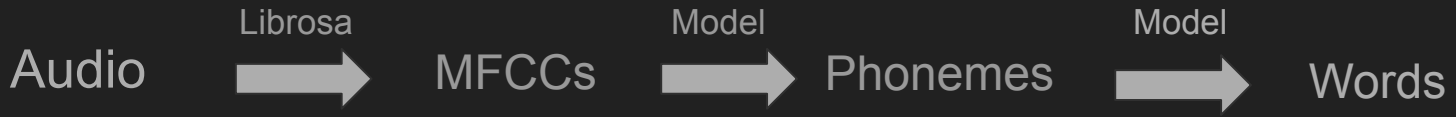
IDC 409
Speech-to-Text

Group 14

Siddharth T (MS21229)
Aniruddha Mondal (MS21182)
Lakshyajeet Samant (MS21120)

Overall Workflow

- Take audio files with transcripts as input
- Perform Montreal Forced Alignment to give a TextGrid file ((word,time period),(phoneme, time period))
- Load the audio (Amplitude with time)
- Process the audio, extract the MFCC features
- The MFCC features are mapped to audio units (phonemes)
- Phonemes are mapped to words, which is the output



Phonemes

- A Phoneme can be regarded as a ‘smallest’ phonetic unit that helps distinguish a word from another by capturing the phonetic differences between them. Thus, they can be regarded as an alphabet of sorts for the phonetic properties of words.
- For example, the words “read” and “read” are homographs which can’t be distinguished by the English alphabet, but they have differing phoneme spellings.
- We primarily use 60 symbols from the IPA dictionary, which is a comprehensive and widely accepted alphabet for phonemes. These 60 symbols capture all the sounds used in the spoken English language.

Montreal Forced Aligner

- This is a tool which helps us in “aligning” the transcript of a given audio file.
- Takes in a transcripted audio file and gives the list of words along with the time at which they are spoken in that file.
- Can also be applied to phonemes.
- We use the MFA to obtain two sets of data from the : one of the words used in the input audio files along with the speaking time, and the other of the phonemes used used in the audio files along with the speaking time.
- The phoneme to word map is obtained by using these two sets of data.

Audio Processing

MFCC Feature Extraction. (Mel-Frequency Cepstral Coefficients)

- Extracting features capturing the formants and timbre of sound.
- Features extraction to highlight the parts of the data the model should learn.
- The audio is divided into different time frames where each is associated with an ordered set of MFCCs.
- Typically the first MFCC represents the overall energy of the frame, giving a measure of the loudness.

Extracting MFCC

- Frame the signal into small overlapping windows.
- Apply Fourier Transform to convert each frame to the frequency domain.
- Apply Mel filter bank to focus on perceptually relevant frequencies.
- Take the logarithm of the filtered spectrum to compress the range.
- Apply Discrete Cosine Transform to extract the final MFCCS.

Phoneme to Word Map

- In our code, we train a model to map features extracted from MFCCs to the aforementioned 60 phonemes.
- In our code, we use the MFA to extract a TextGrid file with time-coordinated transcript of both words and phonemes.
- We can then construct a map from words to phonemes by matching the times of phonemes to words. We train a model to associate sequences of phonemes to words. This model now functions as the required map.

Neural Networks used

- **LSTM** - Stands for **L**ong **S**hort **T**erm **M**emory. It is a type of Recurrent Neural Network that learns when to retain and when to forget information by taking into account ‘context’. This gives it an advantage over other RNNs, which are gap-sensitive.

  In our Speech-to-Text code, we make use of two LSTMs to create accurate models.

  In the first LSTM model which creates a map from MFCCs to phonemes. We use it to ensure that the feature that is associated with a sequence of phonemes is not too long. This allows us to isolate individual phonemes to specific sequence of MFCC features.

  The sequences of phonemes considered for the phoneme-to-word map must not be too long, which is also ensured by the LSTM model we have used to generate this map. This in turn means that words are differentiated in the final output.

Explanation of LSTM

How does an LSTM capture context? Basically, it utilises a hidden state to help take into account the context in which the new information is being added. This is done via 3 Logic gates that control each memory cell: Forget, Input and Output.

We also have an input node takes into account input and previous hidden state.

The LSTM converts output to range (-1,1) using the hyperbolic tan function, tanh.

We take the input node, apply the input operation to it and then apply the forget operation to the cell internal state and update it.

We then apply the output gate and tanh(Cell internal state) to update the hidden state, which is now ready for the next iteration.

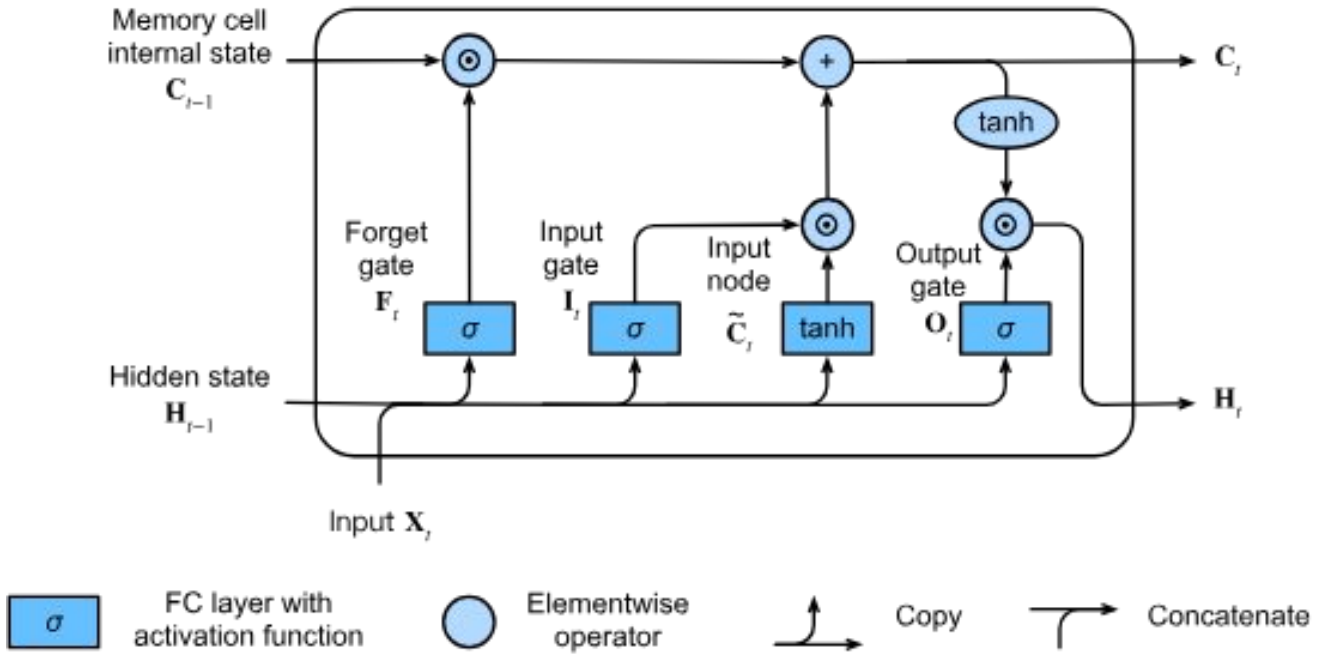
Explanation of LSTM

Hidden State:

Long Term memory

Cell Internal State:

Short Term memory



Cross Entropy Loss

To optimize our model we use a cross entropy loss function, where it is given by:

$$\text{Loss} = -\sum y_i \cdot \log \hat{y}_i$$

Where $\hat{y}_i$ is the true label for class ‘i’

y_i is the predicted probability of class i

Why the Cross Entropy Loss?

From Information Theory, if something has a low probability of occurrence. Occurrence of that event would give you more information compared to an event with higher probability.

1/p(i) ⟶ - log(p(i)) ⟶ $$-\sum_{i=1}^{C} p_i log(p_i)$$

To take into account only significant changes we can convert to a log scale. Now we have multiple classes take the expectation value over the multiple classes.

Now a large value of such an entropy indicates that almost each class is equiprobable

And that we have less information. As our entropy decreases , we are gaining more information and are more certain of our predictions.

Optimizer - Adam

We update our parameters based on 2 factors:

Where our momentum is/where we are currently directed to go and how much we will move is dependent on our previous gradients.

$$x^{n+1} = x^n - \frac{\alpha}{(\sqrt{\hat{v}_n + \epsilon})} \hat{s}_n$$

Sn is a weighted moving average of gradients. with bias correction s^n=1−β1nsn

$$s_n = \beta_1 s_{n-1} + (1 - \beta 1) gn$$

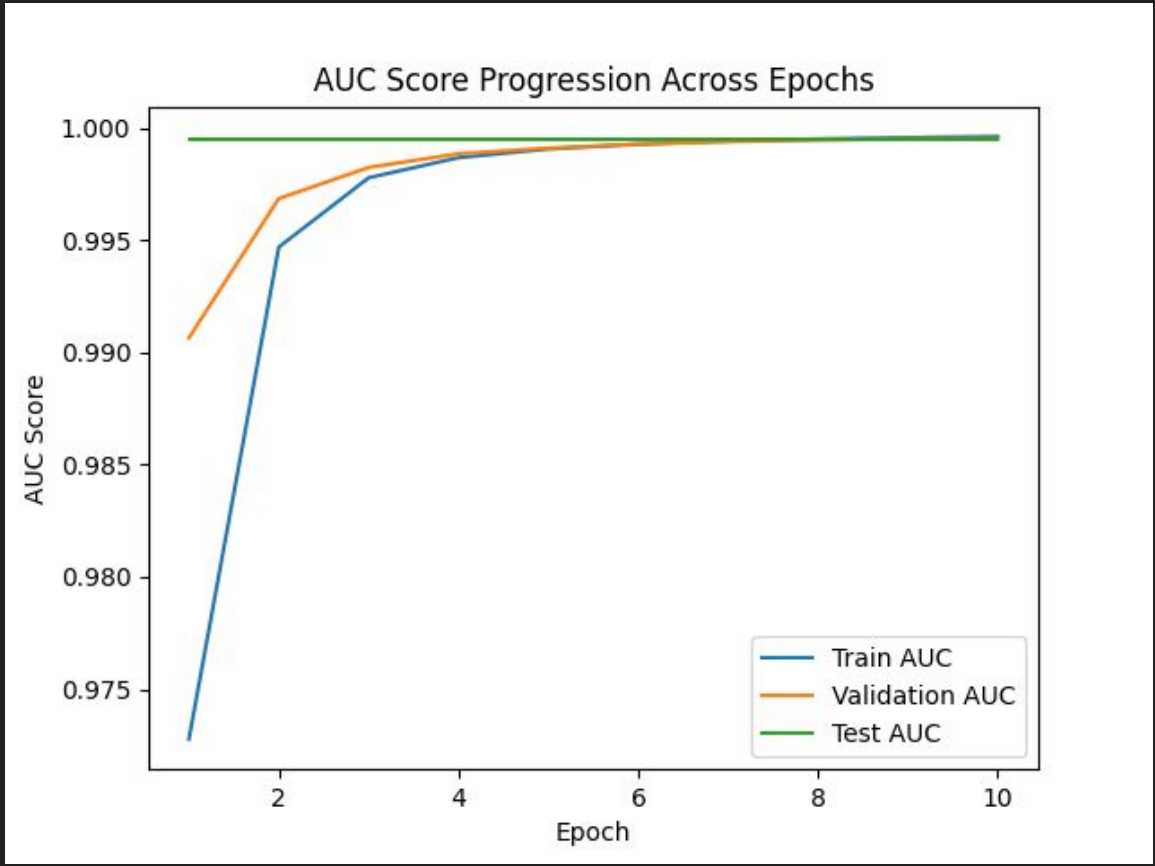
Vn is a weighted moving average of square of gradients , with bias correction v^n=1−β2nvn

$$v_n = \beta_2 v_{n-1} + (1 - \beta_2) g_n^2$$

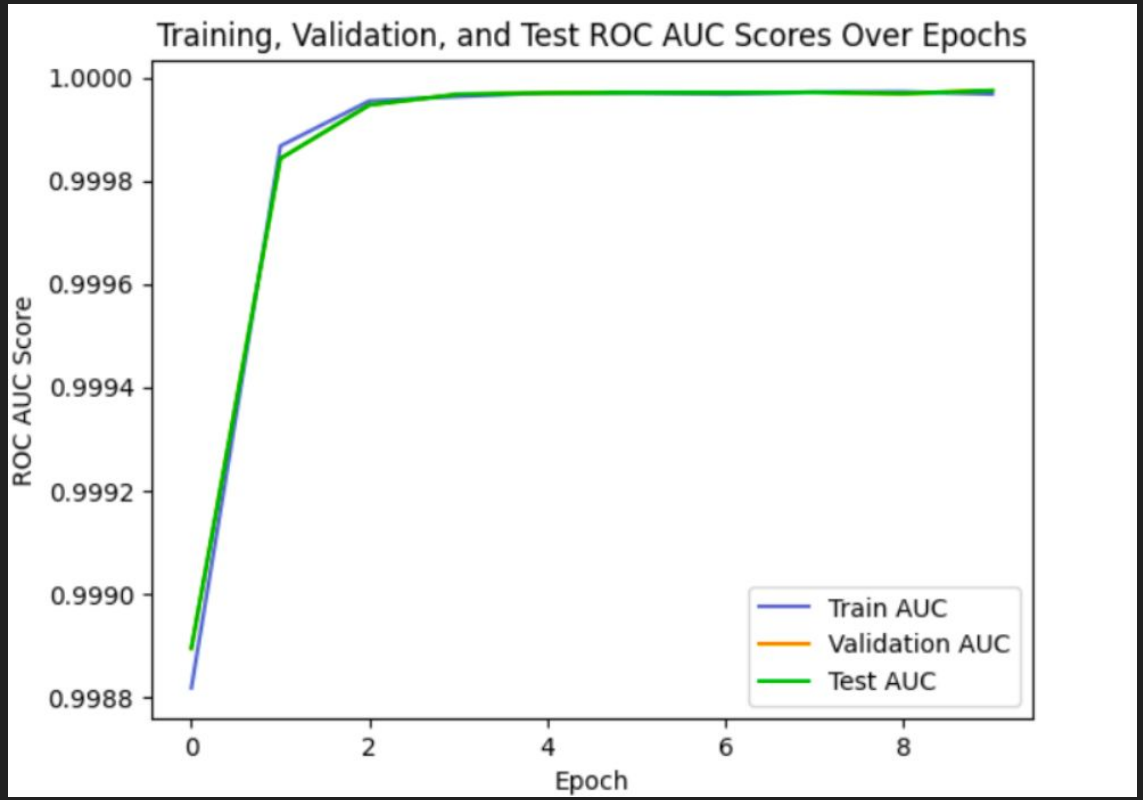
α is learning rate hyperparameter

X^n is parameter value at iteration n and x^{n+1} is parameter value at iteration n+1

AUC variation for MFCC to Phoneme Model



AUC variation for Phoneme Word Model



References

1.Devopedia. 2021. "Audio Feature Extraction." Version 8, May 23. Accessed 2024-06-25. https://devopedia.org/audio-feature-extraction

2. Cross-Entropy Loss — Crucial In ML & How To Use It

3. 10.1. Long Short-Term Memory (LSTM) — Dive into Deep Learning 1.0.3 documentation

4. What is LSTM - Long Short Term Memory? - GeeksforGeeks

5.What is Adam Optimizer? - GeeksforGeeks

6.Adam Optimizer Tutorial: Intuition and Implementation in Python | DataCamp