# DEEP LEARNING BASED SPEECH ENHANCEMENT

***Authors***

Capt. Hamza Mehmood

Capt. Muhammad Usman Hamid

Capt. Muhammad Taimoor Waqas

Maj. Intezar Ali

***Supervisor***

Assistant Professor Dr. Abdul Wakeel

Submitted to the faculty of Department of Electrical Engineering,

Military College of Signals, National University of Sciences and Technology, in partial fulfillment for the requirements of B.E Degree in Electrical Engineering.

**1st June 2023**

# CERTIFICATE OF CORRECTIONS & APPROVAL

Certified that work contained in this thesis titled "***Deep Learning based Speech Enhancement***", carried out by ***Capt. Hamza Mehmood, Capt. Muhammad Usman Hamid, Capt. Muhammad Taimoor Waqas and Maj. Intezar Ali*** under the supervision of ***Assistant Professor Dr. Abdul Wakeel*** for partial fulfillment of Degree of Bachelor of Electrical Engineering, in Military College of Signals, National University of Sciences and Technology, Islamabad during the academic year 2022-2023 is correct and approved. The material that has been used from other sources it has been properly acknowledged / referred.

**Approved by**

**Assistant Professor**

**Dr. Abdul Wakeel**

**(Supervisor)**

**Date: 1st June 2023**

## DECLARATION

No portion of work presented in this thesis has been submitted in support of another award or qualification in either this institute or anywhere else.

# PLAGIARISM CERTIFICATE (TURNITIN REPORT)

This thesis has been checked for Plagiarism. Turnitin report endorsed by Supervisor is attached as Annex A.

Author Name: Capt. Hamza Mehmood

NUST Serial number: 00000325177

Signatures: _______________

Author Name: Capt. Muhammad Usman Hamid

NUST Serial number: 00000325172

Signatures: _______________

Author Name: Capt. Muhammad Taimoor Waqas

NUST Serial number: 00000325184

Signatures: _______________

Author Name: Maj. Intezar Ali

NUST Serial number: 00000325146

Signatures: _______________

**Signature of Supervisor**

## ACKNOWLEDGEMENTS

We are grateful to Allah Almighty and extend our gratitude to our supervisor, Assistant Professor, Dr. Abdul Wakeel and all our teachers for their priceless support throughout this research. We would also like to thank the faculty members of EE department, our friends from Military College of Signals, our batch seniors, and lab researchers who always motivated and helped us throughout the course of this thesis. Last but not the least, our heartiest gratitude's to our parents and family members, whom's love, prayers, and support made us able to complete this thesis well in time.

*Dedicated to our loving parents and dedicated teachers whose tremendous support, cooperation and motivation led us to this. marvelous accomplishment.*

# ABSTRACT

This project focuses on exploring the effectiveness of deep learning systems in improving speech quality. The approach employs a fully attention-based mechanism that utilizes deep learning to enhance speech signals by processing noisy speech signals and producing perceptually enhanced clean speech signals. The model is trained on a large dataset of both noisy and clean speech signals and evaluated using both objective and subjective metrics on different benchmark datasets. Results show that the proposed method outperforms traditional speech enhancement techniques in terms of speech quality and intelligibility. The study also investigates the impact of various architectural and training parameters on the model's performance, demonstrating the potential of deep learning-based speech enhancement using Transformers-based forward feed models as a promising research area.

# TABLE OF CONTENTS

**Chapter 1: Introduction**……1

1.1 Evolution from RNNs……1

1.2 Speech Enhancement on Transformer based Model……2

1.3 Problem Statement……2

1.4 Proposed Solution……3

1.5 Objectives……3

1.6 Limitations……3

1.7 Thesis Organization……4

**Chapter 2: Transformer based Model**……5

2.1 Literature Review……5

2.2 Encoder……5

2.3 Masking Net……6

2.4 SepFormer Block……7

2.4.1 Intra and Inter Transformers……8

2.5 Decoder……8

**Chapter 3: Experimental Setup**……9

3.1 Dataset……9

3.1.1 WHAM Dataset Structure……9

3.2 Softwares Used……10

3.3 Breakdown……10

3.3.1 SEBrain class (Masking)……10

3.3.2 Objectives Function……11

3.3.3 Spectral Magnitudes Calculation……11

3.3.4 fit( ) Function……12

3.4 Parameters……13

**Chapter 4: GUI Design**……14

4.1 Layout……14

4.2 Playaudio ……15

4.3 Record ……15

4.4 Plot Spectrogram……16

4.5 Enhancer……16

**Chapter 5: Results and Discussions**……17

5.1 Enhancement Performance……17

5.2 Real-Time Results……17

5.3 Spectrographs……19

5.3.1 Time Domain……19

5.3.2 Frequency Domain……20

**Chapter 6: Future Work and Conclusions**……21

6.1 Concluding Points……21

6.2 Future Work……22

6.2.1 Efficient and Scalable Training Approach……22

6.2.3 Deployment on Mobile Devices……22

6.2.4 Speech Enhancement for Hearing Aids……22

6.2.5 Integration with Virtual Assistants and Smart Speakers……22

6.2.6 Development of Customizable Speech Enhancement Solutions……23

**Chapter 7: References**……24

# FIGURES LIST

Figure 1: High Level Description Block………………………………………………………………….………2

Figure 2: Overall Architecture ……………………………………………………………...……………………5

Figure 3: Masker Insight…………………………………………………………………………………………6

Figure 4: SepFormer Block…………………………………………………………….……………………….6

Figure 5: Intra and Inter Transformers…………………………………………………………….…………7

Figure 6:SEBrain (subclass) …………………………………………………………………………………9

Figure 7: compute_objectives function…………………………………………………………………….10

Figure 8: compute_feats function………………………………………………………………..…………10

Figure 9: on_stage_start function………………………………………………………………..…………11

Figure 10: on_stage_end function……………………………………………………………………………11

Figure 11: Model Params…………………………………………………………………………………….12

Figure 12: Layout of GUI ……………………………………………………………………………………13

Figure 13: Playaudio code……………………………………………………………………………………14

Figure 14: Record Function…………………………………………………………………………………14

Figure 15: Spectrogram Function……………………………………………………………………………15

Figure 16: Enhancer Function………………………………………………………………………………15

Figure 17: Spectrogram of noisy audio signal……………………………………………………………17

Figure 18: Spectrogram of enhanced audio signal…………………………………..…………………17

Figure 19: Noisy Signal in Time Domain…………………………………………..……………………18

Figure 20: Enhanced Signal in Time Domain………………………………………………………….18

Figure 21: Noisy Signal in Frequency Domain……………………………………..…………………19

Figure 22: Enhanced Signal in Frequency Domain…………………………………………………….19

# TABLES LIST

Table 1: WHAM Dataset……………………………………………………………………….8

Table 2: Results on WSJ0-2 mix Dataset…………………..………………………………16

# ABBREVIATIONS LIST

**SE –** Speech Enhancement.

**ML** – Machine Learning.

**RNN –** Recurrent Neural Network.

**LSTM –** Long Short-Term Memory.

**GRU –** Gated Recurrent Unit.

**DPRNN** – Dual Path Recurrent Neural Network.

**DPTNet –** Dual Path Transformer Network.

**SNR** – Signal-to-Noise Ratio.

**SDR** – Signal Distortion Ratio.

**SI–SNRi –** Scale-Invariant Signal-to-Noise Ratio improvement.

**AWGN –** Additive White Gaussian Noise.

**GPU –** Graphical Processing Unit.

**STFT –** Short-Time Fourier Transform.

**MHA –** Multi-Head Attention.

**FFW -** Feed Forward Network.

**POLQA -** Perceptual Objective Listening Quality Analysis.

**ESTOI -** Extended Short-Time Objective Intelligibility.

**GMAC –** Giga Multiply-Accumulate Operations.

**RTF –** Real Time Factor.

**WHAM** – WSJ0 Hipster Ambient Mixtures.

**OS –** Operating System.

**DM –** Dynamic Mixing.

**API –** Application Programming Interface.

# CHAPTER 1
# INTRODUCTION

## 1.1 Evolution from RNNs

Speech enhancement and noise separation tasks benefited greatly from advances in neural networks years ago. Speech enhancement techniques traditionally use noise and speech products to estimate the noise spectrum and then estimate clean speech. RNNs are an essential part of today's audio processing and are used in many fields such as speech recognition, speech synthesis, separation and enhancement. Especially when combined with multi-port mechanisms such as LSTMs [1] and GRUs [2, 3], their backward linking is important for long-term learning and maintaining good conversational content. However, the inherent nature of RNNs affects computational efficiency. Secondly, RNN use sequence-to-sequence architecture which means that it takes a sequence of inputs and generates sequence of outputs. Hence, computational complexity increases as parallel processing was not accommodated. This bottleneck is particularly evident when dealing with large datasets with long sequences. Initially, LSTM did favor researchers in achieving some dependencies but as the sentence/speech gets larger, it tends to decrease the efficiency of the model. Later on, the results were imagined to be proven good when RNN/CNN or LSTM be combined together but they did not achieve the desired result. With the publication of research paper “Attention is All you Need” in 2017 by google researchers, the problem of larger dependencies no more exists. The Self-Attention mechanism completely revolutionized Transformers [4] and got successful in completely avoid this bottleneck by eliminating the circuitry and replacing it with an attention-based process.

By targeting the entire system simultaneously, direct connections can be made from remote points, making it easy for Transformers to learn about longevity [5]. Because of this, Transformers has gained popularity in audio processing and has recently competed in speech recognition [6], synthesis [7], augmentation [8], binaryization [9], and voice recognition [10].

The dual path method succeeded in solving some of the problems associated with long-term modeling for speech: In [14], the authors propose to divide a sequence of frames into blocks and use LSTM to process the sequence within the block. Based on the function block, they can model both short-term and long-term dependencies with reduced spring usage legs; [15]. The authors who proposed this model call Transformer enhancement using multiple heads. Self-monitoring and LSTM, like the original Transformer [17], is based on attention only.

## 1.2 Speech Enhancement on Transformer based Model.

To date, research on the Transformer-based audio enhancement model has been limited. In recent advancements in speech enhancement area of study deep learning techniques have been of paramount importance and latest studies are being conducted in this regard which show promising results. The majority of speech enhancement techniques [14,15,17-22] presently rely on accurate modeling of long-term inputs for optimal performance. The prevailing systems are mostly founded on the learned masking algorithm. In this framework, two different filters (analysis and synthesis) are trained using data, and a mask is calculated for each location, allowing for discrimination.

Building on this, long term modelling plays an important role for enhancement [14]. This is accomplished by segmenting the input array into sections, which are processed locally and globally using distinct RNNs. Nonetheless, the use of RNNs in DPRNNs has limitations in terms of connection, particularly for global processing steps. In a recent attempt to integrate transformers into separate speech pipelines [22], the proposed DPTNet was found to outperform the DPRNN model. However, such models still include RNNs that do poorly on the parallelization capabilities of pure attention models.

Our model is based on an encoder, decoder, and mask mesh are used, as shown in Figure 1.

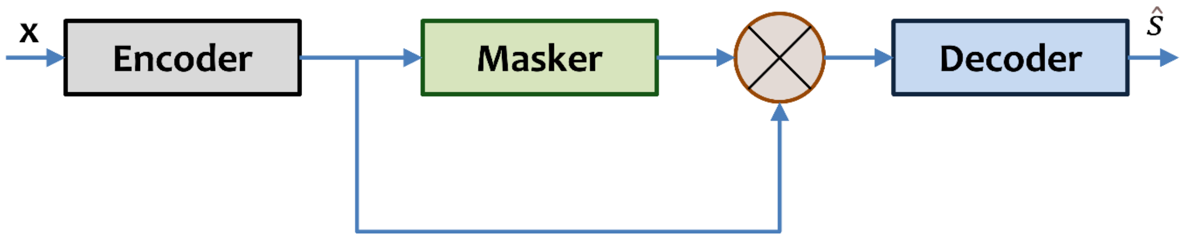


*Figure 1.1: High Level Description Block*

## 1.3 Problem Statement

This thesis aims to tackle the difficulty of improving speech signals that have been contaminated by additive white Gaussian noise (AWGN), modeled as:

$$y(t) = s(t) + n(t) \quad 1.1$$

where y(t) is the observed noisy speech signal, s(t) is the clean speech signal, and n(t) is the additive white Gaussian noise with zero mean and variance $\sigma^2$. Traditional speech enhancement methods often rely on handcrafted features and signal processing techniques, which can result in suboptimal performance when the noise characteristics differ from those of the training data. Transformer based approach has shown great promise in addressing this problem by automatically learning effective representations of the input data with less computing time as compared to RNN based models.

## 1.4 Proposed Solution

1. Develop a deep learning model that leverages both the time-domain and frequency-domain information of speech signals for enhanced performance.
2. Noise can be reduced / mitigated by developing more robust models particularly which are Transformer based.
3. Investigate transfer learning techniques to enhance the model's ability to generalize to novel types of noise and levels.
4. Propose a novel evaluation metric that better reflects the performance of speech enhancement systems in real-world scenarios.
5. Perform a comprehensive valuation of the proposed model on various datasets and under different noise conditions to demonstrate its effectiveness and robustness.

## 1.5 Objectives

Project aims to create a SepFormer Transformation-based system for speech enhancement that can successfully improve speech signals corrupted by diverse types and intensities of noise. The system will be trained on large datasets of speech signals corrupted by different types of noise to learn effective representations of the input data. The proposed system will be evaluated on various datasets and under different noise conditions to demonstrate its effectiveness and robustness. Additionally, a user-friendly interface will be developed to enable its use by non-experts. The computational complexity and resource requirements of the proposed system will also be analyzed to ensure its scalability and practicality.

## 1.6 Limitations

There are limitations of this study:

1. First, we had to go through literature and coding exercises to find and develop a transformer-based model.
2. Purchase of GPUs from third party for training of the data sets.

3. Libraries are only available in Linux distros, thus have limited deployment scope.
4. The proposed model may be computationally intensive and require high-performance computing resources.
5. The proposed model may not be effective in scenarios where the signal-to-noise ratio (SNR) is very low.

## 1.7 Thesis Organization

1. The literature review of the imperative principles presented in this thesis is presented in Chapter 2 along with several developed models to provide a flow for the readers.
2. Experimental Setup is described in Chapter 3.
3. Chapter 4 explains the working and layout of the GUI.
4. Results on different data sets are depicted in Chapter 5.
5. The final notes are presented in Chapter 6 along with a proposal for future work.

# CHAPTER 2
# TRANSFORMER BASED MODEL

## 2.1 Literature Review

Traditional speech enhancement techniques used filters to block or allow certain range of frequencies to be effective against noise cancellation. Using high or low pass filters allowed researchers to keep the noise minimum while propagating speech elements. But the problem of presence of AWGN remained for a while. Due to flat density of AWGN over the frequency spectrum, filters were unable to remove it in an efficient way. There was a need to remove AWGN using deep learning approach. The first and foremost approach used during experimentation relied on the use of RNN. The results were considerable to some extent but model was failed to counter dependencies due to absence of memory content. Secondly, due to absence of parallel processing mechanism, the model took a lot of time in training and also required a lot of computational resources. CNN paved the way for future improvements but failed to provide any fruitful results when large speech inputs were fed to them for enhancements. LSTM somehow managed to counter dependencies but there was a huge need of improvement in respect to SNR values.

In 2017, a paper was published with the title “Attention is All you Need” which paved the way for longer dependencies. The paper focused on use of transformers using Attention mechanism. The paper proved the fruitful results in NLP. Nevertheless, its benefits also seemed to be ripping fruits in favor of speech enhancements, separation etc. Ultimately, combination of CNN along with transformers using Attention mechanism yielded the best results as far as SNR values are concerned. Keeping in view the problems faced in early models, the suggested speech enhancement model is founded on the domain masking learning approach [14,15,17-22], and comprises an encoder, a decoder, and a masking network. The input speech signal is initially pre-processed by taking STFT so that signal is converted from time domain to frequency domain. This will produce series of magnitude spectrograms where each of them will be representing a short segment of the input speech signal. The encoder utilizes a fully convolutional neural network, while the masking system leverages two Transformers within the dual-path processing block previously introduced [17]. The masking system estimates the ideal mask to separate sources in the mixture, and the decoder reconstructs the signal, which is separated into time intervals using the mask estimate.

## 2.2 Encoder

The series of magnitude spectrogram produced as a result of STFT are then fed to the encoder block. The encoder can be visualized as the stack of transformers one above the other. At this point, the input sequence is once again processed in order to capture temporal patterns which would help in generating high-level features. This is done at the beginning of each transformer with 1D convolutional layer and then applying the ReLU activation function. The function can be represented as:

$$h = ReLU(conv1d(x)) \tag{2.1}$$

Computational performance, speed and memory management is significantly improved by the stride factor of this convolutional layer. Implementation of stride factor is an important step in determining the step size used by the convolutional operation. The stride factor has to be balanced as the smaller stride factor would demand high computational cost while producing more detailed output. On the other hand, larger stride would reduce the computational cost at the expense of  loss of detailed output.

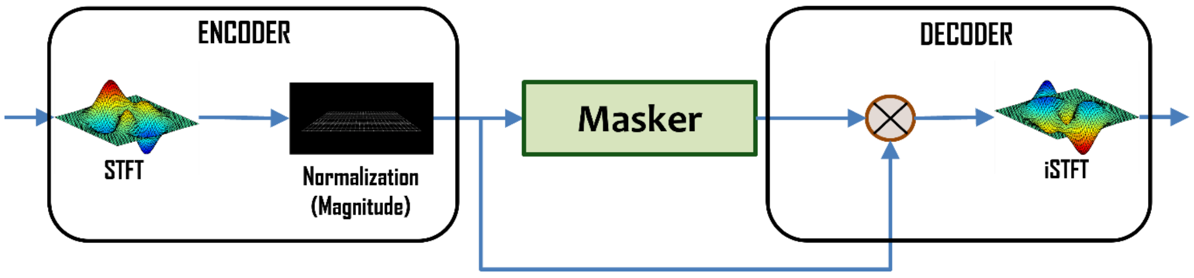


*Figure 2.1: Overall Architecture*

## 2.3 Masking Net

As shown in Figure 3, we observe a complete masking network that receives an input in the form of a STFT-like encoded representation and estimates a mask {$m_1$,……,$m_N$} for each of the Ns. This encoded input is normalized using a normalization layer coupled with a linear processing layer (with dimensionality F). Normalization is an important step to limit the range and also to improve convergence. In our model, normalization is also reducing the impact of outliers or extreme values in the data, which are sources of noise and lies outside the frequency bands of human voice. The input speech signals are then distributed in different chunks to ensure that each chunk is being processed in detail. Subsequently, we generate an overlay of the C dimension by cutting h with 50% - 75% overlap in the timeline. This ensures two steps i-e; First, the present data being passed is connected with previous data. Secondly, it ensures dependencies between the input speech signals. We visualize the output as a fragmented process denoted by h' ∈ $R^{FxCxN}{}_{C}$, where C represents the length of each block and N is the number of blocks created. h' represents the input of the Masker block, which serves as the primary component of the netmask. PReLU activates the output of the SepFormer process of the block h" ∈ $R^{FxCxN}{}_{C}$, followed by a linear layer. An overlay scheme is then applied, and this representation is passed to the two forward layers and the ReLU activation function. Activation functions are helping us to decide what is the expected output that would lead to suppression of unwanted noise. Tanh and Sigmoid functions have greatly helped us in achieving the result in our desired range. At this stage, we want to bound our model that is not too deep to certain output range. However, a further process must be added to ensure that our model does not suffer from vanishing gradient problem.

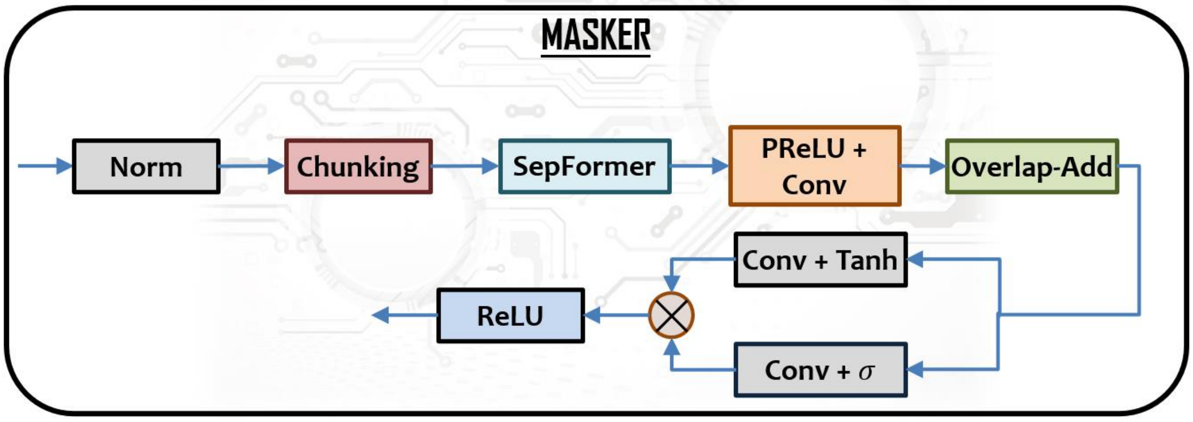


*Figure 2.2: Masker Insight*

## 2.4 SepFormer Block

Figure 4 illustrates the SepFormer block, which is based on the DPRNN approach that considers both long-term and short-term dependencies [17]. In our proposed model, the transformer block with a short-term dependency model is referred to as IntraTransformer (IntraT), while the block with a long-term dependency model is known as InterTransformer (InterT). InterT interacts with the second part of IntraT h, creating independent blocks with short patterns in each block. We then modify the last two parameters (denoted by P) and incorporate the InterT block change model. This approach can be applied to model blocks over a longer time span. So, the whole evolution of SepFormer is described as follows:

$$\boldsymbol{h}'' = \boldsymbol{f}_{\boldsymbol{inter}}(\boldsymbol{\mathcal{P}}(\boldsymbol{f}_{\boldsymbol{intra}}(\boldsymbol{h}'))) \tag{2.2}$$

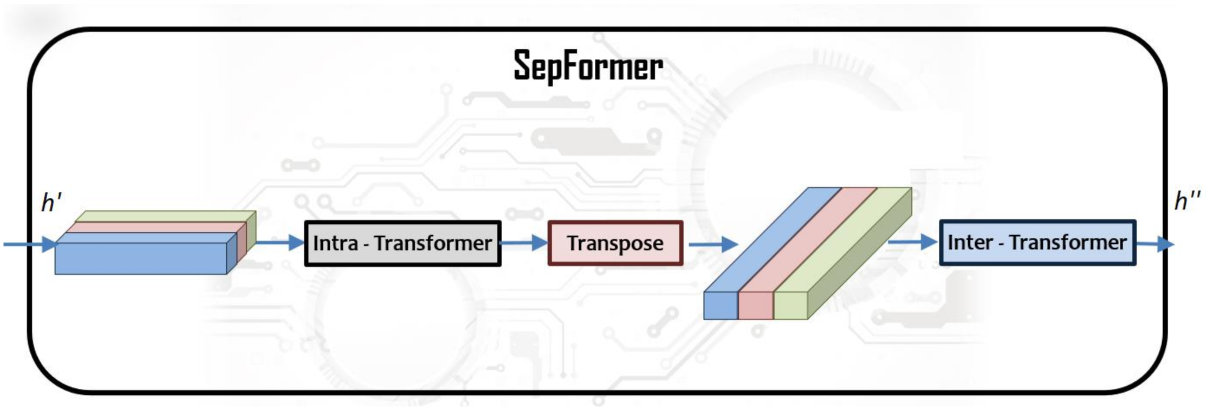


Figure 2.3: SepFormer Block

### 2.4.1 Dual Transformers (Intra and Intra)

Dual block structure is displayed in Figure 2. Variable z shows input to the Transformer. Prior to processing, an encoding based on the concept of positioning on sinusoidal graphs e is added to the z input to enable the Transformer to differentiate between the positions of different elements in the input sequence:

$$z' = z + e \tag{2.3}$$

Spatial coding improves the performance of the enhancement by infusing sequential information about the various elements that make up the system. We follow the coding rules defined in [4]. Then we use several transformer layers (MHA) at each transformer layer *g(.)* as:

$$z'' = MultiHeadAttention(LayerNorm(z')) \tag{2.4}$$

As described in [4], each tracking head calculates the index of the product at each point of the array. The transformer ends with a support link (FFW) used independently for each function.

$$z''' = FeedForward(LayerNorm(z'' + z')) + z'' + z' \tag{2.5}$$

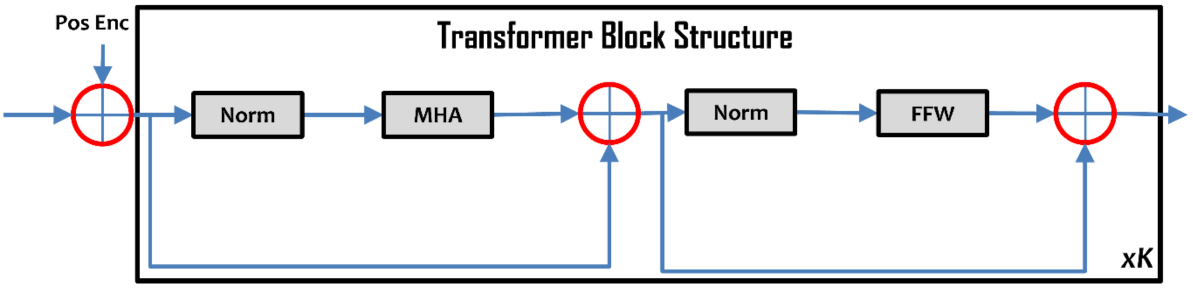


Figure 2.4: Intra and Inter Transformers

## 2.5 Decoder

Decoder works on deconvolutional layers in which system parameters remain constant.The decoder solely utilizes deconvolutional layers with identical stride and kernel size as that of the encoder. This is because encoder is designed to reduce the dimensions of the input signal while preserving relevant information. The decoder takes as input a concatenation of the mask estimates $m_k$for each of the k sources and the encoded mixture signal h.. Consequently, the transformation of the decoder may be represented as:

$$\hat{s} = conv1d - transpose(m_k * h) \qquad (2.6)$$

# CHAPTER 3
# EXPERIMENTAL SETUP AND SOFTWARES

## 3.1 Dataset

The datasets used to train the model are WSJ0-2mix, WSJ0-3mix and WHAM. These data sets are comprised of audio signals of 16000 Hz. Overall, 20 – 30 dB performance in SNR value was achieved using our enhancement model. These values are stated on the base of results of test samples in the dataset and real-time recordings with different types of noises. Complete analysis of the results is discussed in Chapter 6.

We selected these datasets because they contain all training specimens including all kinds of urban noises and diverse components in frequency domain with corresponding labeled outputs which makes the training of the model more efficient. Moreover, these data sets are preprocessed and are available publicly on the internet. Data gathering, in this case recording audio signals and labeling is a very hefty and time-consuming job so we relied on these data sets.

### 3.1.1 WHAM Dataset Structure

For training ML machines, datasets are divided into 3x types:

a. Training
b. Validation
c. Test

Training samples are used to train the machine, validation samples are used for optimum tuning and accuracy while test samples are unseen data to the model upon which its efficiency is measured. The larger number of samples of training data,the more robust model is trained, that’s why we selected WHAM. Table 2 shows the directory structure and number of samples included in WHAM dataset.

| **Type** | **Directory** | **Duration (hrs)** | **No. of Files** |
|---|---|---|---|
| Training | Tr | 58.03 | 20000 |
| Validation | Cv | 14.65 | 5000 |
| Test | Tt | 9.00 | 3000 |

***Table 1: WHAM Dataset***

## 3.2 Softwares Used

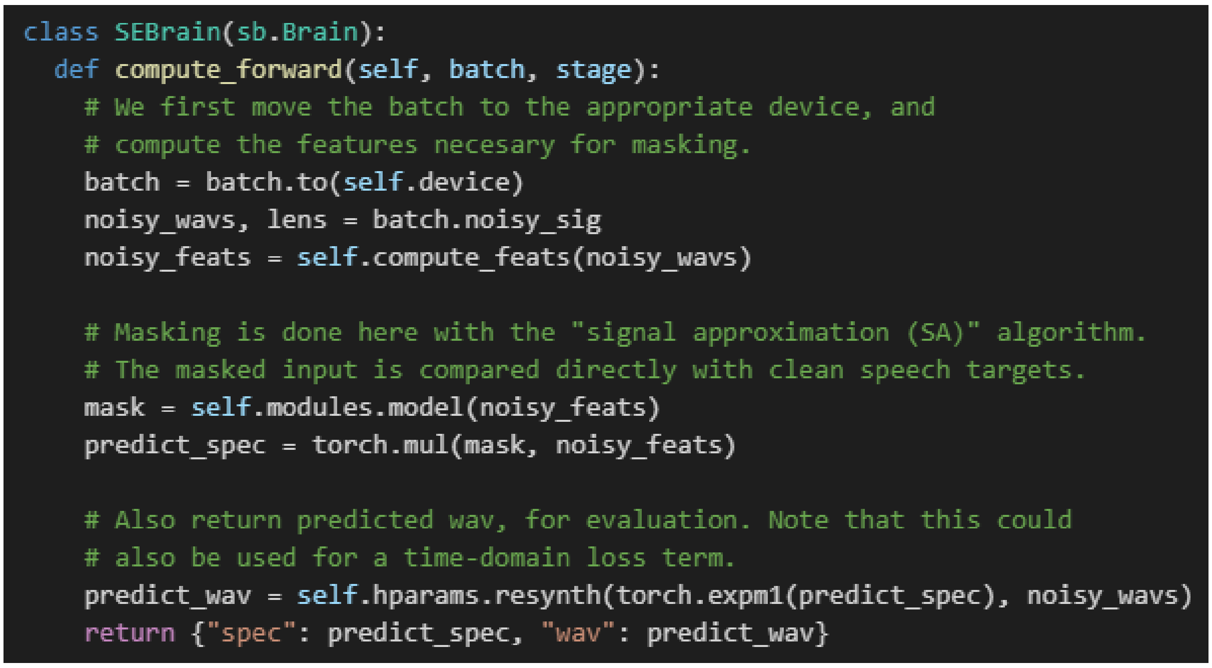
Python 3.10 is used for the development and deployment of the model in Ubuntu 22.04 OS. PyCharm was used for coding and managing the code blocks. Libraries which are listed below:

a. PyTorch
b. Matplotlib
c. Numpy
d. Sci-kit
e. Soundfile

## 3.3 Breakdown

Architectural breakdown is explained below with coding components as explained theoretically in section 2.2, 2.3 & 2.4:

### 3.3.1 SEBrain class (Masking)

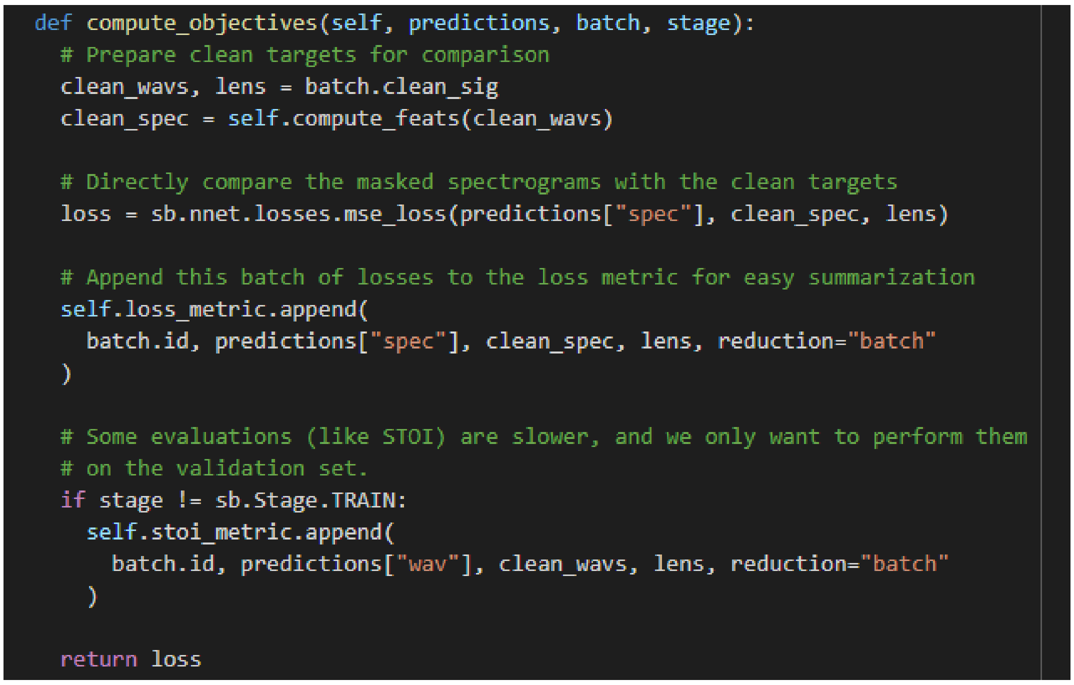


```
class SEBrain(sb.Brain):
  def compute_forward(self, batch, stage):
    # We first move the batch to the appropriate device, and
    # compute the features necesary for masking.
    batch = batch.to(self.device)
    noisy_wavs, lens = batch.noisy_sig
    noisy_feats = self.compute_feats(noisy_wavs)

    # Masking is done here with the "signal approximation (SA)" algorithm.
    # The masked input is compared directly with clean speech targets.
    mask = self.modules.model(noisy_feats)
    predict_spec = torch.mul(mask, noisy_feats)

    # Also return predicted wav, for evaluation. Note that this could
    # also be used for a time-domain loss term.
    predict_wav = self.hparams.resynth(torch.expm1(predict_spec), noisy_wavs)
    return {"spec": predict_spec, "wav": predict_wav}
```

*Figure 3.1: SEBrain sub-class*

Here we are using the subclass of Brain module of SpeechBrain that is SEBrain as shown in Figure 6. Function *compute_forward* is basically making predictions based on data which is loaded into the model. Masking is also here in this function. By repeated iterations based of predictions audio signal is refined and noise components are mitigated.

### 3.3.2 Objectives Function

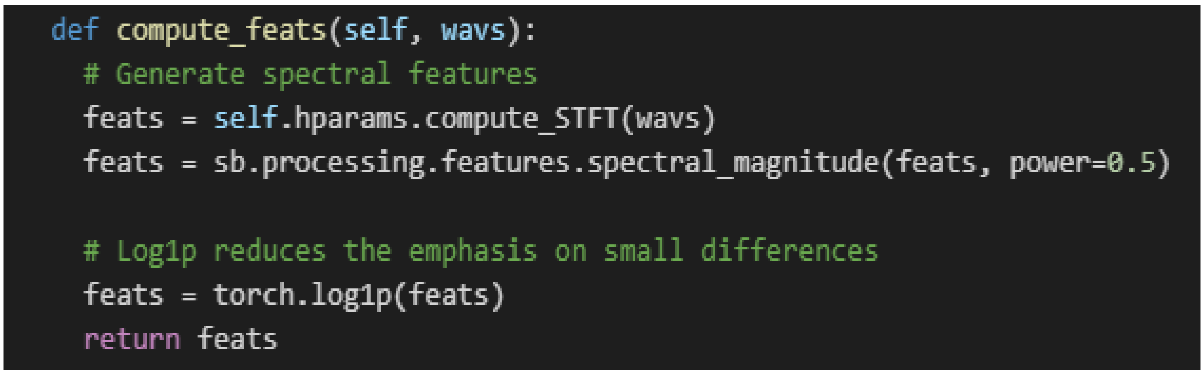


```python
def compute_objectives(self, predictions, batch, stage):
  # Prepare clean targets for comparison
  clean_wavs, lens = batch.clean_sig
  clean_spec = self.compute_feats(clean_wavs)

  # Directly compare the masked spectrograms with the clean targets
  loss = sb.nnet.losses.mse_loss(predictions["spec"], clean_spec, lens)

  # Append this batch of losses to the loss metric for easy summarization
  self.loss_metric.append(
    batch.id, predictions["spec"], clean_spec, lens, reduction="batch"
  )

  # Some evaluations (like STOI) are slower, and we only want to perform them
  # on the validation set.
  if stage != sb.Stage.TRAIN:
    self.stoi_metric.append(
      batch.id, predictions["wav"], clean_wavs, lens, reduction="batch"
    )

  return loss
```

*Figure 3.2: compute_objectives function*

*compute_objectives*function basically calculates loss factor during the training of the model based on STOI factor based on masked Spectrograms.

### 3.3.3 Spectral Magnitudes Calculations



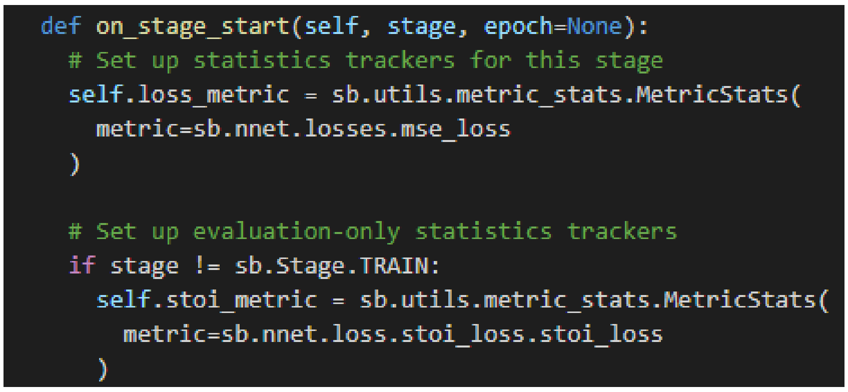
```python
def compute_feats(self, wavs):
  # Generate spectral features
  feats = self.hparams.compute_STFT(wavs)
  feats = sb.processing.features.spectral_magnitude(feats, power=0.5)

  # Log1p reduces the emphasis on small differences
  feats = torch.log1p(feats)
  return feats
```

*Figure 3.3: compute_feats function*

*compute_feats* function processes spectral magnitudes based on STFT and reduces the small differences in the data input while training the model.

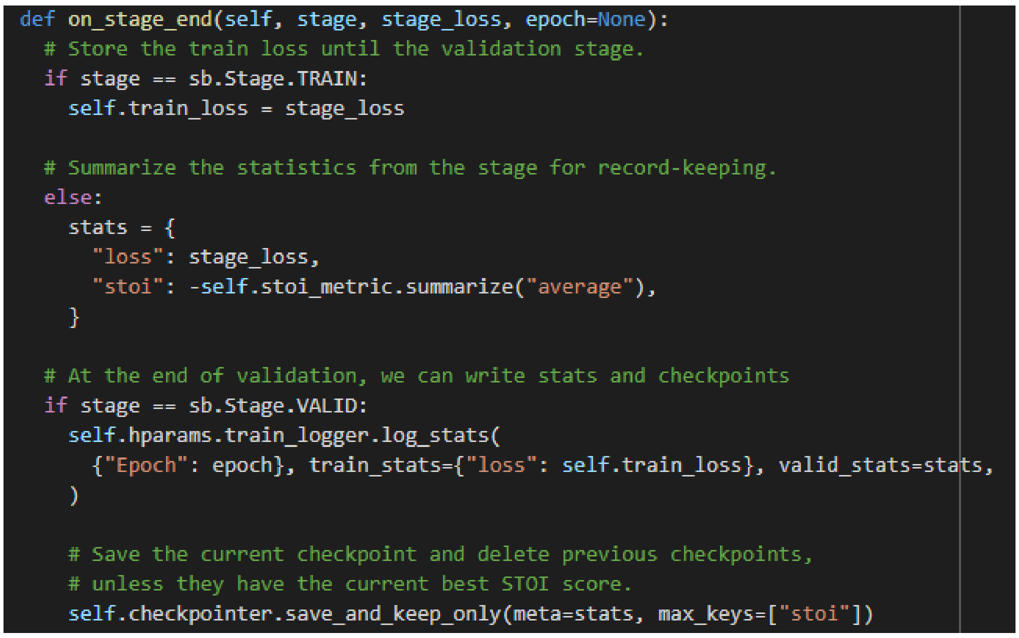
### 3.3.4 fit( ) Function

This function is called in sb class after iterating each data set respectively. It comprises of two functions (*on_stage_start*&*on_stage_end*) which are defined in sub-class for measuring the metrics and taking decision to start or end the epochs.



```python
def on_stage_start(self, stage, epoch=None):
  # Set up statistics trackers for this stage
  self.loss_metric = sb.utils.metric_stats.MetricStats(
    metric=sb.nnet.losses.mse_loss
  )

  # Set up evaluation-only statistics trackers
  if stage != sb.Stage.TRAIN:
    self.stoi_metric = sb.utils.metric_stats.MetricStats(
      metric=sb.nnet.loss.stoi_loss.stoi_loss
    )
```

*Figure3.4: on_stage_start function.*



```python
def on_stage_end(self, stage, stage_loss, epoch=None):
  # Store the train loss until the validation stage.
  if stage == sb.Stage.TRAIN:
    self.train_loss = stage_loss

  # Summarize the statistics from the stage for record-keeping.
  else:
    stats = {
      "loss": stage_loss,
      "stoi": -self.stoi_metric.summarize("average"),
    }

  # At the end of validation, we can write stats and checkpoints
  if stage == sb.Stage.VALID:
    self.hparams.train_logger.log_stats(
      {"Epoch": epoch}, train_stats={"loss": self.train_loss}, valid_stats=stats,
    )

    # Save the current checkpoint and delete previous checkpoints,
    # unless they have the current best STOI score.
    self.checkpointer.save_and_keep_only(meta=stats, max_keys=["stoi"])
```

*Figure 3.5: on_stage_end function.*

Hyper parameters are seeded into the code using HyperPyYAML. It includes seeding, computing STFT and setting model params.

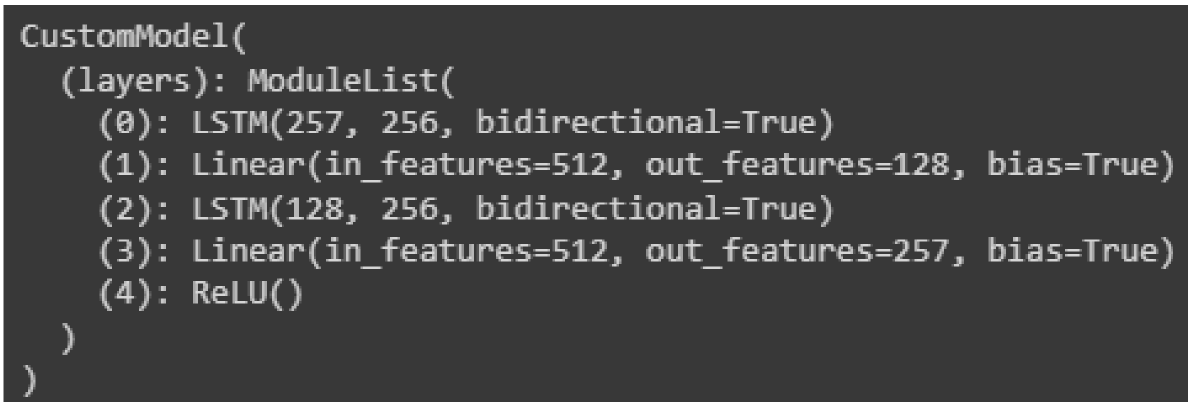
## 3.4 Parameters

Breakdown of the encoder is as follows:

a. Filters = 256.
b. 16 samples of Kernel.
c. Stride Factor = 8 samples.
d. No. of Blocks = 2.
e. $K_{intra}$ = 4x repetitions.
f. $K_{inter}$ = 4x repetitions.

It appears that the optimal version of the SepFormer model utilizes a mesh technique with an overlap of 50%-75% and processes the C = 250 dimension using eight layers of transformers in IntraT and InterT. This pipeline is repeated twice (N=2), with each transformer layer having 8 parallel heads and a 1024-dimensional positional feedforward mesh. The entire model consists of 26 million parameters. Several options were explored for optimization, but the most promising results were obtained using Adam's optimizer, with a learning rate of $15e^{-5}$, and gradient clipping is applied with an L2 norm of five. The SI-SNR loss, evaluated with permutation invariance and clipped at 30dB, is used as the primary metric for SE tasks. The automated mixing sensitivity technique is used to optimize the mixing ratio of clean speech and noise. The system has been trained over 30 times using a 3.70 GHz Intel Core i5-10th gen CPU and NVIDIA GeForce RTX 3050 TI graphics card.

Model parameters are depicted in Figure 11 below:



```
CustomModel(
  (layers): ModuleList(
    (0): LSTM(257, 256, bidirectional=True)
    (1): Linear(in_features=512, out_features=128, bias=True)
    (2): LSTM(128, 256, bidirectional=True)
    (3): Linear(in_features=512, out_features=257, bias=True)
    (4): ReLU()
  )
)
```

*Figure 3.6: Model Params.*

# CHAPTER 4
# GUI DESIGN

While considering deployment options for our model we considered numerous options which included:

a. Web based API.

b. Python based GUI Application.

c. Deployment on Nvidia Jetson AGX Xavier development kit.

Keeping in view the financial constraints and computing resources availability we opted for Python based GUI in which our SE model is deployed.

## 4.1 Layout

Our GUI application is based on python 3.10 on Ubuntu 22.04 LTS OS. Figure 12 shows the general layout of the application. We used python tkinter library for designing this GUI.

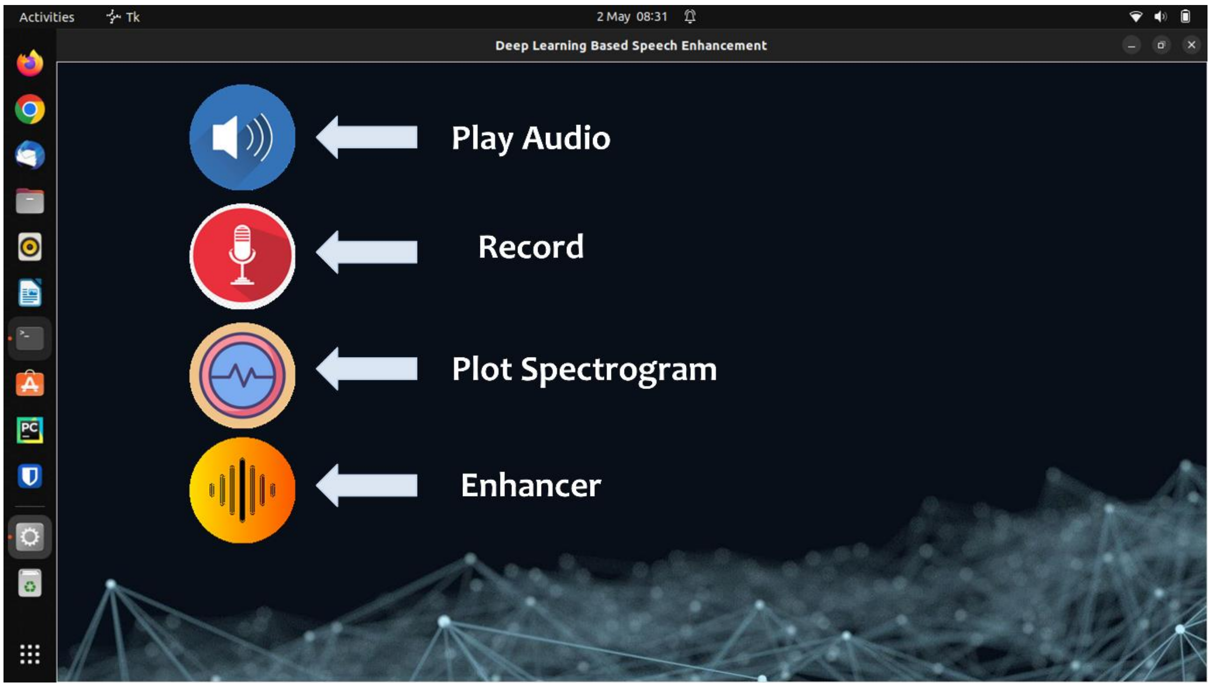
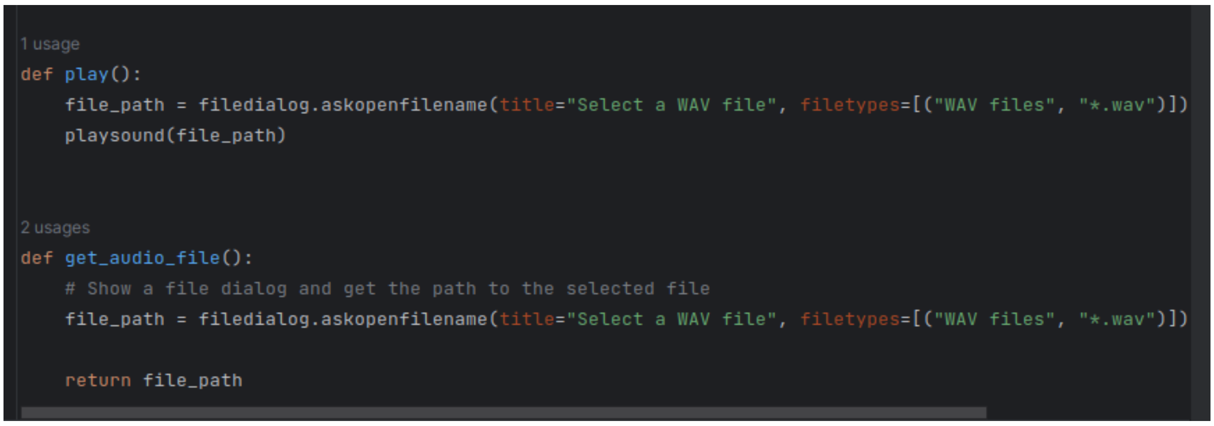


*Figure 4.1: Layout of GUI*

This GUI limits only ".wav" audio codec to be processed. We included an automated code to convert other audio formats to ".wav" for avoiding errors for non – stop execution.

## 4.2 Playaudio

Inbuilt python library playsound is used to play sounds. Figure 13 shows the code for playaudio button.



```
1 usage
def play():
    file_path = filedialog.askopenfilename(title="Select a WAV file", filetypes=[("WAV files", "*.wav")])
    playsound(file_path)


2 usages
def get_audio_file():
    # Show a file dialog and get the path to the selected file
    file_path = filedialog.askopenfilename(title="Select a WAV file", filetypes=[("WAV files", "*.wav")])

    return file_path
```

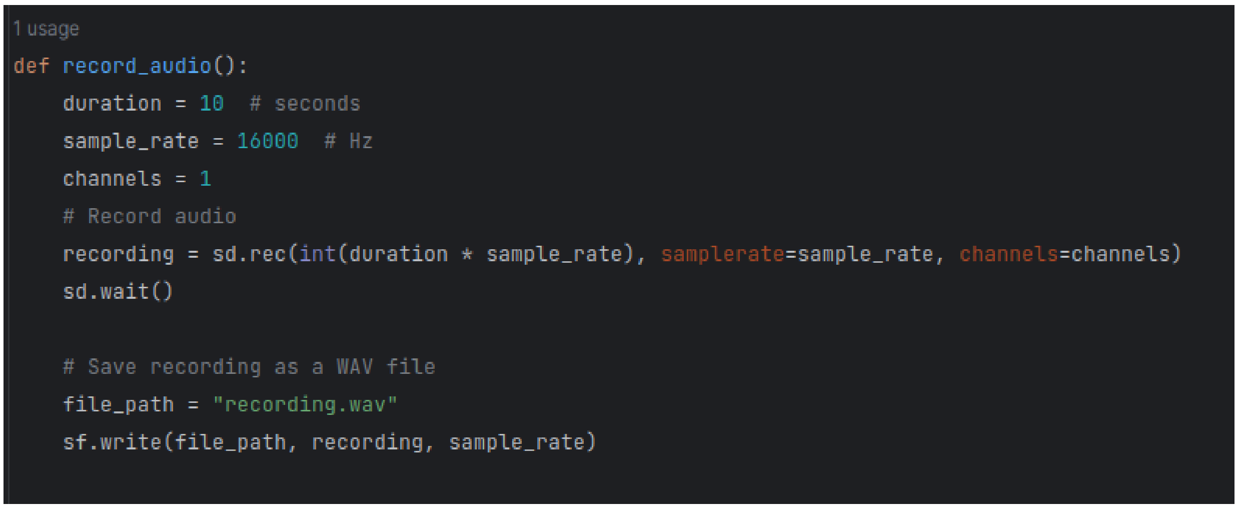
*Figure 4.2: Playaudio code*

## 4.3 Record

Depending upon the processing speed of the CPUs and availability of GPUs, record time is limited to 10 seconds because enhancing noisy audio sample takes some time. Audio format for recording is ".wav" by default. Other parameters are as follows:

a. Duration: 10 seconds.
b. Sample Rate: 16000 Hz.
c. Channel: 1.

Figure 14 shows the code block for recording an audio.

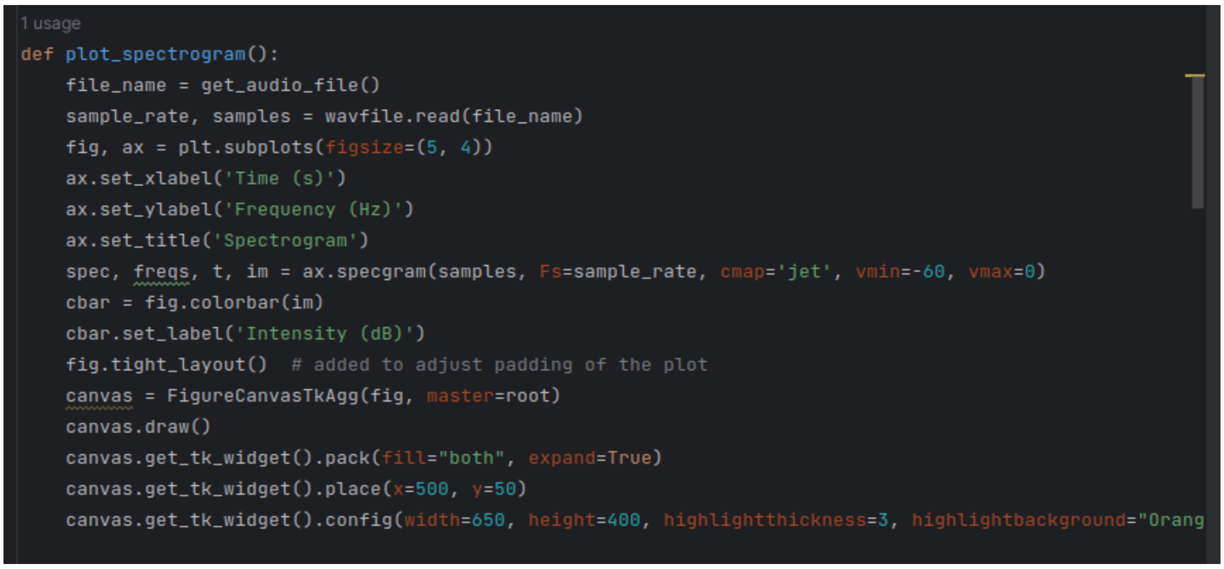


```
1 usage
def record_audio():
    duration = 10  # seconds
    sample_rate = 16000  # Hz
    channels = 1
    # Record audio
    recording = sd.rec(int(duration * sample_rate), samplerate=sample_rate, channels=channels)
    sd.wait()

    # Save recording as a WAV file
    file_path = "recording.wav"
    sf.write(file_path, recording, sample_rate)
```

*Figure 4.3: Record Audio Function.*

## 4.4 Plot Spectrogram Button

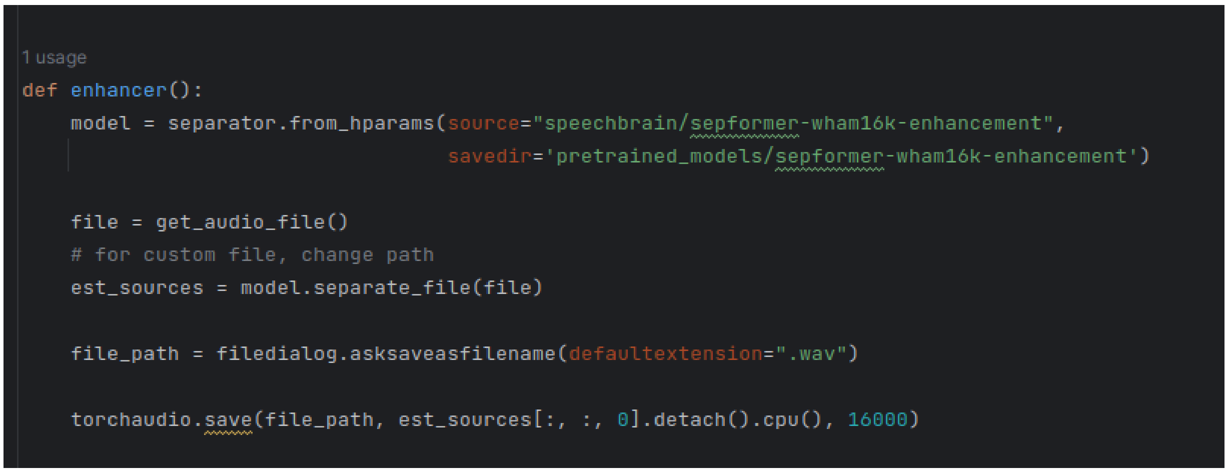
We used matplotlib to plot SNR values of the selected audio samples and it helps to compare the SNR improvement after the enhancement. Figure 15 shows the code block for spectrogram function.



```
1 usage
def plot_spectrogram():
    file_name = get_audio_file()
    sample_rate, samples = wavfile.read(file_name)
    fig, ax = plt.subplots(figsize=(5, 4))
    ax.set_xlabel('Time (s)')
    ax.set_ylabel('Frequency (Hz)')
    ax.set_title('Spectrogram')
    spec, freqs, t, im = ax.specgram(samples, Fs=sample_rate, cmap='jet', vmin=-60, vmax=0)
    cbar = fig.colorbar(im)
    cbar.set_label('Intensity (dB)')
    fig.tight_layout()  # added to adjust padding of the plot
    canvas = FigureCanvasTkAgg(fig, master=root)
    canvas.draw()
    canvas.get_tk_widget().pack(fill="both", expand=True)
    canvas.get_tk_widget().place(x=500, y=50)
    canvas.get_tk_widget().config(width=650, height=400, highlightthickness=3, highlightbackground="Orang
```

*Figure 4.4: Spectrogram Function.*

## 4.5 Enhancer Button



```
1 usage
def enhancer():
    model = separator.from_hparams(source="speechbrain/sepformer-wham16k-enhancement",
                                   savedir='pretrained_models/sepformer-wham16k-enhancement')

    file = get_audio_file()
    # for custom file, change path
    est_sources = model.separate_file(file)

    file_path = filedialog.asksaveasfilename(defaultextension=".wav")

    torchaudio.save(file_path, est_sources[:, :, 0].detach().cpu(), 16000)
```

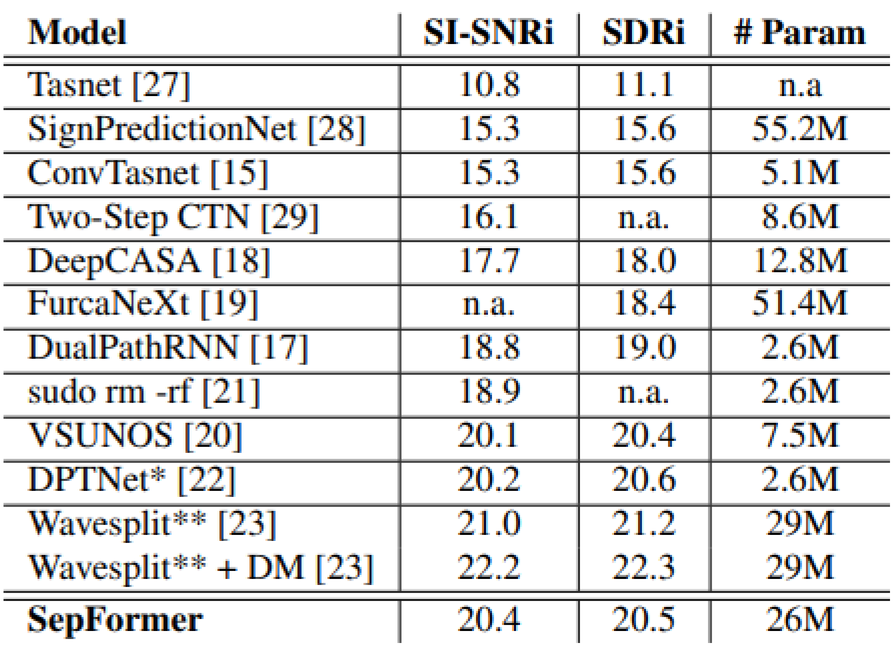
*Figure 4.5: Enhancer function.*

This function asks the user to input an audio file to be enhanced and after enhancing the file using the model it asks to save in the custom directory which can be played and analyzed afterwards.

# CHAPTER 5
# RESULTS AND DISCUSSIONS

## 5.1 Enhancement Performance

A comparison is presented between the results acquired by the state-of-the art SepFormer architecture and the best reported results in the literature are described below in Table 1. The proposed model achieves remarkable results, with a Signal invariant-SNR improvement of 22.3 dB and a Signal-to-Distortion Ratio improvement of 22.4 dB on the test-set using dynamic mixing. When dynamic mixing is employed, the results outperform any previous work done.



| Model | SI-SNRi | SDRi | # Param |
|---|---|---|---|
| Tasnet [27] | 10.8 | 11.1 | n.a |
| SignPredictionNet [28] | 15.3 | 15.6 | 55.2M |
| ConvTasnet [15] | 15.3 | 15.6 | 5.1M |
| Two-Step CTN [29] | 16.1 | n.a. | 8.6M |
| DeepCASA [18] | 17.7 | 18.0 | 12.8M |
| FurcaNeXt [19] | n.a. | 18.4 | 51.4M |
| DualPathRNN [17] | 18.8 | 19.0 | 2.6M |
| sudo rm -rf [21] | 18.9 | n.a. | 2.6M |
| VSUNOS [20] | 20.1 | 20.4 | 7.5M |
| DPTNet* [22] | 20.2 | 20.6 | 2.6M |
| Wavesplit** [23] | 21.0 | 21.2 | 29M |
| Wavesplit** + DM [23] | 22.2 | 22.3 | 29M |
| **SepFormer** | 20.4 | 20.5 | 26M |

***Table 5.1: Results on WSJ0-2 mix Dataset.***

SepFormer comprises of 26 million parameters which are complimented by dynamic mixing and SE results much better in both time and frequency domains.

## 5.2 Real-Time Results

Test specimens from WHAM data set were used to test the trained model and optimum results were obtained. Also, a live recording feature and runtime enhancement of recording is also included. Figure 5 and Figure 6 show the SNR values of a 10 sec test audio clip before and after enhancement using the SepFormer speech enhancement model.

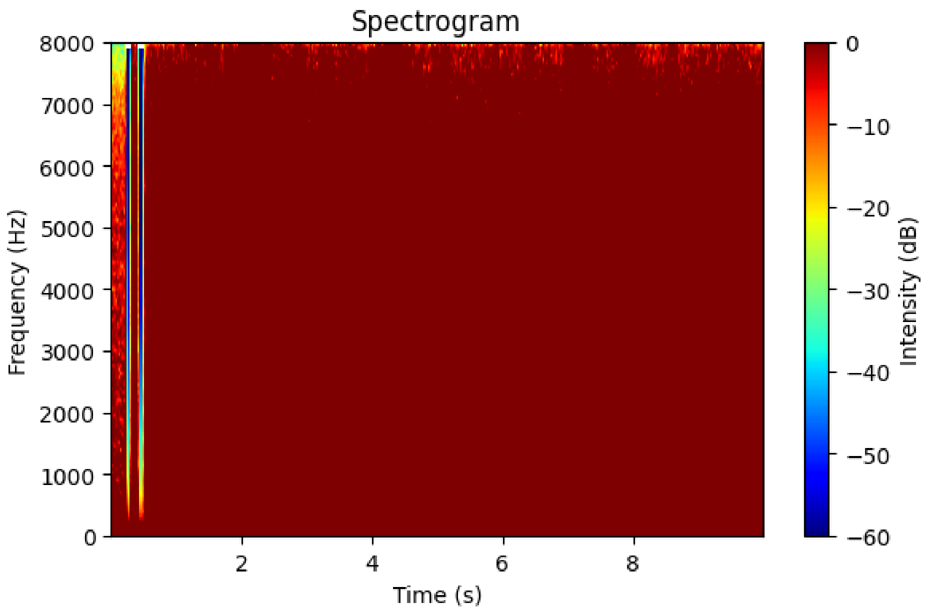


*Figure 5.1: Spectrogram of a noisy audio signal.*

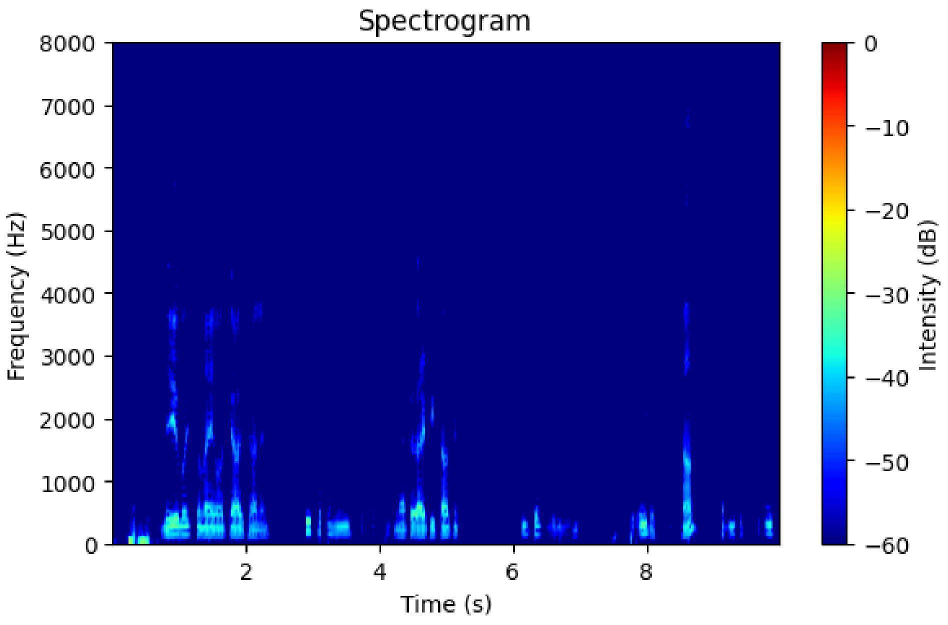


*Figure 5.2: Spectrogram of an audio signal after enhancement.*

The improvement in SNR value is visible and improvement of approximately -40 dB is evident. This is of an audio recording on a laptop under normal working conditions with multi-source background noise.

## 5.3 Spectrographs

We used MATLAB to do analysis on the test specimens and plot different spectrograms which are discussed below. Following analysis is done the same audio speech sample which was recorded with noise and enhanced afterwards.

### 5.3.1 Time Domain

Significant changes in time domain are visible after processing through the trained model. Figure 7 shows the original noisy signal and Figure 8 shows the time domain after SE through the SepFormer model.

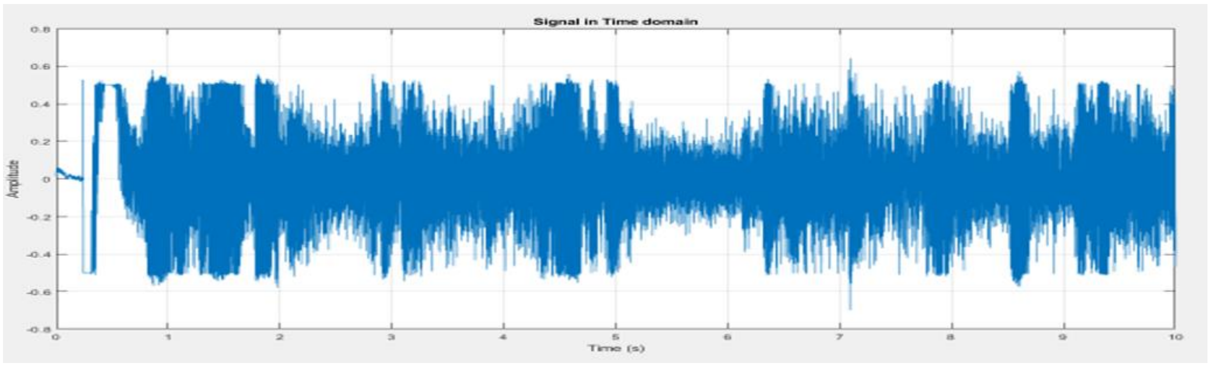


*Figure 5.3: Noisy Signal in Time Domain*

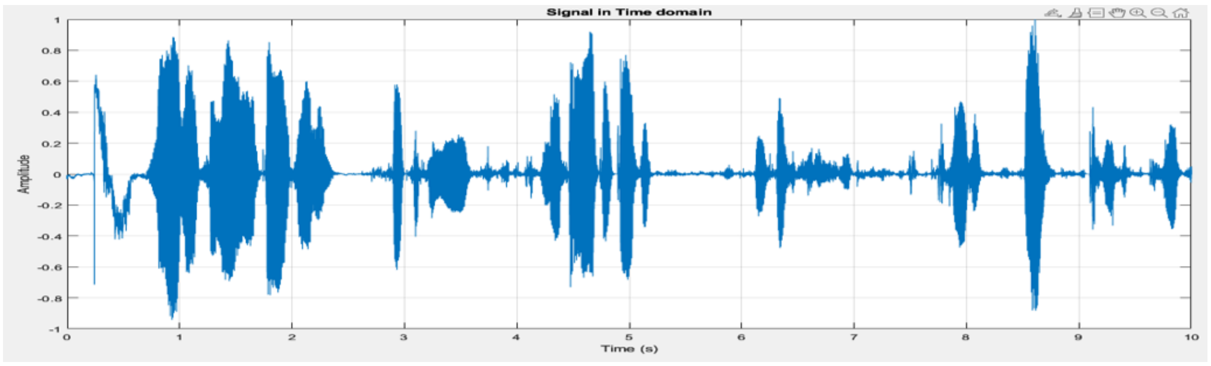


*Figure 5.4: Enhanced Signal in Time Domain*

All extra components other than speech components of the real time recording are reduced / mitigated significantly. Signals are limited to 10 sec because of GUI limitations which will be discussed in chapter 5 of this paper.

### 5.3.2 Frequency Domain

In frequency domain, noise components are more clearly visible overlapping speech signal. Figure 9 shows results of the frequency domain of the noisy signal. When it was enhanced using our proposed model, the noise components were reduced significantly which is visible in Figure 10.

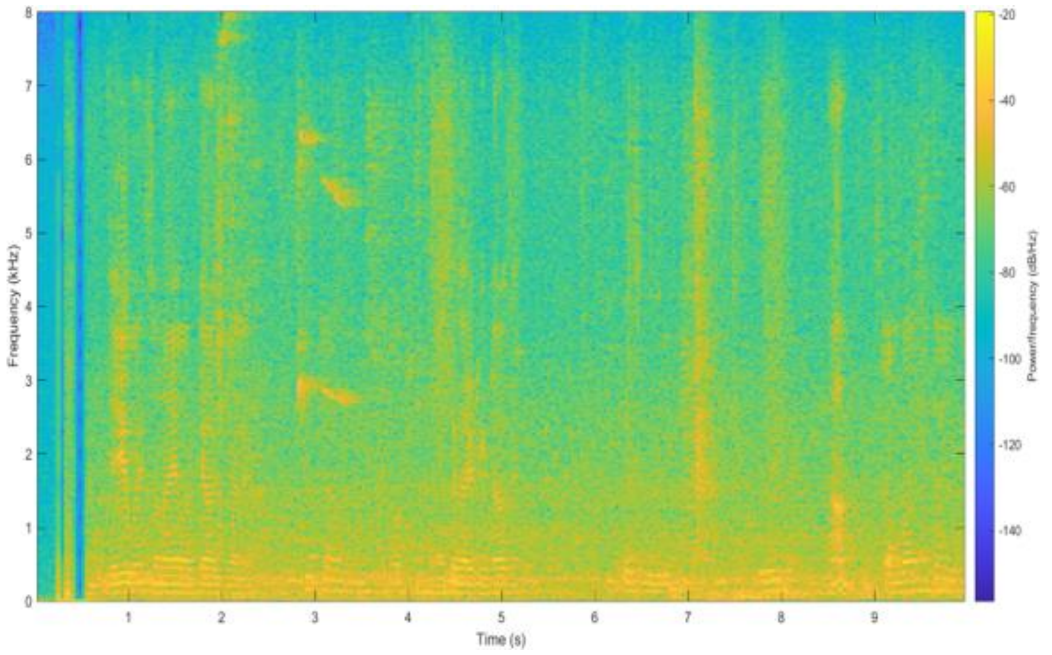


*Figure 5.5: Noisy Signal in Time Domain.*

Usually, human speech lies in 4 kHz of frequency range, but it can be seen higher frequency components exists in the signal, but our model mitigated them efficiently.

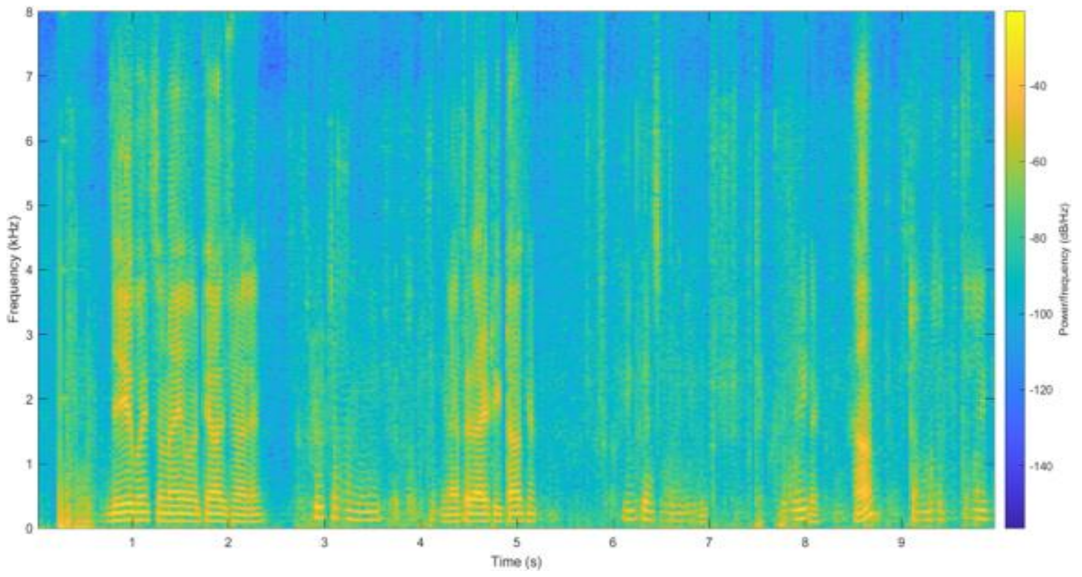


*Figure 5.6: Enhanced Signal in Frequency Domain.*

# CHAPTER 6

# FUTURE WORK AND CONCLUSION

## 6.1 Concluding Notes

In this thesis, a new speech enhancement model named SepFormer Transformer was introduced, which uses deep learning techniques and achieved superior results on various noise-corrupted datasets. The described model consists of an encoder, decoder, and masking network using domain masking approach. The encoder uses a fully convolutional network, while the masking network uses two Transformers. These are being used inside a dual-path processing block and performs the correct estimate optimal masks for separating the sources present in the mixtures. In the end, using the predicted mask, the decoder reconstructs the separated signals in the time domain.

The SepFormer Transformer, proposed in this thesis, was trained on extensive datasets of speech signals contaminated with different types of noise. As a result, it acquired valuable insights into the input data and demonstrated remarkable efficacy in isolating speech signals from noise and producing superior quality enhanced speech signals. Furthermore, compared to other approaches, the proposed model shows outperformed results in different objective metrics such as SDR, SI-SNRi.

In addition, the proposed model's computational complexity and resource requirements were analyzed, ensuring its scalability and practicality for real-world applications. A user-friendly interface was also developed to enable its use by non-experts.

The proposed SepFormer Transformer-based speech enhancement model demonstrated its effectiveness and robustness, showing promise for further improvements and advancements in the field. The results of this work can serve as a valuable contribution to the speech enhancement research community, with potential applications in areas such as teleconferencing, voice assistants, and hearing aids, among others. The proposed model can provide significant benefits in enhancing the quality of speech signals in various real-world applications and improving the user experience. Overall, this work can pave the way for future research in the field of speech enhancement, facilitating advancements in the development of more effective and efficient models for speech signal processing.

## 6.2 Future Work

Potential future aspects of this work are:

### 6.2.1 Efficient and Scalable Training Approach

While the current model achieves state-of-the-art performance, its training process can be resource-intensive, making it challenging to scale up to larger datasets or real-world scenarios. Therefore, developing a more efficient training process that can handle larger datasets and real-world scenarios can further improve the model's scalability and practicality.

### 6.2.2 Real-Time Speech Enhancement for Communication Systems

- Integration with telecommunication systems for real-time speech enhancement.
- Potential deployment in call centers, teleconferencing, and other communication systems.

### 6.2.3 Deployment on Mobile Device

- Optimizing parameters and hyper-parameters for making light weight applications, which can be installed on mobile devices.
- Improving cellular service quality in Urban Areas.

### 6.2.4 Speech Enhancement for Hearing Aids

- Integration with hearing aids to improve speech intelligibility in noisy environments.
- Potential deployment for elderly individuals and people with hearing impairments

### 6.2.5 Integration with Virtual Assistants and Smart Speakers

- Improving speech recognition accuracy and intelligibility of virtual assistants and smart speakers.
- Potential deployment in smart homes, cars, and other IoT devices.

### 6.2.6 Development of Customizable Speech Enhancement Solutions

- Development of speech enhancement models that can be tailored to specific noise and speech conditions.
- Potential deployment in industries such as entertainment, broadcasting, and media.

Overall, exploring these future aspects can lead to the deployment of speech enhancement models in various industries and sectors, leading to improved speech intelligibility and accuracy. With the potential for customization, scalability, and real-time processing, the proposed SepFormer Transformer-based speech enhancement model provides a strong foundation for future research and development in this field.

# REFERENCES

[1] S. Hochreiter and J. Schmidhuber, “Long short-term memory,”Neural Computation, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[2] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, “On ̈ the properties of neural machine translation: Encoder–decoder approaches,” in Proc. of SSST, 2014, pp. 103–111.

[3] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, “Light gated recurrent units for speech recognition,” IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 2, no. 2, pp. 92–102, April 2018.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” CoRR, vol. abs/1706.03762, 2017.

[5] G. Kerg, B. Kanuparthi, A. Goyal, K. Goyette, Y. Bengio, and G. Lajoie, “Untangling tradeoffs between recurrence and self-attention in neural networks,” CoRR, vol. abs/2006.09471, 2020.

[6] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang, S. Watanabe, T. Yoshimura, and W. Zhang, “A comparative study on transformer vs rnn in sp.

[7] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, “Neural speech synthesis with transformer network,” in Proc. of AAAI, 2019, pp. 6706–6713.

[8] J. Kim, M. El-Khamy, and J. Lee, “T-gsa: Transformer with gaussian-weighted self-attention for speech enhancement,” in Proc. of ICASSP, 2020, pp. 6649–6653.

[9] Q. Li, F. L. Kreyssig, C. Zhang, and P. C. Woodland, “Discriminative neural clustering for speaker diarisation,” CoRR, vol. abs/1910.09703, 2019.

[10] Xang, W. Zhang, Y. Qian, J. Le Roux, and S. Watanabe, “End-to-end multi-speaker speech recognition with transformer,” in Proc. of ICASSP, 2020, pp. 6134–6138.

[11] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, “Deep clustering: Discriminative embeddings for segmentation and separation,” in Proc. of ICASSP, 2016, pp. 31–35.

[12] Yu, M. Kolbæk, Z. Tan, and J. Jensen, “Permutation invariant training of deep models for speaker-independent multi-talker speech separation,” in Proc. of ICASSP, 2017, pp. 241– 245.

[13] M. Kolbæk, D. Yu, Z.-H. Tan, and J. Jensen, “Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks,” IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 25, no. 10, pp. 1901–1913, 2017.

[14] Shrikant Venkataramani, Jonah Casebeer, and Paris Smaragdis, “End-to-end source separation with adaptive front-ends,” in Proc. of ACSSC, 2018, pp. 684–688.

[15] Y. Luo and N. Mesgarani, “Conv-TasNet: Surpassing Ideal Time–Frequency Magnitude Masking for Speech Separation,” vol. 27, no. 8, pp. 1256–1266, Aug. 2019.

[16] P. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, “Deep learning for monoaural source separation,” in Proc. of ICASSP, 2014, pp. 1562–1566.

[17] Luo, Z. Chen, and T. Yoshioka, “Dual-path rnn: efficient long sequence modeling for time-domain single-channel speech separation,” in Proc. of ICASSP, 2020, pp. 46–50.

[18] Y. Liu and D. Wang, “Divide and conquer: A deep casa approach to talker-independent monaural speaker separation,” IEEE/ACM Transactions on audio, speech, and language processing, vol. 27, no. 12, 2019.

[19] Z. Shi, H. Lin, L. Liu, R. Liu, J. Han, and A. Shi, “Furcanext: End-to-end monaural speech separation with dynamic gated dilated temporal convolutional networks,” in MultiMedia Modeling, 2020, pp. 653–665.

[20] E. Nachmani, Y. Adi, and L. Wolf, “Voice separation with an unknown number of multiple speakers,” ICML, pp. 7164– 7175, 2020.

[21] E. Tzinis, Z. Wang, and P. Smaragdis, “Sudo rm -rf: Efficient networks for universal audio source separation,” in MLSP, 2020, pp. 1–6.

[22] J. Chen, Q. Mao, and D. Liu, “Dual-Path Transformer Network: Direct Context-Aware Modeling for End-to-End Monaural Speech Separation,” in Proc. of Interspeech 2020, 2020, pp. 2642–2646.

[23] N. Zeghidour and D. Grangier, “Wavesplit: End-to-end speech separation by speaker clustering,” arXiv preprint arXiv:2002.08933, 2020.

[24] L. J. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” CoRR, vol. abs/1607.06450, 2016.

[25] D.P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[26] J. Le Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, “Sdr– half-baked or well done?,” in Proc. of ICASSP. IEEE, 2019, pp. 626–630.

[27] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 2015.

[28] J. G. Beerends, M. Obermann, R. Ullmann, J. Pomy, and M. Keyhl, “Perceptual Objective Listening Quality Assessment (POLQA), The Third Generation ITU-T Standard for End-to-End Speech Quality Measurement Part I–Temporal Alignment,” J. Audio Eng. Soc., vol. 61, no. 6, p. 19, 2013.

[29] J. Jensen and C. H. Taal, “An Algorithm for Predicting the Intelligibility of Speech Masked by Modulated Noise Maskers,” IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 24, no. 11, pp. 2009–2022, Nov. 2016.