# Speech Understanding Major

Neermita Bhattacharya
B22CS092
IIT Jodhpur
b22cs092@iitj.ac.in

## Abstract

*This research tackles core problems in speech processing via an end-to-end pipeline for three principal areas: speech improvement, multilingual transcription, and customized voice synthesis. In the initial section, I implement an end-to-end pipeline for cross-lingual lecture processing. I transcribe a code-switched lecture (English-Hindi), suppress automatically filler words, translate content into Bengali (low-resource language), and regenerate speech with customized voice qualities. I use Whisper and Google STT for transcription, Google Translate for translation, Google TTS for speech generation from text, and voice cloning process with tailored Bengali samples, I generate individualized speech output. Objective measures (WER, CER, PESQ) and subjective assessments (MOS) confirm the quality of every processing step. In the second half, I improve speech readability in noisy scenarios by analyzing and denoising audio recordings and prioritizing the moderator's voice. I perform fine-grained noise analysis using Signal-to-Noise Ratio (SNR) and spectral characteristics to measure disturbances. Spectral Subtraction and Wiener filter-based denoising algorithms reduce back-ground noise without degrading speech quality. Objective evaluation using metrics such as SNR improvement and Perceptual Evaluation of Speech Quality (PESQ) verifies significant audio clarity enhancements. The entire workflow allows for better speech understanding in adverse environments and cross-lingual content accessibility with speaker-specific voice features. Finally, I discuss about a possible research challenge called 'Multimodal Deception Detection in Conversational Speech' explaining the current limitations and giving possible solutions.*

## 1. Question 1

## 2. Introduction

For this question, I have implemented an end-to-end pipeline for cross-lingual lecture processing in four stages. First, I selected a lecture from the Speech Understanding course (40 minutes of the last lecture on Speech Synthesis) and used a speech-to-text model (Whisper) to transcribe its English–Hindi code-switched audio, automatically filtering out filler words after the transcription ("um," "uh," etc.). Next, I used Bengali as the low-resource language and translated the cleaned transcript with Google Translate Client first and then converted it to speech using Google's Text-to-Speech model into a basic Bengali version. Then, I created 10 samples of me talking in Bengali to retrieve specific features to transform the basic Bengali audio into an audio that sounds like me. I evaluated the generated speech quality Mean Opinion Score (MOS) and PESQ (over the first 10 minutes of both audios), and assessed transcription accuracy with WER/CER against a reference that was made using the Google Cloud STT API. The links to the basic Bengali generated audio and the Bengali generated audio in my voice are here - Basic Bengali Audio, Bengali Audio in my voice. The transcriptions can be found at original lecture transcript (Whisper), cleaned lecture transcript (Whisper), base transcript (Google Cloud STT) and cleaned base transcript (Google Cloud STT). The link to github is here.

The capability to handle multilingual speech content efficiently is an important issue in the context of speech understanding, especially for handling code-switched audio and low-resource language. I discuss an end-to-end pipeline that has I have created for cross-lingual lecture processing to handle speech transcription, filler word deletion, translation, and individualized audio generation.

The goal is to transcribe a code-switched lecture (English and Hindi), translate the filtered content into a low-resource language (Bengali), and resynthesize the lecture audio using my own voice in the target language. This process tackles several real-world issues, such as managing code-switching, improving transcription readability, and making inclusive audio generation possible in under-represented languages.

To this end, I used Whisper [5] for precise transcription

and filler word removal with auto-detection, Google Cloud STT for baseline transcription, Google Translate Client for translation, Google TTS for basic Bengali speech generation, and voice cloning procedures with Bengali samples to generate customized speech. Both objective measures (Word Error Rate, Character Error Rate, PESQ) and subjective measures (Mean Opinion Score) were used for evaluation to determine the quality and precision of each phase. In speech and audio processing, various evaluation metrics are used to assess the quality and intelligibility of generated or transmitted speech. Below are some commonly used metrics:

### 2.1. Mean Opinion Score (MOS)

The **Mean Opinion Score (MOS)** is a numerical measure of the perceived quality of speech signals, typically obtained by averaging the subjective ratings given by human listeners. It ranges from 1 (Bad) to 5 (Excellent).

$$\text{MOS} = \frac{1}{N}\sum_{i=1}^{N} R_i \tag{1}$$

where $R_i$ is the rating given by the $i^{th}$ listener, and $N$ is the total number of listeners.

### 2.2. Perceptual Evaluation of Speech Quality (PESQ)

**PESQ** is an objective method to predict the perceived quality of speech, as it would be rated by a human listener. It compares an original (reference) signal with a degraded version and outputs a score usually in the range of 1.0 to 4.5.

PESQ is standardized in ITU-T Recommendation P.862. Although the detailed algorithm is complex, the output can be denoted as:

$$\text{PESQ} = f(x(t), \hat{x}(t)) \tag{2}$$

where $x(t)$ is the original speech signal and $\hat{x}(t)$ is the degraded signal.

### 2.3. Word Error Rate (WER)

**WER** is a common metric for evaluating the performance of automatic speech recognition (ASR) systems. It is calculated as the number of edit operations required to match the hypothesis (predicted text) to the reference (true text), normalized by the number of words in the reference.

$$\text{WER} = \frac{S + D + I}{N} \tag{3}$$

where $S$ is the number of substitutions, $D$ is the number of deletions, $I$ is the number of insertions, and $N$ is the total number of words in the reference.

### 2.4. Character Error Rate (CER)

**CER** is similar to WER but operates at the character level instead of the word level. It is useful for languages without clear word boundaries.

$$\text{CER} = \frac{S_c + D_c + I_c}{N_c} \tag{4}$$

where $S_c$, $D_c$, and $I_c$ are the number of character-level substitutions, deletions, and insertions respectively, and $N_c$ is the total number of characters in the reference.

## 3. Transcription

The model chosen for transcription was Whisper. The lecture on Speech Synthesis was selected as the video. The command `ffmpeg -i "Speech Understanding Course - 2025_04_07 17_52 GMT+05_30 - Recording.mp4" -t 00:40:00 -c copy output_40min.mp4` was used to recover 40 minutes of the video (since transcription took too long otherwise). Next, the command `ffmpeg -i output_40min.mp4 -q:a 0 -map a audio.mp3` was used to convert it into mp3 format. Now, it is suitable for passing through Whisper. This was the unclean transcript with many filler words like ”um”, ”uh”, ”so”, etc. Since there was code switching happening in the audio, both English as well as Hindi filler words were removed. Unfortunately, it was noticed that Whisper wasn't able to produce the Hindi sentences at all. After Sir says 'Can you interpret the second term over here?' he speaks in Hindi 'Peeche se hota hai shayad. Idhar nahi hai.' But, it was translated as 'Which I mentioned previously the name of the second termading will not have any issues.' It is clear that it produced garbage transcriptions.

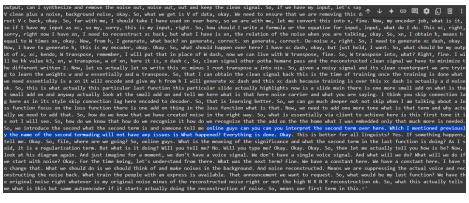

Figure 1. Example of Hindi statements not transcribed

## 4. Audio Generation

The `extract_voice_samples` function takes a zip file of 10 Bengali samples that I spoke, and extracts it to a given destination folder. These audio clips will be utilized to obtain important features needed to transform the Bengali audio into my voice.

## 4.1. Voice Feature Extraction

The voice samples, provided in `.mp3` format, are converted to `.wav` using the `pydub` library. These converted waveforms are then processed using the `librosa` library.

### 4.1.1 Pitch Estimation

I used **librosa's PYIN algorithm** to estimate the fundamental frequency ($F_0$), which provides accurate pitch estimation.

$$F_0 = \text{librosa.pyin}(y, f_{\min}, f_{\max}, sr) \tag{5}$$

Where:

- $y$ is the audio waveform.
- $sr$ is the sampling rate.
- $f_{\min} = \text{librosa.note\_to\_hz('C2')}$
- $f_{\max} = \text{librosa.note\_to\_hz('C7')}$

Unvoiced frames and zero-valued pitches are filtered out to compute the average pitch across the sample:

$$\text{pitch}_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} F_0^{(i)}, \quad \text{where } F_0^{(i)} > 0 \tag{6}$$

### 4.1.2 Speech Rate Estimation

The speech rate is approximated by analyzing the signal envelope and counting the number of syllable-like peaks:

1. The audio amplitude envelope is computed as $|y(t)|$, then pre-emphasized for showing sharp transitions.
2. Peaks in this smoothed envelope are picked using `librosa.util.peak_pick()`, assuming each peak represents a syllable.
3. The speech rate is defined as:

$$\text{Speech Rate} = \frac{\text{Number of Peaks}}{\text{Duration (in seconds)}} \tag{7}$$

### 4.1.3 Feature Normalization

To enable comparison or conditioning of these features in downstream audio generation, we compute normalized pitch and speed factors:

$$\text{Pitch Factor} = \frac{\text{pitch}_{\text{avg}}}{160}, \quad \text{Speed Factor} = \frac{\text{Speech Rate}}{4.0} \tag{8}$$

These reference values (160 Hz for pitch, 4 syllables/sec for rate) are representative of average adult human speech.

### 4.1.4 Statistical Summary

In addition to the normalized values, the following summary statistics are computed:

- Minimum, Maximum, and Variation in pitch
- Minimum, Maximum, and Variation in speech rate

This statistical insight helps determine the speaker's pitch range and speaking dynamics, and will be useful for style transfer.

### 4.1.5 Translate to Bengali

The `basic_bengali_tts` function uses the Google Translate Client to translate the transcribed lecture text into Bengali. Then, that text is used by Google TTS to convert it into Bengali speech. `temp_bengali_tts.mp3`

### 4.1.6 Adapting to my voice

`adapt_voice_simple` function modifies a given audio file by changing its pitch and speed based on specified voice features. It is used to simulate voice adaptation, such as making a voice sound higher/lower or faster/slower.

**Parameters:**

- `tts_audio_path`: Path to the input TTS audio file.
- `voice_features`: A dictionary with two values:
  - `pitch_factor`: Multiplier for pitch adjustment.
  - `speed_factor`: Multiplier for speed adjustment.
- `output_file`: Path to save the final voice-adapted MP3 file.

**How the function works:**

1. Loads the audio file using `librosa`.
2. Converts the `pitch_factor` into semitones and applies pitch shift using `librosa.effects.pitch_shift`.
3. If the `speed_factor` is significantly different from 1, it applies time stretching using `librosa.effects.time_stretch`.
4. Saves the modified audio temporarily as `temp.wav`.
5. Converts the WAV to MP3 using `pydub`'s `AudioSegment`.
6. Deletes the temporary file and returns the output path.

**Returns:** A string containing the path to the output MP3 file with the adapted voice. It is saved as `bengali_with_my_voice.mp3`

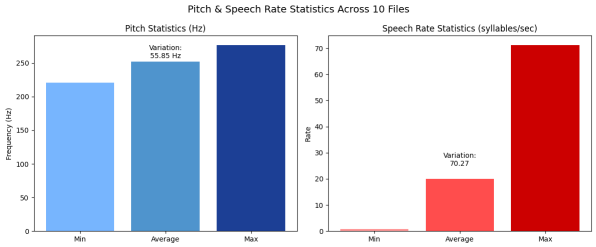


Figure 2. Pitch Statistics and Speech Rate Statistics of my voice

## 5. Performance Evaluation

#### 5.0.1 Bilingual Audio Transcription using Google Cloud Speech-to-Text

To perform transcription of potentially bilingual audio content, we implemented a chunk-based transcription strategy using Google Cloud's Speech-to-Text (STT) API. The function processes long audio files by splitting them into manageable chunks and sending them individually to the STT service.
The steps of the method are outlined below:

- The input audio file is loaded using `pydub`'s `AudioSegment`, and converted to mono to ensure compatibility. The audio is divided into equal chunks of 30 seconds, to avoid exceeding the 10 MB payload limit of the STT API.
- Each audio chunk is exported as a temporary WAV file in Linear16 format, as required by the STT API. This chunk is then read as raw bytes to be used in the request.
- A recognition request is constructed with bilingual support by specifying `en-US` as the primary language and `hi-IN` as an alternative. This encourages the model to understand code-switches in conversation.
- Each chunk is sent to the API using the synchronous `recognize()` method of the Google Cloud STT client. The transcribed text from all chunks is collected and appended to the output file.

Using this method, the lecture audio was transcribed into a reference base transcript which will be used for calculating WER and CER wrt the Whisper model's transcription.

**Final Evaluation Results**

- 10 minutes of the audio was extracted from 3 sources. The original lecture mp3 (English + Hindi), the basic Bengali speech, and the generated Bengali with my voice characteristics.
- PESQ score between the Google TTS Bengali speech and the transformed Bengali (to my voice): 1.041
- PESQ score between original lecturer's speech (English + Hindi) and transformed (to my voice) Bengali speech: 1.282
- The greater PESQ score between the converted Bengali speech (altered to suit my voice) and the original lecturer's audio (English+Hindi, male) than the score with the female Bengali TTS might appear counterintuitive at first. But this can be explained by several reasons. First, both the lecture recording and the converted speech probably have more prosodic and temporal structures, i.e., phrasing, intonation, and rate of speech, which causes higher perceptual similarity in spite of the gender difference.
- The MOS score given to the Bengali speech transformed to my voice is around 3/5. The basic Bengali speech scores around 3.5-4/5 since it has a higher clarity and richness.
- Comparatively, the Bengali TTS audio, although language-matching, likely exhibits synthesized prosody that is less naturally expressive and possesses a very different pacing and rhythm. In addition, the TTS voice features (e.g., pitch contours, articulation) are more dissimilar from the transformed voice, which lowers the PESQ alignment. The PESQ is therefore lower in this situation.
- The word error rate between the clean base transcript (Google STT) and the transcript by Whisper is 0.6751. This is relatively high, suggesting that Whisper struggled to match the Google STT output at the word level. This could be due to accents and audio quality or background noise.
- The character error rate between the clean base transcript (Google STT) and the transcript by Whisper is 0.4673.

## 6. Final Algorithm Overview

The system proposed performs end-to-end voice conversion and language translation with a pipeline that is organized into four stages: transcription, preprocessing, translation, and audio generation.

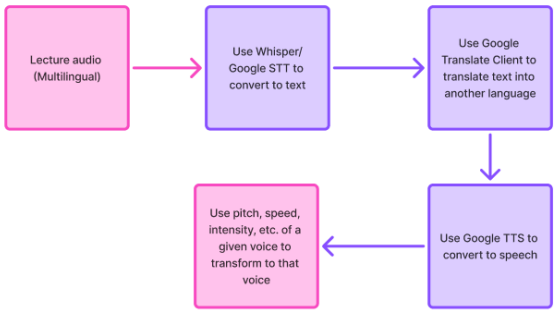


Figure 3. Algorithm Overview (made using figjam)

### 1. Transcription

Lecture audio is transcribed twice, once with Whisper and once with Google STT. After transcription, raw text is sanitized by eliminating filler words (e.g., "um", "uh", "like") and excess pauses utilizing regular expressions. I use the **Google Cloud Speech-to-Text API** to translate bilingual audio (English + Hindi). It is passed to the API with configuration arguments that include auto-punctuation and alternative language support. The resulting transcription is stored to text files. This phase is optimized for code-switching because Google and Whisper provide multilingual input support through `alternative_language_codes` and `language=None` respectively.

### 2. Translation

The filtered transcription is translated into Bengali via the **Google Translate API**, which has broad support for low-resource languages. Sentence delimiters are not removed in order to ensure semantic coherence in the output. Quality of translation in low-resource languages can be inconsistent, but this method has high utility and speed.

### 3. Audio Generation

For producing Bengali speech, I employ **Google Cloud TTS** to convert the translated text into a base Bengali audio waveform. To make the voice personalized, I use a custom post-processing module that scales the pitch and speed of the base audio to the user-specific vocal characteristics. This is performed with the help of `librosa` and `soundfile` libraries:

- **Pitch adjustment**: Executed through `librosa.effects.pitch_shift` with a semitone shift based on the pitch factor.
- **Speed modification**: Executed through `librosa.effects.time_stretch`, limited to a safe ratio (0.8x–1.2x).

### Dealing with Code-Switching and Clearness Assurance

Code-switching is managed at the transcription level by providing support for various languages (e.g., English and Hindi). Whisper can understand different languages with the paramter `language` set to None and `alternative_language_codes` for Google STT. This means that the model can accurately transcribe fragments even when the speakers code-switch in the middle of a sentence. At translation and synthesis, sentence splitting and punctuation serve to maintain intelligibility, while customized pitch and rate changes improve naturalness and speaker identity in the final audio.

## 7. Question 2

## 8. Introduction

This question focused on enhancing speech clarity in audio recordings from different places by analyzing, denoising, transcribing, and evaluating audio samples with a focus on the moderator's voice. A detailed noise analysis was conducted across recordings, where Signal-to-Noise Ratio (SNR) and noise spectral characteristics were used to quantify different disturbances.

Spectral Subtraction-based [1] denoising algorithm and Wiener filter-based denoising algorithms were used to suppress background noise while maintaining the natural quality of the moderator's speech. The algorithm was applied to all noisy recordings, and the enhanced audios were saved. This was followed by evaluation using objective metrics like SNR improvement and Perceptual Evaluation of Speech Quality (PESQ) to assess audio clarity post-denoising.

For speech transcription, Google's Speech-to-Text API [2] was used on both clean and denoised audio. Word Error Rate (WER) and Character Error Rate (CER) were computed to measure transcription accuracy against the clean references. Subjective evaluations, including Mean Opinion Score (MOS) was also checked to assess the clarity and naturalness of the speech.

The analysis revealed that the denoising algorithm significantly improved SNR and intelligibility (PESQ) without introducing noticeable artifacts. Overall, the proposed workflow enhanced moderator speech clarity, improved transcription performance, and preserved the natural tone of the original recordings. The links to the enhanced audio files are here - enhanced audio files, and the transcriptions can be found at clean_transcriptions.txt, ss_transcriptions.txt and w_transcriptions.txt. In speech processing, the presence of background noise very much deteriorates the quality and intelligibility of audio signals. To try to solve this, de-

noising algorithms are applied to suppress noise while preserving the original speech characteristics. Two common techniques for denoising are **Spectral Subtraction** and the **Wiener Filter**. The improvement in signal quality is evaluated using metrics like the **Signal-to-Noise Ratio (SNR)** and **Perceptual Evaluation of Speech Quality (PESQ)**.

### 8.1. Signal-to-Noise Ratio (SNR)

The SNR is a measure of signal strength relative to background noise. It is defined as the ratio of the power of the clean signal to the power of the noise and is usually expressed in decibels (dB):

$$\text{SNR} = 10 \log_{10} \left( \frac{\sum_{n=0}^{N-1} s(n)^2}{\sum_{n=0}^{N-1} (s(n) - \hat{s}(n))^2} \right) \tag{9}$$

where $s(n)$ is the clean speech signal and $\hat{s}(n)$ is the noisy or estimated signal.

### 8.2. Spectral Subtraction

Spectral subtraction is a classical noise reduction technique that assumes the noise is additive and can be estimated during non-speech segments. The magnitude spectrum of the noise is subtracted from the noisy speech spectrum:

$$|\hat{S}(\omega)| = |Y(\omega)| - |D(\omega)| \tag{10}$$

where $|Y(\omega)|$ is the magnitude spectrum of the noisy signal and $|D(\omega)|$ is the estimated magnitude spectrum of the noise. The enhanced speech spectrum is then obtained and the time-domain signal is reconstructed using the inverse Fourier transform.

### 8.3. Wiener Filter

The Wiener filter provides an optimal estimate of the clean speech signal in the minimum mean square error (MMSE) sense. It is based on the power spectral densities (PSD) of the clean signal and noise:

$$H(\omega) = \frac{P_s(\omega)}{P_s(\omega) + P_n(\omega)} \tag{11}$$

where $P_s(\omega)$ and $P_n(\omega)$ are the PSDs of the speech and noise signals respectively. The estimated clean signal is:

$$\hat{S}(\omega) = H(\omega) \cdot Y(\omega) \tag{12}$$

This method is particularly effective when the statistical properties of noise and speech are known or can be estimated.

### 8.4. Perceptual Evaluation of Speech Quality (PESQ)

PESQ is an objective method for assessing the quality of speech as perceived by human listeners. It compares the original clean reference signal with the degraded (noisy or processed) signal using a perceptual model of human hearing. The PESQ score ranges from $-0.5$ to $4.5$, with higher values indicating better perceptual quality.

PESQ is defined by a complex algorithm standardized by the ITU-T Recommendation P.862, which includes time alignment, auditory transform, disturbance computation, and cognitive modeling. The final output is:

$$\text{PESQ}_{\text{score}} = f\left(s_{\text{ref}}, s_{\text{deg}}\right) \tag{13}$$

where $s_{\text{ref}}$ is the clean reference signal, and $s_{\text{deg}}$ is the degraded signal.

## 9. Noise Level Analysis

A database consisting of clean-noisy pairs of audio, as well as only noisy audios was utilized for this task. A function called `calculate_snr()` was created, which would calculate the snr between the clean and noisy audio samples. `analyse_noise_spectrum()` does a short term fourier transform of the noise and the noise spectrum is visualized using `librosa`. The sampling rate for all audio samples is kept at 16000 for reproducibility and fairness in results. The clean files and their matching noisy files are loaded to return snr, noise spectrum, the frequency bin where the noise has the highest energy (peak) in the spectrum, and average noise level. The plots for noise level analysis are all given below.

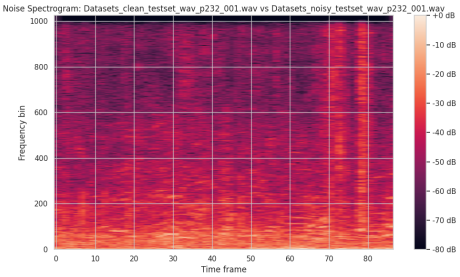


Figure 4. Noise Spectrogram for the first noisy sample

## 10. Denoising Algorithm Design

In this work, we utilize two classical audio denoising techniques: `spectral_subtraction` and `wiener_filter`, both of which operate in the frequency domain using the Short-Time Fourier Transform (STFT).

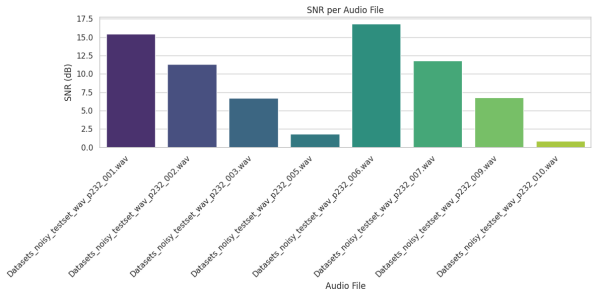


Figure 5. SNR per audio pair

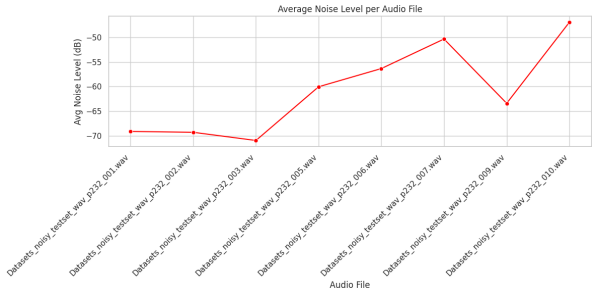


Figure 6. Average noise level per audio pair

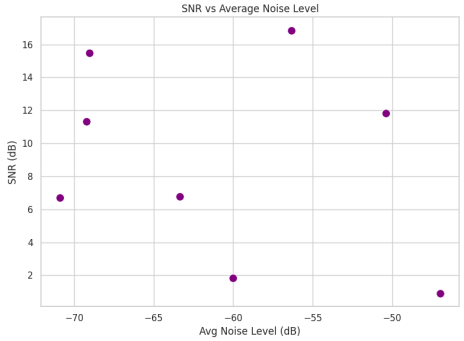


Figure 7. SNR vs Average noise level

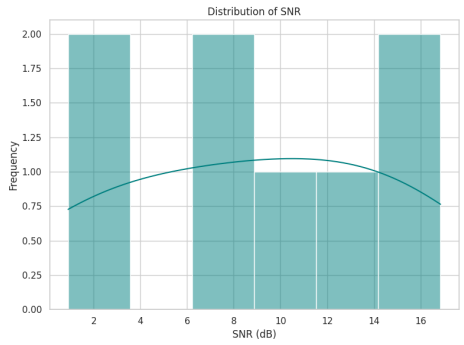


Figure 8. SNR distribution

### 10.1. Spectral Subtraction

The `spectral_subtraction` function reduces noise by estimating the average spectral profile of noise from the first 0.5 seconds of the input audio signal (this was an assumption that its noise only). This noise is then subtracted from the magnitude spectrum of the noisy signal. To avoid artifacts from over-subtraction, a spectral floor is applied by keeping the modified magnitude at a minimum of 1% of the original magnitude. The enhanced spectrogram is reconstructed using the original phase, and the inverse STFT (ISTFT) is used to get the time-domain enhanced signal.

### 10.2. Wiener Filter

The `wiener_filter` function implements a spectral Wiener filter for noise reduction. Similar to spectral subtraction, it estimates the noise power spectrum from the first 0.5 seconds of the audio. It then computes a Wiener gain function based on the ratio of noise power to total signal power in each frequency bin. The gain is constrained to a minimum value ( 0.1) to preserve signal components. The modified magnitude spectrum is combined with the original phase, and the enhanced time-domain audio is reconstructed using ISTFT.

Both techniques assume that the initial segment of the audio signal contains noise only and rely on this assumption to estimate the noise characteristics for suppression. The audio files are all resampled to 16kHz.

### 10.3. Enhancement of Noisy Audio

Following the above functions, a method called `denoise_audio` was created which takes the noisy audio and returns the enhanced version according to the denoising method applied. All noisy samples are enhanced using both spectral subtraction as well as wiener filtering. They were saved in `/content/drive/MyDrive/enhanced_audio`.

## Transcription

The `transcribe_wav_file` function performs automatic speech recognition (ASR) on a given WAV file using Google Cloud's Speech-to-Text API. The steps involved are:

- The input audio file is first loaded using `librosa.load`, keeping its original sampling rate. If the sample rate is not 16 kHz, the audio is resampled to match the required rate for the API.
- The audio data is written to a temporary WAV file using `soundfile.write`, which is later read as bi-

nary data. This temporary file is then removed to avoid clutter.

- The binary audio content is passed to the Google Cloud Speech-to-Text API using the `RecognitionAudio` class. Configuration is specified with `RecognitionConfig`, including audio encoding (`LINEAR16`), sample rate (16 kHz), and language code (`en-US`). Automatic punctuation is also enabled.
- The API returns a response containing transcription results. The function iterates over each result and extracts the most likely one, putting them into a single string, which is then returned.

The clean audio samples, the enhanced audios using spectral subtraction and the enhanced audios using wiener filtering are all transcribed and saved to `clean_transcriptions.txt`, `ss_transcriptions.txt` and `w_transcriptions.txt` respectively. They can be found at clean_transcriptions.txt, ss_transcriptions.txt and w_transcriptions.txt.

## Performance Evaluation

Here, I evaluate speech enhancement quality using objective metrics such as Signal-to-Noise Ratio (SNR) and Perceptual Evaluation of Speech Quality (PESQ), and transcription accuracy using WER and CER.

### 10.4. Denoising Evaluation

#### 10.4.1 Metrics and Setup

- **SNR (Signal-to-Noise Ratio)** quantifies the strength of the clean signal relative to the background noise. Higher SNR indicates better enhancement.
- **PESQ (Perceptual Evaluation of Speech Quality)** assesses the perceptual audio quality by comparing enhanced audio with its clean counterpart.

#### 10.4.2 Workflow

- Clean audio filenames are indexed using unique suffixes (e.g., `p232_001.wav`) to have mapping with corresponding enhanced audio files.
- Both clean and enhanced audio files are read using `scipy.io.wavfile.read`. If their sample rates differ from 16 kHz, they are resampled using `librosa.resample` to meet the requirement of the PESQ metric.
- – `calculate_snr`: Computes the SNR between clean and enhanced signals.
  – `calculate_pesq`: Uses the `pesq` library to compute PESQ scores at 16 kHz.
- Each file's SNR and PESQ score are printed, and both metrics are stored for later visualization. A combined horizontal bar plot and line plot is created:
  – The bar plot visualizes the SNR of each file.
  – A twinned x-axis line plot shows corresponding PESQ scores.

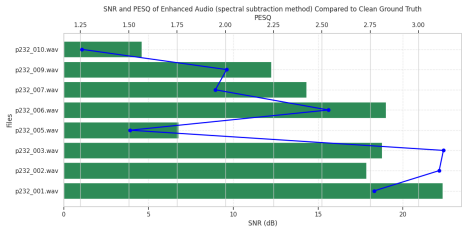


Figure 9. SNR and PESQ values of enhanced audio from spectral subtraction

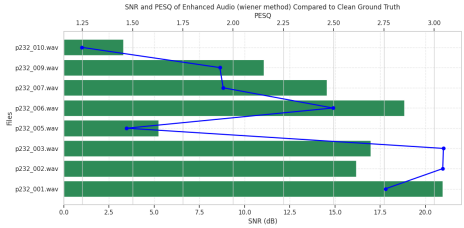


Figure 10. SNR and PESQ values of enhanced audio from wiener filtering

## Transcription Evaluation

Here, I evaluate the accuracy of the transcriptions generated by the speech enhancement pipeline using two metrics: **Word Error Rate (WER)** and **Character Error Rate (CER)**.

#### 10.4.3 Metric Definitions

- **Word Error Rate (WER):** Measures the percentage of words incorrectly predicted, calculated as the edit distance between the reference and hypothesis transcriptions, divided by the number of words in the reference.
- **Character Error Rate (CER):** Similar to WER, but operates at the character level instead of the word level.

#### 10.4.4 Implementation

- **Input:**
  - `clean_transcriptions.txt`: Ground truth reference transcriptions.
  - `ss_transcriptions.txt`: Transcriptions generated from the spectral subtraction method.
  - `w_transcriptions.txt`: Transcriptions generated from the wiener filtering method.
- Both files are read line-by-line, converted to lowercase, and stripped of leading/trailing spaces to normalize the text for proper comparison.
- **Metric Computation:** For each pair of reference and hypothesis sentences:
  - `jiwer.wer`: Calculates the WER.
  - `jiwer.cer`: Calculates the CER.

  These values are stored in lists for further analysis and visualization.
- **Averaging:** The mean WER and CER across all sentences are computed and printed to provide an overall assessment of the transcription quality.

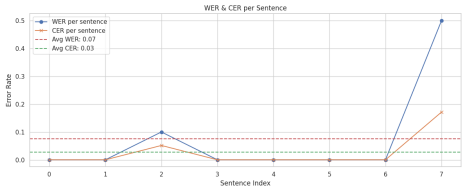


Figure 11. WER and CER of enhanced audio from spectral subtraction

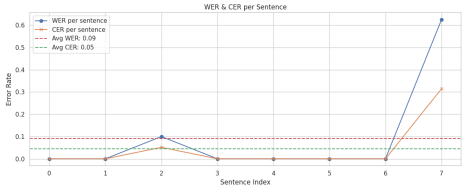


Figure 12. WER and CER of enhanced audio from wiener filtering

## 11. Result Analysis

- The SNR scores of the noisy audio files increased after enhancement using both spectral subtraction and wiener filtering.
- The average word error rate of enhanced audio using wiener filtering is slightly higher than spectral filtering's.
- The Mean Opinion Score (MOS) for all the enhanced audios is given a score of 3.5-4/5. The quality is not excellent, but is still decipherable to a good extent. The content is not altered.
- The moderator's voice remains natural and undistorted.
- For example - 'People look but no 1 ever finds it,' is the clean audio's transcription. 'Keep a look, but no 1 ever claims it?' is the wiener filter's transcription and 'Keep a look, but no 1 ever finds it?' is the spectral subtraction's transcription. It is visible that the latter performed better.
- Average improvement in SNR of spectral subtraction method is 5.52 dB, for wiener filtering is 4.42 dB.
- Overall, in terms of both denoising as well as transcription accuracy, spectral filtering seems to have performed better.

Table 1. SNR Comparison Between Noisy and Enhanced Audio Samples (spectral subtraction)

| File Name | Noisy SNR (dB) | Enhanced SNR (dB) | Gain (dB) |
|---|---|---|---|
| p232_001 | 15.47 | 22.34 | 6.87 |
| p232_002 | 11.31 | 17.86 | 6.55 |
| p232_003 | 6.71 | 18.77 | 12.06 |
| p232_005 | 1.85 | 6.77 | 4.92 |
| p232_006 | 16.86 | 19.01 | 2.15 |
| p232_007 | 11.81 | 14.31 | 2.50 |
| p232_009 | 6.78 | 12.24 | 5.46 |
| p232_010 | 0.91 | 4.60 | 3.69 |

Table 2. SNR Comparison Between Noisy and Enhanced Audio Samples (wiener filtering)

| File Name | Noisy SNR (dB) | Enhanced SNR (dB) | Gain (dB) |
|---|---|---|---|
| p232_001 | 15.47 | 20.94 | 5.47 |
| p232_002 | 11.31 | 16.16 | 4.85 |
| p232_003 | 6.71 | 16.97 | 10.26 |
| p232_005 | 1.85 | 5.23 | 3.38 |
| p232_006 | 16.86 | 18.84 | 1.98 |
| p232_007 | 11.81 | 14.55 | 2.74 |
| p232_009 | 6.78 | 11.07 | 4.29 |
| p232_010 | 0.91 | 3.30 | 2.39 |

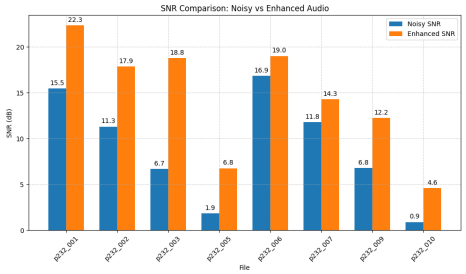


Figure 13. SNR values of noisy audio and enhanced audio from spectral subtraction

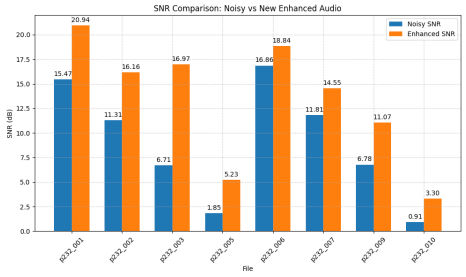


Figure 14. SNR values of noisy audio and enhanced audio from wiener filtering

# 12. Question 3

## 12.1. Multimodal Deception Detection in Conversational Speech

### 12.1.1 Problem Statement & Significance

**The Challenge:** I believe that the the next significant research challenge in Speech Understanding could be about developing robust, reliable systems for Multimodal Deception Detection in Conversational Speech (MDDCS). Whereas today's speech understanding systems can transcribe and extract semantic content from speech, they are not able to reliably detect deception—a key component of human communication that encompasses subtle linguistic, paralinguistic, and physiological cues. The state-of-the-art models generally use either only handcrafted features or CNNs and MLPs with the input as raw spectrograms. For our project, we utilized a Graph Attention Network (AASIST) and finetuned it for this specific task and showed that it achieved state-of-the-art performance on the RODeCAR dataset [3] with 81.81% accuracy.
This challenge involves creating speech understanding systems that can:

- Detect linguistic and acoustic cues of deception in real-life interactions.
- Combine multimodal signals (speech, facial, body language).
- Explain cultural, contextual, and personality differences in deception styles.
- Ensure ethical standards and confidentiality in processing sensitive information.
- Function reliably enough for practical use.

**Critical Gap in Current Research:** Current deception detection approaches face several limitations:

- Traditional polygraph methods show modest accuracy (65-75%) and require controlled settings. They are pretty invasive and easy to fool too (people use medications).
- Existing speech-based deception detection systems have high false positive/negative rates.
- Most methods focus on single-modality features rather than integrating multiple signals.
- Cultural variations in deception markers are poorly understood and rarely incorporated.
- Few systems operate effectively in real-time conversational contexts.

The gap between human intuition (which itself is often unreliable) and computational deception detection represents a significant barrier to advanced speech understanding.

**Potential Impact:** Successfully addressing this challenge would have profound impacts:

**Scientific Impact:**

- Improve understanding of the cognitive and linguistic mechanisms of deception.
- Provide insights into cross-cultural differences in truthfulness markers.
- Utilize information from several fields: linguistics, psychology, computer science, and ethics.

**Commercial Impact:**

- Reorganize security and screening processes at borders and secure facilities.
- Improve fraud detection in insurance claim and financial services.
- Provide information protection measures for social media and journalism integrity.

**Societal Impact:**

- Support criminal investigations while minimizing coercive interrogation methods
- Assist in identifying possible misinformation in public discourse
- Establish more open channels of communication in key areas such as healthcare.

### 12.1.2 Proposed Algorithm & Methodology

I propose a novel architecture called Contextual Fusion Network for Deception Detection (CFN-DD) that combines multiple innovative components:

**Model Architecture:** The proposed system architecture combines many data streams and processing modules:

$$\begin{gathered}\text{Speech Input} \rightarrow \text{Feature Extraction} \rightarrow \{\text{Linguistic Analysis, Acoustic Analysis, Contextual Analysis}\} \\ \text{Video Input} \rightarrow \{\text{Facial Expression Analysis, Micro-expression Detection, Body Language Analysis}\} \\ \text{Contextual Data} \rightarrow \{\text{Baseline Behavior Modelling, Cultural Context Modeling}\} \\ \text{All Features} \rightarrow \text{Multimodal Fusion Module} \rightarrow \\ \text{Confidence-Weighted Decision} \rightarrow \text{Deception Probability}\end{gathered}$$

**Key Components:** These are the key components:

**Speech Feature Extraction Module:**

- Acoustic features: pitch variations, speech rate, hesitations, voice quality
- Linguistic features: lexical diversity, pronoun usage, sentiment analysis
- Temporal patterns: response latency, speech rhythm disturbances

**Visual Analysis Module:**

- Facial micro-expression detection (surprise, fear, anger)
- Eye movement tracking (pupil dilation, blinking rate, gaze aversion)
- Body movement analysis (hand gestures, posture shifts)

**Context Integration Module:**

- Cultural context modeling (Culture-specific deception markers. For different cultures, responses can be different)
- Baseline behavior establishment (individual's normal communication patterns)
- Situational context analysis (stakes, relationship between speakers)

**Multimodal Fusion Network:**

- Early fusion: feature-level integration
- Late fusion: decision-level integration
- Attention mechanisms to weight most relevant features

**Confidence Estimation:**

- Uncertainty/certainty quantification for each prediction
- Calibration of confidence scores across different contexts
- Detection of out-of-distribution samples where model should abstain

**Training Methodology:** The training process involves several phases:

**Pre-training Phase:**

- Train unimodal components on large single-modality datasets
- Develop self-supervised representation learning on unlabeled conversational data
- Pre-train cultural context models on region-specific datasets

**Multimodal Integration Phase:**

- Train fusion mechanisms on multimodal deception datasets
- Use cross-modal attention to identify correlations between modalities
- Develop modality dropout techniques to ensure robustness when modalities are missing

**Calibration Phase:**

- Fine-tune on diverse scenarios with a lot of known ground truth
- Calibrate confidence scores using temperature scaling
- Adapt to different domains (legal, security, commercial)

**Algorithm 1** Multimodal Deception Detection: Initialization

Initialize SpeechEncoder $\theta_S$
Initialize VisualEncoder $\theta_V$
Initialize ContextEncoder $\theta_C$
Initialize MultimodalFusion $\theta_F$
Initialize DeceptionClassifier $\theta_D$

**Algorithm 2** Multimodal Deception Detection: Feature Extraction

**function** EXTRACT_FEATURES(speech_input, visual_input, context_data)
    linguistic_features ← extract_linguistic_features(speech_input)
    acoustic_features ← extract_acoustic_features(speech_input)
    facial_features ← extract_facial_features(visual_input)
    body_features ← extract_body_language(visual_input)
    cultural_context ← encode_cultural_context(context_data)
    baseline_behavior ← retrieve_baseline_behavior(speaker_id)
    situation_context ← encode_situation(context_data)
    **return** {"linguistic": linguistic_features, "acoustic": acoustic_features,
        "facial": facial_features, "body": body_features,
        "cultural": cultural_context, "baseline": baseline_behavior,
        "situation": situation_context}
**end function**

**Algorithm Pseudocode** This discusses the pseudocode in brief.

**Design Justifications:**

- **Multimodal approach:** Humans detect deception using multiple cues; a robust system should similarly integrate linguistic, acoustic, and visual signals. We notice the person's eyes, the way he talks and his gestures too, along with what he says.
- **Cultural calibration:** Deception markers vary significantly across cultures; the system must adapt to different cultural norms.
- **Baseline behavior modeling:** Individual differences in communication style need personalized baselines rather than universal thresholds.
- **Confidence estimation:** Given the high stakes of deception detection, the system must know when it doesn't know.
- **Explainable decisions:** In sensitive applications, the system must indicate which features contributed to its determination.

**Algorithm 3** Multimodal Deception Detection: Main Algorithm

**function** DETECT_DECEPTION(speech_input, visual_input, context_data)
    features ← EXTRACT_FEATURES(speech_input, visual_input, context_data)
    speech_encoding ← SpeechEncoder(features["linguistic"], features["acoustic"], $\theta_S$)
    visual_encoding ← VisualEncoder(features["facial"], features["body"], $\theta_V$)
    context_encoding ← ContextEncoder(features["cultural"], features["baseline"], features["situation"], $\theta_C$)
    speech_reliability ← estimate_reliability(speech_encoding)
    visual_reliability ← estimate_reliability(visual_encoding)
    attended_features ← cross_modal_attention(speech_encoding, visual_encoding, context_encoding)
    fused_representation ← MultimodalFusion(attended_features, [speech_reliability, visual_reliability], $\theta_F$)
    deception_score, confidence ← DeceptionClassifier(fused_representation, $\theta_D$)
    calibrated_score ← calibrate_by_culture(deception_score, features["cultural"])
    **return** {"deception_probability": calibrated_score,
        "confidence": confidence,
        "contributing_factors": identify_contributing_factors(fused_representation, $\theta_D$)}
**end function**

**Algorithm 4** Multimodal Deception Detection: Model Weight Updates

**function** UPDATE_MODELS(feedback_data)
    **if** feedback_data.has_ground_truth() **then**
        update_parameters($\theta_S$, $\theta_V$, $\theta_C$, $\theta_F$, $\theta_D$, feedback_data)
        update_baseline_behavior(feedback_data.speaker_id, feedback_data)
    **end if**
**end function**

### 12.1.3 Evaluation Strategy

**Datasets** There are very few datasets available for deception detection. RLDD (Real-Life Deception Detection [4]) is quite small and is only in English. There are Mandarin and Romanian datasets which do not have videos or gesture annotations. So, to properly evaluate MDDCS, I would create or adapt the following datasets:

**MultiCultural Deception Corpus:** A new dataset containing:

- Recorded deceptive and truthful statements from 1000+ participants.
- Representation from at least 10 different cultural backgrounds.
- Multiple deception scenarios (high-stakes vs. low-stakes).
- Multimodal recordings (audio, video, physiological signals).

**Real-world Deception Evaluation Set:** A dataset of real-world scenarios with ground truth:

- Court testimonies later proven true or false.
- Public statements later verified or debunked.

**Cross-Contextual Lies Database:** A specialized dataset focusing on how context affects deception markers:

- Same individuals lying in different contexts. This would help not overfitting.
- Different types of lies (self-serving, altruistic, malicious).
- Varying relationship dynamics between conversational participants.

**Metrics:** Evaluation would use specialized metrics beyond simple accuracy:

- **Area Under Precision-Recall Curve (AUPRC):** More informative than ROC curves for imbalanced classes like deception detection.
- **Cultural Fairness Gap:** Could measure performance differences across cultural groups to identify bias.
- **Contextual Stability Score:** Could quantify consistency of performance across different contexts.
- **Explainability Concordance:** Measures alignment between model's feature importance and human annotators' judgments.
- **Confidence-Accuracy Calibration:** Evaluates how well confidence scores predict actual accuracy.
- AUROC, Precision, Recall, EER would also be reported.

**Experimental Design:** The evaluation would follow a multi-phase approach:

- **Baseline Comparison:** Compare against current state-of-the-art methods including:
  - Commercial polygraph systems.
  - Linguistic-only deception detection.
  - Human deception detection performance (professional interrogators).
  - Current state-of-the-art deep learning models.
- **Cross-Cultural Evaluation:** Test performance across different cultural groups to ensure fairness.
- **Ablation Studies:** Systematically remove components to quantify the how much a modality contributed.
- **Adversarial Testing:** Evaluate against subjects trained to defeat deception detection.
- **Field Trials:** Deploy in real-world settings (courts, police investigations) to validate laboratory findings.

### 12.1.4 Some Broader Implications:

**Advancement of the Field:** Successfully solving the MDDCS challenge would advance speech understanding in several directions:

- We can move beyond literal meaning to implied meaning and speaker intent.
- Enhance systems' ability to detect and respond to emotional states.
- Improve understanding of cultural variations in speech patterns.
- Establish frameworks for responsible deployment of sensitive AI capabilities.

**Downstream Applications:** The technology would enable help in many applications:

- **Security Screening:** More accurate, less invasive screening at borders and secure facilities (in comparison to polygraphs and lie detection tests).
- **Legal Proceedings:** Tools to assist in evaluating testimony and evidence.

- **Healthcare:** Detection of concealed symptoms or medication non-compliance.
- **Financial Protection:** Fraud detection in insurance claims and financial transactions can be made easier.
- **Media Literacy:** To help identify potential misinformation in news and social media.

## References


[1] Cliston Cole, Marc Karam, and Heshmat Aglan. Spectral subtraction of noise in speech processing applications. In *2008 40th Southeastern Symposium on System Theory (SSST)*, pages 50–53, 2008. 5

[2] Bogdan Iancu. Evaluating google speech-to-text api's performance for romanian e-learning resources. *Informatica Economica*, 2019. 5

[3] Serban Mihalache, Gheorghe Pop, and Dragos Burileanu. Introducing the rodecar database for deceptive speech detection. *2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, pages 1–6, 2019. 10

[4] Verónica Pérez-Rosas, Mohamed Abouelenien, Rada Mihalcea, and Mihai Burzo. Deception detection using real-life trial data. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, ICMI '15, page 59–66, New York, NY, USA, 2015. Association for Computing Machinery. 13

[5] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022. 1