# Unveiling Voices: Deep Learning Based Noise Dissipation with Real-Time Multi-Speech Separation And Speaker Recognition

Chaitra V[1†], Meera Reji[2*], Meghana Aithal[3*], Varshini Gopal[4†], and Shylaja S S[5*]

[†]Department of CSE(AIML), Center for Data Sciences and Applied Machine Learning, PES University, Bangalore, India
[*]Department of CSE, Center for Data Sciences and Applied Machine Learning, PES University, Bangalore, India
[1]chaitrav2020@gmail.com,[2]meerareji48@gmail.com,[3]aithalmeghana1863@gmail.com,
[4]varshinigopal975@gmail.com,[5]shylaja.sharath@pes.edu

**Abstract.** Humans have a remarkable ability to focus on speech even in noisy environments. To replicate this capability computationally, recent research has developed advanced speech enhancement and separation algorithms. This paper presents a comprehensive pipeline integrating several top-notch models for efficient background noise removal, speaker separation, and recognition in audio recordings. Furthermore, our approach exhibits remarkable proficiency in speaker diarization and identification, facilitated by sophisticated feature extraction and classification techniques. Our results demonstrate substantial improvements in audio clarity, speaker separation, and accurate speaker identification. This enhances the capability of our approach for real-time applications in telecommunications, hearing aids, and other audio processing domains. The effectiveness of our method provides a strong proof-of-concept for its real-world applicability, showcasing its potential to revolutionize acoustic applications.

**Keywords:** Audio classification · Background noise extraction · Speech enhancement · Speaker separation · Speaker Recognition

## 1 Introduction

In the modern digital era, the proliferation of audio data across various applications has intensified the demand for advanced techniques that enhance the quality and intelligibility of speech signals. Whether in telecommunication systems, hearing aids, voice-controlled interfaces, or virtual meetings, clear and distinct audio output is essential. However, real-world audio recordings often contend with background noise and overlapping speech, presenting significant challenges to effective communication and information extraction. Addressing these challenges requires innovative solutions that can effectively isolate and enhance speech signals in complex acoustic environments.

Humans possess an extraordinary ability to focus on a particular person's voice amidst noisy environments—a phenomenon known as the "cocktail party effect." Replicating this capability technically has long been a goal in audio signal processing. Traditional methods for noise reduction and speaker separation often struggle in dynamic and complex acoustic scenarios. Recent advancements in deep learning have transformed this field, applying complex neural network algorithms to enhance and separate audio signals with acceptable accuracy. This paper explores these advancements, presenting a panoramic approach that expands the boundaries of possibilities in speech processing.

Our proposed pipeline begins with an audio classifier that identifies the type of background noise present. This step paves the way for a more targeted noise removal process. The detected noise is then amplified, making it more perceivable and thus easier to remove from the original audio. The result is a clean signal with enhanced clarity of the desired speech. The final stage of our pipeline precisely separates the cleaned audio into distinct tracks for each speaker. Importantly, during this separation process, speaker recognition is performed to accurately label each speaker's track. This multi-faceted approach ensures that the output is clear and in order for further processing or analysis.

The models used in our pipeline have been extensively trained on synthetic composite audio mixtures sourced from various datasets to generalize across a broad range of audio scenarios effectively. Our results show improvements in both audio clarity and speaker separation, with promising potential for real-time applications across various domains. This research highlights the effectiveness of deep learning in solving complex audio processing challenges and provides a structured approach for future advancements. By replicating humans' selective focus while listening, we lay the foundation for a new era of acoustic applications that can significantly improve communication in noisy environments.

## 2 Related Work

In this section, we review significant advancements in the field of sound extraction, noise removal, speech separation, and recognition. Recent studies have proposed innovative models and methodologies that enhance performance and efficiency in these areas.

**[1]** presents Waveformer, a pioneering neural network designed for real-time and streaming target sound extraction. The Waveformer architecture features an encoder-decoder setup with dilated causal convolution layers in the encoder and a transformer layer in the decoder. This structure effectively handles large receptive fields while leveraging the generalization capabilities of transformers. The model demonstrates a significant improvement in SI-SNRi by 2.2–3.3 dB over previous models, with the added advantage of being 1.2–4 times smaller and 1.5–2 times faster. The code, dataset, and audio samples can be accessed at `https://waveformer.cs.washington.edu/`.

**[2]** employs spectral subtraction to reduce noise from speech signals within the frequency domain. This technique involves computing the spectrum of noisy

speech using the Fast Fourier Transform (FFT) and then subtracting the average magnitude of the noise spectrum. The method was applied to a speech signal labeled "Real graph" with vacuum cleaner noise added. The noise reduction algorithm was executed in Matlab, with noisy speech data stored in Hanning time-windowed, half-overlapped buffers. FFT was used to compute the spectrums, the noise was subtracted, and the speech signal was reconstructed using the inverse FFT (IFFT). The effectiveness of the algorithm was assessed by calculating the Speech to Noise Ratio (SNR). Seventeen different configurations, varying in Hanning window lengths, buffer overlaps, and frame averaging, were tested. The findings indicated that using one-fourth overlapped data buffers with 128-point Hanning windows and no frame averaging achieved the optimal noise reduction performance.

Recurrent Neural Networks (RNNs) have traditionally dominated sequence-to-sequence learning tasks, but their inherent sequential processing limits the potential for parallelization. Transformers, equipped with a multi-head attention mechanism, are gaining recognition as a more effective alternative. **[3]**, introduces SepFormer, a Transformer-based neural network designed for speech separation that completely omits RNNs. SepFormer leverages a multi-scale strategy to capture both short-term and long-term dependencies, achieving cutting-edge results on the WSJ0-2/3mix datasets. Specifically, it reaches an SI-SNRi of 22.3 dB on WSJ0-2mix and 19.5 dB on WSJ0-3mix. By capitalizing on the parallelization capabilities of Transformers, SepFormer operates more efficiently and with lower memory requirements compared to similar systems.

Self-supervised learning (SSL) approaches, such as WavLM, have demonstrated promising results in speech separation (SS) tasks during smaller-scale experiments. In **[4]**, SSL-based SS is expanded by significantly increasing the scale of pre-training to over 300,000 hours and fine-tuning with 10,000 hours of data. The study also investigates strategies for integrating the pre-trained model with the SS network while adhering to strict computational limits, such as training SSL models at a lower frame rate and employing selective fine-tuning techniques. When compared to a supervised baseline and a WavLM-based SS model trained on 94,000 hours of data, the proposed model achieves relative word error rate (WER) reductions of 15.9% and 11.2% for a simulated far-field speech mixture test set. Additionally, for conversation transcription in real meeting recordings, the model achieves relative WER reductions of 6.8% on the AMI evaluation set and 10.6% on the ICSI evaluation set.correspondingly, while reducing computational cost by 38%.

In 2017, Microsoft enhanced its conversational speech recognition system by incorporating advancements in neural network-based acoustic and language modeling techniques. **[5]** presents the introduction of a CNN-BLSTM acoustic model into the existing framework and improves performance using character-based LSTM and dialog session-aware language models for rescoring. The system utilized a two-stage approach: initially combining acoustic models at the senone/frame level, followed by word-level voting through confusion networks. A further step of confusion network rescoring was applied after the combination. This updated

system reached a significant achievement, recording a 5.1% word error rate on the 2000 Switchboard evaluation set.

## 3 Dataset

Each model in our approach has been trained on exclusive datasets. The CNN classification model, which is trained on dog, keyboard, cat, and cough datasets, has made the classification of these categories easy for noise extraction. This adds to the novelty of our model. This training helps with the efficient spectral subtraction of background sounds. The Waveformer model has been trained on acoustic sounds available in the FSDKaggle2018 dataset and various target sounds obtained from the TAU Urban Acoustic Scenes 2019 Development dataset for noise extraction. The Sepformer models are trained on datasets like WSJ02mix for 2 speakers, WSJ03mix for 3 speakers, and Libri4Mix for 4 speakers separation. The recursion of models helps in the effective separation of speech signals. Speech recognition has been trained on our own dataset to identify the speaker and save the output file in their name, based on speech separation. Thus, the various datasets used to train multiple stages of our project prove beneficial in improving the efficiency of the models.

## 4 Methodology

A detailed step-by-step analysis and research of the audio files present in the above-mentioned dataset has been conducted for speech enhancement, background noise removal, speaker separation, and recognition. The models used are:

1. Convolutional Neural Network (CNN) and feature extraction from Mel-Frequency Cepstral Coefficients (MFCC)
2. Waveformer
3. Spectral Subtraction
4. Sepformer Separation
5. Speaker Recognition and Separation

## 5 Model Descriptions

### 5.1 *Convolutional Neural Network (CNN) and Feature Extraction from Mel-Frequency Cepstral Coefficients (MFCC)*

The code used is divided into two components:

1) Training a Convolutional Neural Network (CNN) Model: This part involves preparing a dataset, extracting features of noise from audio files, building and training a CNN model to classify audio samples into different categories.

2) Using the Trained Model for Prediction: This part involves loading the trained model and using it to make predictions on new mixed audio samples.

**Key Parameters in CNN:**

**[5.1.1] Feature Extraction (MFCC)**

$$\text{MFCC} = \text{DCT}(\log(\text{Mel-Scale}(\text{STFT}(x)))) \tag{1}$$

*The Short-Time Fourier Transform (STFT)* transforms a signal from the time domain to the frequency domain. *The Mel-Scale* mimics the human ear's sensitivity to various frequencies. *The Discrete Cosine Transform (DCT)* decorrelates the logarithmic Mel-spectrogram to generate MFCC features.

**[5.1.2] Building the CNN Model**

*Conv2D Layer:* Performs a 2D convolution operation on the input data.

$$Conv2D(x) = \sigma\left(\sum_{i,j} K_{i,j} \cdot x_{i,j}\right) \tag{2}$$

where: $\sigma$ represents the ReLU (Rectified Linear Unit) activation function, applied to each element to introduce non-linearity, thereby enabling the model to capture intricate patterns. $K_{i,j}$ represents the convolutional kernel or filter weights that the network learns during training. $x_{i,j}$ represents the input data (in this context, it could be a 2D matrix representing the MFCC features).

*MaxPooling2D Layer:* Applies a 2D max pooling operation.

$$MaxPooling2D(x) = \max(x_{i,j}) \tag{3}$$

where $x_{i,j}$ represents the values within the pooling window, and the maximum value is taken as output.

*Flatten Layer*: Flattens the input. It converts the multi-dimensional output of previous layers into a single vector. This vector can then be fed into fully connected layers for classification tasks.

*Dense Layer*: This layer is fully connected, with each neuron establishing a connection to every neuron in the preceding layer.

$$Dense(x) = \sigma(Wx + b) \tag{4}$$

where $W$ signifies the weight matrix and $b$ stands for the bias.

*Dropout Layer*: Used during training to prevent overfitting. It selectively disables a fraction of input units during each training iteration, enhancing the model's generalization and reducing reliance on any single neuron.

**[5.1.3] Model Compilation and Training**

*Sparse Categorical Crossentropy Loss*: Loss function for multiclass classification where the target is integer-encoded.

*Adam Optimizer*: Adaptive moment estimation optimizer.

*Callbacks - Reduce Learning Rate on Plateau*: Reduces learning rate when a metric has ceased improving.

*Loss Function*: Sparse multi-class entropy loss.

$$\text{Loss} = -\sum_{i} y_i \log(p_i) \tag{5}$$

where $y_i$ denotes the actual label and $p_i$ represents the predicted likelihood.

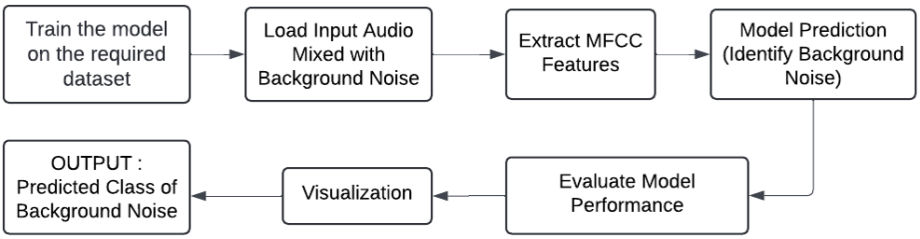


**Fig. 1.** Flowchart of CNN-Based Background Noise Prediction

Fig. 1 outlines the flow of the Convolutional Neural Network (CNN) model used for background noise prediction. The process starts with training the model on a dataset, followed by loading the input audio mixed with background noise. MFCC (Mel-frequency Cepstral Coefficients) features are then extracted, and the model identifies the background noise. The output is visualized, and the model performance is evaluated based on prediction accuracy.

### 5.2 *Waveformer*

Waveformer is a sophisticated audio classification model that employs complex neural network architectures designed for audio processing, often involving convolutional, recurrent, or transformer-based layers tailored to handle spectrogram or waveform inputs.

**Overview of the Waveformer Code:**

It handles downloading model configurations and checkpoints, loading and resampling input audio, constructing query vectors for target classes, performing inference based on specified target classes, and saving the resulting classified output audio. This approach provides a versatile tool for extracting sound from audio files into predefined categories efficiently.

**Resampling**: The code resamples the input audio if its sample rate ($fs$) does not match 44100 Hz using `torchaudio`. This ensures consistency in audio data format before feeding it into the model.

**Model Inference**:

- `'mixture'`: Input audio data after resampling, loaded into a tensor and shifted to the inferred device (CPU or GPU).
- `'query'`: A binary vector indicating which sound classes from `TARGETS` are present in the input audio. If no targets are specified, initializes `'query'` to ones; otherwise, sets indices corresponding to specified targets to ones.

– `'model'`: The Waveformer model instance loaded with pre-trained weights.
– `'.squeeze(0)'`: Removes the batch dimension after model inference.
– `'.cpu()'`: Moves the tensor back to CPU for further processing and saving.

**Argument Parsing**: Handles input arguments using `argparse`, allowing users to specify the input audio file, output path, and optionally, target sound classes (`-targets`).

Here are the typical components and equations involved in this model:

**[5.2.1] Convolutional Layers**: They are fundamental to the feature extraction process in neural networks. In the context of Waveformer, these layers are used to extract relevant features from audio spectrograms or waveforms by applying convolutional filters.

$$output[i, j] = \sum_{m,n} input[i + m, j + n] \cdot kernel[m, n] \tag{6}$$

Where: *input[i + m, j + n]* represents a small region of the input audio data, typically a patch of the spectrogram or a segment of the waveform.
*kernel[m, n]* refers to the convolution filter, which is a matrix of learned parameters that moves across the input to identify particular features like edges or textures in images, or frequency patterns in audio. *output[i, j]* is the result of the convolution operation, which captures the presence of these features in different regions of the input.

**[5.2.2] Recurrent or Transformer Layers**: To capture the temporal dependencies and context in audio sequences, Waveformer may utilize recurrent layers, such as Long Short-Term Memory (LSTM) units, or layers based on transformers. These layers are designed to handle sequential data, making them ideal for audio, where the order of sounds is important.

*Recurrent equation:*

$$h_t = \sigma(W_h \cdot h_{t-1} + W_x \cdot x_t) \tag{7}$$

Where: $h_t$ is the hidden state at time $t$, which captures the current context of the sequence. $h_{t-1}$ represents the hidden state from the preceding time step, $t - 1$, which holds the information from prior steps. $W_h$ and $W_x$ are weight matrices that are learned during training. $x_t$ denotes the input at the present time step $t$. $\sigma$ is an activation function, typically a sigmoid or tanh, that introduces non-linearity.

*Transformer equation:*

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{8}$$

Where: $Q$ (Query), $K$ (Key), and $V$ (Value) are derived vectors from the input data. The term $QK^\top$ is the dot product of the query and key matrices, which determines the relevance of the input sequences to each other. $\sqrt{d_k}$ is a scaling

factor based on the dimensionality of the key vectors, ensuring numerical stability. The *softmax* function normalizes the relevance scores, turning them into probabilities. The output is a weighted sum of the value vectors, enabling the model to concentrate on the key elements of the input sequence.

**[5.2.3] Fully Connected Layers**: Often termed dense layers, these components are responsible for translating the advanced features obtained from convolutional and recurrent layers into the final classification outputs.

$$output = \sigma(W \cdot input + b) \tag{9}$$

Where: $W$ represents the weight matrix that links the input features to the output neurons. *input* denotes the feature vector received from the preceding layer. $b$ is the bias term added to the weighted sum. $\sigma$ is the activation function, commonly a ReLU for hidden layers or a softmax for the output layer.

**[5.2.4] Loss Function (e.g., Cross-Entropy)**: Assesses the disparity between the model's predictions and the actual labels. This metric is essential in training, as it directs the optimization process to modify the model's weights in order to reduce the error between predicted and true values.
Refer to Equation (5)

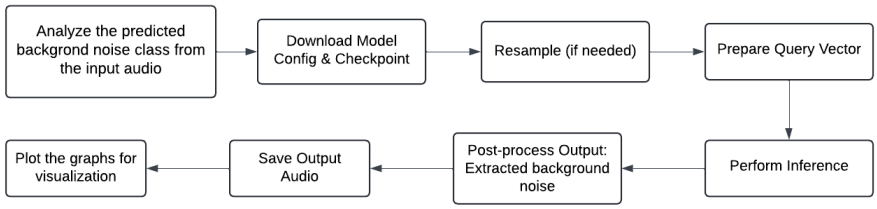


**Fig. 2.** Flowchart for Waveformer Process

Fig. 2 outlines the steps involved in the Waveformer system, which processes input audio to extract background noise. It includes key stages such as analyzing the input audio, downloading the model configuration, preparing a query vector, performing inference, and saving the output audio with optional resampling and post-processing steps.

### 5.3 *Spectral Subtraction*

Spectral subtraction technique mitigates noise by removing an approximation of the noise spectrum from the spectrum of the noisy signal. The process involves STFT, magnitude and phase extraction, noise estimation, spectral subtraction, and ISTFT for signal reconstruction. It performs noise reduction on a given noisy audio file using a noise reference and visualizes the results.

**Key Parameters in Spectral Subtraction:**

**[5.3.1] Padding or Trimming the Signal:**

*Padding*: If the noise signal is shorter than the noisy signal, zeros are appended to the end of the noise signal to match the length. This is crucial because, during the spectral subtraction process, both signals need to have the same number of samples for accurate noise reduction.

$$signal_{padded} = [signal, \underbrace{0, 0, \ldots, 0}_{pad\ length}] \tag{10}$$

*Trimming*: Conversely, if the noise signal is longer, it is truncated to the desired target length. This ensures that the noise estimation process is not affected by extra, irrelevant parts of the noise signal, which could lead to inaccurate noise reduction.

$$signal_{trimmed} = signal[: target\_length] \tag{11}$$

**[5.3.2] Short-Time Fourier Transform (STFT):**

$$STFT(x(t)) = X(k, n) = \sum_{m=0}^{M-1} x(m) \cdot w(n - m) \cdot e^{-j2\pi km/N} \tag{12}$$

Where: $STFT$ is employed to convert the time-domain signal into the time-frequency domain representation. Here, $x(t)$ denotes the input signal, $w(n - m)$ represents the windowing function, $N$ is the total number of FFT points, $k$ indicates the frequency bin, and $n$ denotes the time frame.

**[5.3.3] Magnitude and Phase**:
*Magnitude*: The magnitude $|X(k, n)|$ denotes the strength of each frequency component within a specific time frame. It indicates how strong or prominent a particular frequency is within that segment of the signal.
*Phase*: The phase $\angle X(k, n)$ represents the phase angle of the frequency component, indicating the position of the waveform in its cycle at that specific time frame.
In spectral subtraction, the magnitude is typically the primary focus for noise reduction, as it contains the energy information of the signal. The phase, while less affected by noise, is preserved for signal reconstruction during the inverse process.

**[5.3.4] Noise Estimate**: It is calculated by averaging the magnitude of the noise spectrum over several initial frames. This averaging process helps in obtaining a reliable estimate of the noise that can be subtracted from the noisy signal. The assumption here is that the initial frames are dominated by noise, which provides a good reference for what needs to be subtracted.

$$Noise\ Estimate(k) = \frac{1}{T} \sum_{n=0}^{T-1} |N(k, n)| \tag{13}$$

Where: $N(k, n)$ indicates the magnitude spectrum of the noise at frequency bin $k$ and time frame $n$. $T$ represents the number of time frames used for averaging the noise estimate.

**[5.3.5] Spectral Subtraction:** The core operation where the estimated noise magnitude is subtracted from the magnitude of the noisy signal.

$$|\hat{X}(k,n)| = \max(|X(k,n)| - \alpha|\hat{N}(k,n)|, \beta|\hat{N}(k,n)|) \tag{14}$$

Where: $|\hat{X}(k,n)|$ signifies the amplitude of the noisy signal at frequency bin $k$ and time frame $n$. $|\hat{N}(k,n)|$ indicates the estimated magnitude of the noise spectrum. $\alpha$ is the over-subtraction factor, controlling the amount of noise subtraction. Values of $\alpha > 1$ can help remove more noise but might risk removing parts of the desired signal. $\beta$ is a parameter that sets the minimum allowable magnitude to prevent negative values, which can occur due to over-subtraction.

**[5.3.6] Smoothing Factor:** The smoothing factor helps to stabilize the noise estimate over time, reducing the impact of sudden changes in noise magnitude. It is especially beneficial in practical situations where noise levels may vary, ensuring that the noise subtraction remains effective and consistent.

$$\hat{N}(k,n) = \gamma\hat{N}(k,n-1) + (1-\gamma)|N(k,n)| \tag{15}$$

Where: $\gamma$ is the smoothing factor, typically between 0 and 1. A higher value of $\gamma$ gives more weight to the previous estimate, leading to smoother noise estimation. $\hat{N}(k,n)$ is the updated noise estimate at the current time frame $n$.

**[5.3.7] Inverse STFT (ISTFT):** The ISTFT combines the modified magnitude with the original phase information to reconstruct the time-domain signal. This step is crucial because it produces the final output, a cleaner version of the original noisy signal with significantly reduced noise. The quality of the reconstructed signal depends on the accuracy of the noise estimate and the effectiveness of the spectral subtraction.

$$x(t) = \sum_{n=0}^{N-1} \text{ISTFT}(X(k,n)) \tag{16}$$

Where: *ISTFT* is the process that converts the modified spectral data back into the time domain. The sum is taken over all the time frames to reconstruct the entire signal.

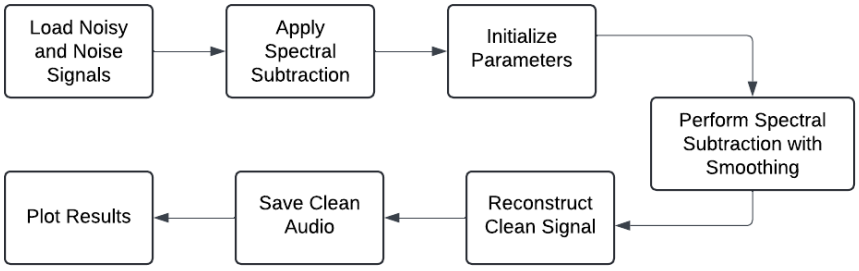


**Fig. 3.** Flowchart for Spectral Subtraction Process

Fig. 3 illustrates the steps in the Spectral Subtraction method used to process noisy audio signals. The process starts by loading noisy and reference noise signals, applying spectral subtraction, and initializing parameters. It then proceeds to perform spectral subtraction with smoothing, reconstruct the clean signal, and save the clean audio. Finally, results are plotted for further analysis.

### 5.4 *Sepformer Separation*

Sepformer model is used for audio source separation based on speaker count detection. The code leverages pretrained Sepformer models, specialized for 2, 3, or 4 speaker scenarios, to effectively separate audio sources. After detecting the number of speakers and selecting the appropriate model, the code applies the Sepformer's capabilities to extract individual audio sources, which are then saved as distinct '.wav' files. This approach ensures that complex audio mixtures containing multiple speakers can be separated accurately, demonstrating the utility of advanced deep learning techniques for audio processing tasks in real-world applications.

**The general principles of Sepformer include**:
**Multi-head Self-Attention**: Allows the model to concurrently focus on different regions of the input audio spectrogram, capturing complex dependencies between time-frequency bins.
**Transformer Encoder Layers**: Transform the input audio features into a latent space conducive to effective source separation. Each encoder layer applies self-attention and feed-forward mechanisms to iteratively refine representations.
**Permutation Invariant Training (PIT)**: Trains the model to handle permutations of separated sources. It uses loss functions that penalize discrepancies between the predicted and ground-truth sources, ensuring consistent output irrespective of input order.
**Training Objectives (Loss Functions)**: Typically include metrics like Mean Squared Error (MSE) or Signal-to-Distortion Ratio (SDR) to quantify separation quality during training. These metrics guide the model towards minimizing distortion and enhancing source separation fidelity.
**Speaker Detection Function**: This function serves as a placeholder for actual speaker detection logic. It prompts the user to input the number of speakers present in the audio file '(1-4)'. In a practical scenario, speaker detection would use pretrained models or heuristic methods for accurate speaker count determination.
**Audio Separation Function**: Depending on the number of speakers detected, this function selects the appropriate pretrained Sepformer model:

- For 2 speakers: Sepformer model trained on WSJ0-2mix dataset (speechbrain/sepformer-wsj02mix).
- For 3 speakers: Sepformer model trained on WSJ0-3mix dataset (speechbrain/sepformer-wsj03mix).
- For 4 speakers: Sepformer model trained on Libri4mix dataset (hahmadraz/sepformer-libri4mix).

**Components of Sepformer**:

**[5.4.1] Encoder Transformation**: The input audio waveform is converted into a high-dimensional feature space that is suitable for processing by the subsequent transformer network. The encoder consists of multiple convolutional layers that convert the raw audio input into a feature set, capturing crucial information about the audio content. This transformation is essential as it decreases the dimensionality of the input while preserving the key features needed for effective source separation.

$$Z = Encoder(X) \tag{17}$$

Where: $X$ represents the input audio waveform, which could be a complex mixture of multiple speakers.
$Z$ is the resulting high-dimensional feature space, a more abstract representation of the input signal.

**[5.4.2] Dual-Path Transformer**: The dual-path transformer is a sophisticated architecture within the Sepformer model that processes the audio signal using two pathways: *the intra-transformer* and *the inter-transformer*. This dual-path strategy enables the model to grasp both local and global contexts, which is essential for accurately separating sources in complex audio mixtures.

- *Intra-Transformer*: Focuses on capturing local context and dependencies within short segments of the audio signal.

$$Z_{intra} = Transformer_{intra}(Z) \tag{18}$$

Where: $Z_{\text{intra}}$ represents the transformed features after processing by the intra-transformer.

- *Inter-Transformer*: Refines '$Z_{intra}$' by incorporating global context across longer segments, crucial for understanding complex audio mixtures.

$$Z_{inter} = Transformer_{inter}(Z_{intra}) \tag{19}$$

Where: $Z_{\text{inter}}$ represents the refined features after processing by the inter-transformer.

**[5.4.3] Decoder Transformation**: It Consists of transposed convolutional layers or other upsampling techniques that convert the latent feature space back into the time-domain waveform. This step reconstructs the separated audio sources, corresponding to each speaker or sound source. The quality of the decoder transformation is critical for ensuring that the separated sources are not only distinct but also closely resemble the original sources in terms of audio quality.

$$\hat{X} = Decoder(Z_{inter}) \tag{20}$$

Where: $\hat{X}$ represents the reconstructed audio signals, corresponding to the separated sources. $Z_{\text{inter}}$ is the feature representation output by the inter-transformer.

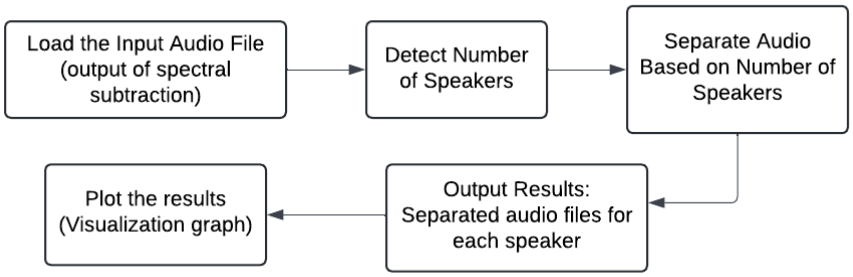


**Fig. 4.** Flowchart for Sepformer Model's Speaker Separation Process

Fig. 4 illustrates the steps in the Sepformer model for speaker separation. The process begins by loading the input audio file, which is the output of spectral subtraction. Then, the number of speakers in the audio is detected, and the audio is separated accordingly. The separated audio files for each speaker are output, and finally, the results are visualized through a graph for further analysis.

### 5.5 *Speaker Recognition and Separation*

Speaker recognition and separation uses a combination of CNN and Sepformer models. The process begins with the extraction of MFCC features from audio recordings, which are subsequently utilized to train a CNN model for speaker classification (Hemachandra and Vidhathri). The Sepformer model is used to separate mixed audio into individual sources. The separated sources are then classified using the trained CNN model, and the results are saved as individual audio files for each speaker.

**Key Parameters used**:

**[5.5.1] Feature Extraction with MFCC**

Function: Transforms an audio signal into Mel-Frequency Cepstral Coefficients (MFCCs), which serve as input features for the CNN model. These features are extensively employed in speech and audio processing due to their ability to accurately reflect the perceptual traits of human hearing, rendering them particularly suitable for tasks such as speaker identification.

***Short-Time Fourier Transform (STFT)***: Transforms the signal from the time domain to the frequency domain through the Short-Time Fourier Transform (STFT). This process divides the signal into brief, overlapping segments, allowing the analysis of its frequency content over time.

$$X(t,\omega) = \sum_{n=0}^{N-1} x[n]w[n-t]e^{-j\omega n} \tag{21}$$

Where: $x[n]$ denotes the incoming audio waveform, a sequence of sampled values over time. $w[n-t]$ is the window function, which selects a short segment of the

signal for analysis. $X(t, \omega)$ is the STFT of the signal, representing its frequency content at each time step $t$ and frequency $\omega$.

***Mel Filterbank***: Applies a series of triangular filters to the magnitude spectrum derived from the STFT. The Mel filterbank outputs a series of values that correspond to the energy in different frequency bands on the Mel scale. These values serve as the foundation for calculating the MFCCs. The Mel scale mimics the human ear's perception of sound, where lower frequencies are perceived with finer resolution than higher frequencies.

$$M_m = \sum_{k=0}^{K-1} |X_k|^2 H_m(k) \tag{22}$$

Where: $H_m(k)$ represents the $m$-th Mel filter, a triangular filter that emphasizes certain frequency bands while attenuating others. $K$ is the number of FFT (Fast Fourier Transform) points, which determines the resolution of the frequency analysis. $|X_k|^2$ indicates the power spectral density of the signal, representing the energy at each frequency.

***Logarithm of Mel Spectrum***: To simulate the human ear's response to sound intensity, the logarithm of the Mel spectrum is computed. This process reduces the dynamic range of the spectrum, enhancing its suitability for speech analysis. The logarithm function is applied to the output of the Mel filterbank, resulting in the log Mel spectrum.

$$log(M_m) \tag{23}$$

***Discrete Cosine Transform (DCT)***: Transforms the log Mel spectrum into the cepstral domain, where the coefficients represent the spectral envelope. The lower-order coefficients capture the broad spectral shape, while higher-order coefficients capture finer details, offering a condensed depiction of the signal's spectral characteristics.

$$MFCC_n = \sum_{m=0}^{M-1} \log(M_m) \cos\left[\frac{M\pi n(m+0.5)}{M}\right] \tag{24}$$

Where: $M$ is the number of Mel bands, corresponding to the number of coefficients extracted from the Mel filterbank. $n$ is the index of the MFCC coefficient, with lower indices representing the most significant features.

**[5.5.2] CNN Model for Speaker Recognition**

Function: Processes the extracted MFCC features to classify the input audio into one of several predefined speaker classes. CNNs are exceptionally suited for this role as they are capable of learning layered representations of input data, enabling them to effectively capture both spatial and temporal patterns within the MFCC features.

***Convolutional Layer***: The input features are processed through convolutional filters to generate feature maps, which identify local patterns such as edges or textures. This layer helps the CNN to build spatial hierarchies, with early

layers recognizing fundamental patterns and higher layers combining them into complex features.

$$y_{i,j,k} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x_{i+m,j+n} w_{m,n,k} + b_k \tag{25}$$

Where: $x_{i+m,j+n}$ represents a local patch of the input feature map. $w_{m,n,k}$ is the convolutional filter applied to the input. $b_k$ denotes the bias term for the $k$-th filter. $y_{i,j,k}$ represents the result of the convolution operation, a feature visualization that emphasizes particular patterns in the input data.

***Activation Function (ReLU)***: The Rectified Linear Unit (ReLU) activation function injects non-linearity into the model, allowing it to understand intricate relationships between input features and target categories. ReLU is frequently utilized in CNNs as it addresses the vanishing gradient issue, enabling the network to learn more rapidly and deliver superior performance.

$$ReLU(x) = max(0, x) \tag{26}$$

Where: $x$ is the input to the activation function, typically the result from a convolutional layer. The ReLU activation function outputs $x$ if it is positive and 0 otherwise.

***Residual Block***: It is a key component of modern deep neural networks that includes skip connections to bypass one or more layers. This design helps the model learn more efficiently by mitigating the degradation problem, where additional layers can degrade performance, and allows the network to focus on important features while avoiding unnecessary complexity.

$$y = ReLU(F(x, \{W_i\}) + x) \tag{27}$$

Where: $F(x, \{W_i\})$ represents the function learned by the residual block, where $x$ is the input and $\{W_i\}$ are the weights of the block. The input $x$ is added to the output of the function $F$, forming a shortcut connection that facilitates easier gradient flow through the network.

***Pooling Layer***: Decreases the spatial size of feature maps, enhancing computational efficiency and reducing overfitting. Max pooling preserves the most significant features while discarding less relevant information, allowing the network to focus on salient patterns and improve generalization.

$$y_{i,j} = \max_{m=0}^{M-1} \max_{n=0}^{N-1} x_{i+m,j+n} \tag{28}$$

Where: $x_{i+m,j+n}$ represents the input to the pooling layer, typically coming from the results of a convolutional layer. The pooling process identifies the maximum value within a localized window of dimensions $M \times N$, thereby decreasing the spatial dimensions of the feature map.

***Dense Layer***: This fully connected layer takes the flattened feature maps from the previous layers and maps them to the output classes, representing the different speakers. The dense layer enables the CNN to make predictions about the

speaker's identity based on the learned features.

$$y = softmax(Wx + b) \tag{29}$$

Where: $W$ is the weight matrix connecting the input features $x$ to the output classes. $b$ is the bias vector incorporated into the weighted sum of the inputs. The softmax function normalizes the output into a probability distribution, assigning each value to the likelihood of the input being in a particular class, with the highest value representing the most likely class.

**[5.5.3] Sepformer for Audio Separation**

Function: Separates mixed audio into individual sources.

By combining the Sepformer model with the CNN-based speaker recognition system, it is possible to achieve both high-quality audio separation and accurate speaker identification. This approach is particularly useful in scenarios where multiple speakers are talking simultaneously, as it allows for the extraction and classification of each speaker's voice.

See Equations (17), (18), (19), and (20) for reference.

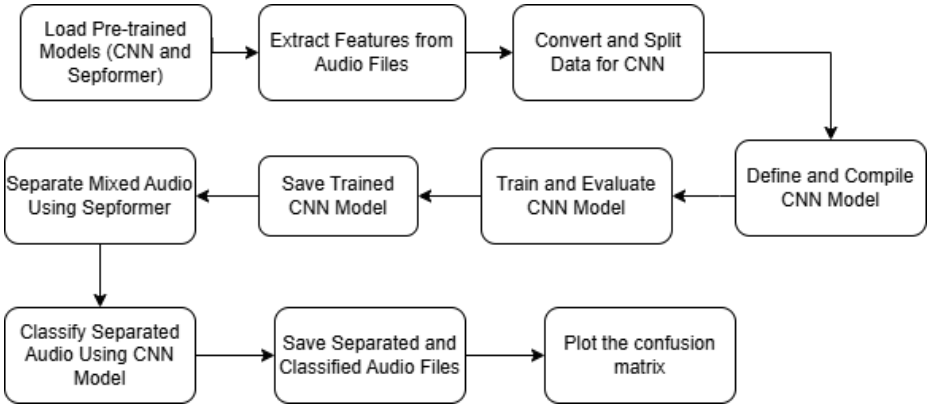


**Fig. 5.** Flowchart for Speaker Recognition and Separation using Sepformer and CNN Models

Fig. 5 outlines the steps involved in the speaker recognition and separation process using Sepformer and CNN models. The process begins by loading pre-trained models, extracting features from audio files, and converting the data for CNN training. The CNN model is defined, compiled, and trained using the prepared data. After training, the mixed audio is separated by Sepformer, and the separated audio is classified using the CNN model. The classified audio files are saved, and the confusion matrix is plotted to visualize the results.

## 6 Suitability of Methodological Framework

The chosen methodology is highly suitable for addressing the complex problem of speech enhancement, speaker separation, and recognition, as it leverages multiple advanced models tailored to different aspects of the task. Starting with

feature extraction using Mel-Frequency Cepstral Coefficients (MFCC) and Convolutional Neural Networks (CNN). MFCC is well-known for its ability to capture the spectral properties of speech, while CNNs are powerful for classifying and distinguishing between various types of audio patterns. This combination ensures that the model can accurately detect and classify different types of background noises, which is crucial for subsequent noise removal processes and enhance speech clarity.

The Waveformer is adept at handling noisy signals and improving the overall quality of the audio by reducing distortions. The Sepformer model is effective in separating audio sources that are intermingled. These models are highly suitable for precise speaker separation even in challenging environments with significant noise interference. The use of these state-of-the-art models ensures that the resulting audio signals are clean and easy to interpret, which contributes to improving speech recognition accuracy.

Finally, the inclusion of a speaker recognition model ensures that post-separation, each speaker is correctly identified and labeled. This step is critical for applications where tracking individual speakers is necessary, such as in meetings or multi-speaker environments. Overall, the chosen methodology demonstrates a comprehensive approach to tackle multi-faceted challenges associated with audio processing, making it highly suitable for the research question.

## 7 RESULTS

1] Convolutional Neural Network (CNN) and feature extraction from Mel-Frequency Cepstral Coefficients (MFCC)

**Table 1.** EVALUATION METRICS

| **Metric** | **Value** |
|---|---|
| **Accuracy** | 0.8243 |
| **Matthews Correlation Coefficient (MCC)** | 0.7407 |
| **Cohen's Kappa** | 0.7377 |
| **Log Loss** | 6.3320 |
| **Mean Squared Error (MSE)** | 0.2162 |
| **Root Mean Squared Error (RMSE)** | 0.4650 |
| **Mean Absolute Error (MAE)** | 0.1892 |
| **Macro-Averaged F1 Score** | 0.7874 |

Table 1 highlights the key evaluation metrics of the model. The accuracy is 82.43%, indicating the model correctly classifies inputs in over 82% of cases across the four classes (cat, dog, cough, and keyboard).

MCC of 0.7407 demonstrates a strong correlation between predicted and actual classes, even in imbalanced data. Cohen's Kappa of 0.7377 shows a high level of agreement between predictions and true labels beyond chance. Log Loss of 6.3320 indicates some uncertainty in the model's predictions. Low values of MSE

(0.2162) and RMSE (0.4650) reflect minimal deviation from the true labels. The Macro-Averaged F1 Score of 0.7874 suggests the model performs consistently well across all classes.

**Table 2.** CLASSIFICATION REPORT

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
|---|---|---|---|---|
| cat | 0.86 | 0.91 | 0.88 | 33 |
| dog | 0.82 | 0.78 | 0.80 | 23 |
| cough | 0.83 | 0.50 | 0.62 | 10 |
| keyboard | 0.73 | 1.00 | 0.84 | 8 |
| **Accuracy** | 0.82 | | | |
| **Macro Avg** | 0.81 (Precision), 0.80 (Recall), 0.79 (F1-Score), 74 (Support) | | | |
| **Weighted Avg** | 0.83 (Precision), 0.82 (Recall), 0.82 (F1-Score), 74 (Support) | | | |

Table 2 provides a detailed breakdown of the model's performance per class: Cat is the best-classified class with an F1-Score of 0.88 (precision: 0.86, recall: 0.91). Dog has a balanced F1-Score of 0.80, with slightly lower recall (0.78). Cough is the weakest class, with an F1-Score of 0.62, indicating difficulty in distinguishing cough sounds. Keyboard has a strong F1-Score of 0.84 but limited support (only 8 samples).

The model's overall accuracy is 82%, with both Macro-Averaged and Weighted-Averaged results showing good performance, though some classes, like cough, are harder to predict, likely due to limited data.

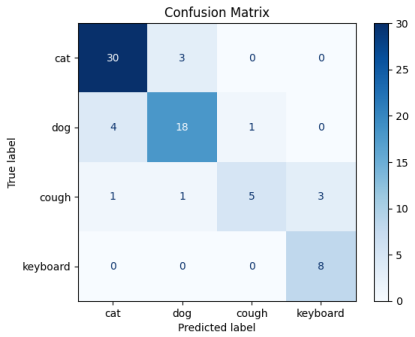


**Fig. 6.** Confusion Matrix for Multi-Class Classification

Fig. 6 evaluates the alignment between actual and predicted labels. It displays the count of accurate and erroneous predictions for each class ('cat', 'dog', 'cough', 'keyboard'). The plot highlights the diagonal cells indicating correct predictions and off-diagonal cells indicating misclassifications. For example, the

entry in the top-left cell shows how many instances the model correctly classified 'cat' when the actual label was 'cat', while the value in the first row and second column shows the frequency with which the model misclassified 'dog' when the actual label was 'cat'.

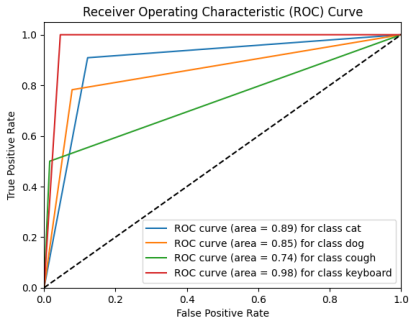


**Fig. 7.** Receiver Operating Characteristic (ROC) Curves for Audio Classification

Fig. 7 displays ROC curves for four audio classes: cat, dog, cough, and keyboard. ROC curves that approach the top-left corner reflect superior performance, signifying a higher true positive rate and a lower false positive rate for the model. A high AUC (Area Under Curve) value of 0.98 for a class like 'keyboard' suggests the model is good at distinguishing 'keyboard' from other sounds. Lower AUC for 'cough' might indicate the model struggles to differentiate 'cough' from other classes, possibly due to overlapping features or less distinct audio characteristics.

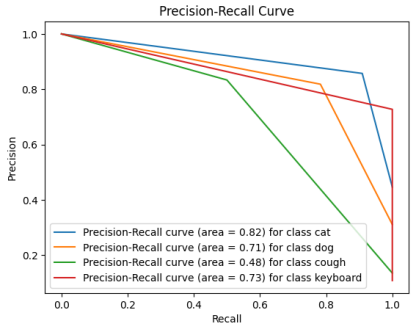


**Fig. 8.** Precision-Recall (PR) curve for multi-class classification

Fig. 8 evaluates the PR curve for multi-class classification across four different categories: 'cat', 'dog', 'cough', and 'keyboard'.The PR curves demonstrate the model's ability to balance precision and recall across multiple classes.

Precision-Recall (PR) Curve curves with higher areas indicate better performance, as both precision and recall are high. A high average precision score for 'cat' suggests the model is proficient in retrieving relevant 'cat' instances and that few non-'cat' instances are incorrectly retrieved. Lower precision-recall area for some classes indicates that for classes like 'cough', the model may either miss many true instances (low recall) or wrongly include many non-instances (low precision).

2] Waveformer

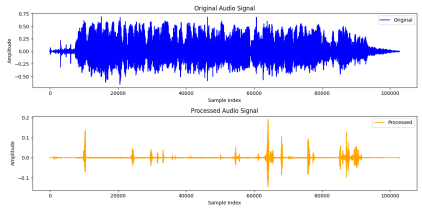


**Fig. 9.** Visualization of Original and Processed Audio Signal Waveforms of Waveformer

Fig. 9 shows two waveform graphs: the top graph represents the original audio signal, while the bottom graph illustrates the processed audio signal after applying noise reduction or other signal processing techniques.

*OriginalAudioSignal*: The waveform of the original audio shows the amplitude variations over time. The original signal serves as the reference point, representing the desired clean signal without any processing anomalies.
*ProcessedAudioSignal*: The waveform of the processed audio shows how the audio signal has been altered by the enhancement or separation process. Ideally, the processed signal should closely match the original signal in both amplitude and pattern. Differences between the original and processed waveforms indicate the presence of remaining noise, anomalies, or any distortions introduced by the processing.

**Table 3.** EVALUATION METRICS

| **Metric** | **Value** |
|---|---|
| SDR (Signal-to-Distortion Ratio) | -12.69 dB |
| STOI (Short-Time Objective Intelligibility) | 0.25 |
| PESQ (Perceptual Evaluation of Speech Quality) | 1.26 |

Table 3 evaluates the quality of the processed audio:
SDR of -12.69 dB indicates significant distortion in the processed signal. STOI of 0.25 shows a notable loss in speech intelligibility, likely due to the noise reduction

or separation methods used. PESQ of 1.26 suggests the audio has low perceived quality, with possible artifacts or unnatural sound.

These metrics highlight that the processing introduced distortion and reduced both intelligibility and overall audio quality.

3] Spectral Subtraction

**Table 4.** EVALUATION METRICS

| **Metric** | **Value** |
|---|---|
| SNR (Signal-to-Noise Ratio) | 13.864 dB |
| MSE (Mean Squared Error) | 0.0009869 |
| PESQ (Perceptual Evaluation of Speech Quality) | 1.369 |
| STOI (Short-Time Objective Intelligibility) | 0.9494 |
| SDR (Signal-to-Distortion Ratio) | 14.713 dB |
| SAR (Signal-to-Artifact Ratio) | 14.713 dB |

Table 4 evaluates the processed audio's quality:
SNR of 13.864 dB shows effective separation between the signal and background noise. MSE of 0.0009869 reflects minimal error, indicating accurate noise reduction. PESQ of 1.369 suggests an improvement in perceived speech quality but leaves room for further enhancement. STOI of 0.9494 demonstrates that speech intelligibility is well-preserved. SDR and SAR both at 14.713 dB indicate strong signal quality with minimal distortion and artifacts.

These metrics confirm that the spectral subtraction method efficiently reduces noise while maintaining high speech intelligibility and minimal processing artifacts.

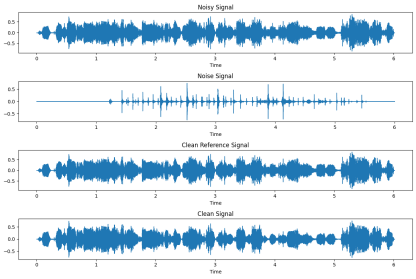


**Fig. 10.** Spectral Subtraction Process in Noise Reduction

Fig. 10 illustrates the stages of noise reduction using spectral subtraction. The top plot shows the original noisy signal, followed by the estimated noise signal in the second plot. The third plot represents a clean reference signal for comparison, and the final plot shows the cleaned signal after spectral subtraction.

$NoisySignalWaveform$: This waveform shows an audio signal that contains both the required signal and background noise. The waveform is expected to show a complex pattern, indicating the presence of multiple sound sources.
$NoiseSignalWaveform$: This represents the isolated noise component. It typically shows periodic or repetitive patterns, characteristic of mechanical noises.
$CleanReferenceSignalWaveform$: This waveform represents the ground truth or the target clean signal. The waveform should have clearer and more distinct patterns compared to the noisy signal, indicating the absence of noise.
$CleanSignalWaveform$: This waveform represents the enhanced or cleaned audio signal obtained after processing. Ideally, this should closely match the clean reference signal waveform, showing that the noise has been effectively reduced.

4] Sepformer Separation

**Table 5.** EVALUATION METRICS FOR SOURCE 1 AND SOURCE 2

| **Source** | **SNR (dB)** | **MSE** | **PESQ** | **STOI** | **SDR (dB)** | **SAR (dB)** |
|---|---|---|---|---|---|---|
| Source 1 | -8.53 | 0.092 | 1.76 | 0.954 | 15.38 | 15.38 |
| Source 2 | 6.38 | 0.002 | 1.70 | 0.969 | 13.57 | 13.57 |

Table 5 evaluates the audio quality for two separate sources:

- Source 1:
  SNR: 8.53 dB (moderate signal improvement). MSE: 0.092 (some deviation from reference). PESQ: 1.76 (low to moderate perceived quality). STOI: 0.954 (well-preserved speech intelligibility). SDR/SAR: 15.38 dB (minimal distortion and artifacts).
- Source 2:
  SNR: 6.38 dB (smaller signal improvement). MSE: 0.002 (closer match to reference). PESQ: 1.70 (similar to Source 1). STOI: 0.969 (excellent intelligibility). SDR/SAR: 13.57 dB (good, but slightly lower than Source 1).

The Sepformer process preserved intelligibility (high STOI) and minimized distortion (SDR/SAR), but perceived quality (PESQ) remains low to moderate for both sources. Source 1 performed better in SNR, SDR, and SAR, while Source 2 had better MSE and STOI. There's room for improvement in overall audio quality.

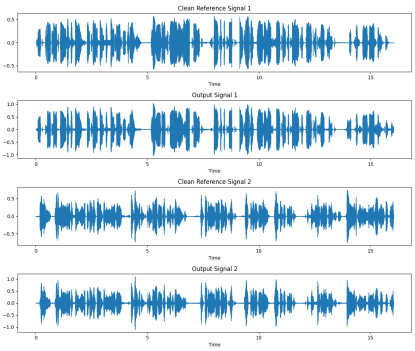


**Fig. 11.** Sepformer Separation: Visualization of Source Separation Waveforms

Fig. 11 illustrates the waveforms for both clean reference signals and their corresponding output signals after processing by the Sepformer model. The waveforms for Source 1 and Source 2 are displayed, with the top two rows showing the clean reference signals and the bottom two rows showing the output signals.

*Clean Reference Signal 1 and 2*: These waveforms represent the ground truth signals that we aim to achieve. The patterns should be clear, indicating the absence of noise.
*Output Signal 1 and 2*: These waveforms represent the signals obtained after the enhancement process. Ideally, they should closely match the clean reference signals. Differences in amplitude patterns between the clean and output signals indicate remaining noise or distortion.

5] Speaker Recognition and Separation

**Table 6.** Classification Metrics for Speaker Recognition Model

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
|---|---|---|---|---|
| (Speaker 1) Hemachandra | 1.00 | 1.00 | 1.00 | 1 |
| (Speaker 2) Vidhathri | 1.00 | 1.00 | 1.00 | 1 |
| **Accuracy** | 1.00 | | | |
| **Macro Avg** | 1.00 (Precision), 1.00 (Recall), 1.00 (F1-Score), 2 (Support) | | | |
| **Weighted Avg** | 1.00 (Precision), 1.00 (Recall), 1.00 (F1-Score), 2 (Support) | | | |

Table 6 shows the performance metrics of the speaker recognition model, including Precision, Recall, F1-Score, and Support for two speakers. Here, we are achieving a perfect score of 1, as the classification model is trained to recognize two known speakers for real-time results. Using this model as a base, we can extend it to train and classify multiple other speakers for recognition. By training the model for different numbers of speakers, we can observe variations in the score.

## 8 Comparative Evaluation and Benchmarking

To assess the performance and effectiveness of the proposed system, it is important to benchmark it against existing state-of-the-art solutions. This theoretical comparison evaluates our system relative to Google's Voice Filter and OpenAI's Whisper, and considers potential benchmarking against well-established datasets like LibriSpeech and VoxCeleb.

**[8.1] Comparison with Google's Voice Filter**
Google's Voice Filter effectively enhances a target speaker's voice in noisy environments, even with interference from another speaker, making it suitable for voice assistants. However, it may struggle in complex multi-speaker scenarios with overlapping voices.

The proposed system utilizes models like Waveformer, Sepformer, and Spectral Subtraction to excel in multi-speaker separation and handle complex audio mixtures more effectively. While Voice Filter is optimized for real-time use, the proposed system offers greater versatility in separating multiple speakers, albeit with higher computational costs.

**[8.2] Comparison with OpenAI's Whisper**
OpenAI's Whisper is effective in speech recognition and transcription, across noisy and multilingual environments.
Our system, on the other hand, focuses on speaker separation and recognition rather than ASR. While Whisper is better suited for large-scale transcription tasks, our system performs tasks involving separating and recognizing multiple speakers, more aligned with real-time speaker isolation needs.

**[8.3] Dataset Benchmarking**
*LibriSpeech*: While LibriSpeech is used for ASR benchmarking with Word Error Rate (WER), the proposed system is not focused on transcription but could still achieve competitive WER due to its noise reduction capabilities.
*VoxCeleb*: For speaker identification, the CNN-based recognition and Sepformer models would likely perform well in multi-speaker environments.

**[8.4] Model Complexity and Efficiency**
The proposed system integrates several models that improve separation and noise reduction but increase computational demand. Although Google's Voice Filter and Whisper are more efficient for real-time performance, the proposed system's strength lies in its superior handling of multi-speaker challenges.

## 9 CONCLUSION

We have presented a comprehensive approach that integrates cutting-edge models such as Convolutional Neural Networks (CNN), Waveformer, Sepformer, and Spectral Subtraction for effective background noise dissipation, speech separation, and speaker recognition. Our results provide considerable improvements in signal clarity, speech separation and accurate speaker identification. By applying these deep learning-based techniques, we have shown that speech enhancement and separation can achieve high accuracy even in real-time, noisy environments, making it suitable for various applications like telecommunications, hearing aids,voice assistants and other acoustic systems.

Despite the promising results, our models still face challenges, mostly in complex audio environments such as multiple speakers and varying noise conditions. Continued refinement and adaptation of the models in real-time could further enhance their performance. Overall, our work provides a strong proof-of-concept for deploying these models in audio processing systems, advancing the field towards more reliable and efficient solutions.

While Google's Voice Filter and Whisper are optimized for single-speaker and transcription tasks, the proposed system offers a more versatile approach for

multi-speaker separation and speaker recognition. Future evaluation on datasets like LibriSpeech and VoxCeleb would further substantiate these claims.

## 10 FUTURE SCOPE

Although our models have yielded satisfactory results in speaker separation, there are several areas where improvements are possible.

The CNN classification model, which classifies given categories of noise signals, can be further advanced to identify and predict a broader range of background noises accurately. Such prediction will be useful for the Waveformer model to extract the noise.

The Waveformer model can be enhanced to identify various background noises instead of specified targets. It needs to be trained on large datasets from numerous fields of noise to improve the efficiency of noise extraction.

Spectral subtraction uses STFT in various layers to remove noise signals and provide a clean mixed audio signal. Adjusting and fine-tuning parameters can improve the quality of subtraction. Complete noise removal helps to achieve clean separation of speech signals.

Our work, which is limited to 4 speakers, requires further advancements in the model and training dataset to achieve speech separation for any number (N)of speakers. An efficient model to accurately identify the number of speakers in mixed audio is required to achieve N speaker separation with higher efficiency. Sepformer models effectively separate speech signals without any ambiguity only with complete noise dissipation.

Speaker recognition can use GenAI models to identify the speaker after separation, which can be trained on large datasets of popular individuals. Training the classification model on a significant number of speakers will help to identify and store the audio files in their respective names. This work will be useful for the media and entertainment industry.

Real-time speech separation with high-efficiency processing and rapid output, surpassing the present state-of-the-art models, has great research potential. This will prove beneficial in the healthcare and automation industries.

## ACKNOWLEDGEMENT

The authors express their sincere appreciation to the Center for Data Sciences and Applied Machine Learning (CDSAML) in the Department of Computer Science and Engineering (CSE) at PES University, Bangalore. Their steadfast support, combined with the university's commitment to fostering a research-friendly atmosphere, has been crucial to the successful completion of this study.

## References


1. Bandhav Veluri, Justin Chan, Malek Itani, Tuochao Chen, Takuya Yoshioka, Shyamnath Gollakota: Real-time target sound extraction. arXiv:2211.02250v3 [cs.SD] (2023)
2. Marc Karam, Hasan F. Khazaal, Heshmat Aglan, Cliston Cole: Noise removal in speech processing using spectral subtraction. Journal of Signal and Information Processing 5(5), 32–41 (2014)
3. Cem Subakan, Mirco Ravanelli, Samuele Cornell, Mirko Bronzi, Jianyuan Zhong: Attention is all you need in speech separation. arXiv:2010.13154v2 [eess.AS] (2021)
4. Zhuo Chen, Naoyuki Kanda, Jian Wu, Yu Wu, Xiaofei Wang, Takuya Yoshioka, Jinyu Li, Sunit Sivasankaran, Sefik Emre Eskimez: Speech separation with large-scale self-supervised learning. arXiv:2211.05172v2 [eess.AS] (2022)
5. Wayne Xiong, Lei Wu, Frank Alleva, Jasha Droppo, Xuedong Huang, Andreas Stolcke: The Microsoft 2017 conversational speech recognition system. arXiv:1708.06073v2 [cs.CL] (2017)
6. Zhifeng Kong, Wei Ping, Ambrish Dantrey, Bryan Catanzaro: Speech denoising in the waveform domain with self-attention. arXiv:2202.07790v3 [cs.SD] (2022)
7. Hao Ma, Zhiyuan Peng, Xu Li, Mingjie Shao, Xixin Wu, Ju Liu: CLAPSep: Leveraging contrastive pre-trained model for multi-modal query-conditioned target sound extraction. arXiv:2402.17455v2 [eess.AS] (2024)
8. Eliya Nachmani, Yossi Adi, Lior Wolf: Voice separation with an unknown number of multiple speakers. arXiv:2003.01531v4 [eess.AS] (2020)
9. Sanyuan Chen, Yu Wu, Zhuo Chen, Jian Wu, Jinyu Li, Takuya Yoshioka, Chengyi Wang, Shujie Liu, Ming Zhou: Continuous speech separation with conformer. arXiv:2008.05773v2 [eess.AS] (2020)
10. Jinyu Li: Recent advances in end-to-end automatic speech recognition. arXiv:2111.01690v2 [eess.AS] (2022)
11. Pavan Nettam, Rakshitha R, Meghana Nekar, Amitram Achunala, Shylaja S S, Anantharaman P N: Human speech extraction from composite audio signal in real-time using deep neural network. In: 2024 11th International Conference on Signal Processing and Integrated Networks (SPIN-2024)
12. Shilin Wan: Research on speech separation and recognition algorithm based on deep learning. In: 2021 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS), Shenyang, China, pp. 722–725 (2021)
13. DeLiang Wang, Jitong Chen: Supervised speech separation based on deep learning: An overview. IEEE/ACM Transactions on Audio, Speech, and Language Processing 25(10), 1705–1721 (2017)
14. Zhuo Chen, Jian Wu, Naoyuki Kanda, Jinyu Li, Sunit Sivasankaran, Sefik Emre Eskimez: Continuous speech separation: Dataset and analysis. In: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, pp. 7284–7288 (2020)
15. Jian Luo, Jianzong Wang, Ning Cheng, Edward Xiao, Xulong Zhang, Jing Xiao: Tiny-Sepformer: A tiny time-domain transformer network for speech separation. In: Interspeech 2022, Incheon, Korea (2022)
16. Cem Subakan, Mirco Ravanelli, Samuele Cornell, Felix Grondin, Mirko Bronzi: Exploring self-attention mechanisms for speech separation. IEEE/ACM Transactions on Audio, Speech, and Language Processing 31, 2169–2180 (2023)
17. Jian Liu, Xue Xiang: Review of the anti-noise method in the speech recognition technology. In: 2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA), Siem Reap, Cambodia, pp. 1391–1394 (2017)

18. Laura Buera, Jasha Droppo, Alex Acero: Speech enhancement using a pitch predictive model. In: 2008 IEEE International Conference on Acoustics, Speech and Signal Processing, Las Vegas, NV, USA, pp. 4885–4888 (2008)
19. Jun Du, Yifan Tu, Li-Rong Dai, Chin-Hui Lee: A regression approach to single-channel speech separation via high-resolution deep neural networks. IEEE/ACM Transactions on Audio, Speech, and Language Processing 24(8), 1424–1437 (2016)
20. Zengmao Nian, Yu-Hao Tu, Jun Du, Chin-Hui Lee: A progressive learning approach to adaptive noise and speech estimation for speech enhancement and noisy speech recognition. In: ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, pp. 6913–6917 (2021)
21. Navneet Upadhyay, Abhijit Karmakar: Speech enhancement using spectral subtraction-type algorithms: A comparison and simulation study. Procedia Computer Science 58, 406–412 (2015)