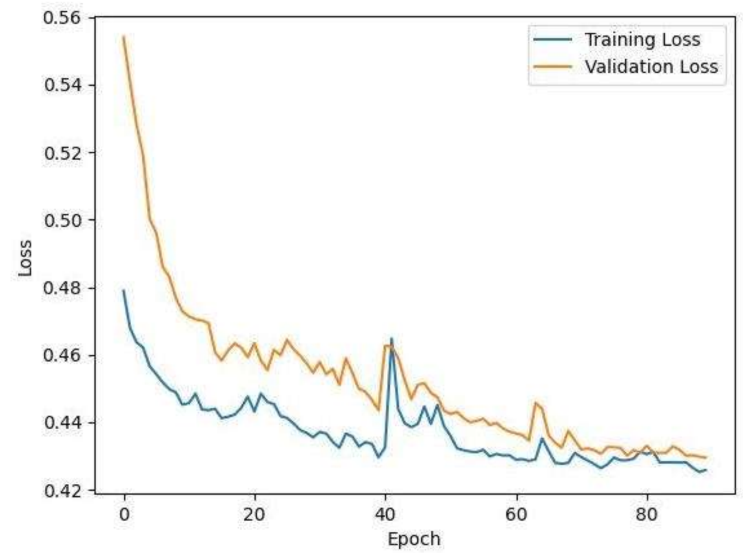


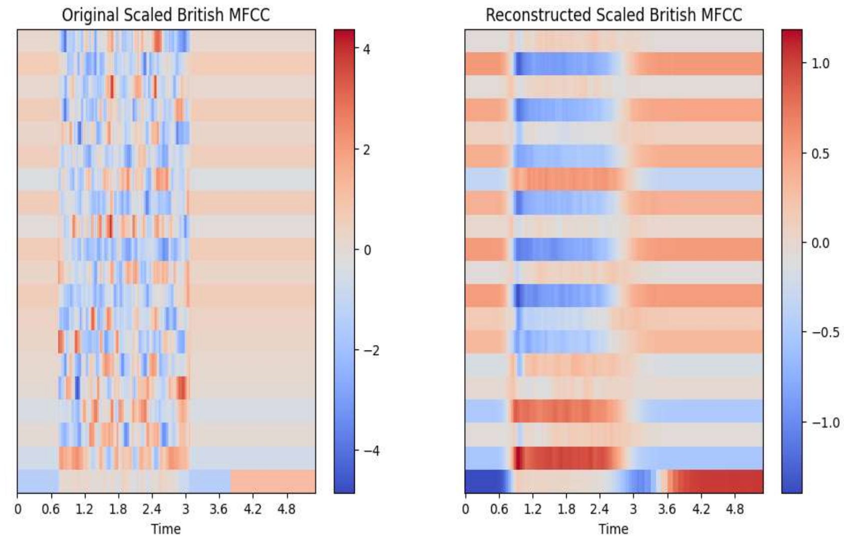
MODEL CONSTRUCTIONS

- The core training ideas were the same, like Dropout, BN, Pooling, etc
- Additionally, we Normalized inputs for smoother training. This was a key step in the process that countered the high dimensionality
- We experimented with MSE as the loss function, with a possible extension to using Huber Loss that combines both L1 and L2 losses.
- We also experimented with multiple model architectures including LSTMs, GRUs, RNNs, etc. into the autoencoder framework.
- The best model chosen had an encoder having 3 CNN layers with 1024, 512, 256 filters and a mirror architecture for the decoder. The latent embedding was a GRU having 8 units.

TRAINING AND EVALUATION

- We observed that the model is learning, and loss reduces to some extent
- Reconstruction shows that model has learnt high level features but is smoothening out the features
- This stems from the nature of choosing MSE as the loss, as a pixel level MSE will start to smoothen the other features out to obtain a lower loss





IMPROVING THE LOSS FUNCTION

- We replaced the vanilla p-norm losses with an adversarial loss that aims to minimize both the p-norm loss and also preserves pixel information
- This new loss was a weighted combination of MAE+ mean pixel-level CCE

$$\text{Loss}(y_{\text{true}}, y_{\text{pred}}) = \alpha \times \text{Spectral Loss}(y_{\text{true}}, y_{\text{pred}}) + \beta \times \text{Adversarial Loss}(y_{\text{true}}, y_{\text{pred}})$$

$$\text{Spectral Loss}(y_{\text{true}}, y_{\text{pred}}) = \text{mean}\left(|y_{\text{true}} - y_{\text{pred}}|\right)$$

$$\text{Adversarial Loss}(y_{\text{true}}, y_{\text{pred}}) = \text{mean}\left(\text{binary\_crossentropy}(y_{\text{true}}, y_{\text{pred}})\right)$$

- We chose alpha=0.7, beta=0.1

IMPROVING THE MODELS

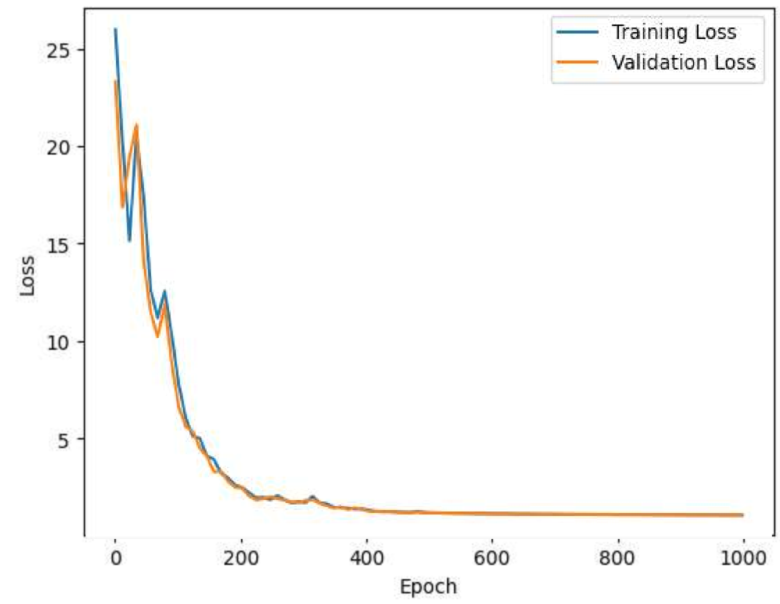
- Skip connections were added to ensure that subsequent model layers did not lose context of the previous information
- We read about different kernel initializations and used the LeCun method where weights are sampled from $\mathrm{Normal}(x \mid \mu = 0, \sigma = \sqrt{\frac{1}{\mathrm{fan\_in}}})$
- Latent Space representation was now 2 GRUs with 16 units. This was to see if deeper bottlenecks could add more information while reconstruction
- ReLU activation was now replaced with GeLU activation to ensure gradients can flow all the way back to encoder

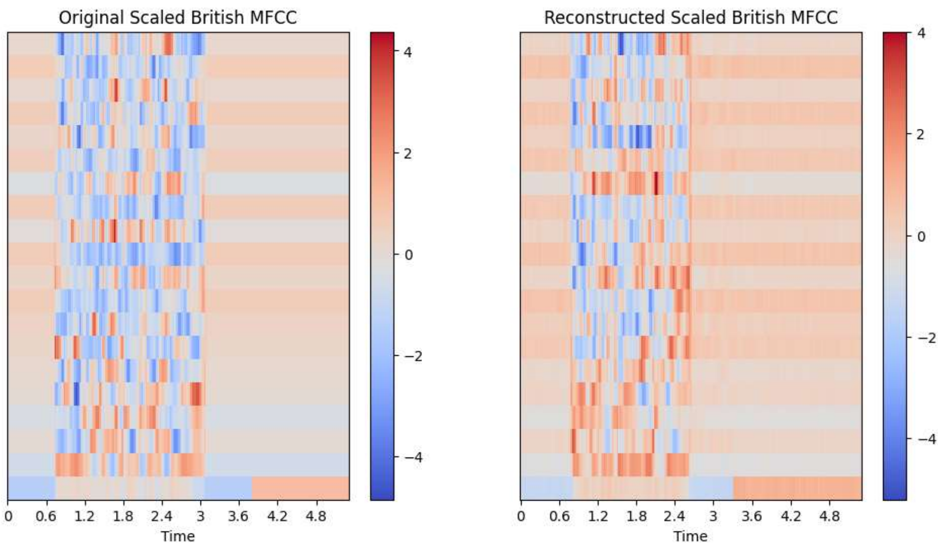
TRAINING AND EVALUATION

- We trained for 1000 epochs, with a patience of 30 epochs
- Reconstruction shows that model has significantly improved learning, and has also matched pixel semantics to a good extent

Original Audio

Reconstructed Audio





FURTHER IMPROVEMENTS

- We achieved significant improvement from the earlier model, but this can certainly be improved. We can search for more datasets, but have to consider the quality as well.
- We hypothesize that using spectrogram data coupled with deeper architectures can improve the model performance significantly. However we cannot explore this due to limited computational resources.
- We also hypothesize that a higher sampling rate with higher fft components can results in wider and higher spectrogram resolutions, allowing the model to capture finer details.
- We can also experiment with transfer learning on audio data (VGGish) for faster and powerful latent representations and then reconstruct it.

Q&A SESSION

Thank you!