DeepSpeech Therapy: A Hierarchical Deep Learning Framework for Real-Time Speech Therapy Assessment

Sahil Shivaji Sawant **sahilshi**

Fagun Nirag Patel **fagunnir**

Introduction

This project aims to develop a Deep Learning based speech-language therapy system that helps individuals with difficulty in speaking & pronunciation improve through interactive and personalized feedback. Inspired by platforms like Duolingo, the system leverages deep learning models to evaluate spoken input, detect pronunciation errors, and offer real-time corrective feedback.

Objectives

- Build an autoencoder deep learning model that can check if a person's speech is clear or defective
- For faulty speech, identify if the speech includes stutter and its related defects
- Give simple and helpful feedback to the user to improve their speaking.
- Create a simple setup that people can use to practice speaking anytime.
- Use a language model to give friendly and encouraging feedback to the user.

Progress

Dataset and Problem Setup

We used the LibriSpeech train-clean-100 subset, which consists of approximately 100 hours of clean, read English speech from 251 speakers. The dataset contains .flac audio files and corresponding .trans.txt transcript files. Each audio sample is assumed to be correctly pronounced, making the dataset ideal for modeling clear speech in an unsupervised manner.
Our goal is to detect deviations from clean speech since no labeled “incorrect” examples are introduced in the dataset.

Dataset links

Download link for train-clean-100:
http://www.openslr.org/resources/12/train-clean-100.tar.gz

Download link for train-clean-360 (used for unseen evaluation):
http://www.openslr.org/resources/12/train-clean-360.tar.gz

Preprocessing Pipeline

Audio Loading and Resampling:

All audio files are loaded using torchaudio.

Any audio not sampled at 16 kHz is resampled to a fixed rate of 16,000 Hz for consistency.

Feature Extraction with MFCC:
We extract Mel-Frequency Cepstral Coefficients (MFCCs), which are widely used in speech recognition and reflect the short-term power spectrum of the audio.

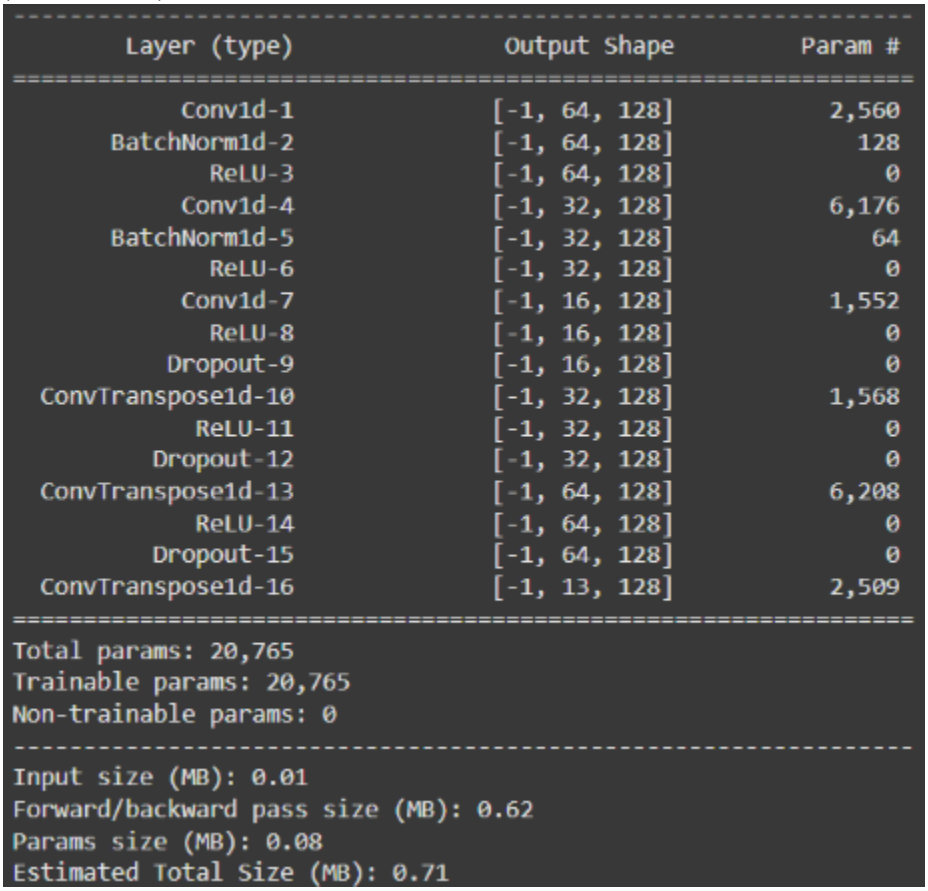
We implemented a 1D Convolutional Autoencoder, consisting of:
Encoder:
Three 1D convolutional layers with nn.Conv1d:

- Input channels: 13 → 64 → 32 → 16
- Activation: ReLU
- Normalization: BatchNorm1d
- Dropout: 0.2 (to reduce overfitting)

Decoder:
Three transposed convolutional layers with nn.ConvTranspose1d:

- Channels: 16 → 32 → 64 → 13
- Activation: ReLU
- Dropout: 0.2

The autoencoder learns to reconstruct its input (MFCCs) and is trained to minimize the Smooth L1 Loss (Huber Loss), which balances between L1 and L2 loss behavior and is robust to outliers.



Layer (type)	Output Shape	Param #
Conv1d-1	[-1, 64, 128]	2,560
BatchNorm1d-2	[-1, 64, 128]	128
ReLU-3	[-1, 64, 128]	0
Conv1d-4	[-1, 32, 128]	6,176
BatchNorm1d-5	[-1, 32, 128]	64
ReLU-6	[-1, 32, 128]	0
Conv1d-7	[-1, 16, 128]	1,552
ReLU-8	[-1, 16, 128]	0
Dropout-9	[-1, 16, 128]	0
ConvTranspose1d-10	[-1, 32, 128]	1,568
ReLU-11	[-1, 32, 128]	0
Dropout-12	[-1, 32, 128]	0
ConvTranspose1d-13	[-1, 64, 128]	6,208
ReLU-14	[-1, 64, 128]	0
Dropout-15	[-1, 64, 128]	0
ConvTranspose1d-16	[-1, 13, 128]	2,509

Total params: 20,765
Trainable params: 20,765
Non-trainable params: 0

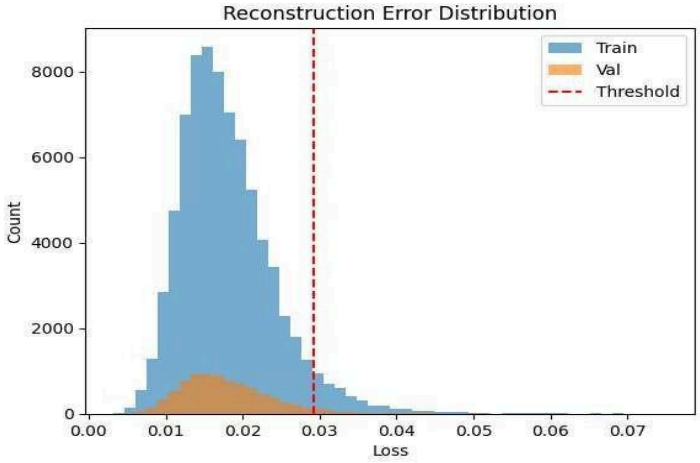
Input size (MB): 0.01
Forward/backward pass size (MB): 0.62
Params size (MB): 0.08
Estimated Total Size (MB): 0.71

Training Details

- Optimizer: Adam with learning rate 1e-3 and weight decay 1e-5.
- Scheduler: ReduceLROnPlateau reduces the learning rate by a factor of 0.5 after 2 stagnant validation loss epochs.
- Training Epochs: Trained for 30 epochs.
- Batch Size: 32.
- Early Stopping: Stops training if validation loss does not improve for 5 epochs.

Evaluation and Thresholding

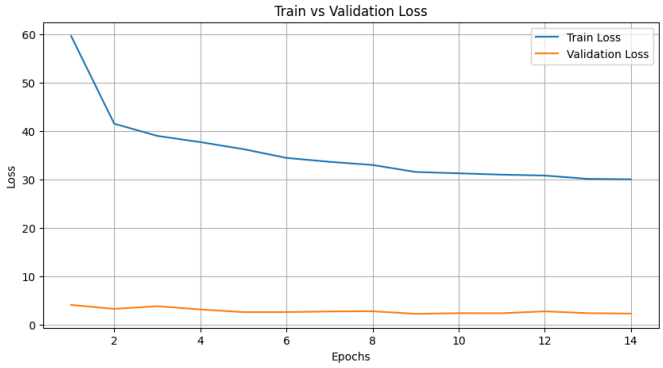
After training, errors are computed across the train and validation datasets. A dynamic threshold is set based on the 95th percentile of the training reconstruction errors, ensuring that 95% of clean speech samples are classified as "clear." Samples with errors above this threshold are flagged as "faulty or mispronounced."

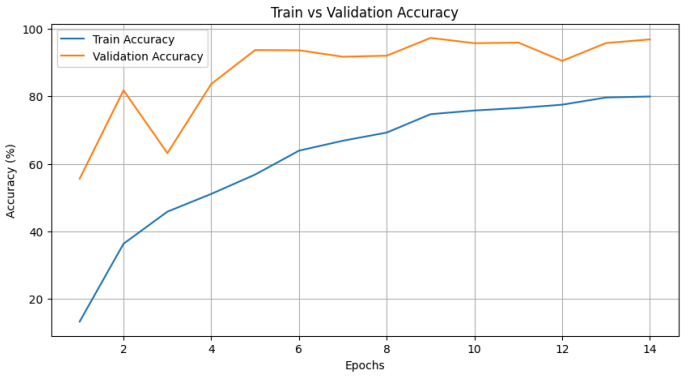


Results

The training run shows stable learning progression, where both loss and accuracy metrics confirm meaningful improvement in the model's ability to reconstruct clean speech and distinguish it based on reconstruction error. The training loss decreased from 59.56 to 29.98 across 14 epochs, while the validation loss also reduced notably from 4.05 to 2.26. The train accuracy improved significantly, rising from 13.21% to 79.96%, and the validation accuracy reached an impressive 96.89%, indicating that the model generalizes well.

There is some mild fluctuation in validation loss between epochs 10 and 13, but early stopping was correctly triggered once it stabilized, avoiding overfitting and preserving the best-performing model. The dynamic threshold of 0.0279, computed as the 95th percentile of training reconstruction errors, effectively delineates clear vs. faulty speech, enhancing the classifier's precision.





Evaluation Report

To assess the generalization ability of the trained speech clarity detection model, we first evaluated it on unseen clean audio samples from the LibriSpeech train-clean-360 dataset. These samples were not used during training or validation. The model demonstrated strong performance on this data, accurately identifying all test samples as "clear" with reconstruction errors well below the learned dynamic threshold of approximately 0.0279. This indicates that the model successfully generalizes to new clean speech recordings from similar distributions and speakers.

To further test the model's capability, we generated synthetic faulty audio samples by applying controlled distortions to clean LibriSpeech recordings. These distortions included adding Gaussian noise, altering playback speed, pitch shifting, and applying random volume changes. These transformations simulate real-world conditions such as mispronunciations, recording interference, or environmental noise. When evaluated on these faulty inputs, the model consistently reported higher reconstruction errors and correctly classified most samples as "faulty." This behavior highlights the model's sensitivity to variations in speech clarity and supports its potential use for automated speech quality assessment in educational or clinical applications.

```
Epoch 01 | Train Loss: 59.5560 | Val Loss: 4.0473 | Train Acc: 13.21% | Val Acc: 55.56%
Epoch 02 | Train Loss: 41.4648 | Val Loss: 3.2318 | Train Acc: 36.30% | Val Acc: 81.80%
Epoch 03 | Train Loss: 38.9487 | Val Loss: 3.7837 | Train Acc: 45.81% | Val Acc: 63.15%
Epoch 04 | Train Loss: 37.6622 | Val Loss: 3.1211 | Train Acc: 51.07% | Val Acc: 83.67%
Epoch 05 | Train Loss: 36.2145 | Val Loss: 2.5650 | Train Acc: 56.81% | Val Acc: 93.73%
Epoch 06 | Train Loss: 34.4032 | Val Loss: 2.5703 | Train Acc: 63.90% | Val Acc: 93.66%
Epoch 07 | Train Loss: 33.5975 | Val Loss: 2.7018 | Train Acc: 66.84% | Val Acc: 91.75%
Epoch 08 | Train Loss: 32.9449 | Val Loss: 2.7387 | Train Acc: 69.26% | Val Acc: 92.05%
Epoch 09 | Train Loss: 31.5036 | Val Loss: 2.2120 | Train Acc: 74.71% | Val Acc: 97.33%
Epoch 10 | Train Loss: 31.2167 | Val Loss: 2.3472 | Train Acc: 75.81% | Val Acc: 95.76%
Epoch 11 | Train Loss: 30.9509 | Val Loss: 2.3352 | Train Acc: 76.54% | Val Acc: 95.95%
Epoch 12 | Train Loss: 30.7564 | Val Loss: 2.7250 | Train Acc: 77.53% | Val Acc: 90.52%
Epoch 13 | Train Loss: 30.0806 | Val Loss: 2.3513 | Train Acc: 79.66% | Val Acc: 95.81%
Epoch 14 | Train Loss: 29.9824 | Val Loss: 2.2603 | Train Acc: 79.96% | Val Acc: 96.89%

Early stopping triggered. Restoring best model.
Dynamic threshold: 0.0279
```

```
Sample Predictions on Unseen LibriSpeech Data:
Clear — error=0.0162 — 2812-160187-0061.flac
Clear — error=0.0115 — 2812-160187-0052.flac
Clear — error=0.0098 — 2812-160187-0053.flac
Clear — error=0.0189 — 2812-160187-0011.flac
Clear — error=0.0166 — 2812-160187-0081.flac
Clear — error=0.0152 — 2812-160187-0034.flac
Clear — error=0.0233 — 2812-160187-0072.flac
Clear — error=0.0213 — 2812-160187-0023.flac
Clear — error=0.0164 — 2812-160187-0086.flac
Clear — error=0.0181 — 2812-160187-0087.flac
```

After adding some form of distortions see results below

```
Clear — error=0.0259 — 2812-160187-0061.flac
Faulty — error=0.0509 — 2812-160187-0052.flac
Faulty — error=0.0500 — 2812-160187-0053.flac
Faulty — error=0.0644 — 2812-160187-0011.flac
Faulty — error=0.1489 — 2812-160187-0081.flac
Clear — error=0.0157 — 2812-160187-0034.flac
Faulty — error=0.0321 — 2812-160187-0072.flac
Faulty — error=0.1085 — 2812-160187-0023.flac
Clear — error=0.0164 — 2812-160187-0086.flac
Clear — error=0.0134 — 2812-160187-0087.flac
```

Part 2: Experimenting with faulty speech - Multiclass stuttering detection

As a foundational step, this work focuses exclusively on a single, well-defined impairment: stuttering. The objective was to experiment with establishing a robust data processing and modeling pipeline capable of classifying different types of stuttering events from audio clips. This focused approach allows for rapid experimentation with several models and the development of a strong methodological framework before scaling up to include other speech disorders.

Dataset and Data Preparation

Dataset Description & Link **https://www.kaggle.com/datasets/ikrbasak/sep-28k**

For this task, we utilized the SEP-28k Stuttering Events in Podcasts Dataset, augmented with data from FluencyBank. This combined dataset provides a large and diverse collection of audio clips sourced from real-world podcasts featuring people who stutter.

- Total Data: The dataset contains metadata for approximately 32,000 3-second audio clips. After cleaning and removing files with poor audio quality, music, or no speech, we obtained a working set of 26,098 usable audio clips.
- Data Format: The data is provided as a set of .wav audio files and corresponding CSV files containing metadata and labels.
- Labels and Classes: Each clip is annotated by multiple raters for the presence of specific disfluencies. For our multi-class classification task, we identified six primary classes:
    - Block: Abrupt pauses or inability to produce sound.
    - Prolongation: Elongation of a sound or syllable.
    - SoundRep: Repetition of a single sound (e.g., "p-p-p-prepare").
    - WordRep: Repetition of a whole word or phrase.
    - Interjection: Use of filler words like "um" or "uh".
    - NoStutteredWords: Fluent speech with no disfluencies.

Data Loading and Preprocessing

The entire pipeline was developed in a Google Colab environment.

- Data Acquisition: The dataset was downloaded directly into the notebook environment using the Kaggle API. This method is efficient and reproducible, avoiding manual uploads.
- Data Cleaning: We combined the label files from SEP-28k and FluencyBank. We then constructed the full file path for each audio clip by concatenating the Show, EpId, and ClipId fields from the metadata. A crucial step was filtering out any entries for which the corresponding .wav file did not exist on disk.

- Label Processing: The dataset provides counts from multiple annotators for each label. To create a single ground-truth label for our multi-class problem, we selected the class with the highest annotator count for each clip (idxmax function).
- Handling Class Imbalance: A preliminary analysis revealed a significant class imbalance, with NoStutteredWords being the dominant class. To address this, we implemented a weighted loss function. The sklearn.utils.class_weight.compute_class_weight function was used to calculate weights that give higher importance to minority classes (like Block and SoundRep) during training, preventing the model from simply defaulting to the majority class.

Methodology: From Baseline to Advanced Pipeline

Our approach was iterative, starting with a simple model to establish a baseline and progressively moving to more complex, state-of-the-art architectures.

Advanced Experimental Pipeline

We developed a multi-phase experimental pipeline aimed at exploring more robust model architectures. Given time constraints, we limited training to just 3 epochs on a randomly selected subset of 4000 clips. The data was divided into training (80%), validation (10%), and a final, unseen test set (10%).

- **CRNN Resnet 18 + Bi-LSTM:** While CNNs are excellent at identifying what features are in a spectrogram, they treat it as a static image. They miss the crucial temporal context of speech. An LSTM is designed to process sequences, allowing the model to understand the timing and order of speech events, which is fundamental to identifying disfluencies. The spectrogram was first passed through the ResNet1s convolutional base to extract a sequence of feature vectors. This sequence was then fed into a Bi-LSTM layer to learn the temporal patterns before making a final classification.

- **Resnet 18:** We transitioned from a simple CNN to a more sophisticated deep convolutional architecture wherein we used a Resnet 18 model. Utilizing a model with strong feature extraction capabilities allowed us to more effectively capture patterns in the spectrogram representations of speech compared to a simple CNN model. When applied to log-mel spectrograms which are visual representations of audiothis model could effectively identify temporal and frequency-based speech patterns, leading to improved learning performance with minimal training.

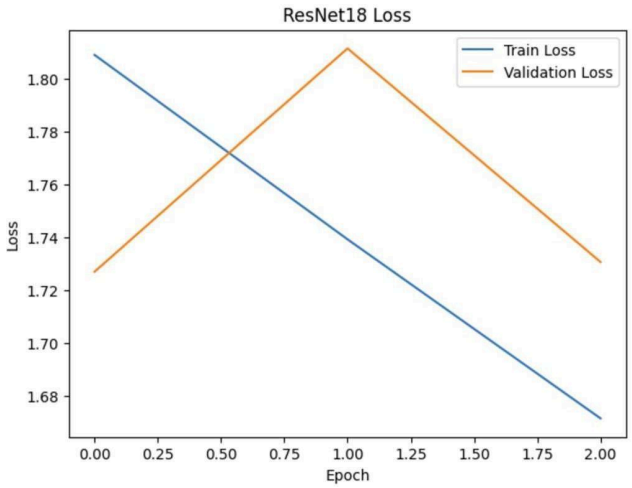
- **CNN with Attention Mechanism:** As CNNs are effective at capturing local frequency-time patterns in spectrograms, they lack the ability to model long-range dependencies in the audio. To address this, we extended a basic CNN architecture by incorporating a self-attention mechanism. The model first applies a series of 1D convolutions over the log-mel spectrogram input to extract local features across time. These feature maps are then passed to a multi-head self-attention layer, which enables the model to learn contextual relationships between distant time steps, a crucial aspect for understanding speech disfluencies that often depend on broader sequence context. The attention-enhanced sequence is refined through a residual connection and layer normalization, followed by global average pooling and fully connected layers to make the final prediction. This architecture combines the local pattern detection of CNNs with the global sequence modeling capability of attention, resulting in a more better understanding of stuttering events spread across time.

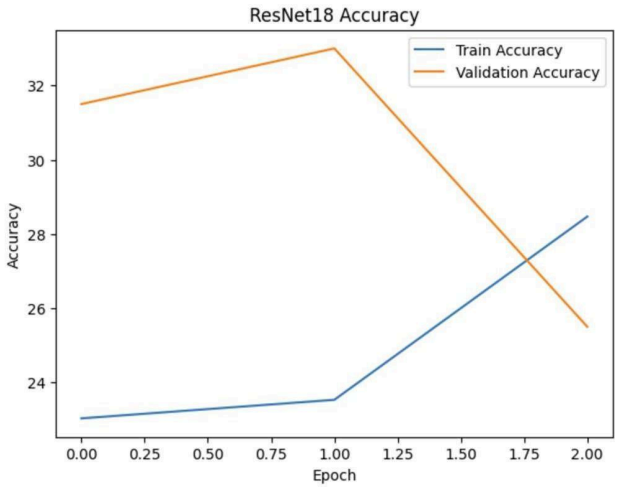
- **Conv1D Autoencoder + Transformer mechanism:** To isolate subtle patterns in speech disfluencies, we adopted a two-stage approach that combines the strengths of unsupervised feature learning and powerful sequence modeling. The first stage uses a convolutional autoencoder to compress log-mel spectrograms into a lower-dimensional sequence of latent embeddings. The encoder extracts key time-frequency features while the decoder reconstructs the original input, ensuring that the learned representations preserve meaningful structure. Once trained, the encoder outputs are passed to a Transformer-based classifier, which models temporal dependencies across the entire audio sequence. Unlike CNNs that focus on local patterns, the Transformer uses self-attention to relate information across time steps, making it well-suited for detecting disfluencies that may depend on both short- and long-range temporal cues.

Results and Discussion

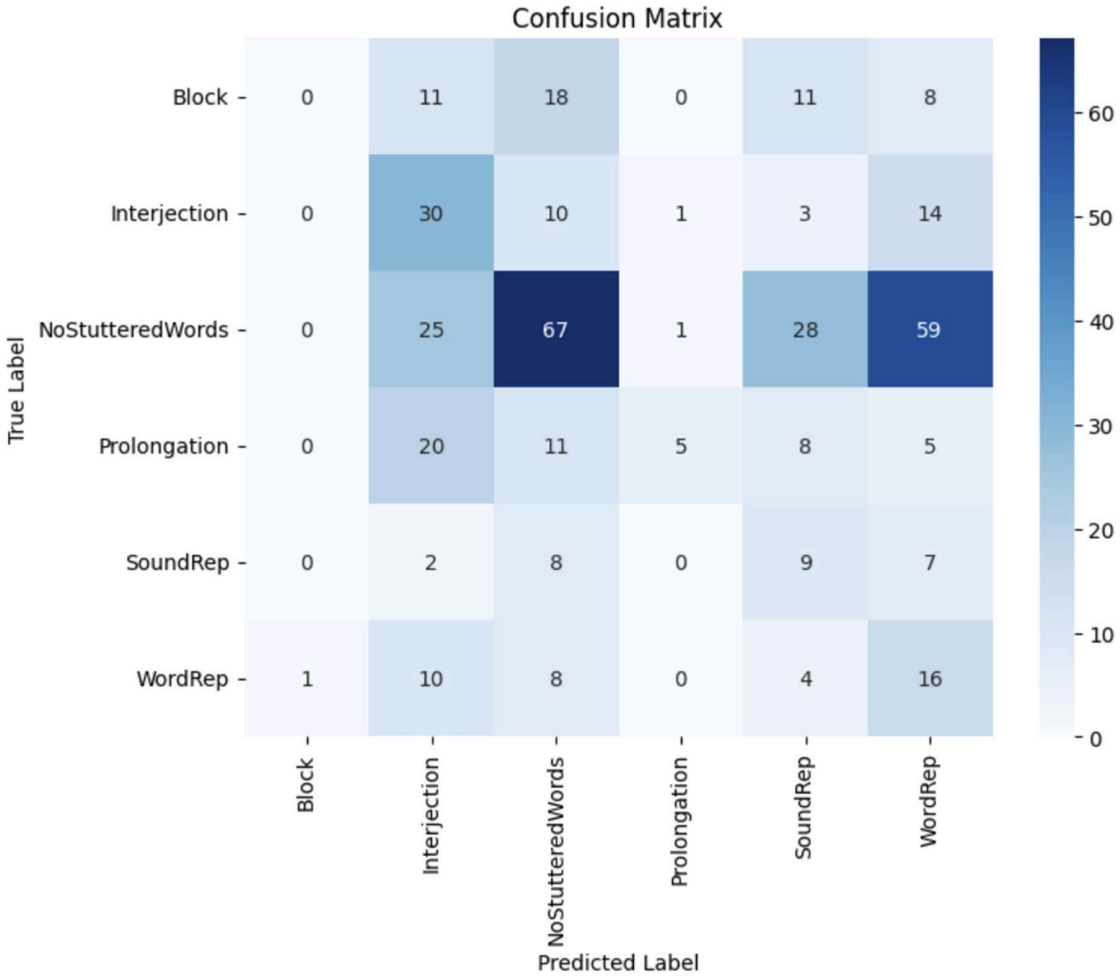
Advanced Pipeline Results
The results from our experimentation provided critical insights

Test Accuracy: 31.75%
Analysis: The ResNet model significantly outperformed random guessing, demonstrating the power of transfer learning. The confusion matrix shows it began to learn features for Interjection and WordRep. However, the training-validation accuracy plots show clear signs of overfitting, where the model started memorizing the training data instead of generalizing. It still struggled with the highly imbalanced nature of the data.



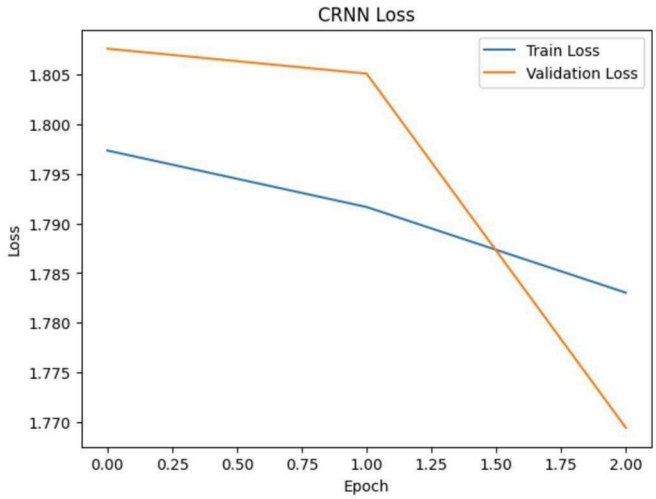


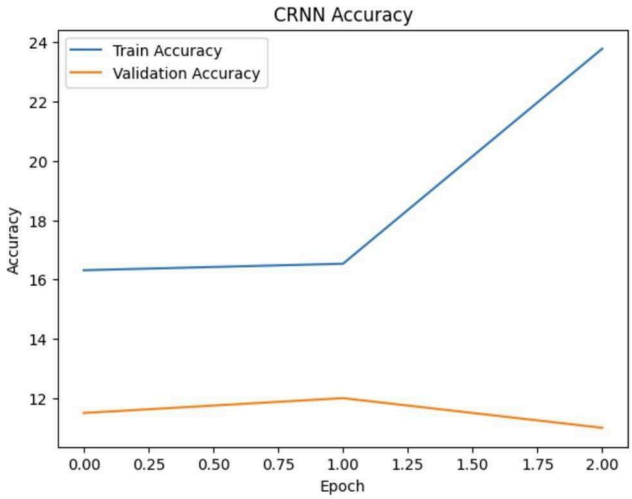
```
Evaluating model from best_resnet_model.pth on the test set
Test Accuracy: 0.3175

Classification Report:
                  precision    recall  f1-score   support

           Block       0.00      0.00      0.00        48
    Interjection       0.31      0.52      0.38        58
NoStutteredWords       0.55      0.37      0.44       180
    Prolongation       0.71      0.10      0.18        49
       SoundRep        0.14      0.35      0.20        26
        WordRep        0.15      0.41      0.22        39

        accuracy                           0.32       400
       macro avg       0.31      0.29      0.24       400
    weighted avg       0.40      0.32      0.31       400
```



Test Accuracy: 11.5%

Analysis: The CRNN performed worse than the ResNet and even worse than random guessing. This is a crucial insight: while theoretically more powerful, this complex hybrid architecture is extremely data-hungry. With only 3 epochs on a small data subset, it did not have sufficient training to learn meaningful temporal patterns and failed to converge.

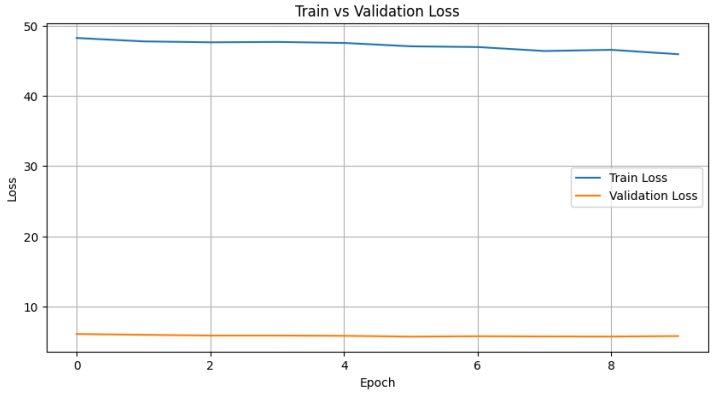


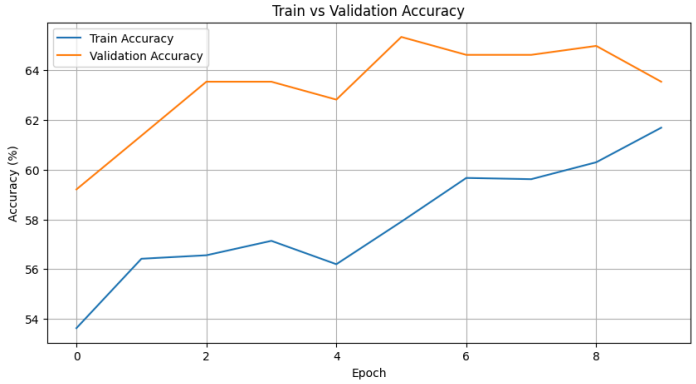


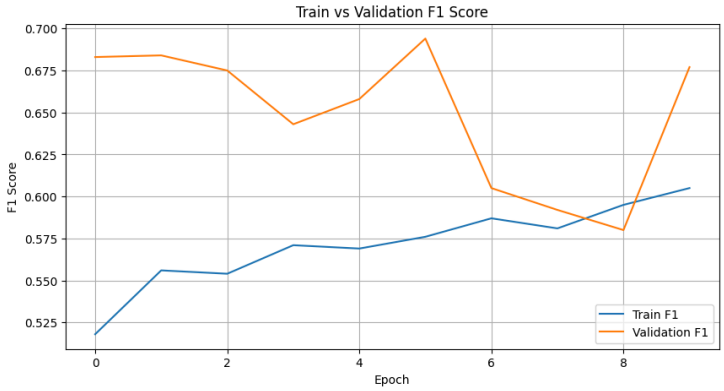

```
Evaluating model from best_crnn_model.pth on the test set
Test Accuracy: 0.115

Classification Report:
                  precision    recall  f1-score   support

           Block       0.08      0.19      0.11        48
    Interjection       0.16      0.31      0.21        58
NoStutteredWords       0.00      0.00      0.00       180
    Prolongation       0.00      0.00      0.00        49
       SoundRep        0.00      0.00      0.00        26
        WordRep        0.11      0.49      0.18        39

        accuracy                           0.12       400
       macro avg       0.06      0.16      0.08       400
    weighted avg       0.04      0.12      0.06       400
```

Test Accuracy: 59.35%
Analysis: The CNN + Attention model steadily improved over the training epochs, reaching a peak validation F1-score of 0.694 before early stopping was triggered. On the test set, the model achieved an F1-score of 0.648, showing a good balance between precision and recall. The confusion matrix highlighted that the model effectively identified disfluent speech, particularly improving recall for the stuttered class.

While the overall accuracy was modest, the attention mechanism helped capture longer-range temporal dependencies that traditional CNNs often miss. This led to a more consistent identification of disfluency patterns such as blocks and prolongations, even in shorter or noisier clips. The results indicate that integrating attention with convolution allows the model to make more informed predictions by considering both local features and global context.





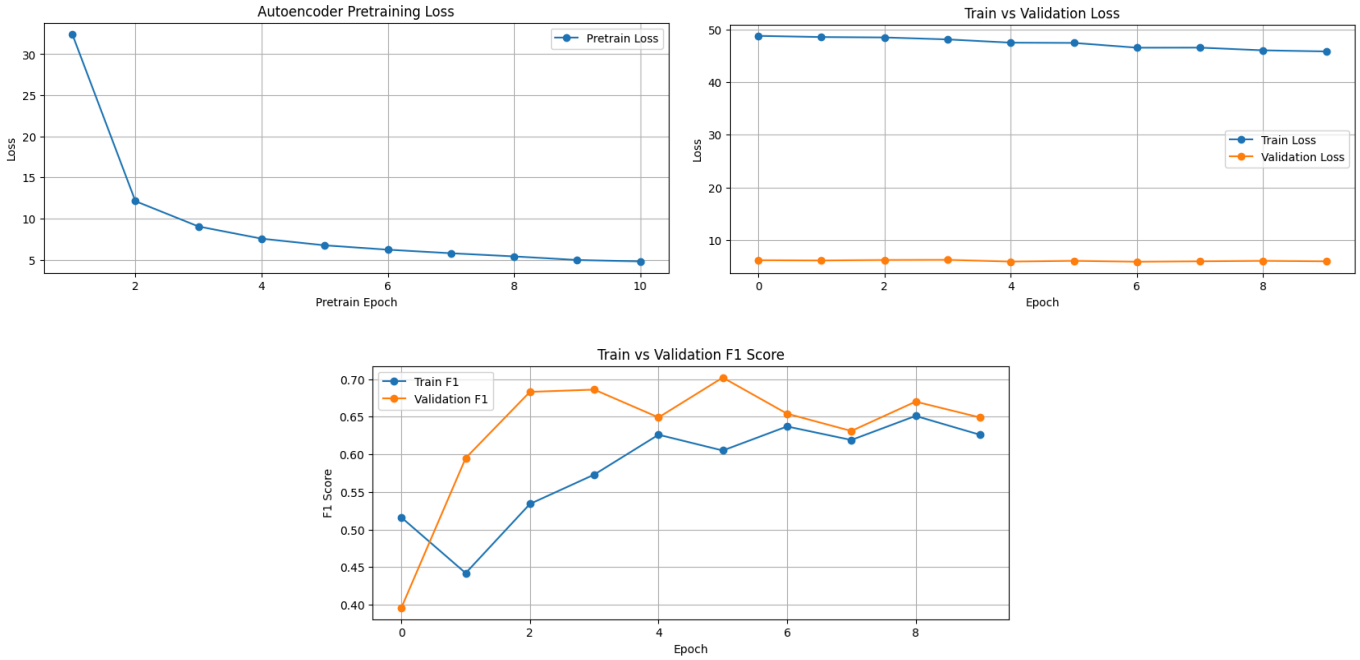


Test Accuracy: 55.76%
Analysis: The ConvAutoencoder + Transformer model achieved a peak validation F1-score of 0.702 before early stopping was triggered. On the test set, it recorded an F1-score of 0.670, indicating strong performance in identifying stuttered speech, particularly with high recall for disfluent segments.

While the overall accuracy was moderate, the model demonstrated a clear advantage in modeling speech disfluencies. The autoencoder learned robust, compressed representations of the spectrograms, which the Transformer then processed to capture long-range temporal patterns. This combination helped the model effectively distinguish stuttered speech even when cues were subtle or distributed across time.

The confusion matrix reflected this strength: although the model had lower precision for the fluent class, it achieved high recall (89.29%) for the disfluent class. This suggests the model was highly sensitive to detecting disfluency, making it particularly useful for screening scenarios where missing stuttered instances is more

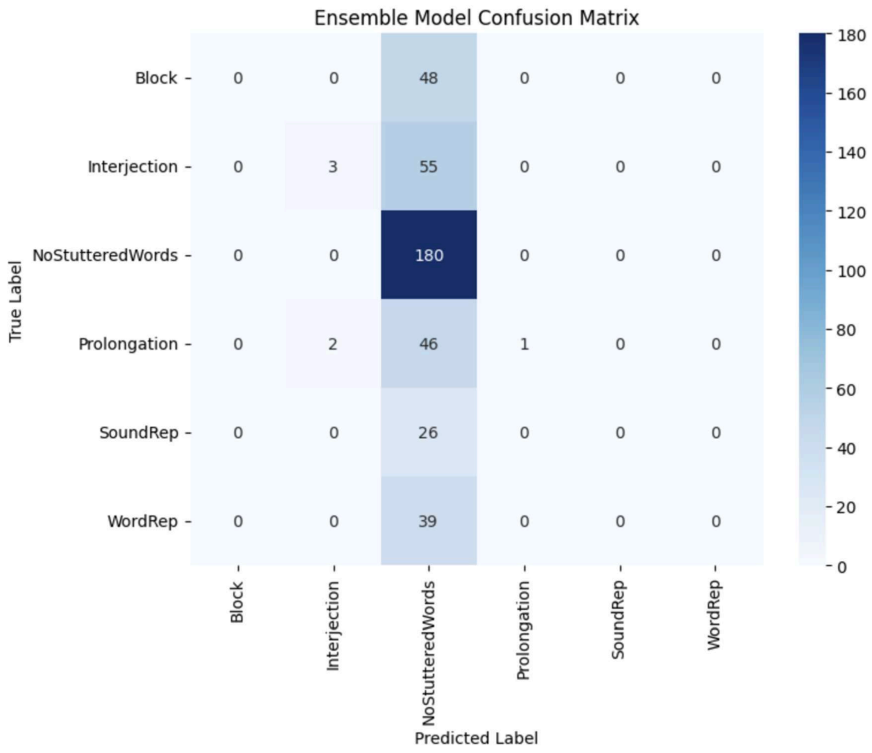
critical than over-predicting them.



Test Accuracy: 46.0%

Analysis: The ensemble model achieved the highest overall test accuracy. The confusion matrix shows a slight but important improvement. While still heavily biased towards NoStutteredWords (due to the influence of Wav2Vec2), it correctly identified a few instances of Interjection and Prolongation that the individual Wav2Vec2 model missed. This demonstrates that even when one model is dominant, the diverse perspectives of other models can contribute to a better, more robust final decision.

```
Ensemble Evaluation Results
Test Accuracy: 0.46

Classification Report:
                precision    recall  f1-score   support

         Block       0.00      0.00      0.00        48
    Interjection     0.60      0.05      0.10        58
NoStutteredWords     0.46      1.00      0.63       180
    Prolongation     1.00      0.02      0.04        49
       SoundRep      0.00      0.00      0.00        26
        WordRep      0.00      0.00      0.00        39

        accuracy                         0.46       400
       macro avg     0.34      0.18      0.13       400
    weighted avg     0.42      0.46      0.30       400
```



Test Accuracy: 93.0%
Analysis: The wav2vec pretrained binary classification model achieved high overall accuracy, driven largely by its ability to correctly identify Control speech. The confusion matrix and classification report revealed that the model was highly biased toward the Control class, achieving perfect recall (1.00) and a strong F1-score (0.97) for it.

However, performance on the Dysarthric class was significantly weaker, with a recall of only 0.06 despite a perfect precision of 1.00. This indicates that while the model is highly cautious about labeling speech as Dysarthric (leading to few false positives), it fails to detect the majority of actual Dysarthric samples.

The results suggest that the model, although pretrained, lacks the necessary generalization for pathological speech patterns. This highlights the need for domain-specific fine-tuning or balancing techniques. The current performance reflects that pretrained models excel at fluent speech classification, but adapting them for clinical speech disorders like Dysarthria requires targeted training and diverse representation in the dataset.

We include this as a supplementary material to use pretrained models to understand how their performance works in our case.

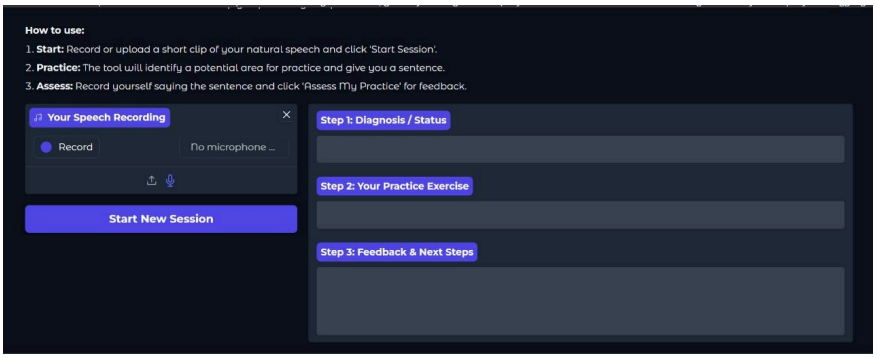
LLM-based Feedback using Gradio:

We implemented a Large Language Model (LLM)-based feedback mechanism designed to provide personalized, real-time guidance to users on how to improve their speech fluency. Instead of just labeling a segment as disfluent, the LLM interprets the context and generates natural language suggestions on how to correct or enhance the delivery.

For example, if a user frequently repeats words (e.g., “I-I-I want to go”), the system might respond with: “Try to take a brief pause before speaking to avoid word repetitions. Think through your sentences mentally before starting.”

By offering these context-aware, constructive suggestions, the LLM-based system acts like a virtual speech coach, helping users not only recognize their speech patterns but also actively work on improving fluency, pacing, and clarity over time.

This was deployed using Gradio



Conclusion and Future Work

This project successfully established a comprehensive, multi-stage pipeline for stuttering event detection. We demonstrated the limitations of basic models and systematically explored the capabilities of modern, state-of-the-art architectures including ResNet, CRNNs, and Transformers. Our experiments confirmed that this is a challenging task due to severe class imbalance.

The key insight is that while advanced models like Wav2Vec2 hold immense promise, they require extensive training on large datasets to perform effectively. Our rapid experimentation on a small subset provided a valuable proof-of-concept and a clear direction for future work.

In future, our plan is to:

- Scale Up: Apply the established pipeline to the entire dataset collection of multiple impairments such as
  DYSPHONIA
  APRAXIA
  DYSARTHRIA
  TBI
  DEMENTIA
  DYSPHAGIA.
- Train Extensively: Increase the number of training epochs significantly (e.g., 20-50) to allow the more complex ConvAutoencoder, CNN+Attention models and Wav2Vec2 models to converge properly.
- Hyperparameter Tuning: Systematically tune parameters like learning rate and batch size to optimize the performance of each model.

- Expand Scope: Ultimately, we aim to apply this robust methodological framework to a larger, custom-collected dataset encompassing a variety of speech impairments, moving closer to our goal of a comprehensive speech therapy system.
- Introduce gamification based Feedback wherein a user gets points for correctly speaking the prompted sentence

Contributions

Name	Project Part	Contribution
Sahil Sawant	Dataset collection for clean speech	100%
Fagun Patel	Dataset collection for various speech problems	100%
Sahil Sawant	Model building, training and evaluation for finding clean speech and faulty speech	100%
Fagun Patel	Model building, training and evaluation for finding anomalies in speech using Resnet & CRNN & ensemble model	100%
Sahil Sawant	Model building, training and evaluation for finding anomalies in speech using CNN+Attention & Conv1D Autoencoder + Transformer	100%
Fagun Patel	LLM Based feedback and model deployment	100%

Trello board

https://trello.com/invite/b/6859bc92be592c12088e8ded/ATTIa9cd61b45d94ed0d39536b78f91d0460FDDCEE88/dl-project

References

1. https://ieeexplore.ieee.org/document/10178807
2. https://ieeexplore.ieee.org/document/10916276
3. https://ieeexplore.ieee.org/document/10922076
4. https://www.kaggle.com/datasets/ikrbasak/sep-28k
5. https://www.clarin.eu/resource-families/corpora-disordered-speech
6. https://www.openslr.org/12
7. https://www.kaggle.com/code/norangalalshehata/wav2vec2-sep-28k
8. https://github.com/jim-schwoebel/voice_datasets?tab=readme-ov-file
9. https://huggingface.co/