A Textless Approach for Speech to Speech Comparison

Aditya Raj (241040002)
Suryansh Singh (241040087)
Shivansh Maheshwari (210987)

Indian Institute of Technology - Kanpur

April 21, 2025

Overview

Problem Definition

- **Compare two audio recordings of the same word (e.g., native vs learner).**
- **Generate a scoring metric for how well the word matches.**
- **Determine a threshold to decide if the pronunciation is acceptable or mispronounced.**

Metrics

- True Positive (TP) – Model predicts correct pronunciation, and it is actually correct.
- False Positive (FP) – Model predicts correct pronunciation, but it is actually wrong (i.e., mispronounced).
- True Negative (TN) – Model predicts wrong pronunciation, and it is actually wrong.
- False Negative (FN) – Model predicts wrong pronunciation, but it is actually correct.

Mispronunciation detection Metrics

- TPR - $\frac{TP}{TP+FN}$
- FPR - $\frac{FP}{TN+FP}$
- TNR - $\frac{TN}{TN+FP}$
- FNR - $\frac{FN}{TP+FN}$

- Precision - $\frac{TP}{TP+FP}$
- Recall - $\frac{TP}{TP+FN}$
- F1 Score - $\frac{2 \times Precision \times Recall}{Precision + Recall}$

Metrics

ASR Metrics

- **Character Error Rate (CER)**:

$$\text{CER} = \frac{S + D + I}{N}$$

- **Word Error Rate (WER)**:

$$\text{WER} = \frac{S + D + I}{N}$$

- **Token Error Rate (TER)**:

$$\text{TER} = \frac{S + D + I}{N}$$

Where:

- S = Substitutions
- D = Deletions
- I = Insertions
- N = Number of words/characters/tokens in reference

Dataset

- We have used Kathbath Hindi datasets . We have rearranged the datasets for our use .
- We have created four columns - Word , Path , Boundary , Group. We have 10 Lakh of rows in csv.
- For each audio path , we are using the boundary to get the segments of each audio .
- Since the audio segments is very small so we are using padding on both side (0.1 sec) .
- After that we have created pairs of datasets in which there is two audio's and label , if both audio represents same word then label is 1 otherwise it is 0.

Metrics

Model	TPR	FPR	TNR	FNR	Precision	Recall	F1	Opt_Threshold
Parakeet_ctc	0.6667	0.3327	0.6673	0.3333	0.6545	0.6667	0.6606	9.5634
Conformer_ctc	0.8196	0.1816	0.8184	0.1804	0.8307	0.8196	0.8251	0.5472
Contrastive 10-04-2025	0.7189	0.281	0.719	0.2811	0.7164	0.7189	0.7177	0
InfoNCE Approach	0.8323	0.1666	0.834	0.1677	0.8312	0.8323	0.8318	0
Contrastive 12-04-2025	0.5952	0.4046	0.5954	0.4048	0.5957	0.5952	0.5955	0
MFCC Approach	0.38	0.6	0.4	0.62	0.3878	0.38	0.3838	0.0112
Conformer (10th layer)	0.7904	0.2086	0.7914	0.2096	0.7934	0.7904	0.7919	-11.3674

Table: Performance metrics and optimal threshold for different models

Results

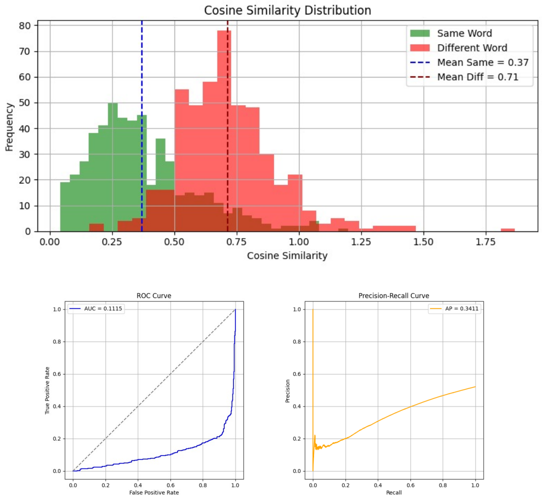


Figure: CTC Conformer

Results

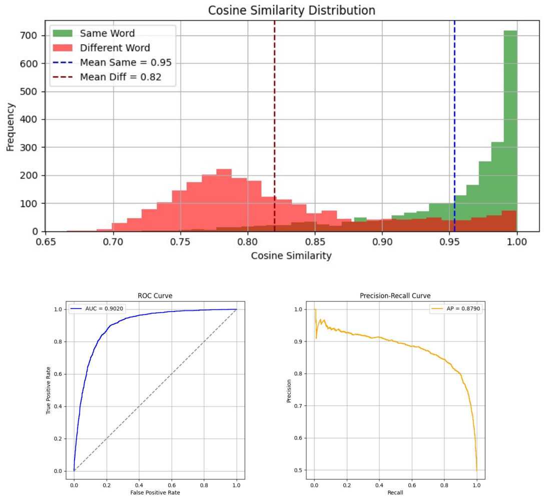


Figure: Wav2vec2 11th April

Results

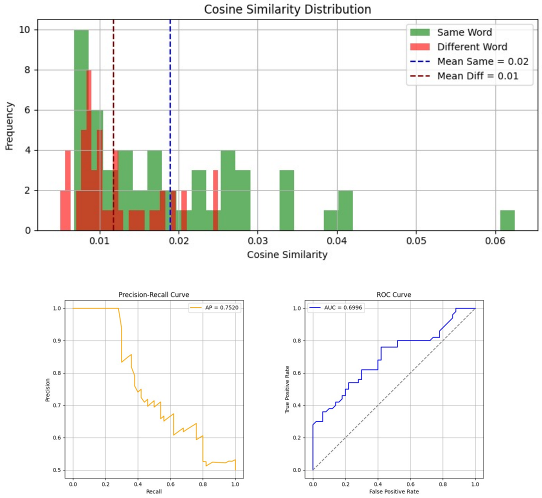


Figure: MFCC Approach

Results

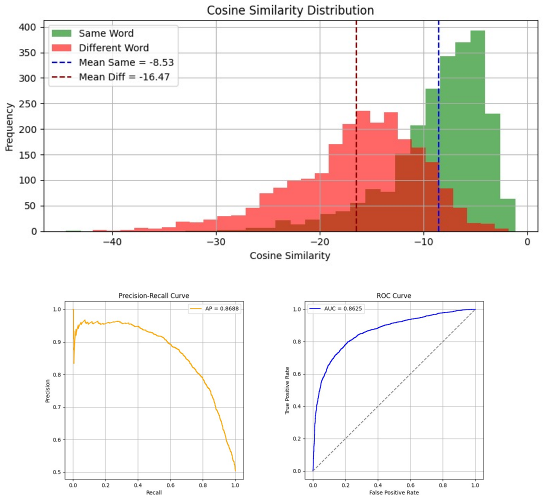


Figure: Intermediate 10th layer CTC hindi

Methods

- We have also tried getting encoder as well as decoder output from conformer CTC hindi model, but it was also giving low score for both the case, and we think that was because of the logits that we were getting from this encoder as well as decoder block of the model were totally different even for similar audio. However, this was not the case when we used the same methods for the English language.
- We have taken the encoder output directly from the parakeet CTC model, and then we applied DTW to get distance. It was not giving us the expected result.
- We have taken encoder output directly from model- Indic conformer-Hindi-RNNT(from AI for Bharat). We applied DTW on top of that. It was giving us low scores even when audios were similar, and also scores were of the same order when the audios were not similar.
- We have also tried to get output from the encoder and joint block output of this RNNT model, but we couldn't find any methods for that. We went through the source code of nemo but there was no method mentioned there by which we could get the output. It was asking for something we didn't have.

Methods

- We have used a traditional signal processing approach for audio like getting the MFCC feature and then applying DTW distance on top of that. Here, we were getting a low score when two audios are not similar and a high score when the audios were similar, but the difference between them was quite low, like 0.596 and so on.
- We have taken encoder output directly from model- Indic conformer-Hindi-RNNT(from AI for Bharat). We applied DTW on top of that. It was giving us low scores even when audios were similar, and also scores were of the same order when the audios were not similar.

Methods

- We have used pre-trained wav2vec model that is trained on many languages as a base model . We are getting embeddings from this models and then we are adding a projection layer on top of that . We are using InfoNCE loss for training . We have tried mean pooling as well as attention pooling here . We have trained model for 15 epoch and got the f1 score as 0.82 .
- We have also used ContrastiveMarginLoss for training on same model but the result was worst , f1 score of 0.62 .
- We have also build SiameseAudioModel with wav2vec2 Bert as base model . We use ContrastiveLossCosine for model training .
- We have taken encoder output directly from model- Indic conformer-Hindi-RNNT(from AI for Bharat). We applied DTW on top of that. It was giving us low scores even when audios were similar, and also scores were of the same order when the audios were not similar.

Final Approach

- For our main approach, we used a pre-trained multilingual **Wav2Vec2** model as the base encoder for extracting audio embeddings. To adapt it for our task, we **froze 80% of the lower transformer layers** and fine-tuned only the top 20% of the model. This strategy helped retain the general acoustic knowledge while allowing the model to specialize for our pronunciation similarity task.
- We added a **projection layer** on top of the encoder to map the extracted embeddings into a lower-dimensional space suitable for similarity comparison.
- The model was trained using the **InfoNCE (Information Noise Contrastive Estimation)** loss, which encourages embeddings from similar audio samples to be closer and dissimilar ones to be further apart in the latent space.
- We trained the model for **20 epochs**, and achieved a promising performance with an **F1 score of approximately 0.80**, indicating good discriminative power between correctly and incorrectly pronounced words.

Leaderboard

Metric	Best Result
TPR	0.8323
FPR	0.1666
TNR	0.8334
FNR	0.1677
Precision	0.8312
Recall	0.8323
F1	0.8318
CER	0.0
TER	-
WER	-
Optimal Threshold	0.9072
Number of Pairs	4768

Table: Results on Kathbath Validation Split.

Training Model using Using Intermediate Layer Output of Pre-trained model

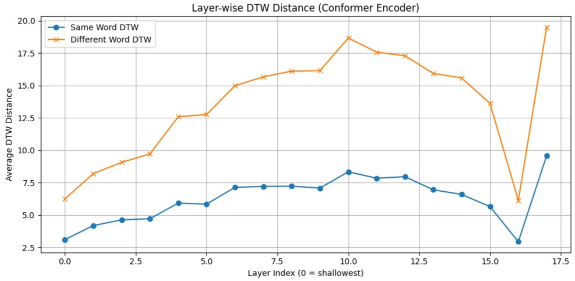
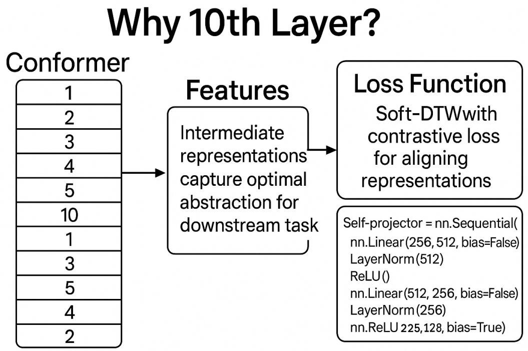


Figure: Layer wise DTW distance of the Output Embeddings

Training Model using Using Intermediate Layer Output of Pre-trained model


Why 10th Layer?
Conformer
1
2
3
4
5
10
1
3
5
4
2
Features
Intermediate representations capture optimal abstraction for downstream task
Loss Function
Soft-DTWwith contrastive loss for aligning representations

```
Self-projector = nn.Sequential(
  nn.Linear(256, 512, bias=False)
  LayerNorm(512)
  ReLU()
  nn.Linear(512, 256, bias=False)
  LayerNorm(256)
  nn.ReLU 225,128, bias=True)
```

Future Work

- We plan to extract **intermediate layer representations** from the Conformer CTC model to analyze whether these layers contain more fine-grained phonetic information compared to the final output.
- These intermediate embeddings will be used as input to a **Soft-DTW (Dynamic Time Warping)**-based alignment approach, which allows differentiable computation of similarity between two audio sequences.
- Using this alignment, we aim to compute a **DTW distance score** that is more sensitive to pronunciation variations and mismatches.
- Additionally, we will perform a **segment-level analysis** to identify which parts of a word are incorrectly pronounced. This will help in giving detailed and actionable feedback to the user for correction.

The End