Self-Supervised Speech Enhancement Model Based on Denosing 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

This thesis proposes a novel method of utilizing the Denoising Diffusion Probabilistic Model (DDPM) [1] to fulfill speech enhancement tasks through self-supervised learning. Compared to traditional speech enhancement models, the proposed model does not require parallel data of matching pairs of clean and noisy audio and is able to maintain its performance with seen and unseen noise.

During the training phase, the procedure and model structure is the same as DDPM. We use clean utterances as the dataset, randomly generating a timestep and a Gaussian noise, then add the Gaussian noise to the clean utterances accordingly. At last, we use a UNet to predict the Gaussian noise based on the noisy input we created.

During the inferencing phase, the noisy utterances are used as the input together with a relatively small timestep. The model will gradually remove the noise from the input and give out a clean utterance. Since it requires fewer timesteps, the processing speed will be faster compared to other DDPM-based models.

The model yields promising results. Based on the subjective and objective evaluation results, by comparing to the award-winning model, DCCRN [2], the model we proposed is capable of performing similarly to DCCRN.

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. Traditionally, speech enhancement is done by calculating a mapping between matching clean and noisy speeches in the time-frequency domain. With the quick 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 [3]. Soon after, as generative neural networks such as VAE [4] and GANs [5] are invented, these techniques are applied to speech enhancement as well, creating promising results [6] [7].

In the past few years, DDPM [1] has become the star of all generative models, it has shown its excellent capability of generating images as well as audios [8] that outperforms GANs [9]. Also, 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. Models involving conditional diffusion probabilistic models [10] have been created and yield excellent performance.

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 shrivel [11]. As such, we propose a self-supervised diffusion model that solely requires clean data to train to solve this problem.

Chapter 2

Related Work

2.1 Speech Enhancement Based on Autoencoder (AE)

Autoencoders [12] is 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.

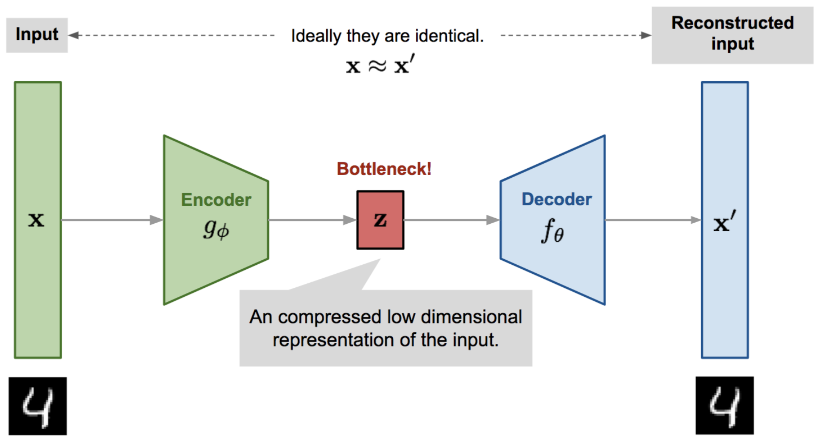

Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
$\mathbf{x}$
Encoder
g_ϕ
Bottleneck!
$\mathbf{z}$
Decoder
f_θ
$\mathbf{x}'$
An compressed low dimensional representation of the input.

Figure 2.1: The structure of AE adopted from [13].

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) [14]. Initially, the DAE is 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 the same way to fight against overfitting. As shown in Figure 2.2, there is not much of a difference in terms of model structure between DAE and AE, just the input changes from clean data to noisy data, and the loss is calculated between the data before adding noise to and the output created by the decoder.

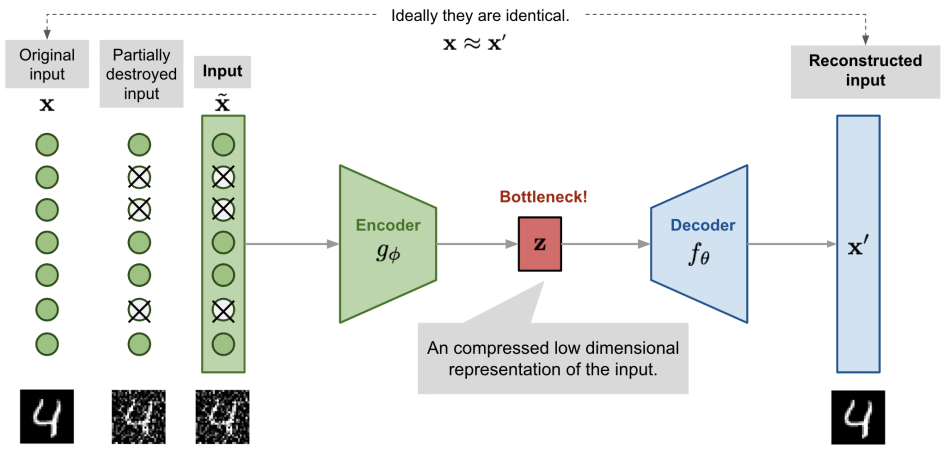

Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Original input
$\mathbf{x}$
Partially destroyed input
Input
$\tilde{\mathbf{x}}$
Encoder
g_ϕ
Bottleneck!
$\mathbf{z}$
Decoder
f_θ
Reconstructed input
$\mathbf{x}'$
An compressed low dimensional representation of the input.

Figure 2.2: The structure of DAE adopted from [13].

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 [3] 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 PESQ metric using a continuous Japanese dataset that contains 50 utterances, showing its capability of completing the task.

2.2 Speech Enhancement Based on Generative Adversarial Networks (GANs)

GANs [5], as the name suggests, is a model that consists 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.

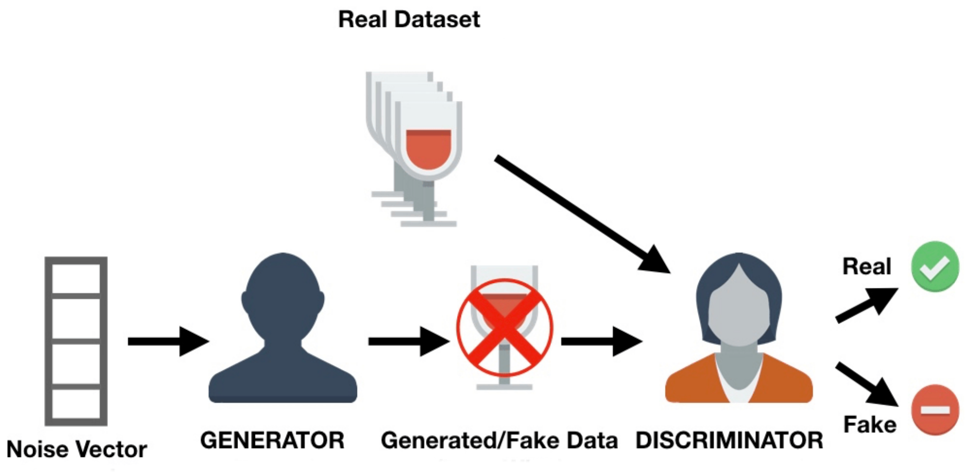


Figure 2.3: The architecture of GANs adopted from [15].

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, its job is to classify whether the input is real or fake (generated by the generator). Its loss function will just be very similar to any other classification model using binary cross-entropy loss, which is shown in Formula 2.1 [5].

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

where x is the real samples, $D(x)$ is the classification result of it made by the discriminator, 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. Of course, the discriminator aims 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 Formula 2.2 [5].

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

GANs indeed did the speech enhancement category a big favor. 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. The generated speech might be perfect but just because the speaker speaks a tad bit slowly, the loss might be huge. This is no longer a problem for GANs because, in GANs, we do not calculate the capability of the generator directly, instead, we 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, SEGAN [7] is developed. It is considered the first speech enhancement model that utilizes GANs. In terms of structures, there is not much of a difference between the original GANs, besides it utilized LSGAN [16] instead of vanilla GANs. LSGAN uses a least-squares function with binary coding instead of binary cross-entropy loss as the loss for the discriminator. The generator is a fully convolutional network with an encoder-decoder structure, and the discriminator adopts the encoder part of the generator. Of course, there will be FC layers to reduce the dimension to 2. The model showed a good result compared to the traditional method using Wiener filters [17].

2.3 UNet

UNet [18] work that 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 some 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 channel 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), and convolutional layers the same 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.

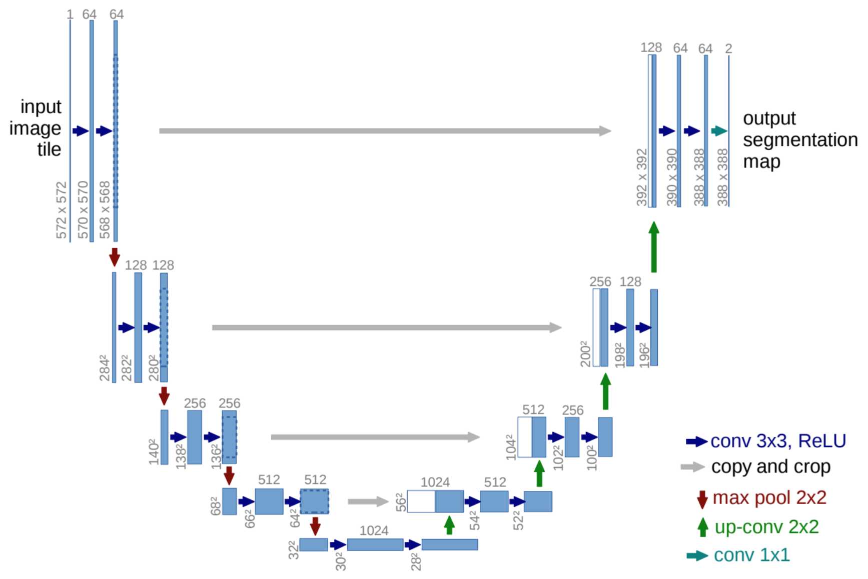


Figure 2.4: The architecture of UNet model from [18].

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 the most basic layers in neural networks. Moreover, it only requires a tiny amount of training data to perform well. Empirical results from [18] showed that only 30 images are needed for training the model. Last but not least, it is very fast during inferencing, only requiring less than one second to process every image with the size 512 by 512 using GPUs in 2015.

UNet has also been used to perform denoising tasks on images, such as RDUNet [19]. 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.

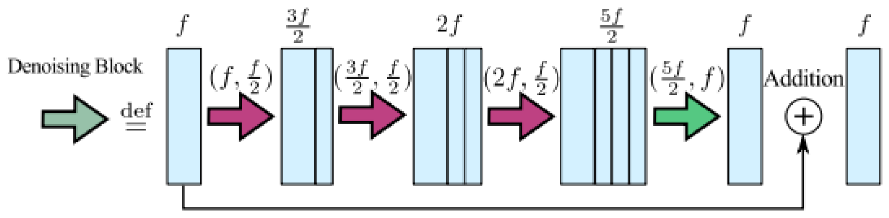


Figure 2.5: The structure of denoising block from [19].

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. UNet's capability of completing denoising tasks might be the reason why scientists choose to modify UNets for DDPM.

2.4 Speech Enhancement based on Denoising Diffusion Probabilistic Model

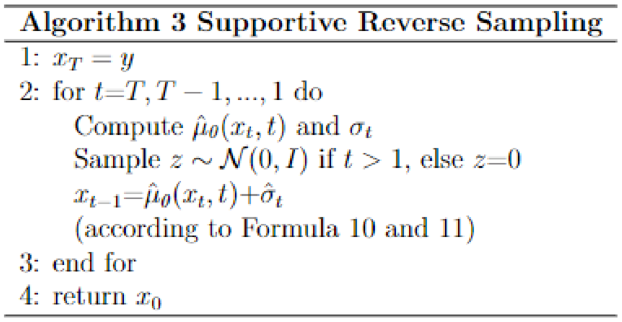
Generating raw audio without any conditions is always a hard task for any kind of model, no matter whether they are flow-based, autoregressive, or GANs. The best they can do is generate some sound similar to a word. The diffusion model could not make much of a difference. Therefore, most of the models 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 [20]. Instead of using the original reverse process, DiffuSE utilizes the condition to create a novel supportive reverse process to replace it. In [20], the author treats the input noisy audio, y, as an addition of clean signal, x_0, and the noise, n, so that $y = x_0 + n$. y is used as part of the new reverse process which is shown in Formula 2.3 and 2.4.

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

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

Besides replacing μ and σ with $\hat{\mu}$ and $\hat{\sigma}$, the rest is the same as the original DDPM. 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 [20]

The model, $\epsilon(x_t, t)$, is trained to calculate the Gaussian noise, ϵ. Its structure, instead of a traditional UNet, follows the structure of Diffwave [21], which is shown in Figure 2.7.

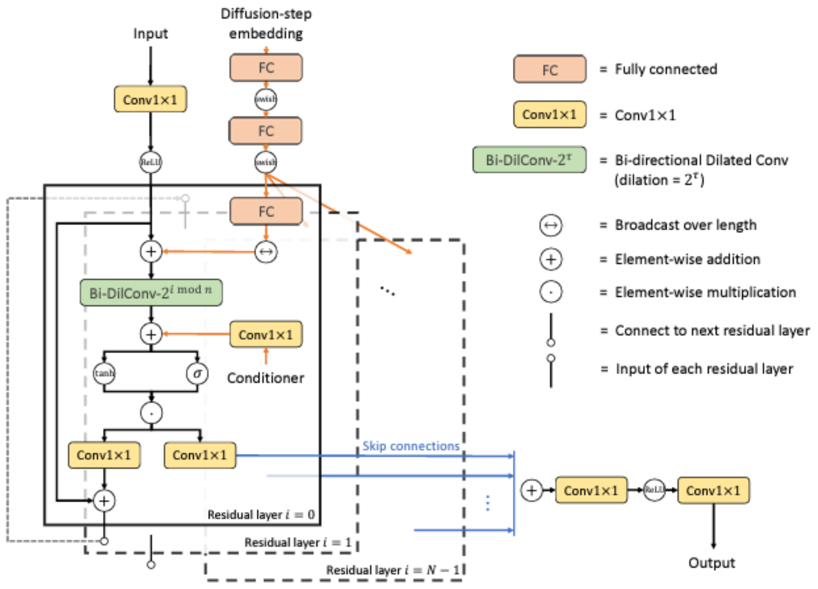

Input
Diffusion-step embedding
FC
swish
Conv1×1
ReLU
Bi-DilConv-$2^{i \bmod n}$
tanh
σ
Conditioner
Skip connections
Residual layer $i = 0$
Residual layer $i = 1$
Residual layer $i = N - 1$
Output
= Fully connected
= Conv1×1
Bi-DilConv-2^{τ} = Bi-directional Dilated Conv (dilation = 2^{τ})
= Broadcast over length
= Element-wise addition
= Element-wise multiplication
= Connect to next residual layer
= Input of each residual layer

Figure 2.7: The structure of the $\epsilon(x_t, t)$ shown in [21].

The structure is originally adopted from WaveNet [22], but since Diffwave and DiffuSE are not autoregressive models, they replaced the one-directional dilating convolutional layers with its bi-directional counterpart, which has been proved in [21] will create better results.

DiffuSE yields a promising result, when compared to popular models like SEGAN, DiffuSE outperforms SEGAN in terms of PESQ, COVL, and CSIG. The paper also provides a smaller version of DiffuSE with less than half the size of the proposed model (50 timesteps, 63 residual channels vs 200 timesteps, 128 residual channels), which is able to create results almost as good as the large one, showing its capability of achieving faster training and generation.

Chapter 3

Proposed Model

As we can see from the previous section, many techniques that are used on speech enhancement tasks were originally developed in the category of computer vision. 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 that allows us to perform all the computer vision techniques on it.

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.

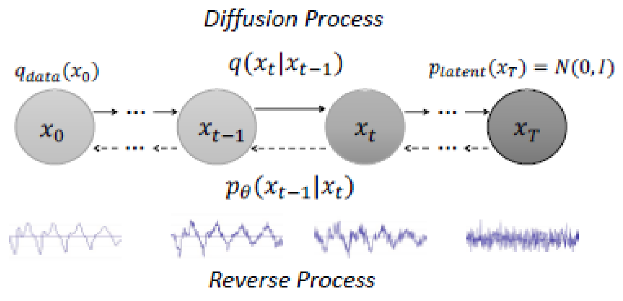


Figure 3.1: The structure of DDPM from [10], the rightward arrows symbol the forward process while the leftward arrows symbol the reverse process.

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 Formula 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 being added noise 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 Formula 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. Therefore, it removes the noise from x_t to recover x_0. Given x_t, a model that combines UNet [18] and transformer [23] 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 Formula 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 after some calculations shown in [1], we have Formula 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 Formula 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, ELBO loss is chosen to be the loss. After some calculations in [1], the loss function is simplified to Formula 3.6.

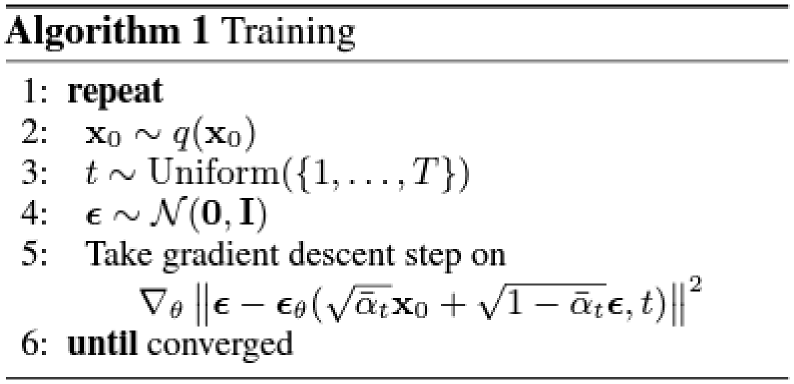
$$L_{simple}(\theta) = E_{t,x_0,\epsilon}[||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t||^2] \tag{3.6}$$

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 Formula 3.7 [1].

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

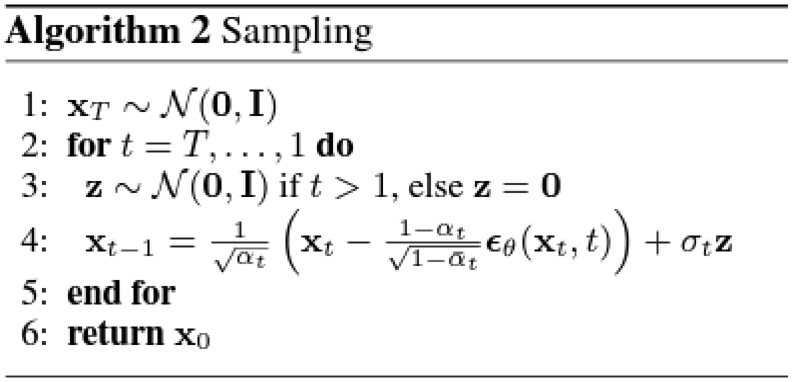
The more detailed training and sampling procedure can be found in Figure 3.2 and Figure 3.3. All the formulas can be found in [1].



Algorithm 1 Training

```
1: repeat
2:   $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3:   $t \sim \text{Uniform}(\{1, \ldots, T\})$
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].



Algorithm 2 Sampling

```
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: for $t = T, \ldots, 1$ 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].

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 we humans can still identify what the clean image should be after denoising (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.

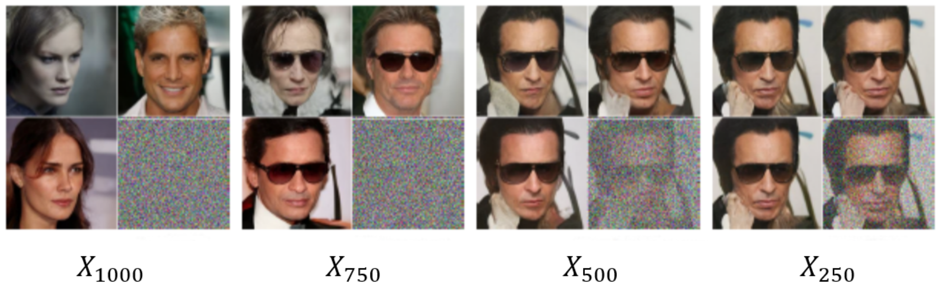


Figure 3.4: When conditioned on the same latent variables, the outputs share more and more details as timestep t decreases. The bottom-right images are the latent x_t and the rest are the outputs of DDPM based on the latent variables[1].

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. We did not change the original structure of the DDPM. We just 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 DDPM perform its denoising magic on the noisy inputs. We believe this will enable to model to give us clean audio.

In the training phase, there is not much difference from training the original DDPM, but instead of using images, we use spectrograms generated from clean audio utterances as inputs. In each training epoch, for every Mel-spectrogram, we randomly generate a timestep and a Gaussian noise, creating the input for the UNet following Formula 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. 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 diffusion model contains 200 steps, the noise scheduler is an arithmetic sequence ranging from 0.0001 to 0.02. 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.

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 and stride both set to (1,1) 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, therefore, we will not lose 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 3.5.

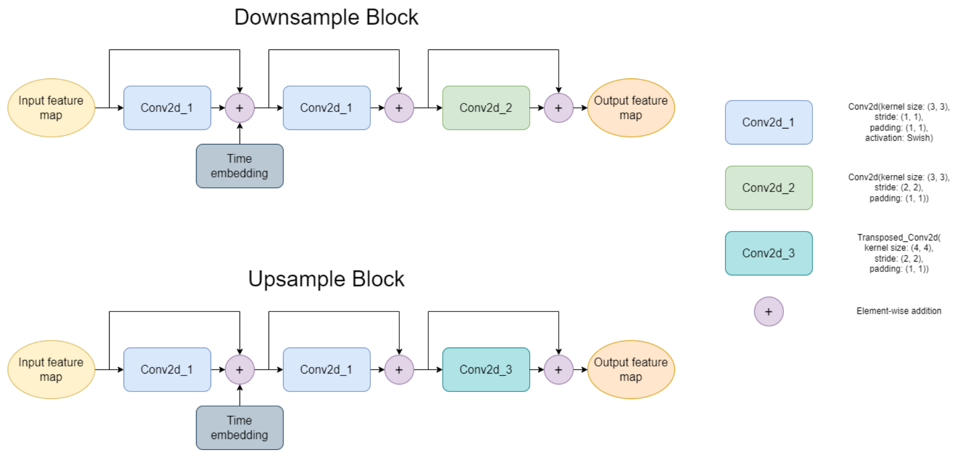


Figure 3.5: The structure of the downsampling blocks and upsampling blocks used in the proposed model.

The attention mechanism is applied to the second last downsampling block and the second upsampling block.

The overall structure of the UNet used in the model we proposed is displayed in Figure 3.6. The implementation of this model can be found on Github[1] with some samples as well.

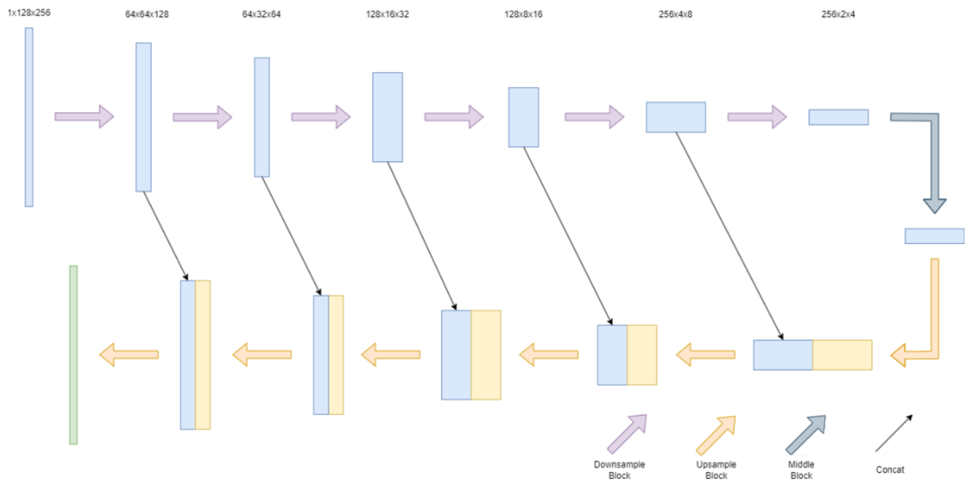
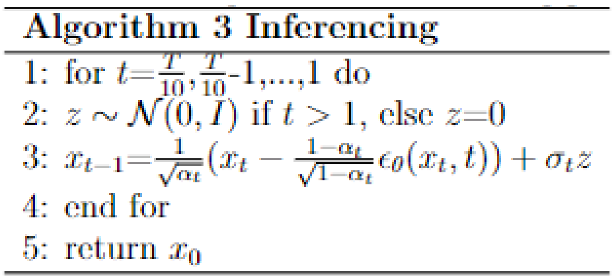


Figure 3.6: The structure of the UNet used in the proposed model. The rectangles with the same shape have the same dimensions.

In the inferencing phase, we do not start with the total timestep, T, instead, we set the timestep, t to a small value, in this case, one-tenth of the total timestep, which is 20. This value is obtained empirically. 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

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

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 noise 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 more detailed algorithm is shown in Figure 3.7.



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 3.7: The algorithm of the inferencing phase.

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.

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 noise obtained from DEMAND dataset to "Centre for Speech Technology Voice Cloning Toolkit(VCTK) Dataset" [25]. The dataset consists 23075 pieces of audio in total created by 56 speakers (28 male and 28 female) reading about 350 utterances each. 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.

We used the test set provided by [24] which contains 824 utterances created by 2 speakers for testing.

4.2 Preprocessing

Firstly, the raw audio is resampled to 16KHz. Due to the difference in time occupancy, the dimension of the raw audio varies. All the audio is either padded with zeros or chopped to a fixed length, 65280.

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. 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.

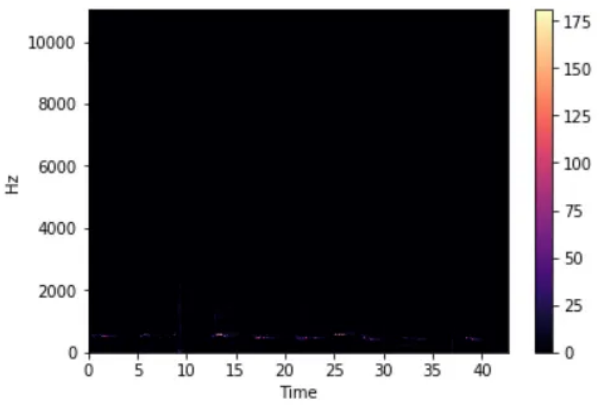


Figure 4.1: The Mel-spectrogram display of the haunting song of humpback whales in power [26]. There are some light spots at the bottom of the image.

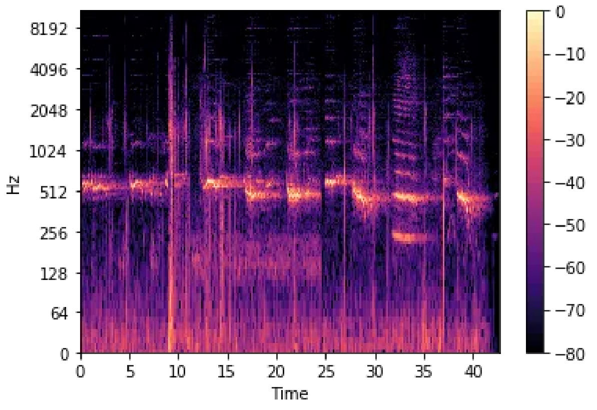


Figure 4.2: The Mel-spectrogram display of the haunting song of humpback whales in decibels [26].

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

We trained the model for 500 iterations with a learning rate set to 0.00002 and batch size set to 16 using Adam optimizer.

Chapter 5

Results

The training and test loss is displayed in Figure 5.1.

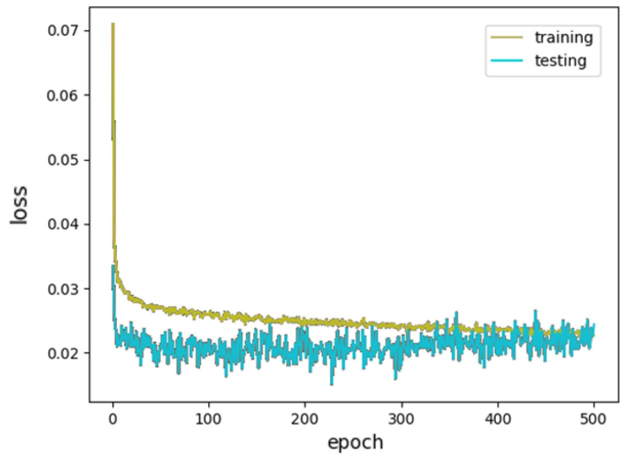


Figure 5.1: The loss of the proposed model.

Although from the plot, we can see that the loss is gradually decreasing throughout the whole training process, after 200 epochs, the model does not improve much after that.

5.1 Objective Evaluation

The models are evaluated using PESQ [27] and STOI [28] and compared to DCCRN [2].

5.1.1 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 Formula 5.1.

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

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 5.2.

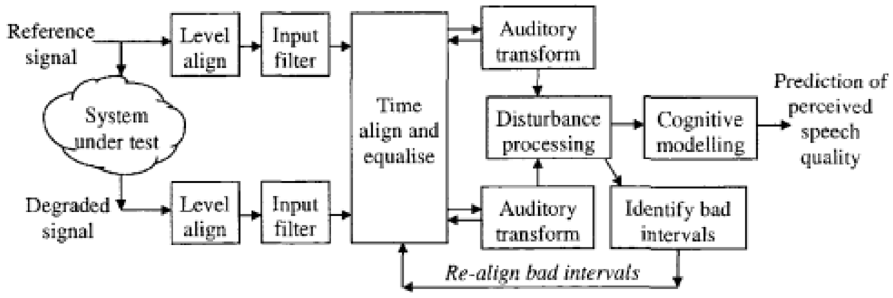


Figure 5.2: The workflow of PESQ evaluation metric adopted from [27] .

5.1.2 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 more detailed procedure of STOI is shown in Figure 5.3.

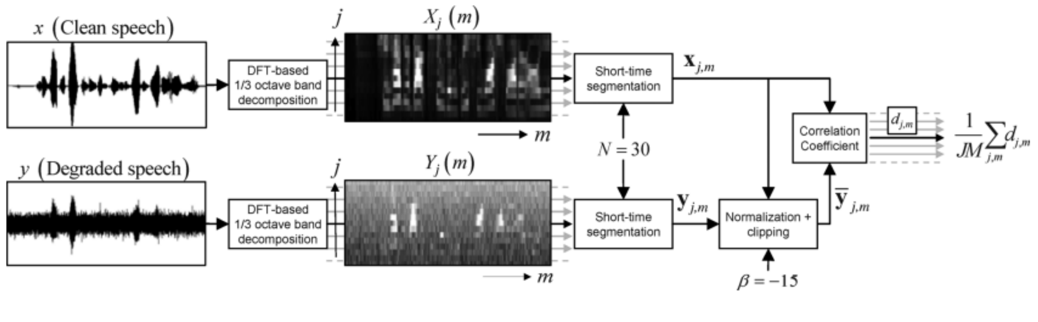


Figure 5.3: The workflow of STOI evaluation metric adopted from [29] .

5.1.3 Objective Evaluation Results

DCCRN [2] is one of the best models existing and achieved the best result in the "Interspeech 2020 Deep Noise Suppression Challenge". It uses a deep convolutional neural network to simulate complex value operations in order to fully utilize complex-valued spectrograms. 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	1.82	**0.75**
DCCRN	**3.05**	0.41

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

5.2 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.

5.2.1 Mean Opinion Score (MOS)

The MOS test contains a total of 20 utterances, 10 from each model. The reviewers are asked to grade the utterances created by the two models during the testing phase in three ways, noise cancelling, intelligibility, and the 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, 1 being the worst and 5 being the best.

5.2.2 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 which one has the better overall quality, and if it cannot be determined, the reviewer can choose X instead.

5.2.3 Subjective Evaluation Results

We received 9 valid results. For the MOS test, we took the average of the scores each model got from the reviewers, 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 1 because there are reviewers choosing X. The reader is invited to listen to samples of our proposed method online[1]

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

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.

5.3 Result Discussion

The objective evaluation gives mixed results probably due to alignment issues, especially for STOI, which does not have an alignment mechanism. If the audio is slightly slower or faster, even though the result might be perfect sound to humans, it will yield terrible results. Therefore, we will focus more on the subjective evaluation. After all, it is how listeners think of the performance that matters most.

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.

Unlike music generation, the Diffusion model has yet to show promising results in the area of unconditional generation of spoken sentences. Top-notch models like Diffwave [21] only achieved a MOS of 3.50 generating one word out of ten in the training set, 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 mishap will be extremely obvious to human ears. A tiny difference in pronunciation will make the sentence hard to understand. As we can see in Figure 3.4, even though the results 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.

5.4 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. 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 deduct 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, like the one used in Diffwave.

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. 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. During the experimentation phase, we tried to train this classifier. We manually created a dataset, adding randomly generated Gaussian noise to the clean utterances following Equation 1 with t ranging from 1 to 50 and fed it to a pre-trained ResNext101 [30] model. Finetuning the ResNext101 model yields training results, shown in Figure 5.4 that look promising but when we apply it to the problem, it gives t=0 most of the time.

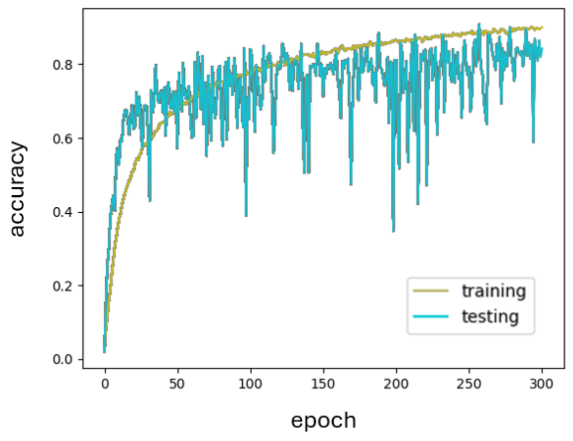


Figure 5.4: The training results of the classifier model, RexNext101. There is some turbulence in terms of the result of the test set, but generally, the accuracy is high enough to be applicable to the task.

Chapter 6

Conclusion

In this thesis, we introduce an innovative unsupervised way, which utilizes the property that diffusion models can be guided by a latent input to deal with speech enhancement tasks, that are usually supervised. We are able to achieve a promising result that is comparable to the top-of-the-league model, DCCRN. Although the performance is not totally satisfying, the Diffusion model is very young and we believe that there still is a lot of potential for us to find out. The zero-shot speech enhancement model that can deal with any noise might be the future of speech enhancement.

Bibliography

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

[2] 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.

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

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

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

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

[14] 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.

[15] Datacamp Team. Demystifying generative adversarial nets (gans). `https://www.datacamp.com/tutorial/generative-adversarial-networks`, May 2018.

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[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] 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.