BRAUDE
College of Engineering, karmiel

Software Engineering Department

Capstone Project Phase A –61998

Speech Enhancement Using U-Net with Compressed Sensing

24-1-R-11

Submitters:
Avihay Hadad – 209286665
Elad Fisher – 318882800

Supervisors:
Prof. Zeev Volkovich
Dr. Renata Avros

Table of Contents

ABSTRACT

This paper presents an approach to speech enhancement task based on deep neural networks, specifically focusing on a time-domain U-Net architecture. While traditional methods achieved good denoising performance, they often ignored the contextual information and detailed features present in input speech signals. To address this, our model includes a time-domain U-Net which combines lightweight Shuffle Attention mechanism for focusing on features of speech and suppressing irrelevant audio information, and a compressed sensing loss (CS loss) by using the measurements of clean speech and enhanced speech, that can further remove noise from noisy speech [9].

Our proposed approach offers a promising solution for speech enhancement task, by achieving high speech quality and intelligibility scores with low number of parameters. Furthermore, it demonstrates good performance in generalization, effectively handling scenarios where the input speech contains new and unfamiliar noise types.

Our project files and source code will be available at: GitHub

1. INTRODUCTION

Speech enhancement is very important subject in the field of speech processing. Its goal is to improve the quality and intelligibility of speech which disturbed by external noises. There are many traditional algorithms and cutting-edge deep learning models that were designed to solve that task. However, many of them were too complex and they had high number of parameters or their denoising performance were mediocre.

Inspired by the Attention Wave-U-Net for speech enhancement [7]; new research and deep learning models of Compress Sensing, we propose a time-domain U-Net model combining lightweight Shuffle Attention mechanism and CS loss [9]. The U-Net model pays attention to the speech contextual information and prevent the detailed features from being lost during down-sampling or up-sampling. The lightweight Shuffle Attention is present in the final output of the encoder for focusing on detailed features of speech and suppressing irrelevant information after down-sampling. The CS loss function is defined using the measurement of speech, which can further remove noise in noisy speech.

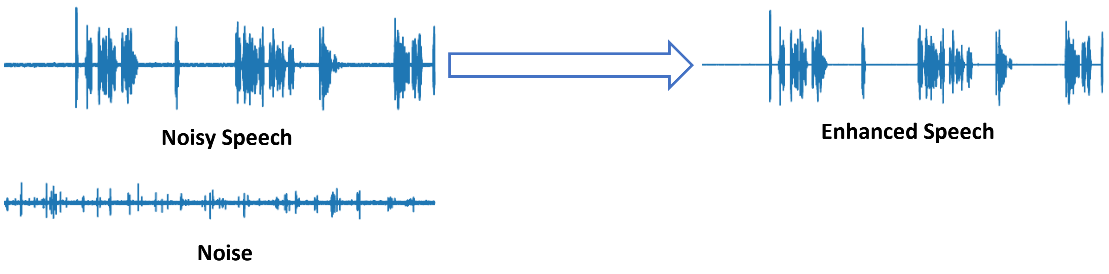


Figure 1. Speech enhancment example [13].

The proposed model can be utilized in various applications. It can enhance audio recordings in editing software, improve accuracy in speech recognition systems, and can even optimize voice calls in mobile communication devices. Furthermore, its potential to enhance speech intelligibility makes it valuable for individuals using hearing aids. Our goal in this paper is to research all the related articles, algorithms, models, and architectures utilized in the development and implementation of the proposed model.

2. BACKGROUND AND RELATED WORK

2.1 Denoising

In computer analyses, information (images, sounds etc.) can be made up by both useful data and noise, the last can reduce clarity for the information analyses [33]. The main goal of the denoising process is to reduce the noise data of the information and preserves more of the useful parts of the data. In audio aspects, denoising is an essential part of the processing, by removing noise and unwanted audio parts of the recordings. By 'filtering' the noise from a recording, we are improving the fidelity and audio quality of the content [16]. One of the classic techniques that are used for noise reduction and enhancement data in audio signals is Wiener Filtering. It works by applying a linear time-invariant filter to the noisy signal and by using the characteristics of the clean and noise speech in order to predict the clean signal in a more accurately way [16].

2.2 Sound Waves

A sound wave is a pattern of disturbances that are caused by the movement of sound, from its source through its entire travel: air, water, and all sorts of surface matter [28]. It is created by an object's vibrations that produce pressure waves – a short period of pressured fluctuation that makes a propagation of sound through the atmosphere [24]. The pressure wave that was created from the object disturbs the particles in the surrounding mediums, and they create a ripple effect of disturbance to the next particles, and so on, until they create a wave pattern. Sound waves have a few characteristics, some of them are amplitude, frequency, time, velocity and wave-length.

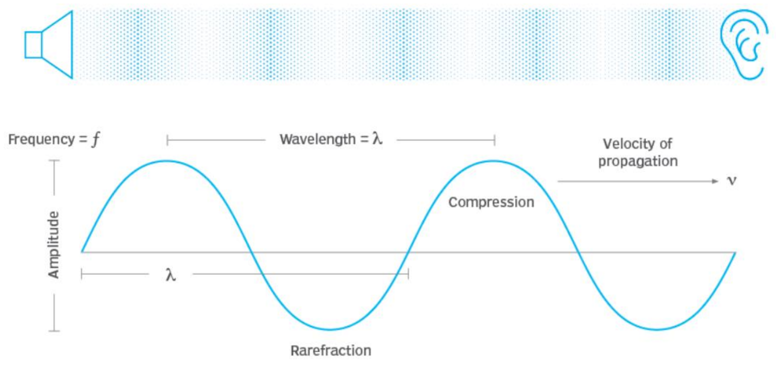


Figure 1. Characteristics of a sound wave [28].

2.3 Representations of audio

Time Domain

A time domain analysis is an analysis of signals; mathematical functions or environmental data, which are in reference to time and can show a graph of the signal changes in that reference [30].

In time domain, a signal will be defined as a function that changes over time (which is represented as a real number): $x(t), t \in \mathbb{R}$

With this definition, we can implement and represent the signal in discreate-time domain or continuous-time domain [27].

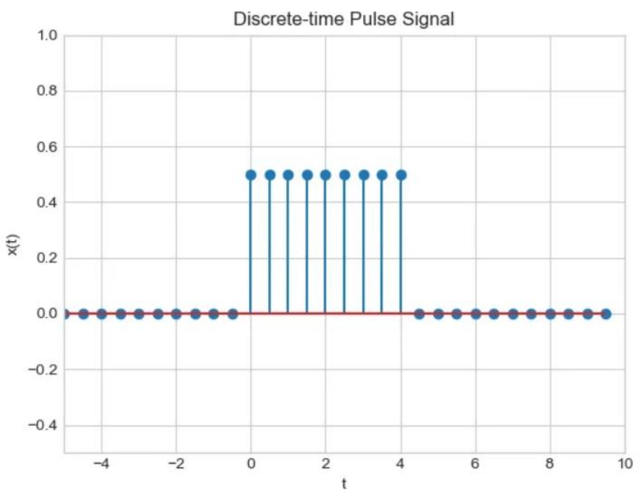


Figure 3. Discrete-time Pulse signal [27].

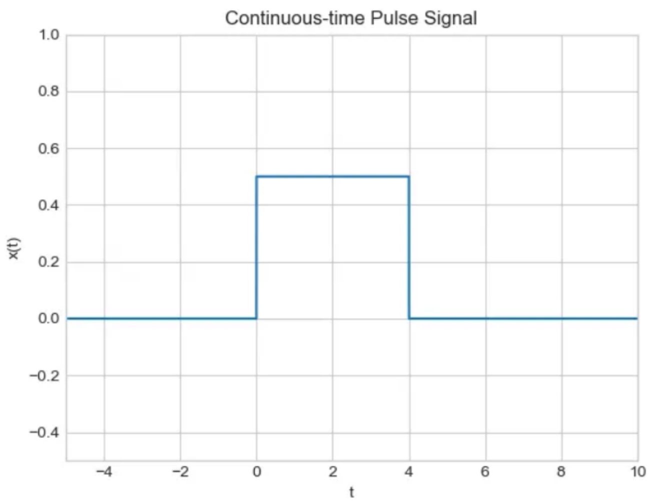


Figure 4. Continuous-time Pulse signal [27].

On the left, the signal will be classified as a discrete-time signal because it has a domain that is a small subset of lines from the real line, representing the signal at specific, isolated time points. On the right, the signal will be classified as a continuous-time signal because it contains at least one interval of the real line, representing the signal continuously over a time interval.

There are some cases where it is beneficial to turn our time-domain signals into time-frequency domain signals. In example, for tasks such as filtering and feature extractions [17].

Time-Frequency domain

The Time-Frequency domain analysis is an analysis of signals or mathematical functions in reference to both time and frequency. The representations of a signal in this domain are called a spectrogram:

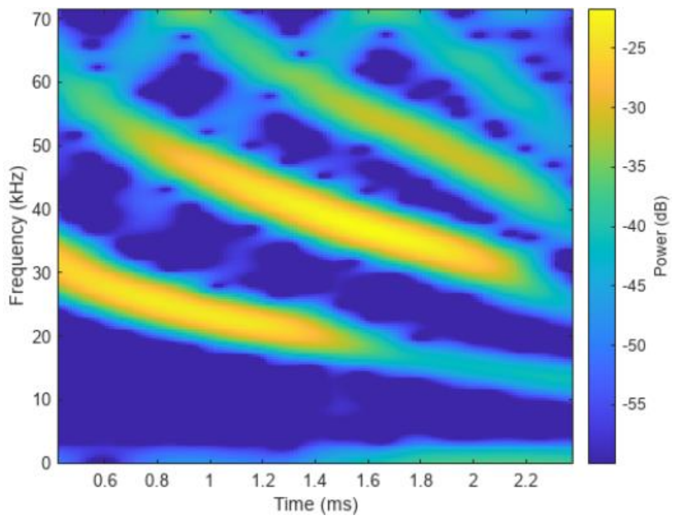


Figure 2. An example of a spectrogram [23].

The different colors representing the amplitude of each frequency at any given time. The advantages of a time-frequency domain representation over time domain lies in the ability to notice the sparsity and structure of different sounds in each timeslot.

Sparsity of different sounds:

We can infer by the yellow time frequency sections (that have a higher amplitude over the rest of the spectrogram), that it is only a small part of the entire spectrogram. with that in mind it is possible to see that many audio signals can have a sparse presence in time frequency domain, which can be helpful as the model will have fewer values/data that needs to be stored for further analysis. In time-frequency domain, the noise or any other disturbing signals, are typically not represented by the same coefficients as the source signal, like in the example, its visible to see that the spectrogram has four frequency ranges of sounds. For example, a compression between the signal's spectrograms:

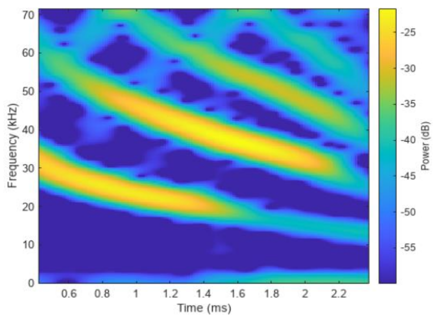


Figure 6. An example of a spectrogram [23].

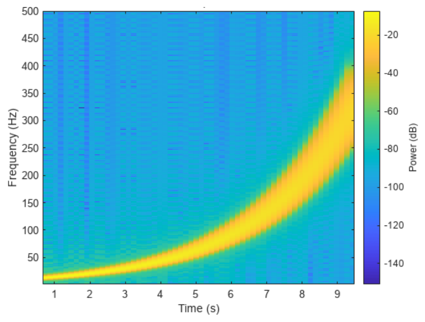


Figure 7. A spectrogram of a signal on a logarithmic scale [23].

The different sparsity in each spectrogram can lead to a conclusion that there's only a small probability that different sound sources are occupying different areas of the time-frequency in a spectrogram. With that, we can say that there is only one source with a significant amplitude in each time-frequency slot. This conclusion is called "W-disjoint orthogonality", which allows us to identify and separate each sound source from each other due to the fact they are mostly in separated areas of the spectrogram.

Different Structures of sounds:

Since naturally created sounds have a structure in both time and frequency, we can differentiate the source signal and noise signal with unique characteristics that make them. If we were to look at Figure 6 again, it is clear to see a structure for all four sound source's ranges. On the other hand, noise like sounds will lead to many coefficients that are distributed over the spectrogram without a clear structure at all.

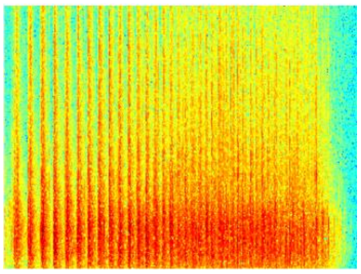


Figure 8. Sound spectrogram of applause [8].

With those characteristics, we can tell sources apart from each other, and can even improve the estimation of individual sources.

2.4 U-Net architecture

The U-Net architecture was proposed in 2015 [11] and was originally designed for segmentation of neuronal structures in electron microscopic stacks. The U-Net relies on the strong use of data augmentation to use the available annotated samples more efficiently, achieving precise segmentations even with minimal training data. The network consists of 2 symmetric paths, contraction and expansion, commonly known as the encoder and decoder, respectively, with a skip connection connecting them. Both paths contain equal number of steps, known as layers, where each layer perform the exact same operation as the other layers in its path. The contracting path performs the typical architecture of a regular convolutional network, downsampling the input to obtain higher level features. Each layer applies two 3x3 convolutions, each of them followed by a rectified linear unit (ReLU) and max pooling operation for downsampling. At each layer we double the depth of the feature maps. In other words:

$$\begin{tikzcd}
{} \arrow[r, "G_0(h,w,ch_{in})"] & \boxed{\text{Conv}(3 \text{ x } 3 \text{ x ch}_{out})} \arrow[r] & \text{ReLU} \arrow[r, "G_1(h,w,ch_{out})"] & \boxed{\text{Conv}(3 \text{ x } 3 \text{ x ch}_{out})} \arrow[r] & \text{ReLU} \arrow[r, "G_2(h,w,ch_{out})"] & \boxed{\text{Max Pooling}}
\end{tikzcd}$$

The Expansive path aims to undo the shrinking caused by the contracting path, focusing on regaining spatial detail. Each layer involves upsampling of the feature map followed by a 2x2 transpose convolution that reduces the number of feature channels by half. The upsampled feature map is concatenated with its corresponding feature map from the contracting path. Then, two 3x3 convolutions are applied, each followed by a ReLU. The concatenation is done due to a loss of details and border pixels during the downsampling.

At the final layer a 1x1 convolution is used to map the output feature vector to the desired number of classes for the segmentation task.

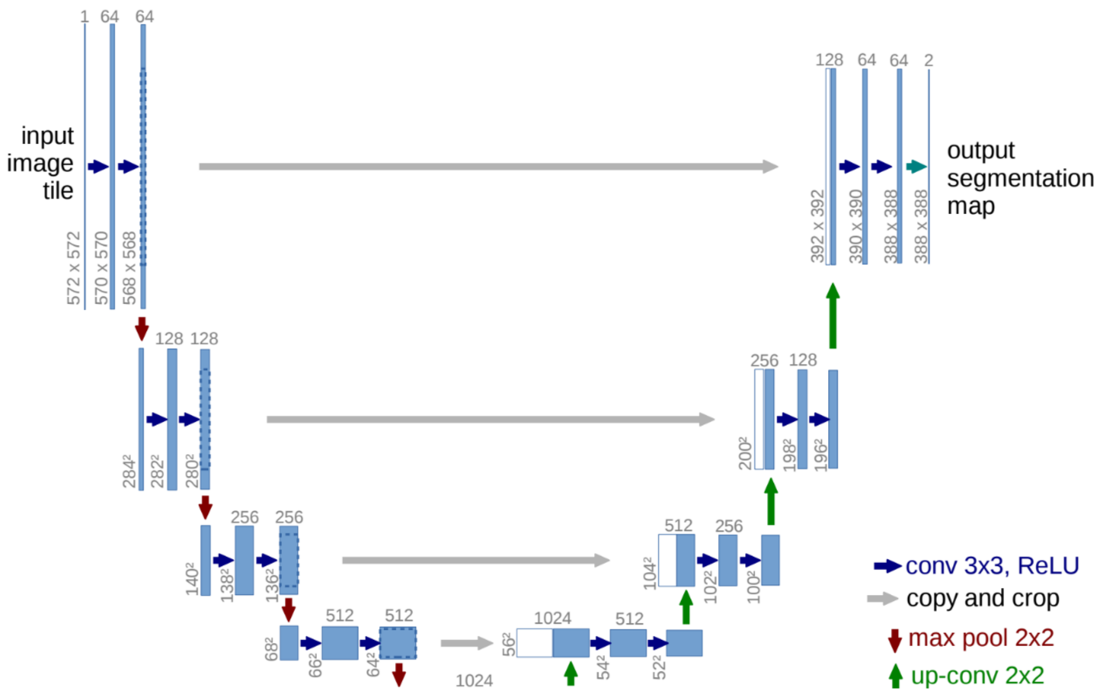


Figure 9. The original architecture of U-Net [11].

Variations of U-Net

Since the introduction of U-Net, many variations of the original architecture have been proposed. These variations mainly focus on modifying the operations within the encoding and decoding blocks; the total number of blocks and the bottleneck block itself, the block in the bottom center of the U-Net. The standard U-Net relies on standard convolutional blocks for feature extraction and upsampling in the encoding and decoding paths. The variations often use alternative operations such as dilated convolutions for capturing long-range dependencies or residual blocks for mitigating the vanishing gradient problem. while the original U-Net uses five blocks in each path, many variations increased or decreased this number depending on the complexity of the task and the available computational resources. The bottleneck block has been changed as well and many alternative methods have been explored for it. The original U-Net applies standard convolutional operations, while other methods used a Long Short-Term Memory (LSTM), attention mechanism, etc. which can further pay attention to the characteristic information output by the encoder.

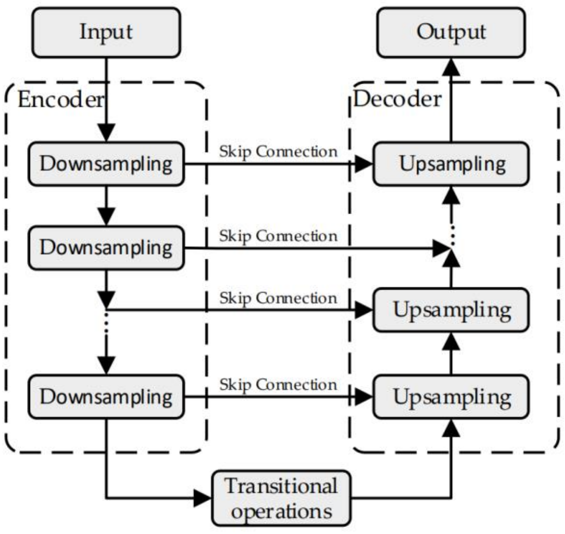


Figure 10. The general architecture of U-Net [9].

Variations of U-Net for Speech Enhancement

Speech enhancement is a very important subject in the field of speech processing. Its goal is to improve the quality and intelligibility of speech which disturbed by external noises. Speech enhancement based on U-Net architecture has been proven effective in various configurations, including the Time-Frequency-Domain and Time-Domain. Time-Frequency Domain U-Net, uses U-Net architecture with speech data represented in the time-frequency domain, often called spectrograms [5, 6]. While achieving promising noise reduction results, the presence of the original noisy phase spectrum in spectrograms can impact the denoising effectiveness. Time-Domain U-Net, also known as Wave-U-Net [12], was initially introduced in the context of audio source separation as an adaptation of the standard U-Net to the one-dimensional time domain. Beyond audio source separation, research on speech enhancement has begun to explore the potential of using the Wave-U-Net architecture for their task [10]. Furthermore, time-domain U-Net addresses the limitations of spectrogram-based methods by directly processing the raw audio signal. This enables the model to leverage both the magnitude (strength) and phase (timing) information of the sound wave, potentially leading to superior noise reduction capabilities.

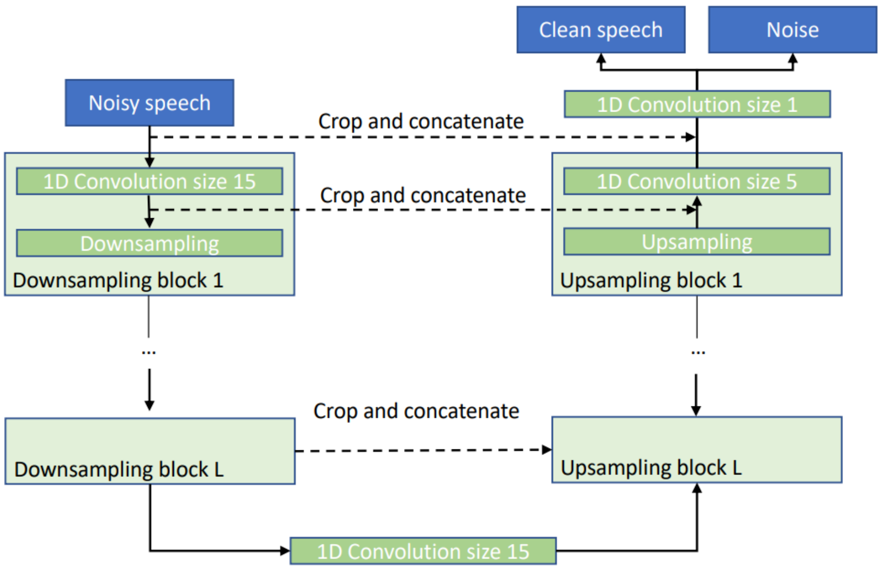


Figure 3. The Wave-U-Net architecture for speech enhancement [10].

2.5 Time-Domain U-Net

Time-Domain U-net differs from regular U-Net in the input and output representation of the model and in the convolution operations. In regular U-Net, the input typically consists of multi-channel images, such as RGB images. In contrast, in a Time-Domain U-Net, the input is usually a one-dimensional time series signal, such as audio waveform data. In the matter of the convolutions, the Time-Domain U-Net applies convolutions directly to the one-dimensional time domain data (1D-Conv).

2.6 Attention Mechanism

The first attention mechanism was first proposed by Dzmitry Bahdanau in 2015 and its purpose is to enhance deep learning models by selectively focusing on important input elements, depending on the contexts. With that, the attention mechanism improves the prediction's accuracy. The Attention mechanism enables the models to have an extra long-term memory and can make the model focus or have attention on all the previously tokens that were generated from the input. The model that was used before the attention was based on encoder-decoder of Recurrent Neural Networks (RNNs) or Lont-Short Term Memories (LSTMs). Even though RNN and LSTM are capable of looking at previous inputs as well, they have a shorter window of reference from any point. When the RNN/LSTMs were analyzing a much longer input, they couldn't access all the words that were generated at the start of the input, and then the translation task for example will turn out poorly [15]. That's why problem was called the "long-range dependency problem of RNN/LSTMs" [14]. In the model that Bahdanau proposed, he introduced the idea of "Attention", which means that when the model will generate a context vector, it will look for a set of positions in the encoder hidden states where the most relevant information is available.

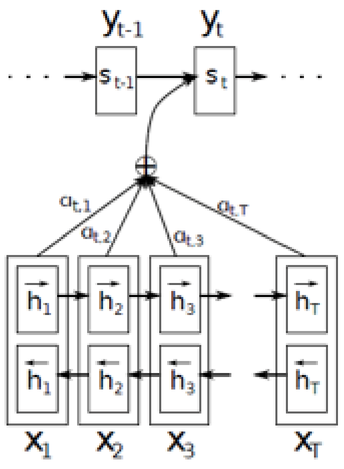


Figure 4. Diagram of the attention model from Bahdanau's paper [1].

As we can see in the diagram above, each block here is a bidirectional LSTM that is used to generate a sequence of annotations for each input sentence. All the vectors (h1, h2, h3 … hTx) are representations of Tx number of words in the original sentence (the input). In addition to that, Bahdanau put an extra emphasis on embeddings to all the words of the input – that are represented by the hidden states, in the process of creating the context vector. It was done by summarizing the weights of the hidden states. In short, for every query (a hidden state of the decoder), the attention will provide a "table" that shows how much attention that query have for each of the keys (encoder's hidden states), and with that can mark the level of focus that query will give for the rest of the input based on each decoded hidden state. With that in mind, we can use this model in order to improve and focus on relevant features in speech input rather than text input.

2.7 Shuffle Attention

As mentioned, the attention mechanism can focus on the important features of speech and lower the value of the unwanted data (noise) in the audio. One way to implement this is by using a shuffle attention mechanism, which will focus on the detailed features of the speech in that manner after the last down-sampling (the last output of the encoder). The mechanism works by dividing the speech feature map into groups along the channel dimension, and with that, the speech feature map of every group is being divided into two sub-feature maps (along the channel dimension as well). Each of the two sub-feature maps are now the input for one of the following: Channel Attention (C-Att) and Spatial Attention (S-Att) [9].

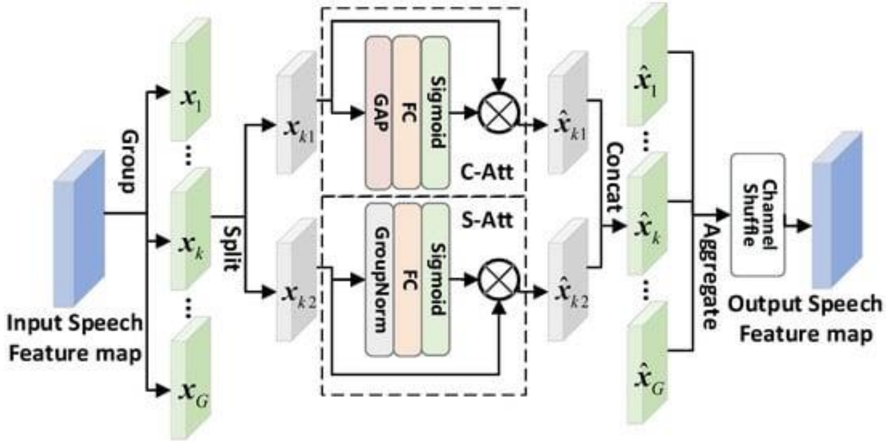


Figure 5. Shuffle attention mechanism with C-Att & S-Att [9].

2.7.1 Channel Attention (C-Att)

C-Att exploits the inner-channel relationships of the features, and with that the model builds a channel attention map [18]. C-Att focuses on the correlations between each of the channels of the speech feature map. From Figure 13, the C-Att uses Global Average Pooling (GAP), a pooling operation that generates one feature map for each corresponding audio characteristic. Instead of a fully connected layers, it averages each feature map.

2.7.2 Spatial Attention (S-Att)

S-Att uses the inter-spatial relationships of features to generate a spatial attention map [29]. Unlike C-Att, S-Att has more focus on the detailed features of the speech feature map.

By combining them together, S-Att and C-Att are complementing each other. In the Shuffle attention of Figure 13, we can see a specific speech feature x_k being split into x_{k1} and x_{k2} which are the speech sub-feature maps. x_{k1} and x_{k2} are then entered into C-Att and S-Att respectively, as can be seen from the following formulas:

$$\hat{x}_{k1} = \sigma(\omega_c f_{GAP}(x_{k1}) + b_c) \cdot x_{k1}$$

$$\hat{x}_{k2} = \sigma(\omega_s g(x_{k2}) + b_s) \cdot x_{k2}$$

$\hat{x}_{k1}$ and $\hat{x}_{k2}$ are the sub-feature maps output generated by C-Att and S-Att. ω_c, ω_s, b_c and b_s are the weights and biases of FC in C-Att and S-Att. $f_{GAP}()$ is the GAP operation, $g()$ is group norm, and $\sigma()$ is for the Sigmoid function. After calculation, the output sub-feature maps, they are concatenated to one another to create the output feature $\hat{x}_k$. After the process happens to all G speech feature group (from x_1 to x_G), the output features of attention from all the groups are aggregated. In addition to the aggregation, the final output is made with the channel shuffle operation – a similar operation to reshape operation. The operation allows information to fuse between different speech sub-features. After that process, the output feature map that was created has the same size as the original feature map input.

2.8 Convolution operation

The convolution operation is the base of all Convolutional Neural Networks (CNNs), and it combines two functions to generate a third function. By sliding a small filter, known as kernel, across the input image and computing the dot product between the filter weights and the corresponding pixel values in the input. This process is repeated across the entire image, generating feature maps that highlight important patterns and structures. The original input can be padded before the convolution begins with surrounding zeros. The convolution kernels are usually a small 2D matrix containing learnable weights. The size of the kernel determines the receptive field, which is the area of the input data considered at each position. The sliding size of the kernel is known as the stride.

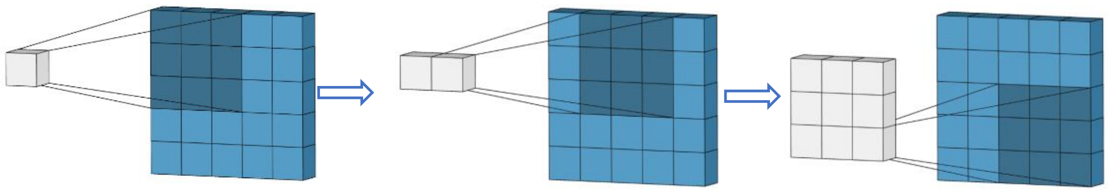


Figure 6. Convolution Operation with stride = 1 [20].

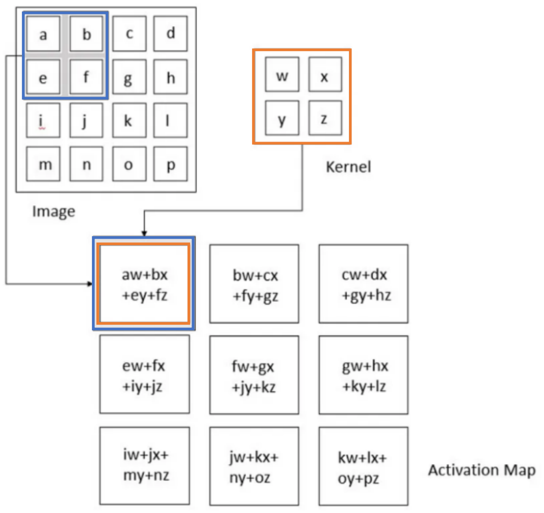


Figure 75. Convolution Operation [20].

However, the application of convolutional neural networks is not limited to two-dimensional data like images. For many tasks, such as time series analysis and audio processing, the data is often sequential. To effectively handle such data, one-dimensional convolution (1D-conv) is taking place instead of the regular convolution operation.

2.8.1 One Dimensional Convolution (1D-Conv)

1D convolution operates similarly to regular convolution but is specifically designed to process one-dimensional sequences. Instead of sliding a filter over a two-dimensional grid, 1D convolution involves moving a filter along a one-dimensional sequence, such as a time series signal or a sequence of words. The convolution formula is as follows:

$$y = x * w \rightarrow y[i] = \sum_{k=0}^{k=m-1} x[i-k] \cdot w[k]$$

Were x is the original input vector, filter w and m is the size of w. According to the formula, we can notice that the vector x is scrolled from right to left and w from left to right. In order to overcame the different directions, we can simply invert the filter vector w, and preform the vector product between x and a rotated w. Illustration of the process of 1D convolution:

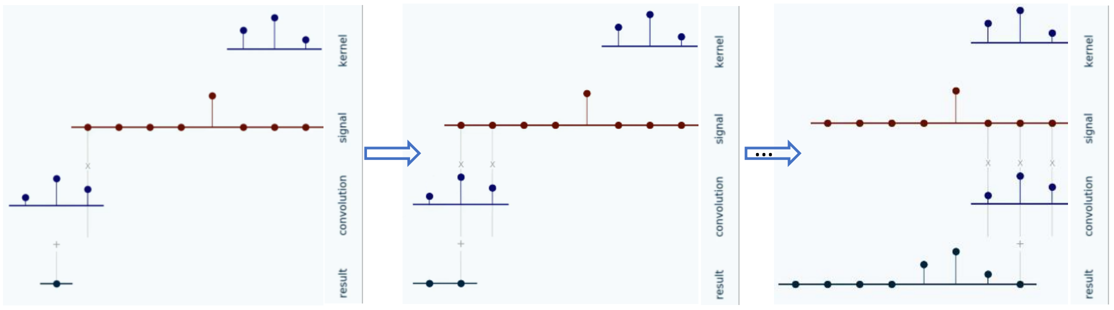


Figure 16. Flip kernel and sliding kernel along signal [19].

Initially, we begin by flipping the kernel. Next, we step the kernel along the signal according to the configured stride. At each position, we compute the dot product between the two by multiplying each pair of aligned values together and then summing up those products.

2.9 Dilated-Convolution

Dilated convolution is a variation of convolution where the kernel is expanded based on a dilation factor (l) before the convolution begins. Dilation involves increasing the size of the kernel by appending zeros to the right, bottom, and diagonals of each entry in the original kernel, except for the entries in the last row and column. The number of zeros added to the kernel is $l - 1$. For instance, a dilation size of 2 would result in one zero added along each direction.

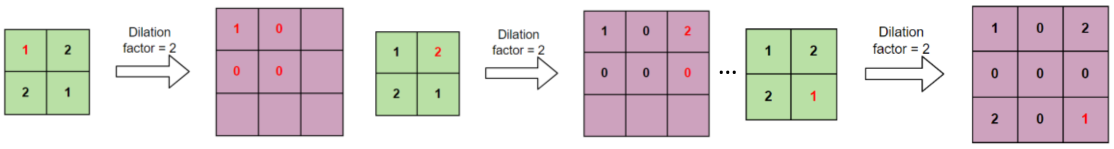
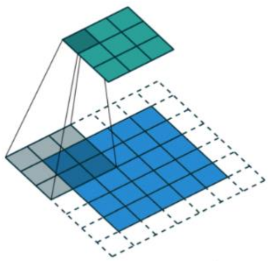


Figure 17. Dilation of a kernel with factor of 2 [34].

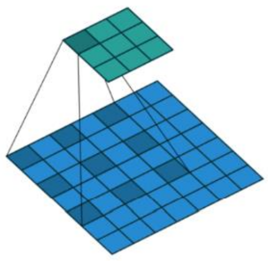
The dilated convolution can be explained as a convolution formula as follows:
Let F be the input, k is the filter, l the dilation factor.

$$\underbrace{(F * k)(p) = \sum_{s+t=p} F(s) \cdot k(t)}_{Standard\ Convolution} \qquad \underbrace{(F *_l k)(p) = \sum_{s+lt=p} F(s) \cdot k(t)}_{Dailated\ Convolution}$$

We can see that at the summation, it is $s + lt = p$ that we will skip some points during convolution. As well, when $l = 1$, it is a standard convolution.



Standard convolution



Dilated convolution (l=2)

Figure 18. Illustration of dilated convolution [25].

2.9.1 1D Dilated-Convolution

1D Dilation in convolution is like 2D operation we explained above, we increase the span of filters without increasing the number of weight parameters in them.

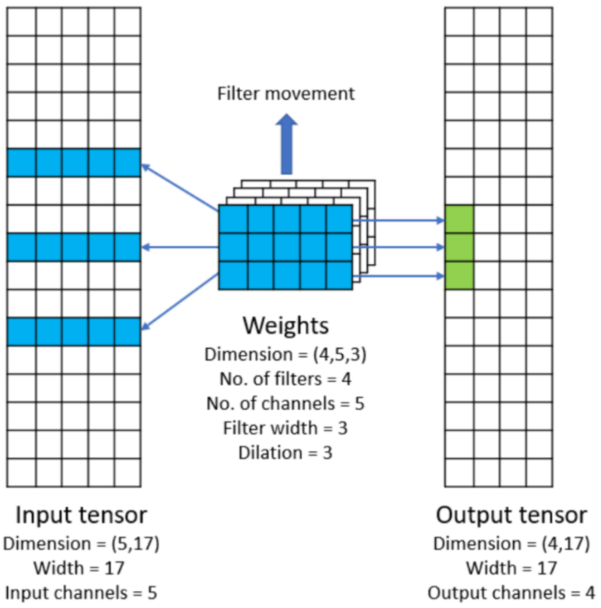


Figure 19. Illustration of 1D dilated convolution [2].

2.10 Transpose-Convolution

Transpose convolution (Trans-Conv) operation is an upsampling operation, generating an output feature map larger than the input feature map. The Trans-Conv operates similar to a regular convolutional operation but in reverse. Instead of sliding the kernel over the input, it moves the input over the kernel, conducting element-wise multiplication and summation. This process expands the output, and the size of the resulting feature map can be adjusted using the layer's stride and padding parameters. [35, 32]

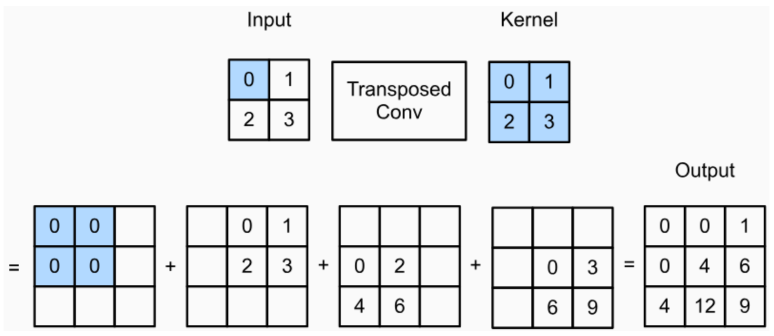


Figure 20. Illustration of Transpose-convolution with a 2x2 kernel [31].

2.11 Gated Linear Unit (GLU)

Gated Linear Unit (GLU) is an activation function for neural networks, aimed to address the issue of vanishing gradient and improving focus on relevant features. Unlike the standard linear activation ($f(x) = x * w + b$), GLU uses a gating mechanism that controls information flow by selectively passing or blocking information and operates as a logical gate. It can be explained by the formula from the paper [3]:

$$GLU(x) = \underbrace{(x * w_a + b)}_{A} \otimes \sigma(\underbrace{x * w_b + c}_{B})$$

Where x is the input vector, w_a, w_b, b, c are learned parameters (weights and biases, respectively), σ is the sigmoid function and $\otimes$ is the element-wise product between matrices. The idea lies in the multiplication between the information (A) and the gating signal $(\sigma(B))$, which is the output of the sigmoid function. The sigmoid function is critical because it outputs values between 0 and 1. As a result, when gating signal is closer to 1, more information from A will pass through. But, when gating signal is closer to 0, it restricts the flow of information. This enables GLUs to capture complex patterns and dependencies within the data [22]. Illustration of the GLU process:

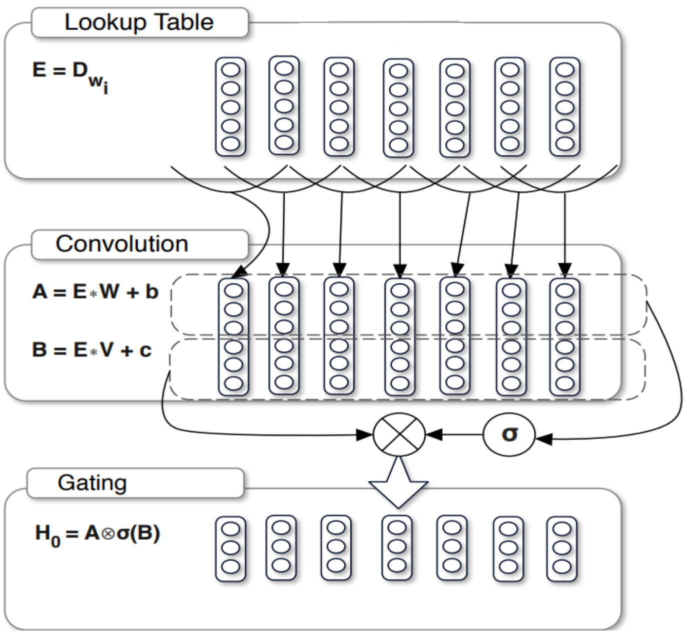


Figure 21. Illustration of operation of GLU [3].

2.12 Compressed Sensing

Image Compression

Regular image compression techniques take advantage of the fact that a lot of information in a picture is repeated. One common approach, as demonstrated below (Figure 22), uses the frequency domain by transforming the image using the Fast Fourier Transform (FFT). This transformation reveals the distribution of image information across different frequencies. In most pictures, the important details we see are usually stored in the lower-frequency components (long and smooth changes in color or brightness), traditional methods discard a portion of the high-frequency coefficients (rapid changes in color or brightness) while keeping the most significant ones, this reduces the file size of the image [21]. Illustration of the process:

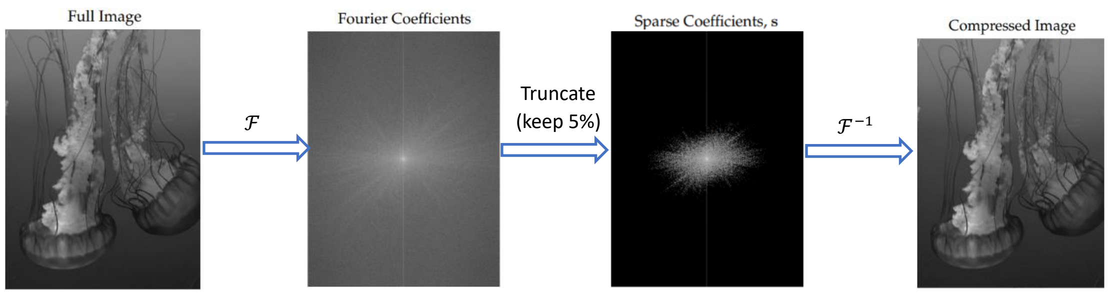


Figure 22. Illustration of compression with FFT $\mathcal{F}$ [21].

The compression follows the equation:

$$x = \Psi s$$

Where $x \in \mathbb{R}^N$ is the original image, $\Psi \in \mathbb{R}^{N\times N}$ is a Fourier transform basis and $s \in \mathbb{R}^N$ is a sparse vector.

Compressed Sensing

Compressed sensing (CS) is a method of compressed sampling [9]. CS requires the signal to be sparse, and the measurements of signal are obtained through sparse and dimensionality reduction of the original signal. Measurements can retain most of the information in the original signal with a very small amount of data. The following equation expresses the process of obtaining measurement signal $y \in \mathbb{R}^M$.

$$y = \Phi x, \ \ (x = \Psi s)$$

Where $x \in \mathbb{R}^N$ is the original signal and $\Phi \in \mathbb{R}^{M\times N}$ $(M \ll N)$ is the measurements matrix. The core problem of CS is to reconstruct x from measurement y, typically by applying constraints on x. A common constraint is that the original signal needs to be sparse. The original signal x can be recovered from measurements by solving an optimization problem for s [21]:

$$\hat{s} = \operatorname{argmin}\|s\|_1, \ \ subject\ to\ \|\Phi x - y\|_2 < \ \varepsilon$$

A schematic illustration of reconstructing x can be demonstrated as below:

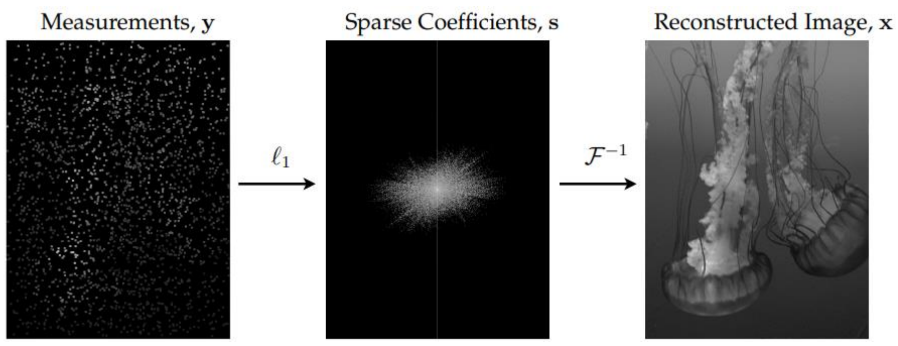


Figure 23. schematic illustration of reconstructing x by solving the optimization problem for s [21].

2.13 Compering Sounds

There are multiple ways to compere the amount of loss of the speech enhancement process. For the following methods we will use the same variables: s is the clean speech, n is the noise in the speech, and the combination of the two will represent the noisy speech signal $x = s + n$. The enhanced speech which is $\hat{x}$ and N is the total amount of samples in the speech signal. Most speech enhancement models that are based on U-Net use either Mean Square Error (MSE) or Mean Absolute Error (MAE) to compare between the enhanced and clean speech signals.

2.13.1 Mean Square Error (MSE)

In speech enhancement, MSE is a common metric used to evaluate the performance of the speech enhancement process. MSE measures the average of the squares of the differences between the enhanced speech and original speech signals.

$$L_{MSE}(s,\hat{x}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{x}_i - s_i)^2$$

Even though it is a common method to evaluate the speech enhancement, it has a significant fault: it is easily affected by unusual, extreme points in the speech signal. since MSE squares the difference between signals, it is possible that large errors can lead to a big impact to the result.

2.13.2 Mean Absolute Error (MAE)

Another common metric that is used to assess the performance of speech enhancement, Mean Absolute Error (MAE) measures the average of the absolute differences between the enhanced and original speech signal.

$$L_{MAE}(s,\hat{x}) = \frac{1}{N}\|\hat{x} - s\|_1 = \frac{1}{N}\sum_{i=1}^{N}|\hat{x}_i - s_i|$$

Since other researches had already proved that MAE is significantly improve the denoising performance [4], and since MAE doesn’t square the differences, it can handle unusual points in a signal better than MSE. Because of that, MAE is considered better to use for this sort of evaluation rather than MSE.

2.13.3 Compressed Sensing (CS)

Using compressed sensing for calculating the amount of loss by using the measurements as the optimizable target. Measurements can retain the main information and even some unobvious features of original speech. Therefore, the enhancement performance of model can be improved by optimizing the MAE between measurements of clean speech and measurements of enhanced speech [9].

$$L_{CS}(s,\hat{x}) = \frac{1}{N}\|\Phi\hat{x} - \Phi s\|_1$$

2.13.4 Short-Time Fourier Transform (STFT)

Short-time Fourier transform (STFT) is a tool that is used to analyze speech signals and helps to understand how the frequency of a signal changes over time. The STFT works by initially splitting a signal into small, overlapping windows, focusing on a small portion of the signal at any given time during analysis. The way to calculate the STFT of a signal with frequency ω and time t is as the following:

$$X_{STFT}[t,\omega] = \sum_{k=0}^{L-1} x[k]g[k-t]e^{-\frac{j2\pi\omega k}{L}}$$

where $x[k]$ represents the k-th window of the signal, and $g[k]$ represents the analysis of the k-th window function that is centered at time t. j represents the imaginary unit ($\sqrt{-1}$). With this method, we transformed each window of the original signal from Time domain into Frequency domain, and by combining all of them in order (of time), the result will be a representation in Time-Frequency domain [26].

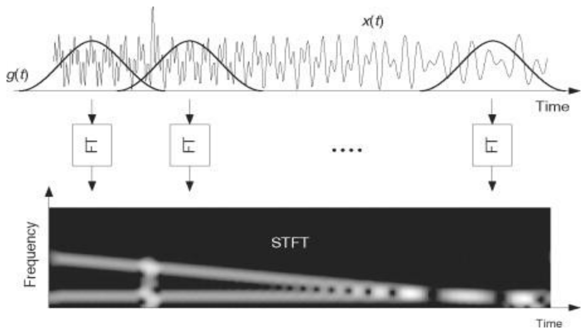


Figure 24. Illustration of computing STFT by taking Fourier transforms of a windowed signal [26].

After there is a way to represent the signals in time-frequency domain, we can optimize the clean and enhanced speech, to learn the time-frequency characteristics of them. With that, the loss function of a STFT can be defined as the following:

$$L_{STFT}(s,\hat{x}) = E_{\substack{s\sim p(s)\\ \hat{x}\sim p(\hat{x})}}\left[\frac{\| |STFT(s)| - |STFT(\hat{x})| \|_F}{\| |STFT(s)| \|_F} + \frac{1}{N}\| log|STFT(s)| - log|STFT(\hat{x})| \|_1\right]$$

Where E represents the expected value, by averaging the loss function of the probability distributions $p(s)$ and $p(\hat{x})$ of the clean speech s and enhanced speech $\hat{x}$. $|STFT()|$ stands for the absolute value of any given speech. $\|val\|_F$ represents the Frobenius norm – a matrix norm that is similar to the Euclidean norm, and $\|val\|_1$ is for the L_1 norm, that measures the absolute differences between the components inside it. The normalization ($\frac{1}{N}$) is used to scale the L_1 norm in the function so that the loss value is averaged over the length of the signal, N.

3. EXPECTED ACHIEVEMENTS

The outcome that we are expecting from this project is centered around the sound enhancement and denoising algorithms. Our goal is to provide a model that can filter out noises and still preserves the quality and clarity of the speech from the audio file. This includes handling a variety of background noises that can potentially lower the audio quality. The model can achieve this by inputting the audio file in the U-Net model, that uses dilated residual blocks in the encoder and decoder for down/up-sampling the feature map, respectively. The model uses as well dilated convolutions to capture context, while the batch normalization stabilizes the process. The GLU (Gated Linear Unit) and the 1D convolution work together to enhance the speech feature map by allowing for a precise noise filtering. The output at the end will be an enhanced, denoised, clearer speech based on the initial input.

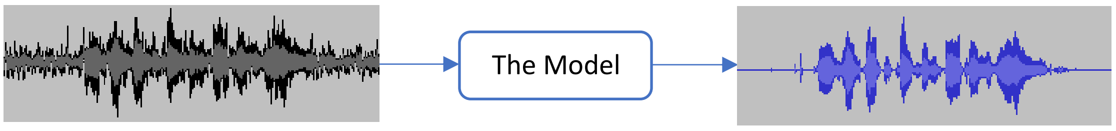


Figure 25. In left: Input noisy speech, that goes through the model. In right: The output of the model, an enhanced speech that has less noise than input.

The criteria with which we will measure the success of the model are the following:

I. Detecting the differences between speech and noise in an audio file.
II. Denoising or filtering the noise from the audio.
III. Enhancing the speech features of the audio.

4. RESEARCH / ENGINEERING PROCESS

4.1 Research Process

The main goal of the project is to optimize and implement a deep neural network model for speech enhancement task [9] that is composed by some advantages in other known speech enhancement models [10, 7].

In the first part the project (part A), the research process started by exploring and learning about the area of speech enhancement and denoising. Later, we read the paper of our base model [9] and preformed research on all related articles, algorithms, models, and architectures employed on our model. As our research progressed and we studied the higher picture of the process, we began to split the different major parts that are taking place in our based model. Our base model, Speech Enhancement with Compressed Sensing U-Net (SECS-U-Net), as any other U-Net model consists of encoder, decoder and skip connections, plus a lightweight Shuffle Attention mechanism and CS loss. The encoder and decoder are using 5 layers each, and time-domain dilated residual blocks are constructed for downsampling and up-sampling, they are used to focus on speech contextual information and detailed features during convolution.

Later in our research we read about the experimental result of the paper and studied the influence of different losses and model components on the results. Finally, a comparative analysis was performed to assess our base model against other competitor models in the speech enhancement task, and SECS-U-Net not only achieved good enhancement effect, but also a good generalization ability.

In part B of our project, we will focus on developing the presented model. The objective is to assess the model's accuracy and overall performance and implement it as part of a software program.

Constraints

The SECS-U-Net model introduces a time-domain U-Net architecture that achieves high speech quality scores with fewer parameters. Unlike the traditional U-Net designed for processing images with three channels and performing regular convolution operations, our model operates differently in terms of the learning process and input-output formats. Additionally, the model operates on speech utterances sampled at 16 kHz, which are then divided into speech chunks of 1 second each (16,000 samples in length) for input. The goal is to face those challenges while also to determine the effectiveness of CS for speech enhancement task.

4.2 Product

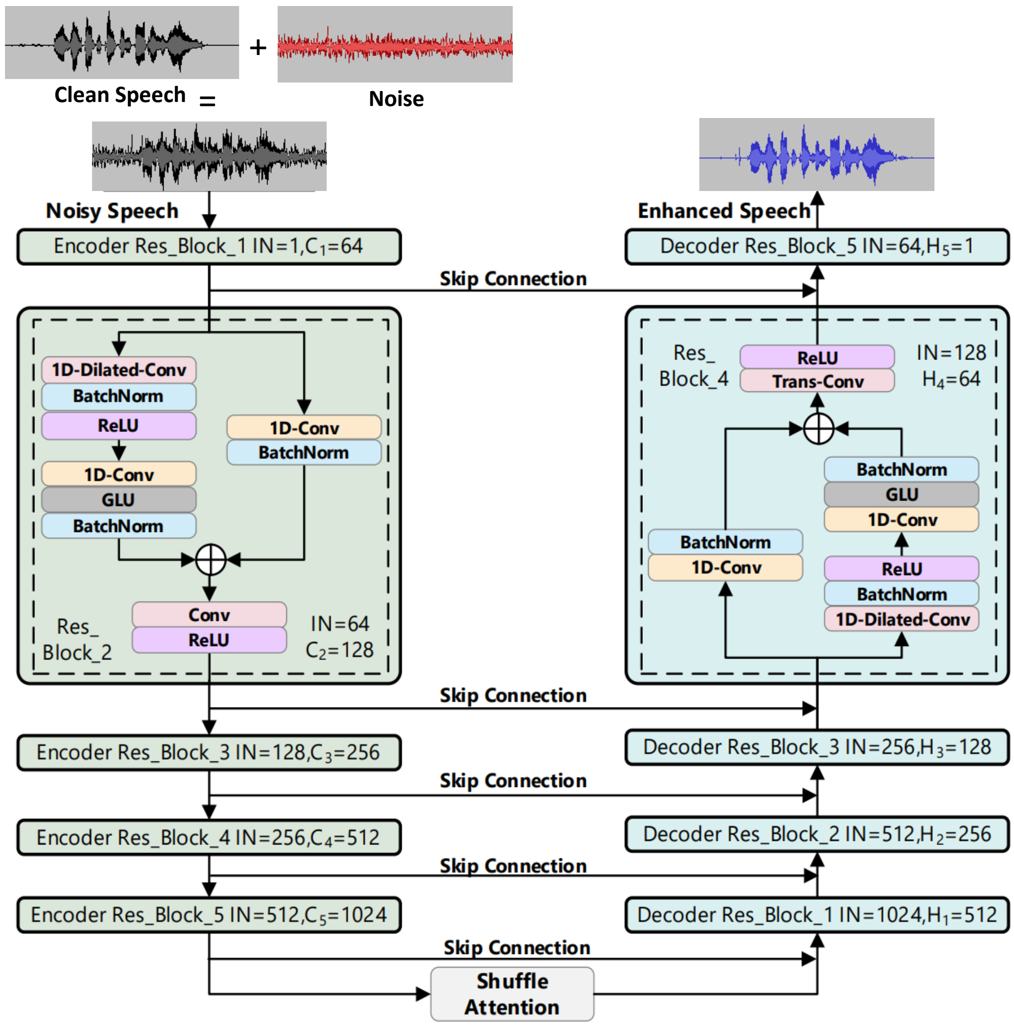


Figure 26. Workflow of the U-Net model. The input is a noisy speech that will pass through SECS-U-Net and output an enhanced speech. The model has 5 encoding layers, 5 decoding layers, and 1 Shuffle Attention. IN represents the channel of input speech feature map in each layer, Ci and Hi, respectively, represent the channel of speech feature map after down-sampling and up-sampling at the i-th layer [9].

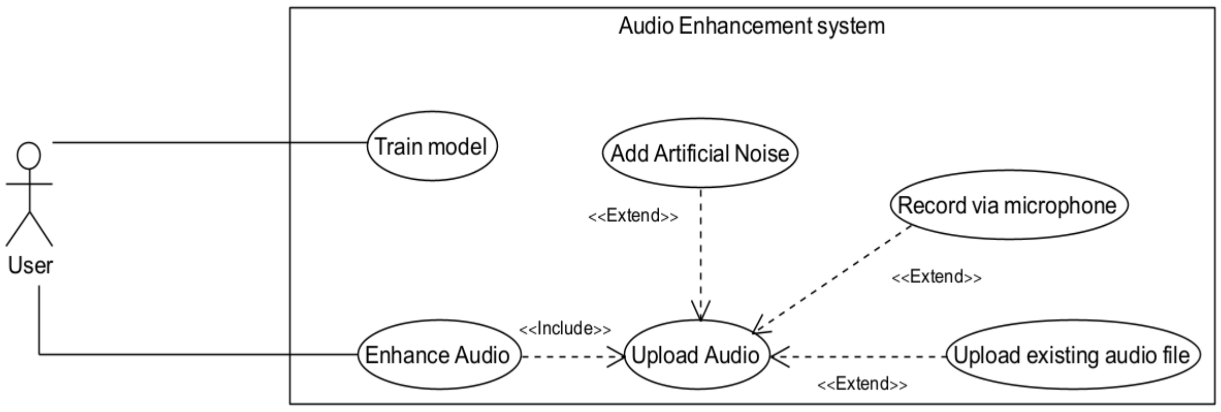
SECS-U-Net

The model is taking as an input a noisy speech signal $x = s + n$, where s is the clean speech, n is the noise, and outputs as a result enhanced speech $\hat{x}$. The model pays attention to the speech contextual information and prevents the detailed features from being lost during down-sampling or up-sampling, by implementing a time-domain dilated residual blocks as shown in Res Block in Figure 26 in the encoder and decoder of the U-Net. When down-sampling or up-sampling, the input speech feature map is fed to the next components according to the structure shown in the Res Blocks above. Only one block of the down-sampling and up-sampling operations is shown separately in Figure 26, except for the number of channels (i.e., Ci and Hi), the other layers use identical sampling operations.

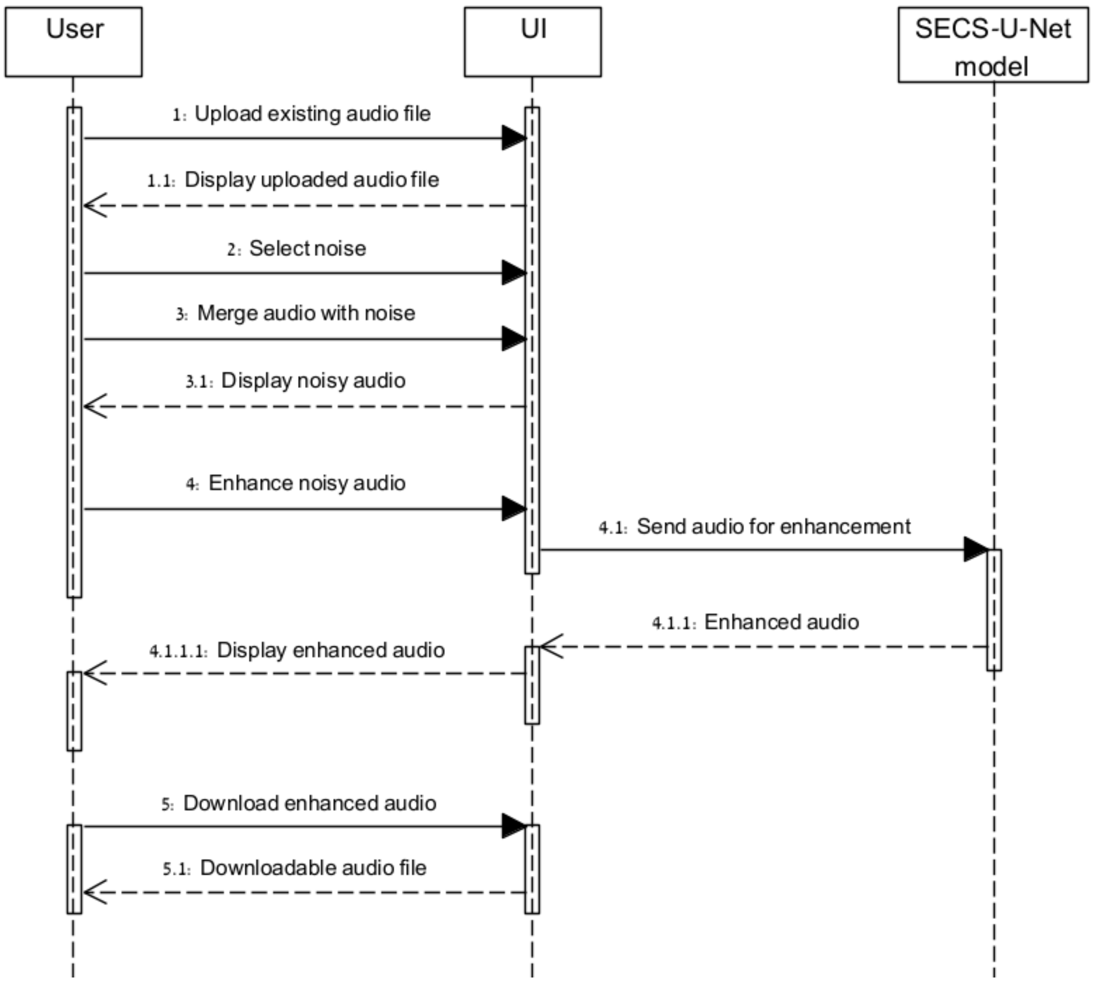
The Attention mechanism, as explained in the background section, can focus on the features of speech, and suppress irrelevant information. We are using a lightweight Shuffle Attention in the model in the final output of the encoder. It is used to focus on detailed features of speech and suppress irrelevant information after down-sampling.

4.2.1 UML

Use-Case diagram:



Sequence diagram:


User
UI
SECS-U-Net model
1: Upload existing audio file
1.1: Display uploaded audio file
2: Select noise
3: Merge audio with noise
3.1: Display noisy audio
4: Enhance noisy audio
4.1: Send audio for enhancement
4.1.1: Enhanced audio
4.1.1.1: Display enhanced audio
5: Download enhanced audio
5.1: Downloadable audio file

4.2.2 Requirements

Table 1: Functional Requirements

ID	Requirement Description
1	The system will load the latest training data of the model.
2	The system will provide a home page to navigate from.
3	The system will be able to return to the home page.
4	The system will be able to train the model.
5	The system will save the training weights locally.
6	The system will be able to receive audio files from user.
7	The system can marge artificial noises to a given audio file.
8	The system will enhance the noisy audio by using the trained model.
9	The system will provide a mechanism to save the enhanced audio.
10	The system can play any of the audio files (original, noisy and enhanced).
11	The system can clear current enhancement.

Table 2: Non-Functional Requirements

ID	Requirement Description
1.1	The system will look for training data from local memory.
1.2	The system will block enhancement in case there no training data.
2.1	The user can navigate to "Train the model" and "Enhance audio"
2.2	The system will alert the user to train the model if it can enhance audio.
3.1	Each page will have a close button.
3.2	Every page except the home page, will have a home button.
4.1	The system will present the date of the last training.
4.2	The system will show in percentages the progress of the trading.
5.1	The system automatically will save the latest trained model in local storage.
6.1	The system can receive audio by recording or uploading files.
6.2	The system will use the computers recorder to record and save audio.
6.3	The system will check the uploaded file is of type audio (MP3, MWV, etc.).
7.1	The system will provide the user with a selection of artificial noises to choose.
8.1	The system will send the noisy audio to the trained model to run the enhancement algorithm.
9.1	The "Enhance Audio" page will have a download button.
9.2	The system will use a standard file saving mechanism to store the file locally.
10.1	The system will use standard media player.
11.1	The "Enhance Audio" page will have a clear button.
11.2	The clear button will reset the page in order to start the process again.

Requirements Gathering Process

After we researched on multiple audio enhancement systems that are available online, and by interviewing people who are working with audio editing applications, we came to the understanding that people use to listen to the audio file through the enhancement process, in order to hear the difference between the audio before and after the enhancement. We also learned from the interviews that an option to record audio with microphones is needed.

4.2.3 GUI

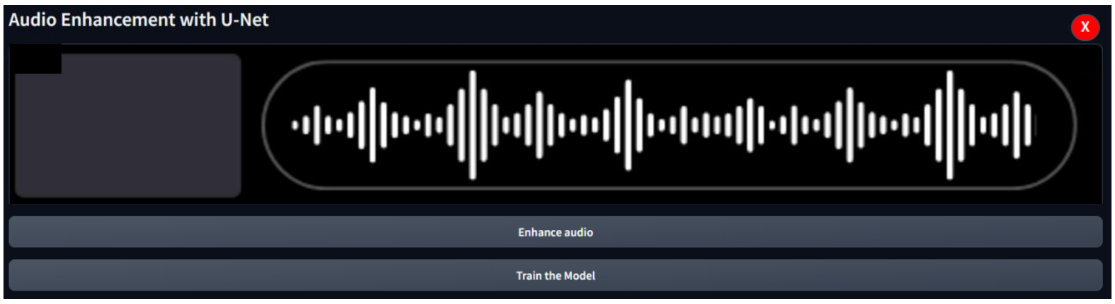
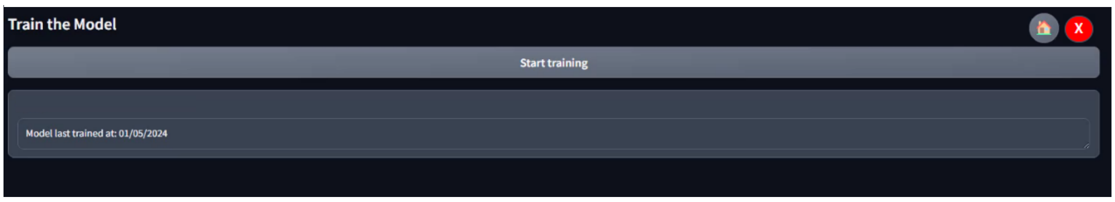


Figure 27. Home Page for the app. can train the model or enhance audio.



Model last trained at: 01/05/2024

Figure 28. Train Model Page. Can see the last time the model was trained.

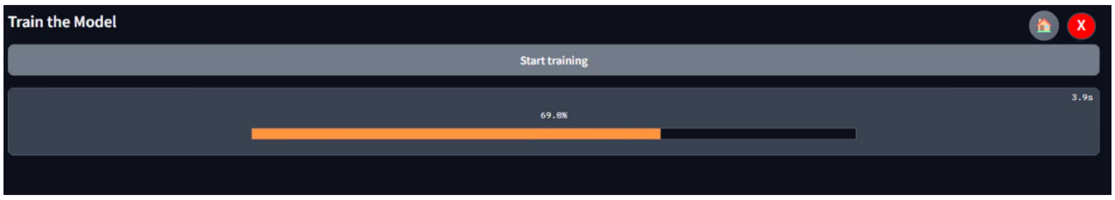
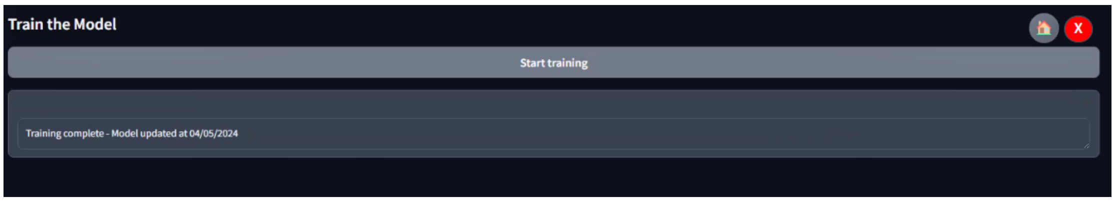


Figure 29. Training the Model.



Training complete - Model updated at 04/05/2024

Figure 30. Finished Traning the model. updated the date of training.

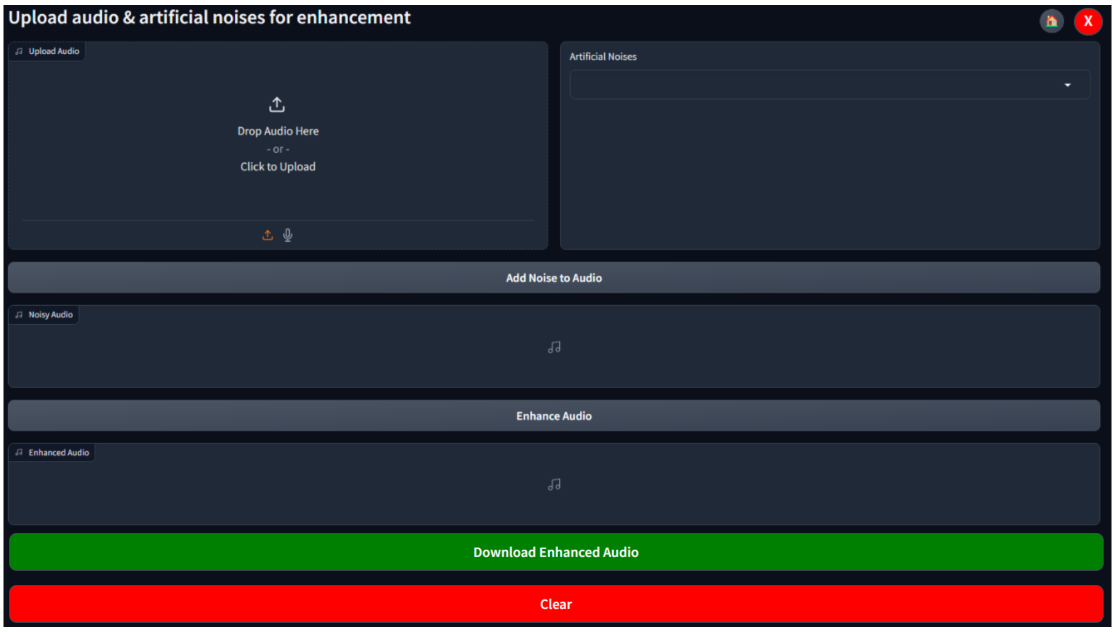


Figure 31. Enhance Audio Window, with the option insert Audio to enhance.

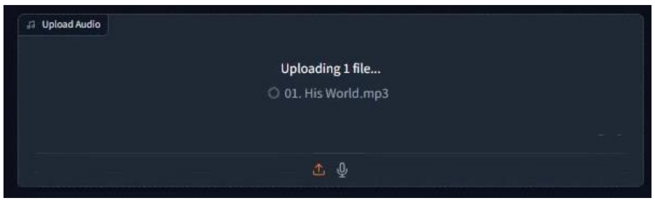


Figure 32. Uploading an audio file.

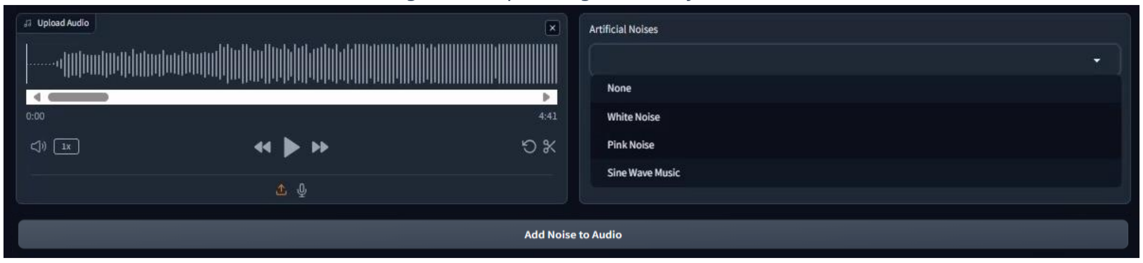


Figure 33. Choosing Artificial Noise to add to the audio.

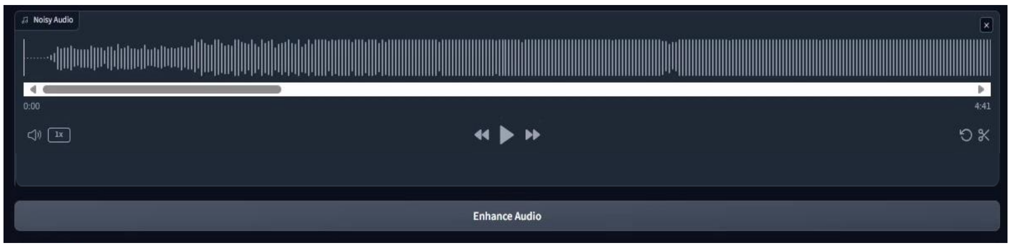


Figure 34. The new Noisy audio that will go through the enhancment.

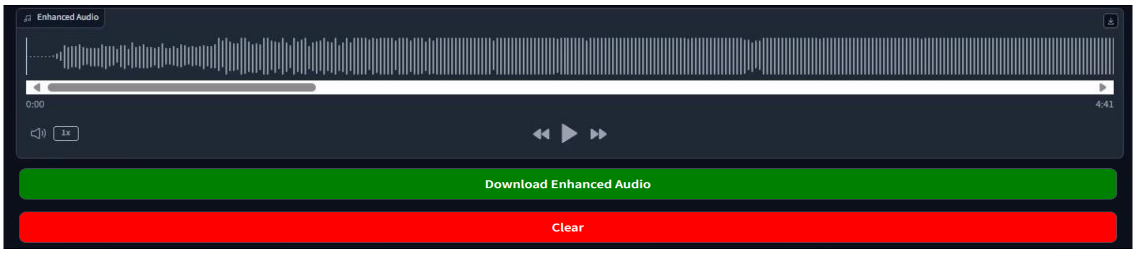


Figure 35. The enhancement Audio that can be downloaded.

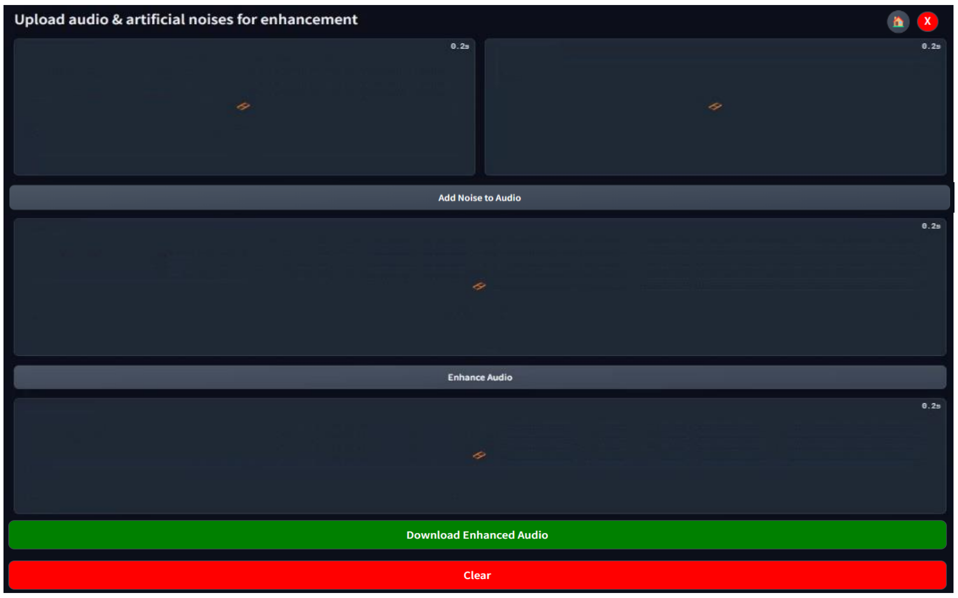


Figure 36. Clear button clears the entire app with small animation.

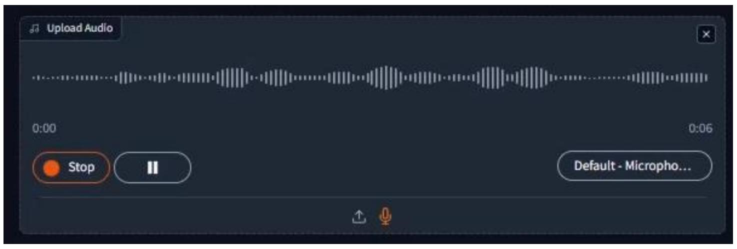


Figure 37. Recording audio insted of uploading a file.

4.2.4 Testing Plan

ID	Module	Test Description	Expected Result
1	Any Page	Click "X" button.	Closes the program.
2	Home Page	Click "Train the Model" button.	Opens "Train the Model" page.
3	Home Page	Click "Enhance Audio" button, when the model isn't trained.	An Error Pop-up appears with a notification to train the model.
4	Home Page	Click "Enhance Audio" button, when the model has a trained data.	Opens "Enhance Audio" page.
5	Train / Enhance Page	Click "Home" button.	Opens "Home" page.
6	Train the Model Page -> before train.	Checking date of last training.	Should be "NONE" or a previous date.
7	Train the Model Page -> before train.	Click "Start Training" button.	Initiating model training.
8	Train the Model Page -> after train.	Checking date of last training.	Matches the current date.
9	Enhance Audio Page -> upload audio box.	Clicking the "Click to Upload" button.	Opens window to select file.
10	Enhance Audio Page -> upload audio box.	Dropping a file that isn't audio type	An Error Pop-up appears, telling the user to drop an audio file.
11	Enhance Audio Page -> upload audio box.	Dropping an audio file.	Uploading the audio file.

12	Enhance Audio Page -> upload audio box -> select file.	Selecting a file that isn't audio type.	An Error Pop-up appears, telling the user to select an audio file.
13	Enhance Audio Page -> upload audio box -> select file.	Selecting an audio file.	Closing the file select window, uploading the audio file.
14	Enhance Audio Page -> Any box with audio uploaded.	Clicking "Play" button.	Plays the audio file.
15	Enhance Audio Page -> Any box with audio uploaded.	Clicking "Trim" button.	Enables the user to trim parts of the audio file.
16	Enhance Audio Page -> Any box with audio uploaded.	Clicking "forward" or "backward" button.	Moves the audio listener forward or backwards.
17	Enhance Audio Page -> Artificial Noises box.	Click Drop-down and selecting "None".	Option "None" should be chosen.
18	Enhance Audio Page -> Artificial Noises box.	Click Drop-down and selecting "Whie Noise".	Option "White Noise" should be chosen.
19	Enhance Audio Page -> Artificial Noises box.	Click Drop-down and selecting "Pink Noise".	Option "Pink Noise" should be chosen.
20	Enhance Audio Page -> Artificial Noises box.	Click Drop-down and selecting "Sine Wave Music".	Option "Sine Wave Music" should be chosen.
21	Enhance Audio Page.	Clicking "Add Noise to Audio" button, when no audio and noise selected.	An Error Pop-up appears telling the user to select audio and noise.
22	Enhance Audio Page.	Clicking "Add Noise to Audio" button, when no audio selected.	An Error Pop-up appears telling the user to select audio.
23	Enhance Audio Page.	Clicking "Add Noise to Audio" button, when no noise selected.	An Error Pop-up appears telling the user to select noise.
24	Enhance Audio Page.	Clicking "Add Noise to Audio" button when both noise and audio are selected.	Starts merging the audio with the artificial noise.
25	Enhance Audio Page.	Clicking "Enhance Audio" when no audio has uploaded in the "Noisy Audio" box.	An Error Pop-up appears telling the user to first add noise to audio.
26	Enhance Audio Page.	Clicking " Enhance Audio " button when there is a noisy audio in the box.	Sends the noisy audio as input to the model to start the enhancement process.
27	Enhance Audio Page.	Clicking "Download Enhanced Audio" when there isn’t any audio in the "Enhanced Audio" box.	An Error Pop-up appears telling the user to first enhance a noisy audio.
28	Enhance Audio Page.	Clicking "Download Enhanced Audio" when there is an audio in the "Enhanced Audio" box.	Opens a selection window to select download location for the file.
29	Enhance Audio Page -> Download enhanced audio window.	Selecting a path and a file name to save the enhanced audio.	Converting the audio to an audio file and saving it locally.
30	Enhance Audio Page.	Clicking "Clear" button.	Clears the entire window and all audio boxes.
31	Computer	Starting the program when we have a saved trained model saved locally.	The Home page should open and load the previous trained model.
32	Computer	Starting the program when we don’t have a saved model locally.	The Home page should open without loading anything.

5. EVALUATION

The evaluation plan for a speech enhancement model contains multiple steps to guarantee a comprehensive evaluation of its performance. The first step involves selecting the appropriate dataset for training and testing, which includes multiple speakers for clean speech and various types of noise at different levels of signal-to-noise ratio (SNR). Secondly, we should perform data preprocessing of the speech signals, which involves dividing each speech utterance (sampled at 16kHz) into speech chunks of 1 second each and to use overlap of 0.5 seconds between them. In addition, we apply a random shift between 0 to 1 second for input speech. The next step involves training the model using the preprocessed data and using a validation set to monitor the model's performance during the training process. After completing the training phase, the model is tested on a test set with pairs of clean and noisy utterances. Some of these noises are unfamiliar to the model, serving to test its generalization ability. During testing we utilize metrics such as PESQ (Perceptual Evaluation of Speech Quality), MOS (Mean Opinion Score), SSNR (Segmented SNR) to measure the quality of the speech enhancement.

6. AI TOOLS

We have used Gemini (gemini.google.com) and ChatGPT (chatgpt.com) primarily to enhance the grammar and flow of our writing.

Prompt for example: “*Refine the grammar and flow of the following text: <Text here>*”

7. REFERENCEES

[1] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

[2] Chaudhary, N., Misra, S., Kalamkar, D., Heinecke, A., Georganas, E., Ziv, B., ... & Kaul, B. (2021). Efficient and generic 1d dilated convolution layer for deep learning. arXiv preprint arXiv:2104.08002.

[3] Dauphin, Y. N., Fan, A., Auli, M., & Grangier, D. (2017, July). Language modeling with gated convolutional networks. In International conference on machine learning (pp. 933-941). PMLR.

[4] Defossez, A., Synnaeve, G., & Adi, Y. (2020). Real time speech enhancement in the waveform domain. arXiv preprint arXiv:2006.12847.

[5] Deng, F., Jiang, T., Wang, X., Zhang, C., & Li, Y. (2020, October). NAAGN: Noise-Aware Attention-Gated Network for Speech Enhancement. In Interspeech (pp. 2457-2461).

[6] Geng, C., & Wang, L. (2020, June). End-to-end speech enhancement based on discrete cosine transform. In 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA) (pp. 379-383). IEEE.

[7] Giri, R., Isik, U., & Krishnaswamy, A. (2019, October). Attention wave-u-net for speech enhancement. In 2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA) (pp. 249-253). IEEE.

[8] Gu, J., Lu, L., Cai, R., Zhang, H. J., & Yang, J. (2005). Dominant feature vectors based audio similarity measure. In Advances in Multimedia Information Processing-PCM 2004: 5th Pacific Rim Conference on Multimedia, Tokyo, Japan, November 30-December 3, 2004. Proceedings, Part II 5 (pp. 890-897). Springer Berlin Heidelberg.

[9] Kang, Z., Huang, Z., & Lu, C. (2022). Speech enhancement using u-net with compressed sensing. Applied Sciences, 12(9), 4161.

[10] Macartney, C., & Weyde, T. (2018). Improved speech enhancement with the wave-u-net. arXiv preprint arXiv:1811.11307.

[11] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18 (pp. 234-241). Springer International Publishing.

[12] Stoller, D., Ewert, S., & Dixon, S. (2018). Wave-u-net: A multi-scale neural network for end-to-end audio source separation. arXiv preprint arXiv:1806.03185.

[13] Audio Denoiser: A Speech Enhancement Deep Learning Model (analyticsvidhya.com)

[14] Attention Mechanism In Deep Learning (analyticsvidhya.com)

[15] Attention Networks: A simple way to understand Self Attention (medium.com)

[16] Audio Enhancement and Denoising Methods (medium.com)

[17] Audio journey part 5 : Audio, time or frequency domain (medium.com)

[18] Channel Attention Module (paperswithcode.com)

[19] Convolution in one dimension for neural networks (e2eml.school)

[20] Convolutional Neural Networks, Explained (towardsdatascience.com)

[21] Data Driven Science & Engineering Machine Learning, Dynamical Systems, and Control - University of Washington (databookuw.com)

[22] Gated Linear Unit — Enabling stacked convolutions to out-perform RNNs (medium.com)

[23] Practical Introduction to Time-Frequency Analysis (mathworks.com)

[24] Pressure Wave (ametsoc.org)

[25] Review: DilatedNet — Dilated Convolution (towardsdatascience.com)

[26] Short-Time Fourier Transform (sciencedirect.com)

[27] Signal Representation — Time Domain versus Frequency Domain (medium.com)

[28] Sound Wave (techtarget.com)

[29] Spatial Attention Module (paperswithcode.com)

[30] Time Domain Analysis vs Frequency Domain Analysis (cadence.com)

[31] Transposed Convolution (d2l.ai)

[32] Understand Transposed Convolutions (towardsdatascience.com)

[33] What Is Denoising? (blogs.nvidia.com)

[34] what-is-dilated-convolution (educative.io)

[35] What is Transposed Convolutional Layer? (geeksforgeeks.org)