Self-Supervised Speech Enhancement Model Based on Denoising Diffusion Probabilistic Model

Submitted by:
Xiao Tianqi

Supervised by:
Dorien Herremans
Berrak Sisman

Computer Science and Design (CSD)

The thesis presented for the degree of
Master of Engineering (MEng)

SINGAPORE UNIVERSITY OF
TECHNOLOGY AND DESIGN

Abstract

The objective of speech enhancement is to remove the background noise from the corrupted audio signals. Traditionally, it is a supervised task, using the noisy audio as input and the clean audio as output. However, supervised learning creates some problems. Supervised learning requires matching pairs of clean and noisy data which is hard to obtain in real life, therefore, most datasets are created by adding noise to clean audio. However, there are almost infinite types of noise in real-life situations, the dataset can only cover a limited number. Even with data augmentation, the problem can be relieved a bit, but not fully solved. When facing unseen noise in the training phase, the performance of the model will shrivel.

In order to tackle these problems, this thesis proposes a novel method of based on the Denoising Diffusion Probabilistic Model (DDPM) [1] to fulfill speech enhancement tasks through self-supervised learning. Compared to traditional speech enhancement models, instead of parallel data of matching pairs of clean and noisy audio, the proposed model only requires clean data for training and is able to maintain its performance with seen and unseen noise. The proposed model utilized an overlooked property that the output of DDPM can be guided by a latent input. Therefore, instead of using a Gaussian noise as the input and setting the timestep to the maximum timestep and randomly generating an output that we cannot control, we use the noisy audio as the guidance and set the timestep to a relatively small number compared to the maximum timestep. The model also has an extra benefit that it has faster inferencing speed as unlike traditional DDPM-based models, it only uses part of the total timestep instead of all of it. Compared to CDiffuSE [2], the number of floating-point operations (FLOPs) of our model is significantly lower with a huge margin. Our model requires only 0.757G FLOPs for inferencing, while CDiffuSE requires 252G FLOPs[1].

The model yields promising results. Based on the subjective and objective evaluation results, by comparing to the award-winning model, DCCRN [3], the model we proposed is capable of performing similarly to DCCRN.

[1]The data is obtained using thop python library

Contents

Chapter 1

Introduction

Speech enhancement aims to improve the quality and intelligibility of degraded audio signals so that listeners can have a better experience. The speech enhancement technique, also known as noise canceling, is vastly used in categories such as hearing aids, telecommunication, and speech recognition. It is a challenging task as the output usually has a very large dimension (if the sample rate is 22,050 and the audio is 5 seconds long, the dimension will be 11,0250, which is equivalent to an image with dimensions 332 by 332), and humans are very perceptive to audio, meaning one tiny mistake will make the quality of the audio very low.

Traditionally, speech enhancement is done by calculating a mapping between matching clean and noisy speeches in the time-frequency domain. With the recent development of neural networks, traditional methods are replaced by deep neural network (DNN) models, as they can learn the non-linear mapping between clean and noisy speeches faster and better by introducing non-linear activation functions to the model [4]. Soon after, as generative neural networks such as Variational Autoencoder (VAE) [5] and Generative Adversarial Networks (GANs) [6] are invented, these techniques are applied to speech enhancement as well, creating promising results [7] [8]. The pipeline of VAE and GANs in the speech enhancement category is very similar to its counterparts in image denoising, but instead of raw audio, we have to convert it into spectrograms, which can be seen as an image representation of the raw audio.

In the past few years, Denoising Diffusion Probabilistic Model (DDPM) [1] has become the star of all generative models, it has shown its excellent capability of generating images as well as audio files [9] that outperform GANs [10]. DDPM contains two processes, forward and backward. The forward process adds Gaussian noise to clean audio until it is just Gaussian noise, while the backward process tries to regain the clean audio step by step using a neural network that predicts the noise added. In terms of random generation, it has yet to show any satisfying outcomes in the audio category. Compared to VAE and GANs, DDPM only involves training one single neural network instead of two, therefore, the model is much easier to train as we do not have to worry about the mode or posterior collapsing. Even though random generation is not a success, models involving conditional diffusion probabilistic models [11] have been created and yielded excellent performance, which inspired us to conduct research on modifying the DDPM in a different way to see whether it can be used in speech enhancement.

It is worth mentioning that all of the methods mentioned above are supervised learning, requiring parallel clean and noisy data. As it is an almost impossible task to collect parallel data directly, currently, the datasets are created by adding noise to clean speech, which makes the type of noise limited and stationary. When facing unseen or non-stationary noise in testing or real-life applications, the performance will decrease [12]. As such, we propose a self-supervised diffusion model that solely requires clean data to train to solve this problem.

The proposed model uses UNet[13] as the noise prediction model which has various advantages. First of all, its ability to denoising images has already been proven [14]. Furthermore, it does not require a lot of data to train. Last but not least, it is a fully convolutional network. There will be fewer parameters as the kernels are shared by the feature maps. As such, we think UNet might be the perfect model for this task.

In a nutshell, this thesis proposes an unsupervised method to deal with speech enhancement. The model has three main contributions. Firstly, it accelerates the speed of inferencing. Long inferencing time is a problem for DDPM-based models as the total timestep is usually a large number (about 200 to 1000). It means that the same prediction model has to run for hundreds, if not, thousands of times to complete the inferencing, which is very time-consuming. For the proposed model, instead of using all the steps in the DDPM model, the proposed model only utilizes a fraction of it, therefore, the inferencing speed will be faster. Secondly, it does not require datasets consisting of matching pairs of clean and noisy data, so it is easier to obtain the data as there are plenty of datasets prepared for other audio tasks, such as voice conversion, available. Lastly, as noise is not a part of the training dataset, it can work equally well on all kinds of noise, whether they are seen or unseen. The same cannot be said for other supervised models.

The rest of this thesis is organized as follows. In Chapter 2, we talk about some previous models that are used on speech enhancement tasks. In Chapter 3, we show the details of our proposed model, including the modification of DDPM and its implementation. In Chapter 4, we talk about the training process of the model. In Chapter 5, we show the performance of the model, including subjective and objective evaluation. Chapter 6 is the conclusion of this thesis.

Chapter 2

Related Work

Speech enhancement tasks are very similar to image-denoising tasks. Instead of using raw audio, we convert the audio into spectrograms, which can be seen as an image representation of the audio. As such, removing the background noise from the audio can be done the same way as removing the noise from an image. To tackle this problem, there are many models invented based on AE, GANs, and UNet. A detailed explanation of these three models can be found in this chapter.

2.1 Speech Enhancement Based on Autoencoder (AE)

Autoencoders [15] are made up of an encoder, and a decoder. The encoder's job is dimension reduction, which is transforming the input data into an encoding with lower dimensionality. The decoder's job is to generate an output similar to the training set fed to the encoder based on the given encoding. The architecture of autoencoders is displayed in Figure 2.1.

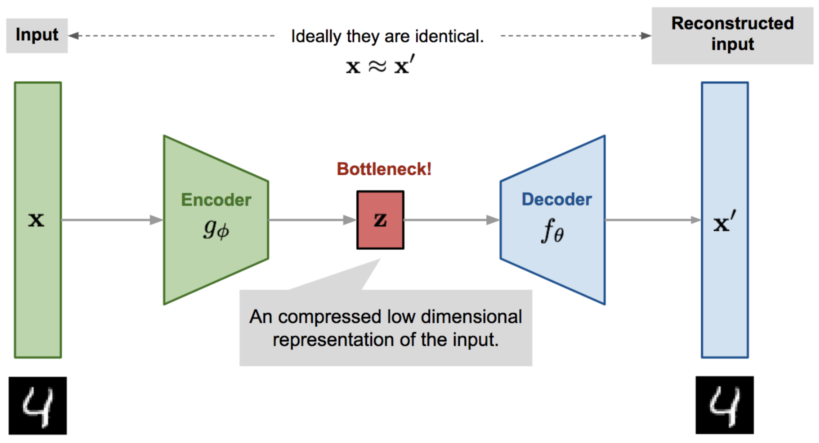

Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
$\mathbf{x}$
Encoder
g_ϕ
Bottleneck!
$\mathbf{z}$
Decoder
f_θ
$\mathbf{x}'$
An compressed low dimensional representation of the input.

Figure 2.1: The structure of AE adopted from [16].

During the training phase, the training data, x is first converted to the latent expression, z, by the encoder, g_ϕ. Then, the decoder, f_θ, tries to restore the encoding back to the original input. The loss function is usually the L1 loss or the L2 between the original input, x, and the generated output, x'.

There are many variants of autoencoders, such as the famous variational autoencoders (VAE), one among them that can be used on speech enhancement tasks is denoising autoencoders (DAE) [17]. DAE is originally proposed to increase the robustness of the performance of vanilla autoencoders. Instead of clean data, the data feed to DAE is intentionally corrupted either by adding some noise or applying a mask that will hide some values from the input stochastically. Like the dropout layer, DAE uses a similar approach to prevent overfitting. As shown in Figure 2.2, there are no major differences in terms of model structure between DAE and AE, except that the input changes from clean data to noisy data, and the loss is calculated between the data before noise is added to and the output created by the decoder.

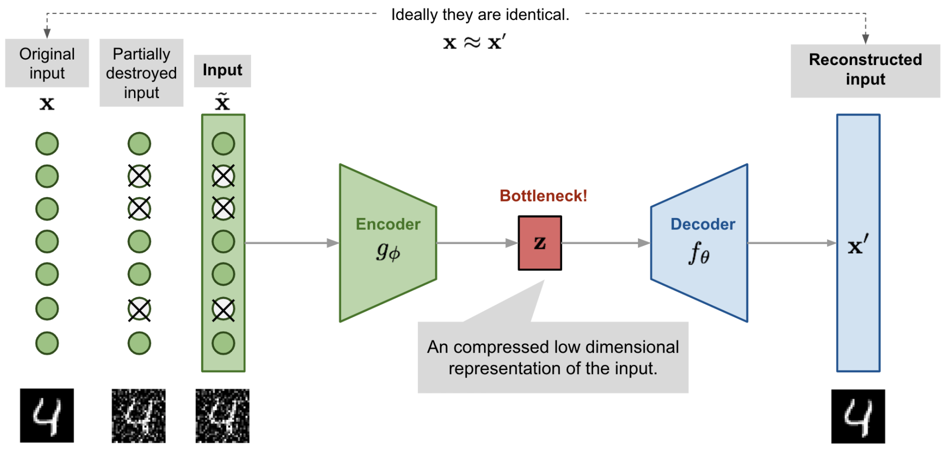

Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Original input
$\mathbf{x}$
Partially destroyed input
Input
$\tilde{\mathbf{x}}$
Reconstructed input
Encoder
g_ϕ
Bottleneck!
$\mathbf{z}$
Decoder
f_θ
$\mathbf{x}'$
An compressed low dimensional representation of the input.

Figure 2.2: The structure of DAE adopted from [16].

Modifying DAE to suit the speech enhancement task is fairly simple. Instead of adding randomly generated noise or masking, the corrupted input is now created by adding different types of noise that we hear every day in real-life situations.

There are also some improvements in DAE in terms of structures, namely deep DAE [4] that stacks multiple DAEs with different parameters together or recurrent DAE that uses the same DAE multiple times. Deep DAE achieved a relatively good score of 3.52 on speech enhancement tasks by testing with the Perceptual Evaluation of Speech Quality (PESQ) metric using a continuous Japanese dataset that contains 50 utterances. PESQ is a subjective evaluation metric that simulates the Mean of Opinion (MOS) test done by human listeners, the score ranges from -0.5 to 4.5, 4.5 being the best. The model outperforms the traditional method Minimum Mean-Square Error with improved minima controlled recursive averaging [18], showing its capability of completing the task.

2.2 Speech Enhancement Based on Generative Adversarial Networks (GANs)

GANs [6], as the name suggests, is a model that consists of two smaller models that compete with each other. There is a generator that creates the outputs we expect, and a discriminator, whose task is to determine whether the generated output created by the generator are authentic ones. The architecture is displayed in Figure 2.3.

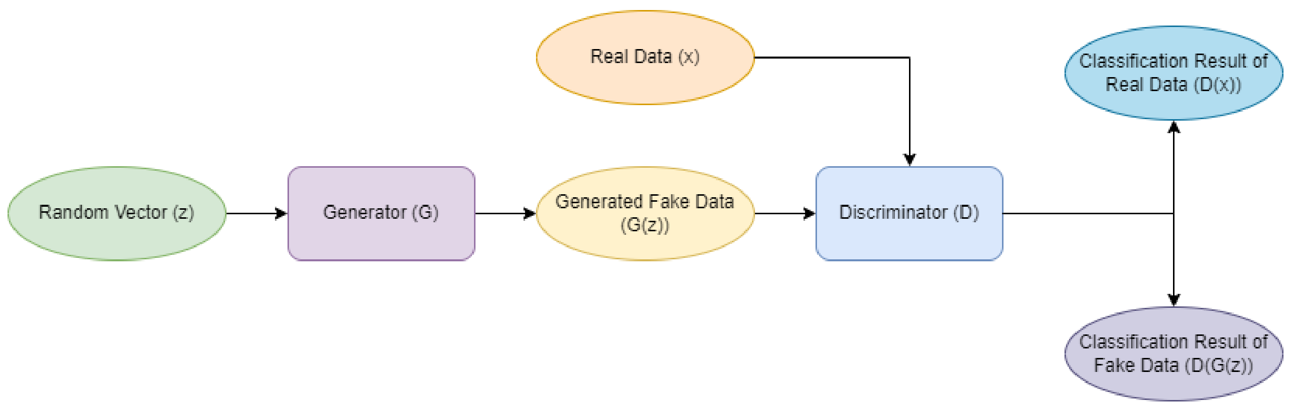

Real Data (x)
Classification Result of Real Data (D(x))
Random Vector (z)
Generator (G)
Generated Fake Data (G(z))
Discriminator (D)
Classification Result of Fake Data (D(G(z))

Figure 2.3: The architecture of GANs.

During the training phase, the two neural networks are trained alternatively while maintaining the parameters of the other one fixed until the loss of the discriminator converges. Theoretically speaking, the accuracy of the discriminator will be 50%.

The discriminator is a classification network, whose main job is to classify whether the input is real or fake (generated by the generator). The loss function is typically similar to other standard classification model using binary cross-entropy loss, which is shown in Equation 2.1 [6].

$$L_D = log(D(x)) + log(1 - D(G(z))) \tag{2.1}$$

where x is the real samples, $D(x)$ is the classification result of x, generated by the discriminator, where 0 means fake and 1 means real, z is the input feed to the generator, and $G(z)$ is the fake result made by the generator. The main objective of the discriminator is to minimize the loss, L_D.

The generator is usually structured as a generative neural network such as autoencoders. Its job is to fool the discriminator and make it classify the output made by the generator as real. Therefore, the generator will want L_D to be as large as possible. Therefore, the loss function of the whole GANs system can be formulated as Equation 2.2 [6].

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}} log(D(x)) + E_{z \sim p_z} log(1 - D(G(z))) \tag{2.2}$$

where x follows the distribution of the real dataset, p_{data}, and z is a randomly generated vector following the distribution of p_z, most of the time is a Gaussian distribution.

GANs have a significant benefit compared to previous deep learning methods, such as VAE. In previous methods, the loss is usually calculated using mean squared error or mean absolute error. However, due to alignment issues, the two loss functions mentioned above cannot reflect the quality of generated outputs properly. For instance, the generated speech might be perfect in terms of content and intonation, but due to a temporal alignment issue where the speaker speaks at a slower pace, the loss might be huge. This is no longer a problem for GANs as they do not calculate the capability of the generator directly but instead use a discriminator. Our task shifts from minimizing a loss that may or may not represent the capability of the model properly to fooling a network with a very simple task that has been proven to be very feasible.

Based on GANs, Speech Enhancement Generative Adversarial Network (SEGAN) [8] is developed. It is considered the first speech enhancement model that utilizes GANs. Similar to vanilla GANs, SEGAN also contains a generator and a discriminator. The training process is the same as vanilla GANs as well. The generation is used to conduct the speech enhancement task. It is a fully convolution network that has a decoder-encoder structure. During inferencing, the noisy signal is fed to the encoder to create a compressed representation, c, which is concatenated with a randomly generated latent representation, z. The result is passed to the generator to create the clean signal, completing the task. The discriminator is adopted from Least Square GAN (LSGAN) [19]. LSGAN uses a least-squares function with binary coding instead of binary cross-entropy loss as the loss for the discriminator. The discriminator adopts the encoder part of the generator. Just like any classification neural network, a fully connected layer is added at the end to reduce the dimension to 2, as there are two classes, real and fake. The model showed a good result compared to the traditional method using Wiener filters [20].

2.3 Convolutional Networks for Biomedical Image Segmentation (UNet)

UNet [13] was originally used on semantic segmentation tasks. It is a fully convolutional neural network that does not contain any FC layers. The model is made up of a number of downsampling blocks, the same number of upsampling blocks, and one middle block.

The downsampling blocks consist of several convolutional layers with stride (1,1), ReLU function as activation and batch normalization, and a max pooling layer with kernel size (2,2). The number of channels doubles but the dimensions of the feature maps are reduced by half. The output of each downsampling block is saved for later use.

The middle block contains a few convolutional layers with stride (1,1), so the dimension does not change much.

The upsampling block is made up of a transposed convolutional layer with stride (2,2), with the same number of convolutional layers as its matching downsampling block. The first downsampling block matches the last upsampling block, the second downsampling block matches the second-last upsampling block, etc. The feature maps from the previous block first go through the transposed convolutional layer, double its width and height, and then the result is concatenated with the output from its matching downsampling block. After that the feature maps pass through the rest of the convolutional layers, reducing its number of channels by half.

In the end, there is one final CNN layer with kernel size and stride both set to (1,1) that converts the result of the last upsampling block to the final output which has the number of channels that we want. The detailed structure of UNet is displayed in Figure 2.4.

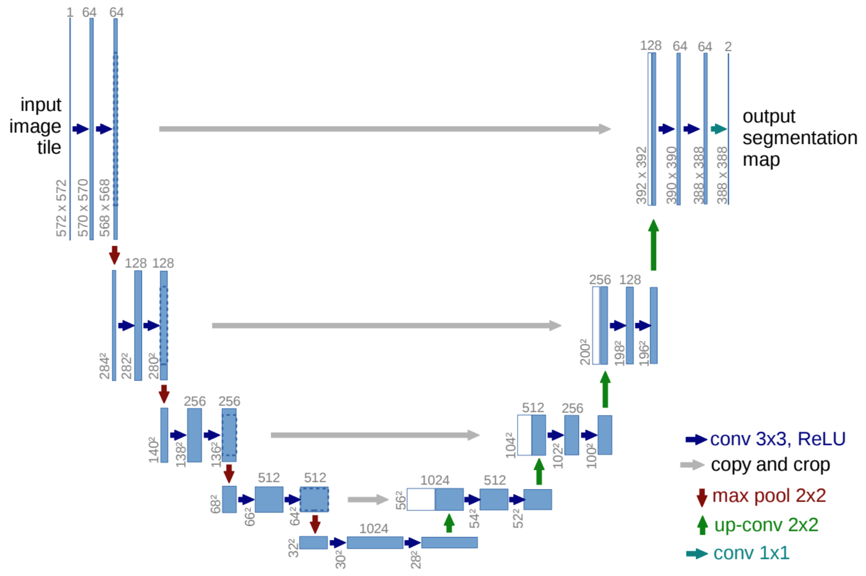


Figure 2.4: The architecture of UNet model from [13].

Besides its performance, UNet has some other advantages. First of all, UNet is extremely easy to implement, it only consists of convolutional layers, transposed convolutional layers, and max pooling, which are basic types of layers in neural networks. Moreover, it only requires a small amount of training data to perform well. Empirical results from [13] showed that only 30 images are needed for training the model to rank first in terms of the warping error in the 2015 EM Segmentation Challenge. Last but not least, it has a relatively short training time that only requires 10 hours to train using a Nvidia Titan GPU (6GB).

UNet has also been used to perform denoising tasks on images, such as RDUNet [14]. The structure of RDUNet is still the same as the original UNet but the author introduced residue connection to the sampling blocks, renaming it the denoising block. The architecture of the denoising block is shown in Figure 2.5.

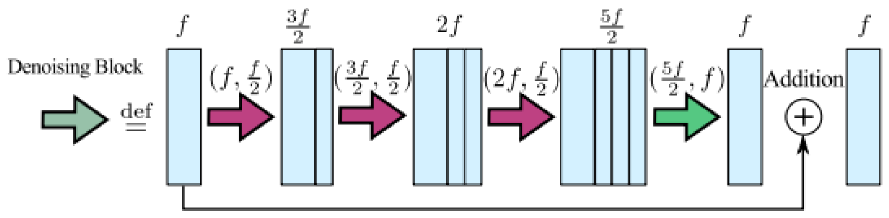


Figure 2.5: The structure of denoising block from [14].

When tested using both colored (CBSD68, Kodak24, Urban100) and greyscale datasets (BSD68, Kodak24, Set12), RDUNet outperforms the original UNet in terms of PSNR and SSIM. The strong performance of UNet in denoising tasks is one of the reasons why scientists choose to modify UNets for DDPM.

2.4 Speech Enhancement based on Denoising Diffusion Probabilistic (DDPM)

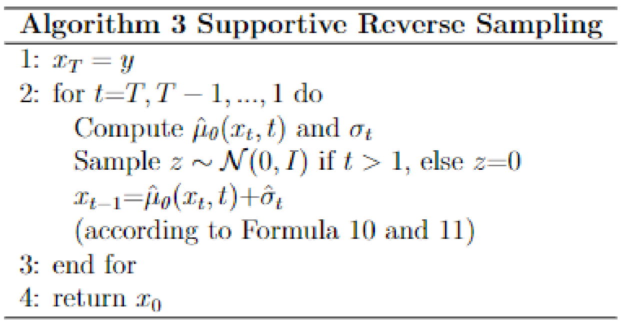
Generating raw audio without any conditions is always a hard task for any kind of model, regardless of whether they are flow-based, autoregressive, or GANs. Currently, the best result we have is to randomly generate a word but not a sensible sentence, even with DDPM. Therefore, most of the models based on the diffusion model are conditioned, meaning that they will feed the condition, usually a Mel-spectrogram to the model at every step, more specifically, for speech enhancement tasks, the condition is usually the Mel-spectrogram of the noisy audio.

One example is DiffuSE [21]. Instead of using the original reverse process, DiffuSE utilizes the condition to create a novel supportive reverse process to replace it. DiffuSE treats the input noisy audio, y, as a combination of clean signal, x_0, with noise, n, added to it, so that $y = x_0 + n$. y is then used as part of the new reverse process which is shown in Equations 2.3 and 2.4.

$$\hat{\mu}_\theta(x_t, t) = (1 - \gamma_t)\mu_\theta(x_t, t) + \gamma_t\sqrt{\bar{\alpha}_{t-1}}y \tag{2.3}$$

$$\hat{\sigma}_t = \sqrt{\sigma_t^2 - \gamma_t^w \alpha_{t-1}^{-}} \tag{2.4}$$

Besides replacing μ and σ with $\hat{\mu}$ and $\hat{\sigma}$, the rest is the same as the original DDPM. μ and σ represent the mean and standard deviation of the distribution of x_t, which is predicted by the trained model, $\epsilon(x_t, t)$. A more detailed version of the supportive reverse sampling is displayed in Figure 2.6.



```
Algorithm 3 Supportive Reverse Sampling
1: $x_T = y$
2: for $t$=$T, T-1, ..., 1$ do
       Compute $\hat{\mu}_\theta(x_t, t)$ and $\sigma_t$
       Sample $z \sim \mathcal{N}(0, I)$ if $t > 1$, else $z$=0
       $x_{t-1}$=$\hat{\mu}_\theta(x_t, t)$+$\hat{\sigma}_t$
       (according to Formula 10 and 11)
3: end for
4: return $x_0$
```

Figure 2.6: The algorithm of the supportive reverse sampling from [21]

Part 1 shows the input for the model, which is the noisy audio. Part 2 shows how the input, x_T is denoised step by step to create the clean audio, x_0, as the output.

The model, $\epsilon(x_t, t)$, is trained to calculate the Gaussian noise, ϵ. Instead of a traditional UNet, DiffuSE follows the structure of Diffwave [22], which is shown in Figure 2.7.

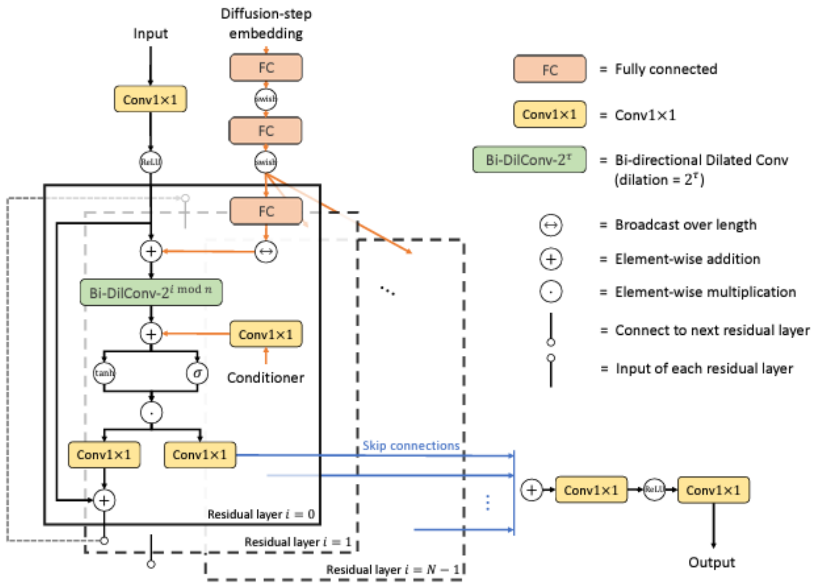


Figure 2.7: The structure of the $\epsilon(x_t, t)$ shown in [22].

The structure was originally adopted from WaveNet [23]. WaveNet and Diffwave are both audio-generative models. They can generate audio randomly without any condition, or work as a neural vocoder when given a Mel-spectrogram to condition on. WaveNet is an autoregressive model that utilizes dilated convolutional layers, meaning that the stride of the kernel is larger than the size of the kernel. The benefit of dilated convolutional layers is that it has a very large receptive field with only a few layers stacked together, so more information from the input can be included at each timestep of generation. Diffwave integrates the diffusion step embedding into the WaveNet model as the model needs to output different results based on different timesteps. The embedding is a 128-dimension vector, the first 64 elements are $sin(10^{\frac{4(n-1)}{63}}t)$ and the last 64 elements are $cos(10^{\frac{4(n-65)}{63}}t)$, where n is the position in the embedding vector ranging from 1 to 128. The embedding then passes through 3 fully connected layers. The dimension of the timestep embedding will be the same as the input vector. Since Diffwave and DiffuSE are not autoregressive models but instead, they replaced the one-directional dilating convolutional layers with its bi-directional counterpart, which produces better results as shown in earlier works [22].

DiffuSE offers a promising performance in comparison to popular models like SEGAN, DiffuSE outperforms SEGAN in terms of PESQ, COVL, and CSIG. The paper [21] also provides a smaller version of DiffuSE with less than half the size of the original DiffuSE model. The original DiffuSE has 200 timesteps and 128 residue channels while the smaller version only has 50 timesteps and 63 residual channels. The smaller model is able to produce results almost as good as the large one, showing its capability of achieving faster training and generation.

2.5 Discussion

In this section, we explained how AE, GANs, and UNet work, and their application to speech enhancement. All the methods above are capable of completing the task of speech enhancement, even though their performance differs from one another. Although the model architecture differs from each other, the core idea is still to calculate the L1 or L2 loss between the generated output and the clean audio signal. This leads to a problem that the generated audio might have a perfect quality, but there might be a higher loss due to a slightly slower speed of voice. Also, as there are only limited types of noises presented in the dataset, the model might not perform well on unseen kinds of noise. These are the problems we will deal with with our proposed model.

Chapter 3

Proposed Model

As we can see from the previous section, many techniques that are used on speech enhancement tasks were originally developed for various computer vision tasks. In fact, dealing with audio tasks is very similar to dealing with images. Instead of using raw audio directly, we usually convert them to spectrograms using Fourier Transform first. The spectrogram can be seen as a way of showing the signal strength of an audio utterance over time at various frequencies in an image, which allows it to be treated as an image, thus allowing us to apply computer vision techniques to audio data in the form of spectrograms.

The problem can be defined as this, given a noisy audio signal, x, and $x = x' + n$, where x' is the clean audio we want and n is the background noise, we created a model, F, that $F(x) = x'$.

3.1 Denoising Diffusion Probabilistic Model

The DDPM [1] is a generative model that consists of a forward process and a reverse process. An illustration of the processes is displayed in Figure 3.1.

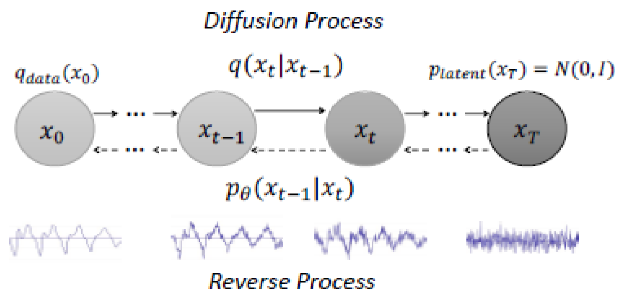


Figure 3.1: The structure of DDPM from [11], the right arrows indicate the forward process while the left arrows indicate the reverse process.

For a DDPM with a total of T timesteps, during the forward process, at timestep t, a randomly generated Gaussian noise will be added to the previous x_t after being modified by the variance scheduler, illustrated in Equation 3.1 [1].

$$q(x_t|x_{t-1}) = N(x_t; \sqrt{1 - \beta_t} x_{t-1}, \beta_t I) \tag{3.1}$$

where x_t represents the result of the input x_0 with noise being added for t times, and β_t is the noise scheduler. β is an arithmetic sequence with a β_1 as its smallest value and β_T as its largest value. β_t is the $t-th$ term in the sequence.

Because x_t is only dependent on x_{t-1}, the forward process is the same as a Markov Chain, as such, we have Equation 3.2 [1], which allows us to directly calculate x_t from x_0.

$$q(x_t|x_0) = \prod_{s=0}^{t-1} q(x_{s+1}|x_s) = N(x_t; \sqrt{1-\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I) \tag{3.2}$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$.

The value of β is relatively small, in [1], $\beta_1 = 0.0001$ and $\beta_T = 0.02$. Eventually, as the noise has been constantly added to x_0, we assume that at the end, x_t follows Gaussian distribution, because α_t is very close to zero.

The reverse process reverses the process of adding noise, which is akin to removing the noise from x_t to recover x_0. Given x_t and t, a model is trained to predict the Gaussian noise added to x_{t-1}.

$$p_\theta(x_{t-1}|x_t) = N(x_{t-1}, \mu_\theta(x_t, t), \sigma_\theta(x_t, t)) \tag{3.3}$$

where θ is the parameters we will obtain during training.

In Equation 3.3 [1] , $\sigma_\theta(x_t, t)$ is set to be $\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$ and $\mu_\theta(x_t, t)$ is shown in Equation 3.4.

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)) \tag{3.4}$$

where $\epsilon_\theta(x_t, t)$ is the output of the model we trained that is used to predict the Gaussian noise added to x_{t-1}.

Since x_{t-1} only depends on x_t but not any time step before it, The reverse process can be treated as a Markov chain as well. Therefore, we have Equation 3.5 [1].

$$p_\theta(x_{0:T}) = p_\theta(x_T) \prod_{t-1}^{T} p_\theta(x_{t-1}|x_t) \tag{3.5}$$

As such, $p_\theta(x_0) = \int p_\theta(x_{0:T})dx_{1:T}$,which is not tractable. Therefore, evidence lower bound (ELBO) loss is chosen to be the loss. The loss function can be simplified to Equation 3.6.

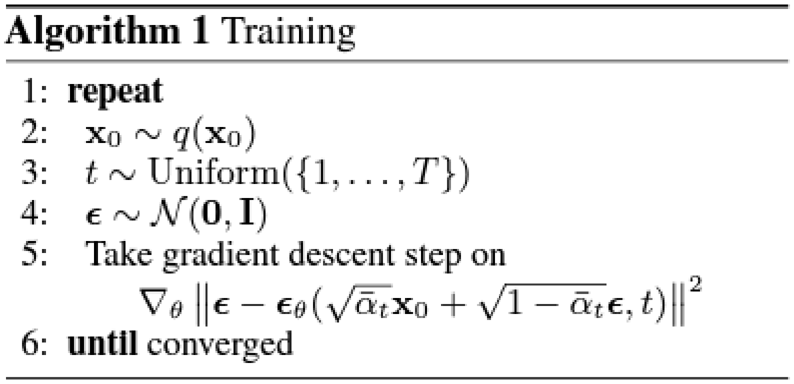
$$L_{simple}(\theta) = E_{t,x_0,\epsilon}[||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||^2] \tag{3.6}$$

which can be understood as the mean squared loss between the generated Gaussian noise and the noise predicted by the model trained.

In the application phase, the input is step-by-step recovered using the trained model following Equation 3.7 [1].

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)) + \sigma_t z \tag{3.7}$$

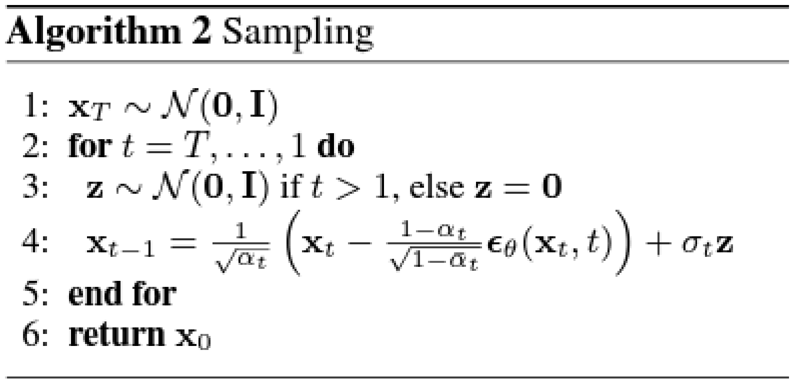
More details on the training and sampling procedure can be found in Figure 3.2 and Figure 3.3. All the equations can be found in [1].



Algorithm 1 Training

```
1: repeat
2:   $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3:   $t \sim \mathrm{Uniform}(\{1, \dots, T\})$
4:   $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5:   Take gradient descent step on
         $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
6: until converged
```

Figure 3.2: The training algorithm for DDPM from [1]. Line 2 to 4 are the inputs used to generate the noisy image, x_t, from the clean image, x_0. Line 5 is the backpropagation of the L2 loss between the actual noise and the predicted noise.



Algorithm 2 Sampling

```
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: for $t = T, \dots, 1$ do
3:   $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4:   $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
5: end for
6: return $\mathbf{x}_0$
```

Figure 3.3: The sampling algorithm for DDPM from [1]. Line 1 is the randomly generated Gaussian noise, which is used as the input. Line 2 to 4 describe the reverse process, how noise is step by step removed from the noisy input, x_T, to generate a clean image, x_0.

3.2 Proposed Modification

DDPM has a property that “when t is small, all but fine details are preserved, and when t is large, only large-scale features are preserved [1]”, where t is the timestep. As shown in Figure 3.4, as the value of t reduces, the outputs become more identical and much closer to what we humans expect them to be. In other words, when t is relatively small, we can feed the diffusion model a noisy image as the latent to conditioned on, and it will give us the clean image we expect after the reverse process. Of course, the noisy image should be only slightly contaminated by the noise and it should still resemble the original clean image to a certain extent (just like X_{250} shown in Figure 3.4). This is because that t is an indication of the noise level, if the input image is too noisy, the corresponding t will be a large number. In that case, we cannot preserve the fine details and the model will fail. In conclusion, we can use slightly noisy images as guidance for DDPM to generate the clean image we want.

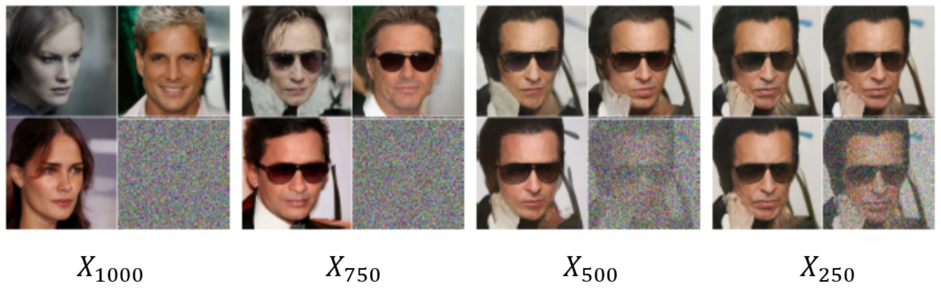


Figure 3.4: When conditioned on the same latent variables, the outputs share more and more details as timestep t decreases. The bottom-right images in each group are the latent x_t and the other three images in each group are the outputs of DDPM based on the latent variables[1].

The same idea can be applied to speech enhancement tasks as well. The target of speech enhancement is to remove the background noise from audio utterances while maintaining the quality of human speech. Using X_{250} as a reference, the Gaussian noise in X_{250} can be seen as the background noise in the noisy audio utterances, and the clean image of the man wearing a pair of sunglasses can be seen as the clean audio we want to obtain. It is even more convenient that these audio utterances that need to be enhanced can be identified as slightly noisy because the human voice is relatively intelligible even with the background noise interfering. As such, our model utilizes the property mentioned above. Our proposed approach builds upon the original architecture of DDPM. During the inferencing phase, we treated the input noisy audio utterances as the latent variables mentioned above for the model to be conditioned on (just like X_{250}), and let the trained DDPM to denoise the noisy inputs. Unlike other speech enhancement models based on conditioned DDPM that utilize all the timesteps, we use a t that matches the noise level of the input. We believe this will enable the model to give us clean audio.

In the training phase, the input to DDPM is in the form of spectrograms generated from clean audio utterances, instead of images for the original DDPM. In each training epoch, for every Mel-spectrogram, we randomly generate a timestep and a Gaussian noise, creating the input for the UNet following Equation 3.2. The UNet processes the input and gives us a predicted noise. The predicted noise is used to calculate the L2 loss between the randomly generated Gaussian Noise and itself, the loss is used for backpropagation.

UNet is chosen for various reasons, besides its capability in terms of speed and performance mentioned in the previous chapter, it is also very convenient for UNet to generate an output that has the same dimension as the input. Originally, UNet is used for semantic segmentation, having the same size of both the input and the output is a requirement. As such, we can easily calculate the L2 loss between the generated ground truth and the predicted value, which will help the UNet to improve itself and make better predictions along the training process.

Noted that, in the training phase, we only use clean audio utterances but not parallel datasets that contain matching clean and noisy audio utterances. The training process is shown in Figure 3.5.

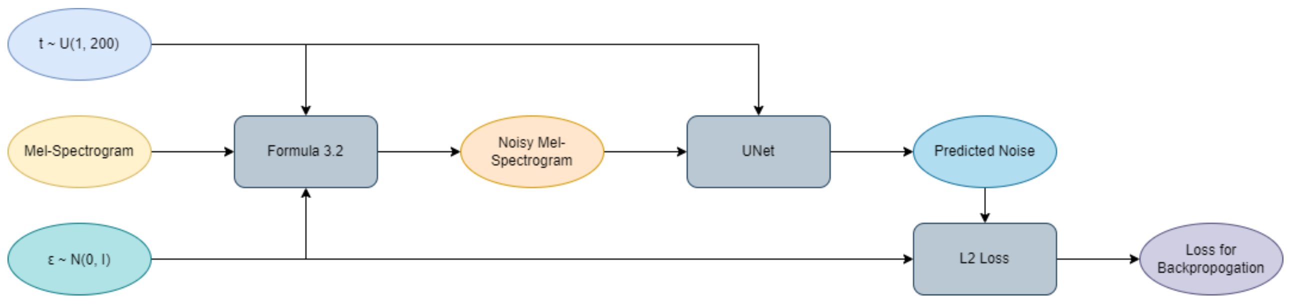


Figure 3.5: The training process of the proposed model, the ovals represent the data for each stage and the rectangles represent the calculation using the data input to them.

Randomly generating the timestep makes sure that all possible noise levels are considered. We tried to make the model more robust and able to handle any level of noise, but empirical results showed that when t is larger than 40, the results start to lose their identity. The sentence is no longer recognizable but the background noise is still removed.

It might seem counter-intuitive that we add Gaussian noise to the clean audio and train the model to predict the noise added, since most of the time, the background noise does not follow a standard Gaussian distribution. However, there is no way to create a model to find the distribution that fits all kinds of noises as it is extremely difficult to collect the data for such a task. Therefore, we chose Gaussian noise to simulate the distribution of noises as it is very commonly found in the living environment.

The diffusion model contains 200 steps, the noise scheduler is an arithmetic sequence ranging from 0.0001 to 0.02. These values are commonly used in audio-related models that are based on DDPM, such as CDiffuSE and Diffwave. The model we trained to calculate the noise added is a UNet that contains 12 blocks, 6 downsampling blocks, and 6 upsampling blocks. The channels for each downsampling block are [64, 64, 128, 128, 256, 256] respectively, and the reverse for upsampling blocks. These values are obtained empirically, when we increase the dimension and the number of blocks, the performance of the model does not improve, but when we decrease them, the performance becomes worse. Therefore, we believe that it is the optimum structure under current circumstances.

The overall structure of the UNet used in the model we proposed is displayed in Figure 3.6.

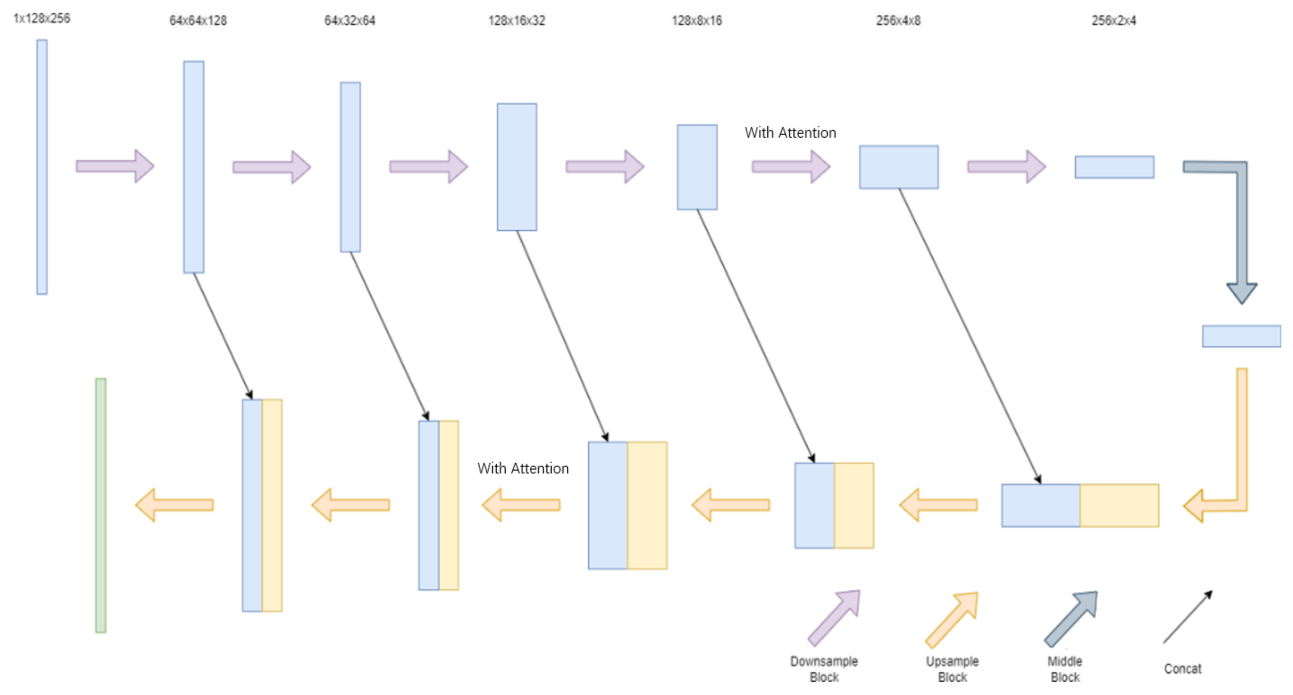


Figure 3.6: The structure of the UNet used in the proposed model. The rectangles with the same shape have the same dimensions.

In each downsampling block, there will be two convolutional layers with stride (1,1) and kernel (3,3), and another convolutional layer with the same kernel size but a different stride (2,2). The layers are connected using skip connection. If the dimension does not match, there will be another convolutional layer with kernel size (1,1) and stride set to (1,1) as well to adjust the number of channels. The output after each convolutional layer is padded with 0 to maintain its dimension. The dimension of the feature map will be halved at the end of each downsampling block.

At each upsampling block, there will be two convolutional layers with stride (1,1) and kernel size (3,3) as well, followed by a transposed convolutional layer with stride (2,2) and kernel size (4,4). The output after each layer is padded with 0 too. After going through the upsampling block, the width and height of the feature maps will be doubled. During the whole process of downsampling and upsampling, the dimensions of the feature maps are kept as a power of 2 to prevent the loss of any information and avoid dimensional problems when concatenating the feature maps during the upsampling phase. The structure of the downsampling and upsampling blocks are shown in Figure 3.7.

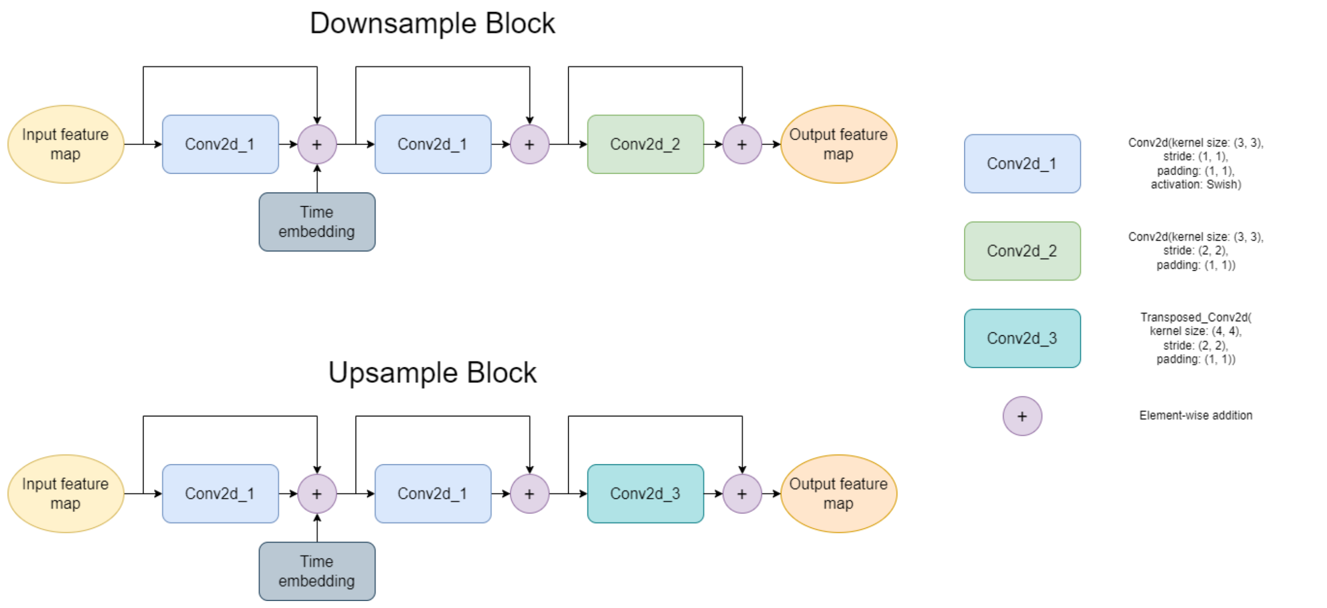


Figure 3.7: The structure of the downsampling blocks and upsampling blocks used in the proposed model. The downsample block is represented by the purple arrows and the upsample block is represented by the yellow arrows in Figure 3.6

The self-attention mechanism is applied to the second last downsampling block and the second upsampling block. The input feature map is first flattened and then passes through 3 different fully connected layers to get the query, Q, key, K, and value, V. The three matrices have the same dimension. The attention is calculated following Equation 3.8.

$$Self - Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.8}$$

where d_k is the number of channels of the input feature map. The output of the equation is reshaped to the same size as the input.

The implementation of this model can be found on Github[1] with some samples as well.

In the inferencing phase, instead of starting with the total timestep, T, we set the timestep, t to a small value, in this case, one-tenth of the total timestep, which is 20. On the one hand, if the value is too small, the output utterance will still be noisy, on the other hand, if the value is too large, the output will lose its intelligibility, and we will not be able to identify what the speaker is saying. We feed the noisy audio utterance to the UNet at the start of the reverse process. Instead of starting with timestep T and a Gaussian noise like the original DDPM, we started with timestep 20 and the noisy audio utterance. The model trained in the previous part will use it as a latent condition and work from there to generate the clean audio utterances we want. The training process is shown in Figure 3.8

[1]https://github.com/BobXiao97/Unsupervised-Zero-shot-Speech-Enhancement-based-on-DDPM

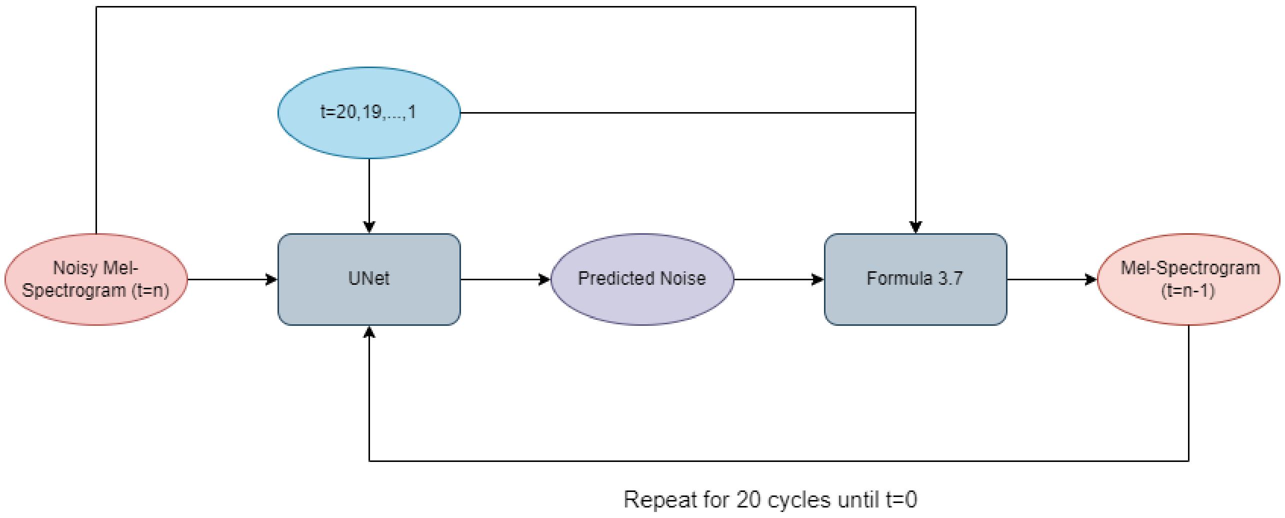


Figure 3.8: The inferencing process of the proposed model, the ovals represent the data for each stage and the rectangles represent the calculation using the data input to them.

The detailed algorithm is shown in Figure 3.9.

Algorithm 3 Inferencing

```
1: for $t=\frac{T}{10},\frac{T}{10}$-1,...,1 do
2:   $z \sim \mathcal{N}(0, I)$ if $t > 1$, else $z$=0
3:   $x_{t-1}=\frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\alpha_t}}\epsilon_\theta(x_t, t)) + \sigma_t z$
4: end for
5: return $x_0$
```

Figure 3.9: The algorithm of the inferencing phase. Line 1 is the timesteps that the model needs to start with, which is $\frac{1}{10}$ of the total timestep, T. Line 2 and 3 are the equations to remove the noise from the input noisy audio, where x_t is the noisy input and ϵ_θ is the UNet we trained to remove the noise. Line 5 is the result we obtained at last.

Since we only use one-tenth of the timestep, the speed of processing will be significantly faster than other diffusion-based speech enhancement models that utilize all T timesteps. The proposed model also has a reasonable training time of 20 hours thanks to UNet. Its FLOPs are significantly smaller than other DDPM-based models such as CDiffuSE.

Chapter 4

Experiment

4.1 Dataset

The dataset we used is a combination of "Noisy Speech Database for Training Speech Enhancement Algorithms and TTS Models" [24] and LJSpeech. [24] is created by adding 5 kinds of noise obtained from DEMAND dataset (Bus, Cafe, Living, Office, and Square) to "Centre for Speech Technology Voice Cloning Toolkit(VCTK) Dataset" [25] with different SNRs (0.25dB, 0.75dB, 1.25dB, and 1.75dB). The dataset consists 23,075 pieces of audio in total created by 56 speakers (28 male and 28 female) reading about 350 utterances each. For training the proposed model, we only use clean audio but not the noisy ones. LJSpeech consists of 13,100 short audio utterances read by a single speaker. The content is extracted from 7 non-fiction books. In conclusion, the training set contains 36,175 utterances.

The test set provided by [24] contains 824 utterances created by 2 speakers for testing. It is split into two parts, 100 and 724 as the validation set and the test set respectively. The validation set is used during training to see if there is overfitting or not and the test set is used to evaluate the performance of the model.

4.2 Preprocessing

Firstly, the raw audio is resampled to 16KHz. Due to the difference in audio duration, the dimension of the raw audio varies. All the audio is either padded with zeros or truncated to a fixed length of 65,280.

Since neural networks do not learn low-level features like raw audio easily, the audio is then converted to Mel-spectrograms by applying short-time Fourier transform (STFT) using Librosa, with the following parameters: length of FFT window: 1024, hop length: 256, number of Mel bands to generate: 128, max db: 80, which are the standard value used for audio related task. The Mel-spectrogram is further converted from power to decibel to make the Mel-spectrogram easier for the model to learn as the pattern is much clearer, as shown in Figure 4.1 and Figure 4.2.

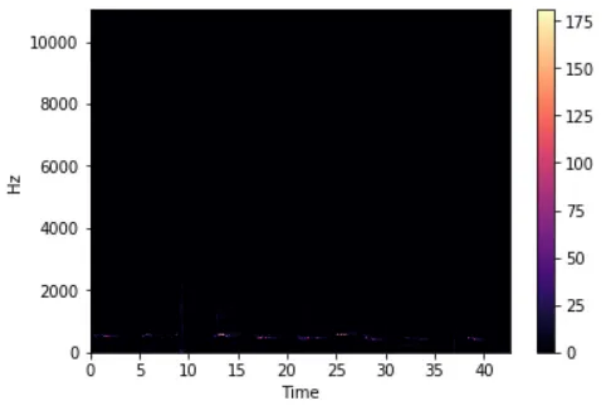


Figure 4.1: The Mel-spectrogram display of the haunting song of humpback whales in power [26]. There are some light spots at the bottom of the image.

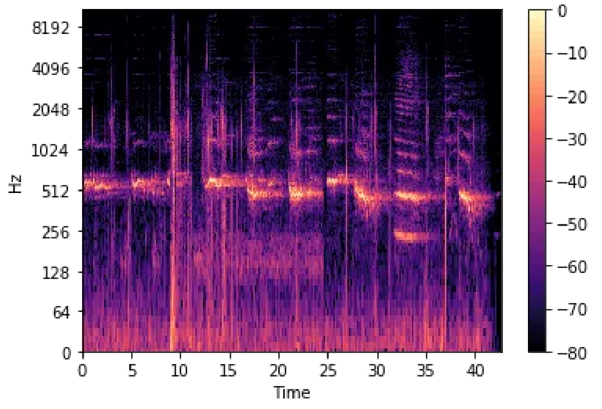


Figure 4.2: The Mel-spectrogram display of the haunting song of humpback whales in decibels [26].

In terms of the number of Mel bands, we chose 128 as the optimum value as it is the power of 2, so it is easier to calculate during the downsampling when the dimensions are reduced by half. If we change the value to 64, there will be missing information, and if we change the value to 256, there will be too many empty rows with 0 in the spectrogram, making it hard for the network to backpropagate.

Lastly, we normalize the Mel-spectrograms to 0 and 1 by dividing everything by 80. As a result, we convert every raw audio into a Mel-spectrogram with dimension [128,256] and every element ranging from 0 to 1.

4.3 Model Training

The dataset contains only clean audio. We trained the model for 500 iterations with a learning rate set to 0.00002 and batch size set to 16 using Adam optimizer. After every epoch, we used the validation set to evaluate the performance of the model to see if the validation loss exceeds the training loss. The parameters of the model are frozen during the validation process. If the validation loss is higher, we will stop the training to avoid overfitting. The training and validation loss is displayed in Figure 4.3. The test set is not involved in the training phase.

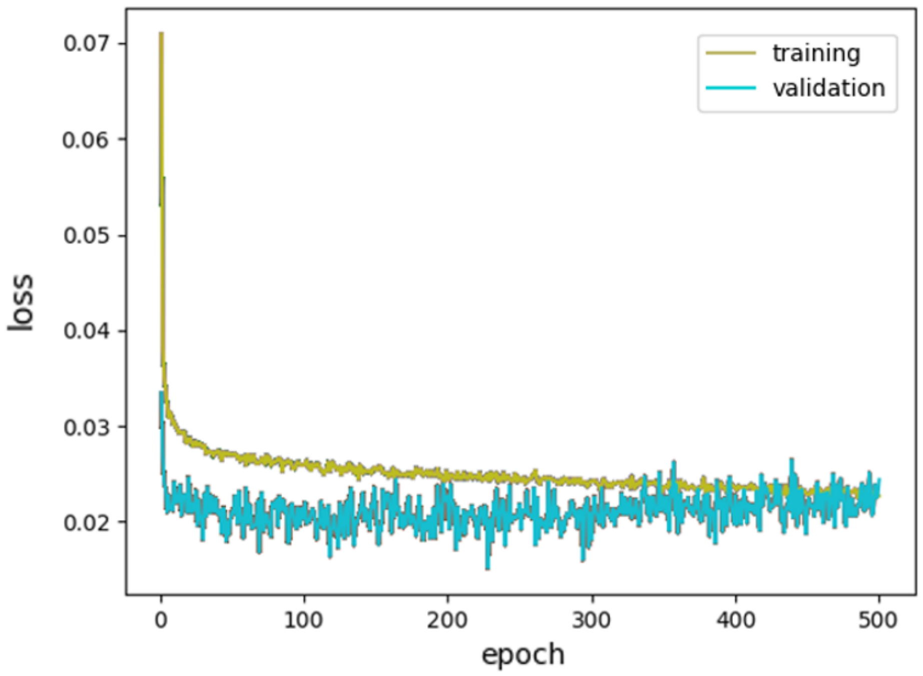


Figure 4.3: The loss of the proposed model.

We stopped the training process when the testing loss generally exceeded the training loss to avoid overfitting. In hindsight, although from the plot, we can see that the loss is gradually decreasing throughout the whole training process, the performance of the model did not improve much since 200 epochs based on the human listening test. It might be due to the fact that we cannot identify such tiny differences as the loss decreases very little through the training process after 200 epochs. However, since the model contains residual blocks, we decided to extend the training process to determine whether the performance will improve.

During the training phase, we have adjusted the number of sampling blocks. When we increase the number of sampling blocks, the performance remains more or less the same, but when we reduce the number of sampling blocks, the performance worsens. Therefore, we think that 6 sampling blocks are the optimum structure under current circumstances. We've also tested the total timesteps of the model ranging from 1000 to 200. Again, there is not much of a difference in terms of performance, but since fewer timesteps mean faster inferencing speed, we decided to use the smaller one.

To prepare for the evaluations for the next chapter, we convert the output Mel-spectrograms from decibels to power, followed by the inverse short-time Fourier transform using Librosa to obtain the audio in .wav form.

Chapter 5

Results

5.1 Baseline

The baseline that is used in this thesis is the Deep Complex Convolution Recurrent Network for Phase-Aware Speech Enhancement (DCCRN) [3]. DCCRN is one of the best existing models and achieved the best performance in the 'Interspeech 2020 Deep Noise Suppression Challenge'. It is a supervised model that uses a deep convolutional neural network to simulate complex value operations in order to fully utilize complex-valued spectrograms. While it is based on the encoder-decoder structure, it uses two decoders instead of one, with one for real values and one for imaginary values. Besides DCCRN, we also compared the performance of Speech Enhancement Generative Adversarial Network (SEGAN) and our model but with more sampling blocks.

5.2 Objective Evaluation

The models are evaluated using Perceptual Evaluation of Speech Quality (PESQ) [27] and Short-Time Objective Intelligibility (STOI) [28] and compared to DCCRN [3].

5.2.1 Perceptual Evaluation of Speech Quality (PESQ)

PESQ is a metric that objectively evaluates speech quality. It was standardized as Recommendation ITU-T P.862 in 2001, and widely used for objective voice quality testing by network equipment vendors, phone manufacturers, and telecom operators. The model requires two inputs, the clean audio that works as the ground truth, and a testing signal that is required to be evaluated. Firstly, the model level aligns the clean and testing signals and passes through a filter that uses the Fast Fourier Transform (FFT). The processed signals are then passed through a time alignment module. The module calculates an estimated delay time for each pair of inputs and the delay time is used to find the frame-by-frame delay used in the auditory transform. The auditory transform module is a psychoacoustic module that converts the input signals into a representation of perceived loudness in time and frequency. The output from the auditory transform module is then passed to the disturbance processing module, followed by the cognitive modeling module. These two modules calculate the absolute difference between the ground truth and the testing signal and give a measure of audible error. Symmetric disturbance, d_{SYM}, and Asymmetric disturbance, d_{ASYM}, are obtained from the previous step and the final result of PESQ is calculated following Equation 5.1.

$$PESQMOS = 4.5 - 0.1d_{SYM} - 0.0309d_{ASYM} \tag{5.1}$$

From the equation, we can see that the audio will score higher if the quality is better, and the best result is 4.5. The result ranges from -0.5 to 4.5, but it is highly unlikely to score less than 1.0 unless the distortion is extremely high. The full workflow of PESQ is shown in Figure 5.1.

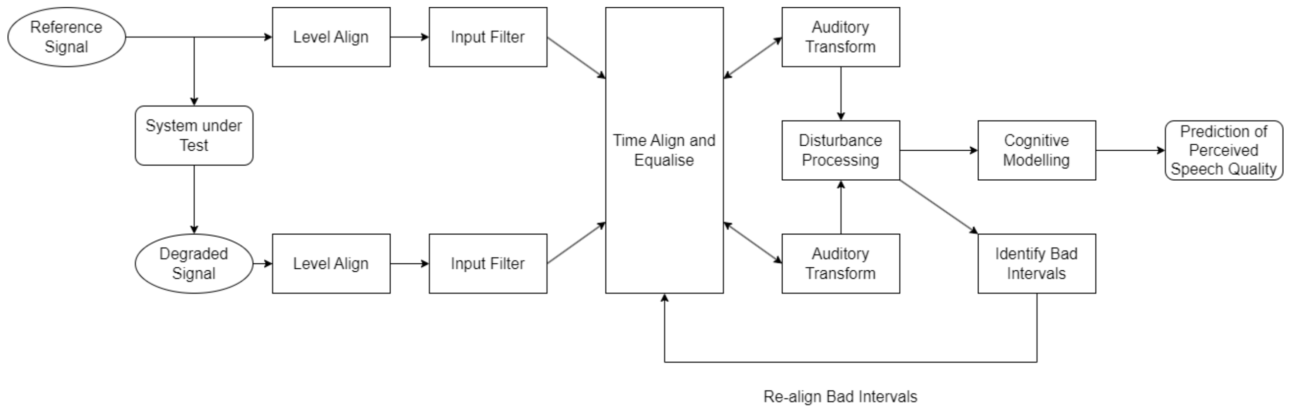


Figure 5.1: The workflow of PESQ evaluation metric adopted from [27] .

5.2.2 Short-Time Objective Intelligibility (STOI)

STOI is the metric that evaluates the intelligibility of the input signals. It calculates how much the testing audio is correlated with the ground truth. The testing signal and the ground truth are first decomposed into DFT-based one-third octave bands. Then, temporal envelope segments with a time-lapse of 384 ms are obtained from both the ground truth and the testing signal. These segments are first normalized and clipped, then compared by means of a correlation coefficient. Lastly, the short-time intelligibility measures are averaged to a scalar value, which is the final result. Again, the quality is better if the score is higher. The detailed procedure of STOI is shown in Figure 5.2.

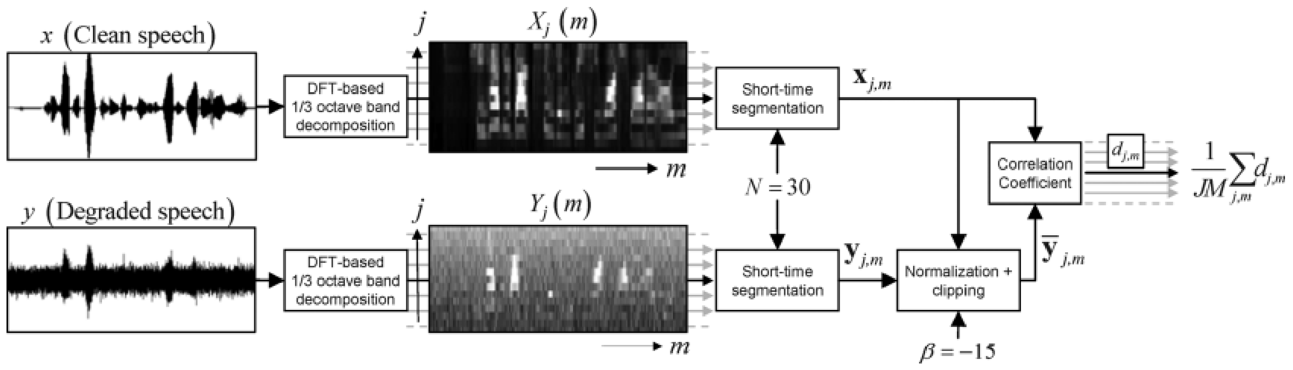


Figure 5.2: The workflow of STOI evaluation metric adopted from [29] .

5.2.3 Objective Evaluation Results

The data used to evaluate the two models is the test set provided by 'Noisy Speech Database for Training Speech Enhancement Algorithms and TTS Models' which contains 824 pieces of audio.

A higher score means better performance. The result is shown in Table 5.1.

Metric	PESQ	STOI
Proposed Model (6 blocks)	1.82	**0.75**
Proposed Model (8 blocks)	1.80	0.73
DCCRN	**3.05**	0.41
SEGAN	1.40	0.08

Table 5.1: Test Results of the proposed model compared to DCCRN and SEGAN.

5.3 Subjective Evaluation

For the subjective evaluation, we created a questionnaire that contains two parts, the MOS test and the ABX test. All the utterances used are the result of the proposed model and DCCRN using the noisy utterances in the test set.

5.3.1 Mean Opinion Score (MOS)

The MOS test contains a total of 20 utterances, 10 each produced by our proposed model and DCCRN. The noisy utterances are randomly selected from the test set. The utterances processed by the proposed model follow the inferencing algorithm described in the previous part and the utterances processed by DCCRN follow its own requirements. The reviewers are asked to grade the utterances created by the two models during the testing phase in terms of three aspects, noise canceling, intelligibility, and overall performance. Noise-canceling is to evaluate how well the model can remove the background noise and intelligibility is to see whether the model can maintain the identity of the human voice in the original utterance. The scores range from 1 to 5, with 1 being the worst and 5 being the best.

5.3.2 ABX test

The ABX test contains 10 pairs of utterances. These utterances are generated by the two models based on the same noisy utterance. The reviewers are asked to compare and indicate which of the two models has the better overall quality, and if it cannot be determined, the reviewer can choose X instead.

5.3.3 Subjective Evaluation Results

We received 9 results from 9 participants respectively, all of them are researchers working in the area of signal processing. In order to avoid bias, they do not know whether audio is created by the proposed model or DCCRN. For the MOS test, we took the average of the scores assigned by the reviewer for each model, and for the ABX test, we calculated the percentage of the reviews that prefer a certain model. Based on the valid answers we collected, Table 3 is created. The MOS results are shown in the first three columns and the ABX result is shown in the last column. The percentage does not add to 100% because there are reviewers choosing neither of the models. The reader is invited to listen to samples of our proposed method online[1]

Metric	Noise Canceling	Intelligibility	Overall	Preferred
Proposed Model	**3.50**	4.00	3.09	8.89%
DCCRN	3.40	**4.16**	**3.46**	**84.44%**

Table 5.2: Subjective Results of the proposed model compared to DCCRN.

5.4 Result Discussion

The objective evaluation gives mixed results probably due to alignment issues. If the audio is slightly slower or faster, even though the resulting audio might be audible and clear to humans, it will yield a low score due to temporal misalignment. Therefore, the subjective evaluation gives a more realistic measure of the output audio quality and is the main metric we focus on. This is also aligned with our main users, which are the listeners of the output audio and hence, their opinions and performance evaluation are important.

From the subjective evaluation results, we can clearly see that the proposed model is able to fulfill its task of noise canceling. In terms of the MOS test, it is able to achieve a very close score to DCCRN, even slightly better than it in terms of noise canceling. The generated audio is quite intelligible but not as perfect as expected. However, when compared against one another via the ABX test, DCCRN is much preferred compared to the proposed model, showing that the proposed model is slightly inferior to DCCRN.

During the experimentation phase, we tried to increase the number of downsampling and upsampling blocks from 6 to 8 and apply the attention mechanism more often, but the result remained relatively the same, or even slightly worse. However, there is a huge drop in performance if we reduce the number of blocks from 6 to 4. Therefore, we deduce that under current circumstances, the model has reached its peak performance. To further improve its performance, instead of the original UNet, we could adopt another model that may yield better results.

Also, in the dataset, the level of noise in each audio is different, intuitively, their timestep, seen as an indicator of noise level, should be different. Therefore, if we can adjust the timestep according to the input noisy audio instead of a fixed value, the performance might be much better.

There are successful models based on DDPM that generate beautiful music, such as [30]. This is probably because we do not require music to follow very strict rules. Unlike music, generating a human voice is much more difficult. Not only the words should be pronounced correctly, but the grammar also has to be correct as well. The diffusion model has yet to show promising results in the area of unconditional generation of spoken sentences. Based on [22], the overall performance of state-of-the-art models like Diffwave only achieved a MOS of 3.50 generating one word out of ten in the training set using the speech command 0-9 dataset [31], generating a whole sentence even with a guiding latent will only be harder.

Compared to image generation, audio generation is a more delicate matter as a small error will be extremely obvious to human ears. A few wrong patches or pixels might still result in an overall understandable image, the error might be negligible. However, a tiny difference in pronunciation will make the sentence hard to understand. As we can see in Figure 3.4, even though the results

[1]`https://github.com/BobXiao97/Unsupervised-Zero-shot-Speech-Enhancement-based-on-DDPM`

are very close when t is set to 250, there are still tiny differences in the results and these tiny differences will deteriorate the quality of the audio.

5.5 Further Improvement

Based on the result, there might be two potential improvements that can be made. The first one is to change the structure of the UNet in the diffusion model. One potential model is the WaveNet model that is used by DiffuSE and Diffwave mentioned in Chapter 2. The capability of the model has already been proven. It is more time-consuming compared to our proposed model, but since it uses bi-directional dilated convolution, it can extract high-level features from a larger window without missing any continuous information.

Another improvement that can be made is to add a classifier that can be used to determine timestep t given a piece of noisy audio. As part of our experimentation, we attempted to develop and train such a classifier for determining the ideal timestep for noisy input audio. We manually created a dataset, adding randomly generated Gaussian noise to the clean utterances from the VCTK dataset following Equation 3.1 with t ranging from 1 to 50. In the end, we will have a dataset of noisy audio and their matching timesteps, t. In other words, it is a standard classification dataset that has 50 classes. The dataset is fed to a pre-trained ResNext101 [32] model. ResNext is an image classification model that has been upgraded from ResNet. The model expands the residue blocks from ResNet, instead of just one path, it has multiple paths for each residue block. ResNext performed better than ResNet on the ImageNet-1K dataset even though the two models have similar parameters and FLOPs. Finetuning the ResNext101 model yields training results, shown in Figure 5.3 that look promising but when we fed it the original noisy data from the VCTK dataset whose noise is not Gaussian but captured in real-life situation, it gives t=0 most of the time.

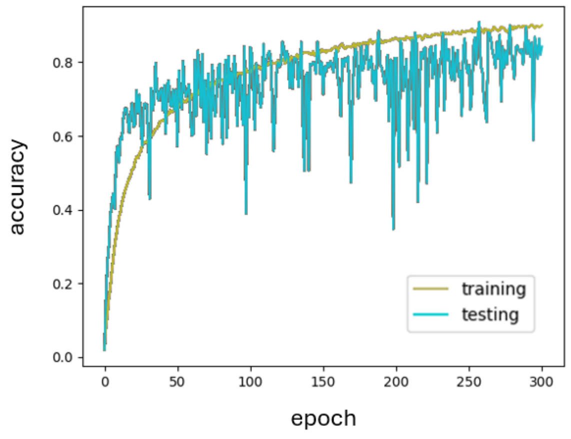


Figure 5.3: The training results of the classifier model, RexNext101. There is some turbulence in terms of the result of the test set, but generally, the accuracy is high enough to be applicable to the task.

Chapter 6

Conclusion

In this thesis, we introduce an innovative unsupervised way, that utilizes the property that diffusion models can be guided by a latent input to deal with speech enhancement tasks. In contrast to existing models for speech enhancement that are usually supervised, our proposed model has the following advantages. Instead of using an artificial parallel dataset with a limited number of noises, we only used the clean audio but not their noisy counterparts, and were able to achieve a promising result that is comparable to the state-of-the-art model, DCCRN. Compared to other DDPM-based models, the proposed model also has a higher inferencing speed as it only utilized one-tenth of the timestep. Although our proposed model is in its preliminary stages, we obtained promising performance in terms of noise canceling and STOI while the performance of PESQ, and intelligibility are not too far off from the state-of-the-art. We believe that there are various promising enhancements based on our preliminary experiments.

In the future, we could rework the noise prediction neural network and add a timestep prediction model to the proposed model to make the model more precise and help it achieve better performance. As a result, we believe that the zero-shot speech enhancement model that can deal with any noise has a bright future.

Bibliography

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[2] Yen-Ju Lu, Zhong-Qiu Wang, Shinji Watanabe, Alexander Richard, Cheng Yu, and Yu Tsao. Conditional diffusion probabilistic model for speech enhancement. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7402–7406. IEEE, 2022.

[3] Yanxin Hu, Yun Liu, Shubo Lv, Mengtao Xing, Shimin Zhang, Yihui Fu, Jian Wu, Bihong Zhang, and Lei Xie. Dccrn: Deep complex convolution recurrent network for phase-aware speech enhancement. *arXiv preprint arXiv:2008.00264*, 2020.

[4] Xugang Lu, Yu Tsao, Shigeki Matsuda, and Chiori Hori. Speech enhancement based on deep denoising autoencoder. In *Interspeech*, volume 2013, pages 436–440, 2013.

[5] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[7] Simon Leglaive, Xavier Alameda-Pineda, Laurent Girin, and Radu Horaud. A recurrent variational autoencoder for speech enhancement. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 371–375. IEEE, 2020.

[8] Santiago Pascual, Antonio Bonafonte, and Joan Serra. Segan: Speech enhancement generative adversarial network. *arXiv preprint arXiv:1703.09452*, 2017.

[9] Songxiang Liu, Yuewen Cao, Dan Su, and Helen Meng. Diffsvc: A diffusion probabilistic model for singing voice conversion. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 741–748. IEEE, 2021.

[10] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022.

[11] Yen-Ju Lu, Zhong-Qiu Wang, Shinji Watanabe, Alexander Richard, Cheng Yu, and Yu Tsao. Conditional diffusion probabilistic model for speech enhancement. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7402–7406, 2022.

[12] Meng Sun, Xiongwei Zhang, Thomas Fang Zheng, et al. Unseen noise estimation using separable deep auto encoder for speech enhancement. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(1):93–104, 2015.

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 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*, pages 234–241. Springer, 2015.

[14] Javier Gurrola-Ramos, Oscar Dalmau, and Teresa E Alarcón. A residual dense u-net neural network for image denoising. *IEEE Access*, 9:31742–31754, 2021.

[15] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[16] Lillian Weng. From autoencoder to beta-vae. `https://lilianweng.github.io/posts/2018-08-12-vae/`, August 2018.

[17] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[18] Israel Cohen. Noise spectrum estimation in adverse environments: Improved minima controlled recursive averaging. *IEEE Transactions on speech and audio processing*, 11(5):466–475, 2003.

[19] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

[20] Jae Lim and Alan Oppenheim. All-pole modeling of degraded speech. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(3):197–210, 1978.

[21] Yen-Ju Lu, Yu Tsao, and Shinji Watanabe. A study on speech enhancement based on diffusion probabilistic model. In *2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 659–666. IEEE, 2021.

[22] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

[23] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[24] Cassia Valentini-Botinhao et al. Noisy speech database for training speech enhancement algorithms and tts models. *University of Edinburgh. School of Informatics. Centre for Speech Technology Research (CSTR)*, 2017.

[25] Cassia Valentini-Botinhao, Xin Wang, Shinji Takaki, and Junichi Yamagishi. Investigating rnn-based speech enhancement methods for noise-robust text-to-speech. In *SSW*, pages 146–152, 2016.

[26] Dalya Gartzman. Getting to know the mel spectrogram. `https://towardsdatascience.com/getting-to-know-the-mel-spectrogram-31bca3e2d9d0`, August 2019.

[27] Antony W Rix, John G Beerends, Michael P Hollier, and Andries P Hekstra. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*, volume 2, pages 749–752. IEEE, 2001.

[28] Cees H Taal, Richard C Hendriks, Richard Heusdens, and Jesper Jensen. A short-time objective intelligibility measure for time-frequency weighted noisy speech. In *2010 IEEE international conference on acoustics, speech and signal processing*, pages 4214–4217. IEEE, 2010.

[29] Cees H. Taal, Richard C. Hendriks, Richard Heusdens, and Jesper Jensen. An algorithm for intelligibility prediction of time–frequency weighted noisy speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2125–2136, 2011.

[30] Curtis Hawthorne, Ian Simon, Adam Roberts, Neil Zeghidour, Josh Gardner, Ethan Manilow, and Jesse Engel. Multi-instrument music synthesis with spectrogram diffusion. *arXiv preprint arXiv:2206.05408*, 2022.

[31] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[32] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.