THE HONG KONG POLYTECHNIC UNIVERSITY

CAPSTONE PROJECT

Comparison of the Performance of GAN and Diffusion Models in the Vocoder Field

Author:
XUE, Bingxin

Supervisor:
Dr. Jiang, Binyan

A thesis submitted in fulfillment of the requirements
for the degree of Bachelor of Science
in the
Data Science and Analytics

May 12, 2024

Contents

1 Introduction

The vocoder is mainly responsible for reconstructing parameters of characterising sound segments, frequently the short-term spectrogram of sounds, such as the mel-spectrogram, into a waveform perceptually similar to the original audio. (Di Giorgi et al., 2022; Alencar and da Rocha Jr, 2022) Converting the frequency domain features into a time domain waveform is one of the vital processes of speech synthesis and directly affects the quality of the final synthesised speech.

The reconstruction of audio is affected by the phase estimation problem. The traditional method enhances the consistency of the spectrogram to iterative phase estimation using the Grin-Lim algorithm, but the synthesised speech is unnatural.

People discovered that the neural network could solve this problem better currently, and these vocoders using neural networks for speech synthesis are called "neural vocoders". GAN-based vocoders and Diffusion-based vocoders are two of the representatives. (Tan et al., 2021)

The Generative Adversarial Network (GAN) model produces speech similar to actual speech by adversarial training. (Goodfellow et al., 2014) The Diffusion model iteratively improves the sample nature by adding noise to generate realistic speech. (Ho et al., 2020

This research aims to explore the differences between these two generative models, GAN and Diffusion, in the vocoder domain. By comparing their performance and characteristics, we can better understand and evaluate their applicability in speech synthesis and provide a reference for future development of vocoder technology.

2 Background

The development of deep learning gradually makes the (deep) neural network a significant component of speech synthesis. For example, Aaron van den Oord proposed WaveNet in 2016,, which directly converts linguistic features into waveforms using neural networks. Unlike previous vocoders, neural network-based vocoders are more intelligible and natural, requiring less manual preprocessing and feature development. (Tan et al., 2021) As popular neural network-based generative models in recent years, the GAN and Diffusion models have also been used as vocoders. To better understand the differences between the two models, the following will briefly describe the basic construction of the two models and their operation.

2.1 Generative Adversarial Network

The GAN model was first proposed by Ian Goodfellow et al. in 2014, and its uniqueness lies in introducing a generator and discriminator, which is an adversarial mechanism. The generator generates false samples, while the discriminator distinguishes the false and actual samples. (Goodfellow et al., 2014)

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \quad (1)$$

In the above function, G represents the generator, D represents the discriminator, x is the actual sample, and z is the synthetic sample. The generator aims to maximise the discriminator's error rate and generate samples almost identical to the actual sample. The discriminator seeks to minimise the identification error rate and accurately distinguish the actual and synthetic samples. The GAN model achieves a non-cooperative game through adversarial training between them, which will eventually reach an equilibrium state, i.e., the Nash equilibrium. (Goodfellow et al., 2014)

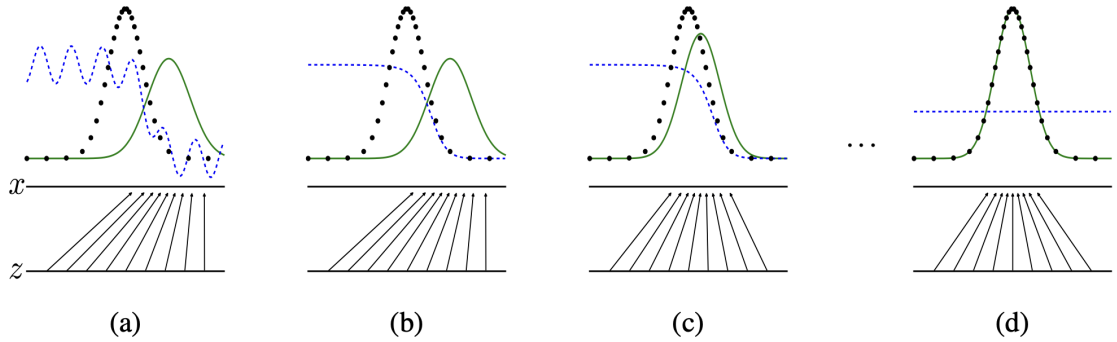

Figure 1: The green curve representing the generator distribution converges with the black dotted line of the actual sample distribution such that the discriminator cannot distinguish them gradually. Hence, the distributions of the discriminators finally become a straight line, i.e., $D(x) = \frac{1}{2}$. (Goodfellow et al., 2014)

The following is the detailed process (Goodfellow et al., 2014):

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

2.2 Diffusion

The Diffusion model was first proposed by Sohl-Dickstein et al. in 2015 but did not gain attention until the publication of *Denoising Diffusion Probabilistic Models* in 2020. (Ho et al.) It is a latent variable model that exists unobserved variables(latent variable). (Cai, 2012) The principle of operation of Diffusion models can be divided into two steps: forward process and reverse process.

2.2.1 Forward Process

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \quad q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{2}$$

The forward process can also be called the diffusion process. $x_1, ..., x_T$ are latents of the same dimensionality as the data $x_0 < q(x_0)$. It utilizes the approximate posterior $q(x_{1:T}|x_0)$, which is fixed to a Markov chain, to add Gaussian noise with variance $\beta_1, ..., \beta_T$ to the data so that the samples gradually become pure noise.

2.2.2 Reverse Process

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \tag{3}$$

The reverse process, i.e., denoising, starts from $p(x_T) = \mathcal{N}(x_T; 0, \mathbf{I})$ and gradually learns a Gaussian distribution, which recovers the noise to the original audio through a Markov chain. The detailed process can be seen in the following two figures; this is an example of adding noise and denoising:

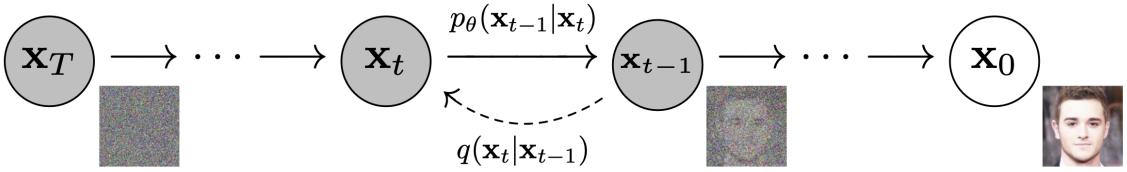

Figure 2: This figure shows how the model works in a Diffusion model. (Ho et al., 2020)

Example of detailed model operation process:

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
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

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: This figure shows the detailed diffusion process, where x_0 denotes the original image, t denotes the time step, ϵ denotes the noise added by the diffusion process, ϵ_θ denotes the noise predicted by the inverse diffusion process, $\alpha_t = 1 - \beta_t, \bar{\alpha}_t = \Pi_{i=1}^{t} \alpha_i$ and $\sigma_t^2 = \beta_t$. (Ho et al., 2020)

The forward and reverse processes help the model learn the sample features so that the final synthesised image becomes more and more similar to the original image.

3 Project Goals and Objectives

To compare the two differences in the vocoder field, I chose the GAN-based model, Hi-Fi GAN (Kong et al., 2020), and the Diffusion-based model, WaveGrad (Chen et al., 2020), to make a detailed comparison. Although they belong to different vocoder types, their inputs and outputs are consistent: Inputs are mel-spectrogram, and the outputs are audio.

3.1 Hi-Fi GAN

3.1.1 Data Preprocessing

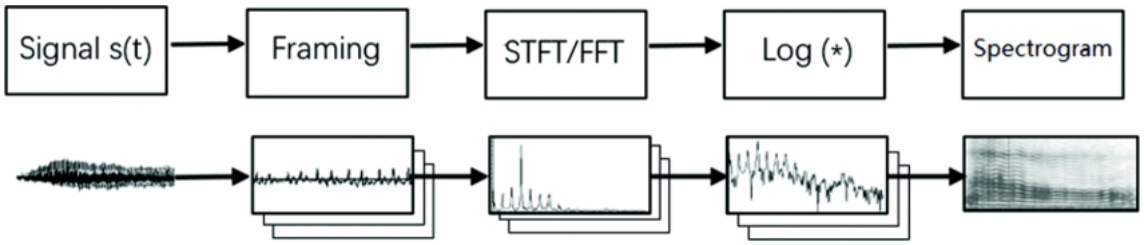

Figure 4: From Signal to Mel-Spectrogram (Ji et al., 2021)

Before the data is fed into the vocoder, the audio must be converted into a mel-spectrogram in the preprocessing stage. The transformation process from audio to mel spectrum can be simply defined in the following steps:

Step 1 The audio data is converted into one-dimensional and normalised vectors.

Step 2 The digital audio signal is divided into frames, i.e., the complete long-time audio data is split into multiple short-time frames. If the length of the tensor is insufficient, it will be padded.

Step 3 The audio data is windowed, i.e., the original signal data multiplied by a window function to reduce spectral leakage, i.e., spectral distortion.

Step 4 The time-frequency features of the audio, i.e., the representation of the audio signal in frequency and time, are obtained by a short-time Fourier transform (STFT).

Step 5 The time-frequency features are processed to obtain the amplitude spectrogram. Compared with the previous time-frequency feature, the amplitude spectrogram removes the phase information but retains the energy information of the time-frequency feature.

Step 6 The amplitude spectrogram is transformed into the mel-spectrogram and normalised again.

All these steps above are the classical process of processing audio-digital signals. It is also the pre-stage of many speech synthesis, speech recognition, and speech enhancement.

3.1.2 Generator

The mel-spectrogram data obtained by preprocessing will be used as input into the generator for synthesising speech.

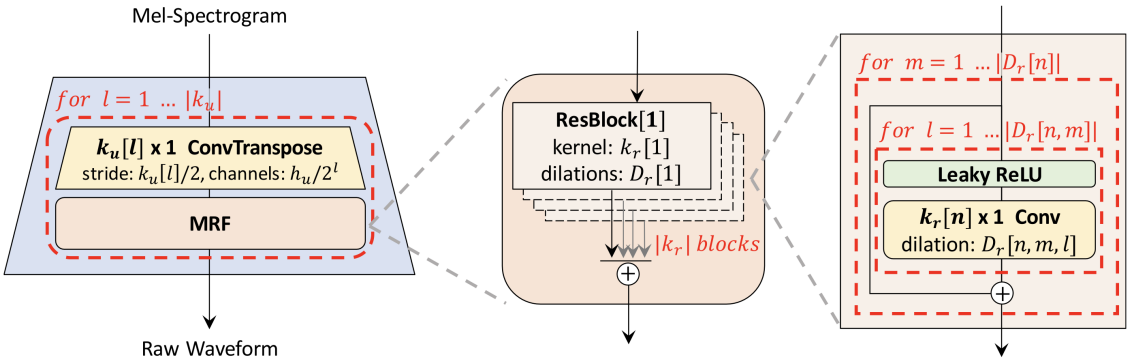

Figure 5: The generator upsamples the mel-spectrogram up to $|k_u|$ times to match the temporal resolution of the original waveform. The Multi-Receptive Field Fusion (MRF) module adds features using residual blocks, which have different sizes and dilation rates; the detailed workflow of the residual blocks is illustrated in the third figure. (Kong et al., 2020)

The mel-spectrogram data will be input to extract the features through a convolutional layer. After upsampling, i.e., inserting new samples into the original signal, the audio data's features are enriched with details. Next, the features are further learned and enhanced by residual blocks. Finally, the one-dimensional synthesised synthetic audio data is obtained after a series of convolutional layers and activation functions processes. The generated and actual audio will be the inputs, entered into a multi-period discriminator (MPD) and a multi-scale discriminator (MSD) to determine. (Kumar et al., 2019)

3.1.3 Multi-Period Discriminato

First, the one-dimensional audio data is converted into two dimensions; specifically, the one-dimensional raw audio of length T is reshaped into two-dimensional data of height T/p and width p. The following is an example of an MPD with a period of 3:

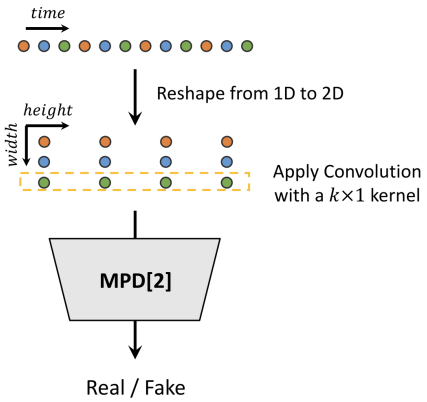

Figure 6: The second sub-discriminator of MPD with period 3 (Kong et al., 2020)

After that, it will be processed with a series of weight normalization, convolution operation, and Leaky ReLU activation functions, and the feature map will be obtained.

3.1.4 Multi-Scale Discriminator

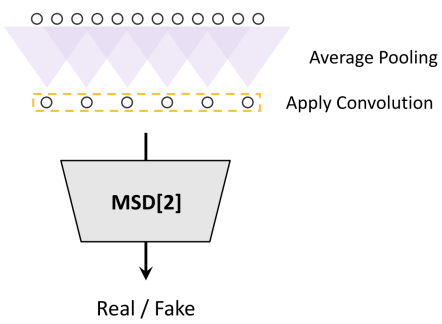

Figure 7: The second sub-discriminator of MSD (Kong et al., 2020)

The three scale discriminators are applied here. The first scale discriminator applies spectral normalisation, and the second and third scale discriminators apply weight normalisation and average pooling to allow the model to focus on higher-level features, improving the model's generalisation ability.

These two discriminators extract the feature maps based on different feature extraction methods to determine whether the audio data belongs to the actual audio. The discrimination results and feature maps will be saved to the corresponding lists, which are used to calculate the final discrimination probability and loss function. Ultimately, the feature map is flattened into a one-dimensional vector, representing the probability that the input audio is actual or generated. A higher value means the audio is more likely to be actual. The discrimination results, and feature maps for actual and synthesised audio are returned.

3.1.5 Loss Function

GAN Loss

GAN Loss for the generator G and the discriminator D are defined as

$$\mathcal{L}_{Adv}(D;G) = \mathbb{E}_{(x,s)}\left[(D(x)-1)^2 + (D(G(s)))^2\right], \tag{4}$$

$$\mathcal{L}_{Adv}(G;D) = \mathbb{E}_{s}\left[(D(G(s))-1)^2\right] \tag{5}$$

, where x denotes the ground truth audio and s denotes the input condition, the mel-spectrogram of the ground truth audio.

Mel-Spectrogram Loss

$$\mathcal{L}_{Mel}(G) = \mathbb{E}_{(x,s)}\left[||\phi(x) - \phi(G(s))||_1\right] \tag{6}$$

Mel-Spectrogram Loss is the L1 distance between the mel-spectrogram of a synthesized waveform synthesized and that of a truth waveform.

Feature Matching Loss

$$\mathcal{L}_{FM}(G;D) = \mathbb{E}_{(x,s)}\left[\sum_{i=1}^{T}\frac{1}{N_i}||D^i(x) - D^i(G(s))||_1\right] \tag{7}$$

, where T denotes the number of layers in the discriminator; D^i and N_i denote the features and the number of features in the i-th layer of the discriminator, respectively. Feature Matching Loss is the L1 distance between the truth sample and the generated sample in each intermediate feature space.

Final Loss

The Final Loss function consists of the GAN Loss, Mel-Spectrogram Loss, and Feature Matching Loss. Therefore, the final objectives for the generator and discriminator are as

$$\mathcal{L}_G = \mathcal{L}_{Adv}(G;D) + \lambda_{fm}\mathcal{L}_{FM}(G;D) + \lambda_{mel}\mathcal{L}_{Mel}(G) \tag{8}$$

$$\mathcal{L}_D = \mathcal{L}_{Adv}(D;G) \tag{9}$$

3.2 WaveGrad

WaveGrad has a similar preprocessing approach to audio as Hi-Fi GAN, which converts the audio data to mel-spectrogram data. Afterward, the gradient and loss are calculated based on the mel-spectrogram. (Chen et al., 2020)

Algorithm 1 Training. WaveGrad directly conditions on the continuous noise level $\sqrt{\bar{\alpha}}$. l is from a predefined noise schedule.

1: **repeat**
2: $y_0 \sim q(y_0)$
3: $s \sim \text{Uniform}(\{1, \dots, S\})$
4: $\sqrt{\bar{\alpha}} \sim \text{Uniform}(l_{s-1}, l_s)$
5: $\epsilon \sim \mathcal{N}(0, I)$
6: Take gradient descent step on $\nabla_\theta \left\| \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}}\, y_0 + \sqrt{1-\bar{\alpha}}\, \epsilon, x, \sqrt{\bar{\alpha}}) \right\|_1$
7: **until** converged

Algorithm 2 Sampling. WaveGrad generates samples following a gradient-based sampler similar to Langevin dynamics.

1: $y_N \sim \mathcal{N}(0, I)$
2: **for** $n = N, \dots, 1$ **do**
3: $z \sim \mathcal{N}(0, I)$
4: $y_{n-1} = \frac{\left(y_n - \frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_n}} \epsilon_\theta(y_n, x, \sqrt{\bar{\alpha}_n})\right)}{\sqrt{\alpha_n}}$
5: if $n > 1$, $y_{n-1} = y_{n-1} + \sigma_n z$
6: **end for**
7: **return** y_0

As seen from the above figure, the specific implementation steps of WaveGrad are only slightly different from the original Diffusion model. In the Diffusion model, x_0 represents the original picture, while x_t represents the picture that has been deconstructed into pure noise. WaveGrad is the same, except that the processing object of the model has been changed from images to speech.

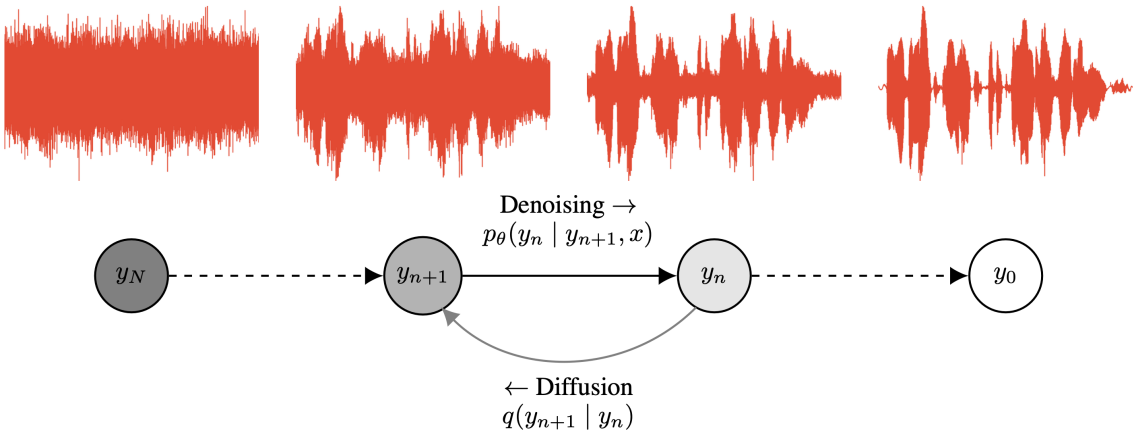

Figure 8: This figure show the diffusion process and the denosing process, where y_0 represents the original audio waveform, y_n is the pure noise, and x is the mel-spectrogram conditioning signal. (Chen et al., 2020)

First, the model is established with a noise schedule, $\beta_1, ..., \beta_N$, $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, $\alpha_n = 1 - \beta_n$, and $\bar{\alpha}_n = \Pi_{s=1}^{n} \alpha_s$. The model generates the noise for the current time step by sampling the noise level at each time step and combining the information from previous time steps. This constructed noise is added to the original audio to produce a new signal. Afterwards, the loss between the noise reconstructed by the model from the diffusion process and the actual noise is calculated.

Repeat the above steps until the audio is entirely changed to pure noise. A new audio signal is generated by subtracting the noise predicted by the model from the pure noise audio signal. Subsequently, the mean and variance of this audio signal are calculated using posterior probability calculation.

The mean and a posteriori variance determine the Gaussian distribution's central location and dispersion. Combining the posterior mean, posterior variance, and random noise, subject to a Gaussian distribution, can generate an estimate of the original audio signal. The steps above are repeated until the original audio signal is reconstructed based on the previous audio estimate.

The model continuously calculates the difference between the predicted audio signal and the actual audio signal through the above two steps and, learns the acoustic characteristics of the audio signal and adjusts the parameters by optimising the loss function so that the reconstructed noise is as close as possible to the actual noise.

Eventually, the model can synthesize the desired audio directly from the pure noise by inverse diffusion.

3.3 Object

This experiment aims to understand the GAN and Diffusion models' performance in the vocoder domain by training the Hi-Fi GAN and the WaveGrad under the same conditions and comparing their differences in various aspects.

4 Research Plan

4.1 Experiment Setup

4.1.1 Dataset

The LJ Speech Dataset is the data source of the two models in this experiment. It is a public domain speech dataset comprising 13,100 short audio clips of a single speaker reading passages from 7 non-fiction books, 12950 data samples as a training set, and the rest as a test set. A transcription is provided for each clip. Clips vary in length from 1 to 10 seconds and have a total length of approximately 24 hours. (Ito and Johnson, n.d.)

4.1.2 Training

	Hi-Fi GAN	WaveGrad
Sampling Rate (Hz)	225050	
Hop Size	300	
Batch Size	16	
Learning Rate	0.0002	0.003

Sampling Rate: the number of samples per second taken from a continuous signal to generate a discrete or digital signal, measured in hertz (Hz) or cycles per second for time-domain signals such as sound waveforms. (Federal Agencies Digitization Guidelines Initiative, n.d.)

Hop Size: the number of samples the frame moves forward in each step in the data sequence. The frame can be conceived as a sliding window with the hop length defining the step length of the movement on the signal. The window performs feature extraction on a new signal part at each step. Thus, hop length determines the degree of overlap between consecutive audio frames. (Agrawal, 2023)

Batch Size: the number of samples in one epoch to train a neural network. (Zvornicanin, 2024) Suppose there are 1000 samples of data and a batch size of 3, which means that the model will process three samples at a time. This process will repeat until all samples have been processed; the number of iterations is approximately 334 times (by dividing 1000 by 3).

Learning Rate: determines how quickly a model learns from the data during training. It controls the size of the step taken in each iteration of the optimization algorithm, affecting the convergence and accuracy of the model. (Dremio, n.d.) Smaller learning rate values may lead to long training time and stagnation. In comparison, larger values may lead to local rather than global optimization, and the training process may be unstable. (Brownlee, 2020)

Since sampling rate, hop size, and batch size can directly affect the quality of the final synthesized speech, I unified the values of these three hyperparameters to ensure the fairness and reliability of the experiment.

4.1.3 Evaluation Metrics

This experiment's results are assessed using various objective indicators, including Short-time objective intelligibility (STOI) (Taal et al., 2011), wideband perceptual evaluation speech quality (WB PESQ) (Rec, 2005), Scale Invariant Source-to-Noise Ratio (SISNR) (Le Roux et al., 2019), and Real-time Factor (RTF) (Li et al., 2022).

STOI is a speech intelligibility metric. It measures the correlation of short-time temporal envelopes between the clean and separated speech, which ranges in [0, 1]. (priyanka, 2023)

PESQ is a speech-perceived quality metric. It measures disturbance between the clean and separated speech using cognitive modeling, which ranges in [-0.5, 4.5]. (Chang and Hung, 2022) Since the sampling rate is greater than 8000 and the audio belongs to broadband audio, using WB PESQ as the evaluation criteria is more appropriate.WB PESQ is an extension of the PESQ algorithm. It is specifically designed to address wideband audio, which has a broader frequency range and higher audio bandwidth than the narrowband audio. (priyanka, 2023)

SISNR is also a speech intelligibility metric.

$$\text{SISNR} = \frac{\text{the clean speech signal ' s energy}}{\text{the energy of the residual noise or interference}} \tag{10}$$

, while it is insensitive to the scaling of the signals. (priyanka, 2023)

RTF is the time of the model operation.

$$\text{RTF} = \frac{\text{the time taken to transcribe the audio}}{\text{the duration of the audio}} \tag{11}$$

(Henderson, 2019) They are all standard objective metrics, and high values indicate better performance except for RFT. Therefore, these metrics can objectively inflect the model's operation efficiency and quality of synthesized speech.

4.2 Result

Results for both models at the same number of training steps:

Model	STOI	WB PESQ	SISNR	RTF
Hi-Fi GAN	0.939	2.507	-29.228	0.002
WaveGrad	0.148	1.045	-46.356	20.469±1.788

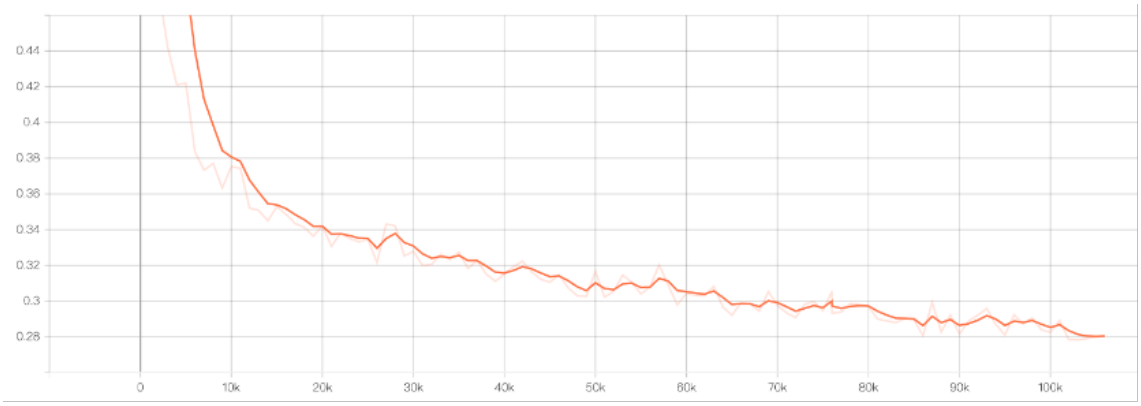

Figure 9: Hi-Fi GAN: mel_spec_error (validation/mel_spec_error)

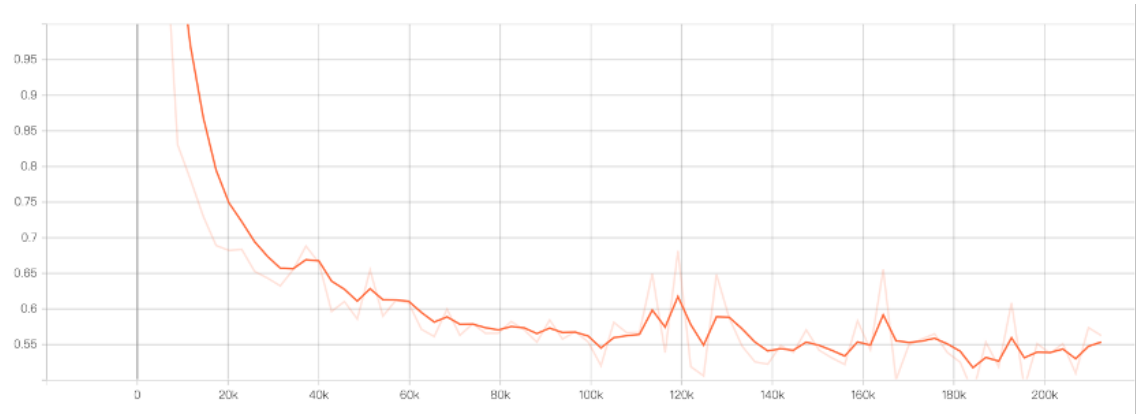

Figure 10: WaveGrad: l1_spec_test_batch_loss (test/l1_spec_test_batch_loss)

5 Diccusion

According to the results, the synthesized speech from Hi-Fi GAN has better voice quality than WaveGrad under the same number of training steps. Therefore, regarding short-term benefits, it is evident that Hi-Fi GAN is more efficient and can synthesize audio that is perceptually almost identical to the original audio in a short period. However, due to the short training time, we cannot determine whether the final synthesized speech of the Hi-Fi GAN model is still better than the speech synthesized by the WaveGrad model over the long training time.

In addition, this experiment only compares the performance of the two representatives of the GAN and Diffusion models in Hi-Fi GAN and WaveGrad in the vocoder field, which implies that the GAN model has a comprehensive and absolute advantage over the Diffusion model in this field because other advanced Diffusion mode vocoders may not be included in this comparison.

However, a problem commonly exists in the GAN model: The Discriminator must be synchronized well with G during training; otherwise, the model may collapse, and this situation is irreversible. Thus, compared to the GAN Model, the Diffusion Model is more stable. (Goodfellow et al., 2014)

6 Conclusion

In conclusion, in the vocoder domain, the GAN model exceeds the Diffusion model in terms of train time and synthesized speech quality. However, the Diffusion model will be more stable than the GAN model. In other words, the GAN model has higher short-term benefits and risks than the Diffusion model.