Diffusion Models for Audios

A Project Report Submitted
in Partial Fulfillment of Requirements
for the Degree of

Bachelor of Technology

by

Sagalpreet Singh (2019CSB1113)
Aman Palariya (2019CSB1068)

Department of Computer Science & Engineering

Indian Institute of Technology Ropar

Rupnagar 140001, India

May 8, 2023

Abstract

Diffusion models have been extensively used for various computer vision tasks such as image denoising, super-resolution, and inpainting. Due to their ability to model complex data distributions, researchers have started exploring their potential for audio generation tasks.

In this work, we propose two novel diffusion models for audio generation: the cross-diffusion and the double-diffusion. Our models are designed to be memory and speed efficient, which makes them suitable for real-time applications.

Cross-diffusion technique allows for conditional generation of high-quality audio without compromising on the output quality. We evaluated the effectiveness of this technique for instrument style transfer and generating background music for a given piece of lyrical vocal audio. The double-diffusion technique, on the other hand, is designed to produce an unconditional paired set of instrumental sounds for chorus generation. In both techniques, models for each instrument can be trained independently unlike the existing conditional diffusion models. We have also analyzed the effect of different hyperparameters on the performance of the model.

The experimental results show that the proposed models are comparable in terms of perceptual quality of the generated audio when compared to the existing models while being significantly simple in terms of architecture and having lower memory and computational requirements.

While cross-diffusion has the potential to be used for various applications like music production and sound design, double-diffusion can offer creative starting points for music producers.

Acknowledgements

We would like to express our deepest gratitude to our advisor, Dr. Abhinav Dhall, for his invaluable guidance, support, and mentorship throughout the course of our thesis. His expertise and insights have been instrumental in shaping our work and we are truly indebted to him for the opportunity to work under his supervision.

We would also like to extend our heartfelt thanks to Gulshan Sharma from LASII Lab, for his continuous support and assistance throughout our research. His guidance and valuable inputs have been instrumental in helping us navigate through the challenges of our research.

We would like to acknowledge the kind support of our institute, IIT Ropar, for providing us access to the High-Performance Computing facility, which has enabled us to perform complex simulations and computations.

We are also grateful to our advisor for providing us access to the GPU from his lab, which has significantly accelerated our work and made our research possible.

Finally, we would like to thank the panel members, Dr. Deepti R. Bathula and Dr. Shashi Shekhar Jha who evaluated our progress and provided us with their valuable feedback and suggestions. Their insights have helped us improve the quality of our work and have enriched our learning experience.

Honor Code

We certify that we have properly cited any material taken from other sources and have obtained permission for any copyrighted material included in this report. We take full responsibility for any code submitted as part of this project and the contents of this report.

Sagalpreet Singh (2019CSB1113)

Aman Palariya (2019CSB1068)

Certificate

It is certified that the B. Tech. project "Diffusion Models for Audios" has been done by Sagalpreet Singh (2019CSB1113), and Aman Palariya (2019CSB1068) under my supervision. This report has been submitted towards partial fulfillment of B. Tech. project requirements.

Dr. Abhinav Dhall
Project Supervisor
Department of Computer Science & Engineering
Indian Institute of Technology Ropar
Rupnagar-140001

Contents

List of Figures

List of Tables

Nomenclature

Roman Symbols

D_{KL}	KL divergence

Greek Symbols

α_t	$1 - \beta_t$
β_t	Variance of noise added at t^{th} step
$\overline{\alpha_t}$	$\prod_{s=0}^{t} \alpha_s$
ε	Standard normal random variable
ε_i	IID standard normal random variables, $i \geq 0$

Other Symbols

$q(x)$	Real data distribution
T	Maximum diffusion time step, $q(x_T) \sim \mathcal{N}(0, 1)$
t	Diffusion time step

Chapter 1

Introduction

1.1 Generative Artificial Intelligence

Generative models have seen a significant boost in recent years due to the advancements in deep learning and artificial intelligence like Generative Adversarial Networks in Goodfellow et al. [2020] and Diffusion Models in Ho et al. [2020]. These models have the ability to produce new and original content by learning from existing datasets and have been used for a wide range of applications, including computer vision, natural language processing, and music generation. Generative models can be classified into two categories: unsupervised and supervised models Bishop and Nasrabadi [2006]; Goodfellow et al. [2016, 2020]; Kingma and Welling [2013]; Kulkarni et al. [2015]; LeCun et al. [2015].

Unsupervised generative models Blei et al. [2017]; Hinton [2002]; Hinton et al. [2006] learn to generate data without any supervision from the training dataset. These models attempt to learn the underlying data distribution by modeling the joint probability distribution of the data. Examples of unsupervised generative models Goodfellow et al. [2020] include Variational Autoencoders (VAEs) Kingma and Welling [2013], Generative Adversarial Networks (GANs), and Autoregressive Models Van den Oord et al. [2016]. These models have been used for various applications, such as image generation, image translation, and image inpainting.

Supervised generative models Gatys et al. [2016]; Mirza and Osindero [2014]; Sohn et al. [2015]; Yu et al. [2017], on the other hand, learn to generate data by conditioning on a specific input or label. These models are trained using supervised learning, where the input data is labeled with a specific output. Examples of supervised generative models include Conditional VAEs Sohn et al. [2015] and Conditional GANs Mirza and Osindero [2014]. These models have been used for tasks such as text-to-speech conversion, style transfer, and image captioning.

One class of generative models that has recently gained a lot of attention in the computer vision community is diffusion models . Diffusion models, also known as denoising diffusion probabilistic models, are a class of generative models that have shown remarkable success in generating high-quality images and videos. These models are based on the idea of iteratively diffusing a noise signal to generate a sample from the target distribution.

Diffusion models perform the generative process by adding noise to the input data and then learning to remove it through a series of diffusion steps. At each diffusion step, the noise level is gradually reduced until the input data is transformed into a sample from the target distribution Ho et al. [2020]. This approach allows diffusion models to capture complex patterns and structures in the data, leading to high-quality and diverse samples.

Diffusion models have achieved remarkable success in computer vision tasks, including image generation, image inpainting, and image denoising Dhariwal and Nichol [2021]. One of the key advantages of diffusion models over other generative models is their ability to generate samples that are not only visually appealing but also exhibit a high degree of diversity. This is particularly useful in scenarios where the goal is to generate a wide range of samples that can capture the variability in the data distribution.

1.2 Audio Generation

The success of diffusion models is not limited to the computer vision domain. Recently, there has been growing interest in applying diffusion models to other domains, including music generation Zhang et al. [2023]. The potential benefits of diffusion models in the audio domain are substantial, given the similarities between the temporal and spectral structures of images and music Dhariwal and Nichol [2021]. By extending the diffusion framework to the audio domain, researchers can generate new and original music compositions.

One application of diffusion models in the audio domain is chorus generation. Chorus generation involves the generation of multiple copies of an audio signal with slight delays and frequency variations. This process can create a richer and fuller sound, enhancing the listening experience for the listener. To generate a chorus, the diffusion model is trained to learn the distribution of a single audio signal and then to add variations to the signal by diffusing noise.

Another application of diffusion models in the audio domain is instrument style transfer. Instrument style transfer involves transforming the characteristics of an audio signal to resemble those of a different instrument. This task requires the model to learn the underlying structure of the audio signal and generate new audio samples that are coherent and natural-sounding. To achieve this, the diffusion model is trained to learn the distribution of a specific instrument's sound and then to transform the input audio signal into the desired instrument's sound.

Despite the potential benefits of diffusion models in the audio domain, there are several challenges that need to be addressed before these models can achieve the same level of success as in the computer vision domain. One of the major challenges is the complexity of the audio signal, which has both temporal and spectral components. This complexity makes it difficult to learn the underlying distribution of the data and generate coherent and natural-sounding audio samples.

To overcome this challenge, researchers have proposed various modifications to the diffusion model architecture, such as incorporating convolutional and recurrent neural networks Mehri et al. [2016]. These modifications allow the model to capture the temporal and spectral components of the audio signal and generate high-quality audio samples.

Another challenge in applying diffusion models to the audio domain is the lack of large-scale audio datasets. Unlike image datasets, which are readily available, audio datasets are often smaller in size and may not contain enough diversity to capture the full range of audio signals. This makes it difficult to train and evaluate diffusion models on audio data.

To address this challenge, researchers have proposed various techniques for augmenting and synthesizing audio datasets Bird et al. [2020]; Wei et al. [2020].

One approach is to use data augmentation techniques, such as pitch shifting and time stretching Raffel and Ellis [2014], to create additional variations of the audio signals. Another approach is to use synthesis techniques, such as additive synthesis and subtractive synthesis, to generate new audio signals that can be used to train and evaluate diffusion models Charles [2008].

Despite these challenges, the potential of diffusion models in the audio domain is immense. With the increasing availability of large-scale audio datasets and the development of more advanced diffusion model architectures, it is expected that diffusion models will become a powerful tool for audio generation tasks in the future.

Chapter 2

Diffusion Models

Diffusion models are a type of stochastic process that describes the dynamics of a random variable over time. Diffusion models are particularly powerful because they can capture complex dependencies and generate high-quality samples by gradually diffusing noise through a sequence of transformations. This chapter aims to provide a comprehensive overview of diffusion models and their applications. We begin with a theoretical foundation of diffusion models, covering probability theory, stochastic processes, and Markov Chain Monte Carlo.

2.1 Overview

Diffusion models operate by applying Gaussian noise incrementally to the input image through a sequence of T steps known as the forward process. Forward diffusion is used to produce the targets for the neural network, namely the image after undergoing $t < T$ noise steps.

Subsequently, a neural network is trained to invert the noise process and recover the original data. By effectively modeling the reverse process, the generative model is able to produce novel data, referred to as the reverse diffusion process or the sampling process.

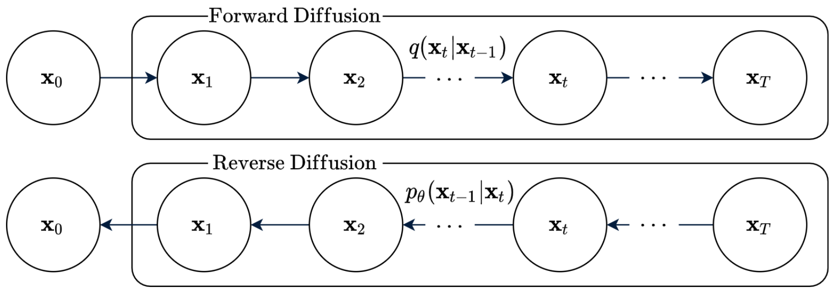

Figure 2.1: Forward and reverse diffusion processes Das [2021]

2.2 Forward Diffusion Process

Diffusion models are latent variable models that operate in a hidden continuous feature space. This feature space is referred to as ”latent” because it is not directly observed. Consequently, diffusion models share similarities with variational autoencoders (VAEs).

Unlike flow-based models, diffusion models utilize a Markov chain consisting of T steps. In this context, a Markov chain refers to a process where each step only depends on the previous one, which is a mild assumption. Additionally, diffusion models are not restricted to specific types of neural networks.

When presented with a data-point x_0 sampled from the real data distribution $q(x)$ (where $x_0 \sim q(x)$), diffusion models define a forward diffusion process by adding noise. At each step of the Markov chain, Gaussian noise with variance β_t is added to the previous latent variable x_{t-1}, resulting in a new latent variable x_t with distribution $q(x_t|x_{t-1})$.

Thus, we have the following equation for the forward diffusion process:

$$q(x_t|x_{t-1}) = \mathcal{N}\left(x_t; x_{t-1}\sqrt{1-\beta_t}, \beta_t I\right) \tag{2.1}$$

It turns out that the forward diffusion process need not be simulated for each time step as there is a closed-form solution for obtaining x_t. We show the derivation for the same:

$$\begin{aligned}
q(x_t|x_{t-1}) &= \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\varepsilon_1 \\
&= \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\varepsilon_1 \\
q(x_t|x_{t-2}) &= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-1}}\varepsilon_2\right) + \sqrt{1-\alpha_t}\varepsilon_1 \\
&= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1})\varepsilon_2} + \sqrt{1-\alpha_t}\varepsilon_1 \\
&= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\varepsilon \\
&\vdots
\end{aligned}$$

Thus, we obtain the closed-form solution for obtaining the diffused image x_t directly from x_0.

$$q(x_t|x_0) = \sqrt{\overline{\alpha_t}}x_0 + \sqrt{1-\overline{\alpha_t}}\varepsilon \tag{2.2}$$

2.3 Reverse Diffusion Process

As T approaches ∞, the distribution of the latent variable x_T becomes almost isotropic Gaussian. This means that if we are able to learn the reverse distribution $q(x_{t-1}|x_t)$, we can sample x_T from $\mathcal{N}(0, I)$, run the reverse process, and obtain a sample from $q(x_0)$, which generates a new data point from the original data distribution. The challenge, however, is how to model the reverse diffusion process.

It turns out that computing $q(x_{t-1}|x_t)$ is intractable. Thus, we approximate $q(x_{t-1}|x_t)$ with a parameterized model p_θ which could be a neural network. $q(x_{t-1}|x_t)$ will also be a Gaussian distribution. Thus, for small enough β_t, we can parameterize p_θ by its mean and variance as follows:

$$p_\theta\left(x_{t-1}|x_t\right) = \mathcal{N}\left(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta\right)$$

Since the goal is to have x_0 belong to the underlying distribution, we can maximize the log likelihood and define the loss function as $-\log(p_\theta(x_0))$

The likelihood is intractable since we cannot directly evaluate the distribution $q(x_t|x_{t-1})$ for each t. Instead, we can use a tractable approximation to the likelihood, which is the evidence lower bound (ELBO) obtained using a variational autoencoder (VAE) framework. The derivation for the same can be referred in Appendix A.1. The derivation leads to the following loss function and denoising step:

$$L_t = \|\varepsilon - \varepsilon_\theta(x_t, t)\|^2 \tag{2.3}$$

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\varepsilon_\theta(x_t, t)\right) + \sqrt{\beta_t}\varepsilon \tag{2.4}$$

Ho et al. [2020] proposed to keep the variance fixed and train the neural network to only learn the mean in diffusion models. This approach was further enhanced by Dhariwal and Nichol [2021] by allowing the network to learn the covariance matrix as well, leading to better results. The reason behind this improvement is that modeling the covariance matrix can capture more complex relationships between the data and the diffusion process. By allowing the network to learn the full covariance matrix, the model can better capture the true distribution of the data, leading to higher quality samples.

In our discussion of the theoretical setting of diffusion models and the proofs, we have restricted ourselves to the simple approach of keeping the variance fixed. However, the generalization follows very naturally. The pseudo code for learning process can be referred from Algorithm 1. The pseudo code for sampling process can be referred from Algorithm 2.

Algorithm 1 Training

```
1: repeat
2:     $x_0 \sim q(x_0)$
3:     $t \sim \mathcal{U}(\{1, \dots, T\})$
4:     $\varepsilon \sim \mathcal{N}(0, I)$
5:     Take gradient step $\nabla_\theta \left\|\varepsilon - \varepsilon_\theta\left(\sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\varepsilon, t\right)\right\|^2$
6: until converged
```

Algorithm 2 Sampling

```
1: $x_T \sim \mathcal{N}(0, I)$
2: for $t = T, \dots, 1$ do
3:     if $t > 1$ then
4:         $z \sim \mathcal{N}(0, I)$
5:     else
6:         $z = 0$
7:     end if
8:     $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t, t)\right) + \sigma_t z$
9: end for
10: return $x_0$
```

2.4 Network Architecture

Although, any neural network architecture can be used for approximating the distribution parameters, we present an overview of the architectures proposed in literature that have been found to work well while laying special emphasis on the distinctive features of these networks.

The first thing to notice is that the input and output dimensions of the network must be of the same size. Ho et al. [2020] utilized a U-Net architecture Ronneberger et al. [2015] to accomplish this goal. A U-Net is a symmetric architecture with input and output of the same spatial size that employs skip connections between encoder and decoder blocks of corresponding feature dimension. Typically, the input image is initially downsampled and then upsampled until it returns to its original size.

In diffusion models, the diffusion timestep t specifies how much Gaussian noise is added to the input image in the forward process. To capture this positional information, a sinusoidal position embedding is added to one or multiple residual blocks of the neural network. The sinusoidal position embedding is a technique commonly used in modeling spatial/temporal relationships in the data Vaswani et al. [2017]. It involves adding sinusoidal functions of different frequencies and phases to the input data. These sinusoids can capture different patterns in the data at different spatial scales, allowing the network to learn more robust representations. By adding a sinusoidal position embedding to a residual block in

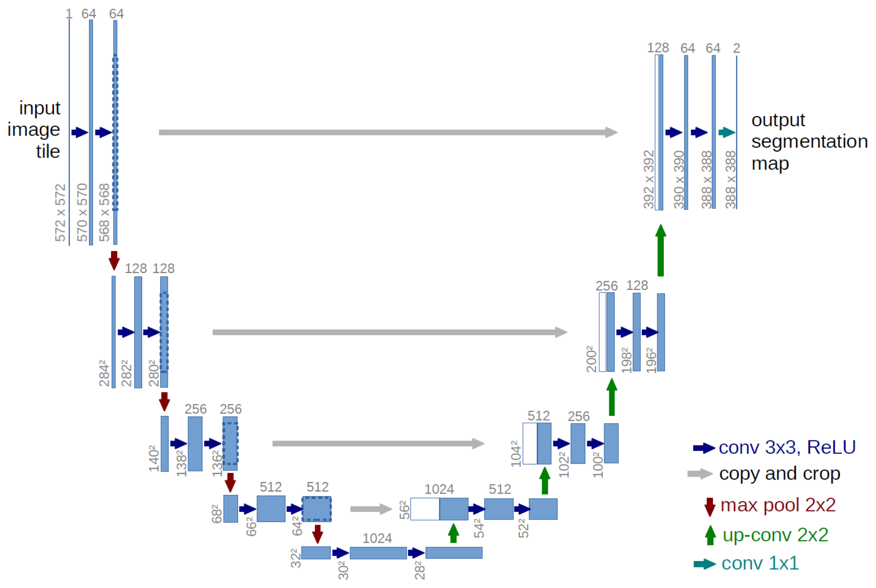

Figure 2.2: A U-Net architecture that takes a single-channel 572×572 image and outputs a dual-channel 388×388 image Ronneberger et al. [2015]

the diffusion model, the network can learn to capture the timestep representation and use it in predicting the noise.

2.5 Conditional Diffusion Models

Conditioning the sampling process is a vital factor in image generation, also known as guided diffusion. Some methods have incorporated image embeddings into diffusion to manipulate the generation process Dhariwal and Nichol [2021]; Sohn et al. [2015]. The guidance concept mathematically refers to conditioning the prior data distribution p(x) with a condition y, such as class labels or an image/text embedding, leading to the distribution $p(x|y)$. To make a diffusion model p_θ a conditional diffusion model, we can introduce conditioning information y at each diffusion step as follows:

$$p_\theta(x_{0:T}|y) = p_\theta(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t, y) \tag{2.5}$$

Chapter 3

Audio – Representation and Processing

Humans primarily communicate through verbal interactions – by speaking and listening. Sound is the phenomenon that makes all of this possible. In this chapter, we will understand the basics of sound and digital audio.

3.1 Acoustics

Sound is a form of energy that travels in the form of waves. Sound, being a mechanical wave, cannot travel without matter. Sound waves are created when the source vibrates causing changes in the air pressure. This pressure propagates through the medium carrying the wave from one place to another. Human ears can sense these vibrations and convert them into electrical signals that the brain interprets as sound.

3.1.1 Characteristics of Sound Waves

A sound wave has four characteristics

- **Velocity**: It is the speed and direction in which the sound wave is propagating. The speed of a sound wave is determined by the medium in which it travels. For example, the speed of sound in dry air at room temperature is 343 meters per second.

- **Frequency**: It is the number of complete cycles of pressure changes that occur in unit time. Frequency is measured in Hertz (Hz). When expressed in Hertz, the frequency is the number of pressure change cycles completed in one second by a sound wave. Humans can hear sounds in the frequency range of 20 Hz to 20K Hz Frequency of a sound wave is perceived by the human ear as the pitch. A high-frequency sound has a higher pitch and sounds shrill while a lower-frequency sound is softer.

- **Wavelength**: Wavelength is the distance between two successive peaks or troughs in a sound wave. It is related to the frequency and the speed of the sound. If v is the speed, f is the frequency, and λ is the wavelength, then the relation between these is given by eq. (3.1).

$$v = f \times \lambda \tag{3.1}$$

- **Amplitude**: Amplitude is the magnitude of pressure changes in a sound wave and determines the loudness of the sound. The larger the amplitude, the louder the sound. Amplitude is usually measured in decibels (dB), which is a logarithmic unit that compares the pressure of a sound wave to a reference pressure. The higher the amplitude of a sound wave, the louder it is perceived to be.

3.1.2 Fourier Transform

Fourier transform is a fundamental mathematical technique that is used to break a signal into simpler components. The Fourier transform works by decomposing a signal into a series of sine and cosine waves, each with a different frequency and amplitude. This is done by representing the signal as a sum of complex exponential functions, which are then analyzed using Fourier analysis. In practical terms, the Fourier transform allows us to analyze the frequency content of a signal and identify the individual components that make it up. For example, in the context of sound waves, the Fourier transform can be used to identify the fundamental frequency of a musical note and the harmonics that contribute to its timbre.

3.1.3 Spectrogram

A spectrogram is a visual representation of the frequency content of a signal over time. It is a type of frequency-domain representation that displays the distribution of frequency components of a signal with respect to time. Spectrogram is obtained by breaking a sound wave into its sinusoidal components and then plotting the magnitude of each component with time.

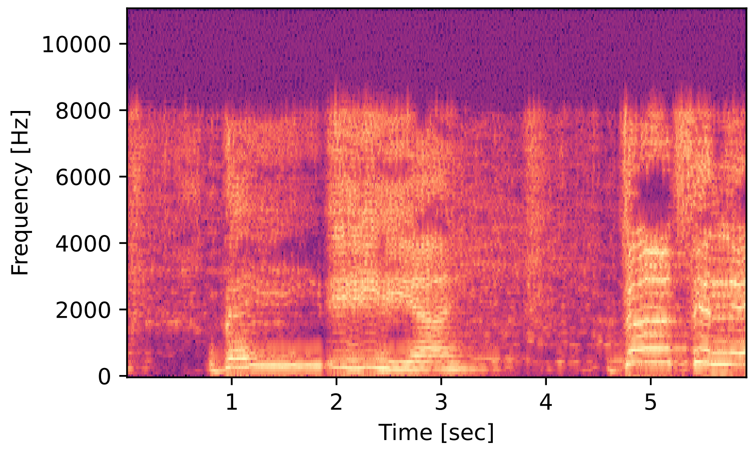

Figure 3.1: An example spectrogram showing the time on x-axis and frequencies on y-axis

Spectrograms are commonly used in the field of audio processing to analyze and visualize the frequency content of sound waves. They are particularly useful for identifying patterns and features in sound that are difficult to detect by ear alone.

3.2 Digital Audio

Digital audio is the representation of sound in a digital format which can be stored and manipulated using computers. In contrast to analog audio, which represents sound as continuous waveforms, digital audio breaks down sound into discrete samples. To convert an analog signal to digital, analog-to-digital converter (ADC) is used. Similarly, digital-to-analog converter (DAC) is used when the digital signal has to be converted to analog.

In modern computer systems, digital audio is typically represented using pulse code modulation (PCM), which works by sampling the analog signal at regular intervals and quantizing the amplitude of each sample into a digital value. The resulting digital values are then stored as binary data, typically using 16 or 24 bits per sample. The sampling frequency (also called sample rate) is usually dictation by Nyquist criteria.

3.2.1 Fast Fourier Transform (FFT)

Fast Fourier Transform (FFT) is an efficient algorithm that calculates the Fouier transform of a discrete signal such as digital audio. This is achieved by breaking the signal down into smaller parts and calculating the Fourier Transform of each part separately, then recombining the results. In practice, the FFT is much faster than the FT and is frequently used in digital signal processing applications.

3.2.2 Digital Spectrogram

There are different algorithms available for spectrogram generation from audio. Constant Q-Transform (CQT) spectrogram and Mel spectrogram are popular choices for converting audio to spectrogram. To turn a spectrogram back into audio, the inverse Fourier transform is used to convert the frequency-domain representation back into the time domain. This process is known as synthesis and requires some additional steps, such as windowing and overlap-add processing, to avoid artifacts in the resulting audio signal.

3.3 Music

Music is an audio signal that is composed of a mixture of sound sources, including vocals and instruments. The audio signal can be analyzed using techniques such as Fourier analysis to decompose it into its constituent frequencies.

Vocals in a song typically consist of a sequence of phonemes, which are the individual sounds that make up words. These phonemes are produced by the vocal cords, and their frequencies are typically concentrated in the range of 100 Hz to 1 kHz. The sound of a voice can be modified by changing the shape of the mouth and the position of the tongue and lips, which alters the resonant frequencies of the vocal tract.

Instruments in a song can be broadly categorized as either pitched or unpitched. Pitched instruments produce notes that have a clear frequency, while unpitched instruments produce sounds that do not have a clear pitch. Pitched instruments, such as guitars or pianos, produce a sequence of notes that can be analyzed using Fourier analysis to determine the fundamental frequency of each note.

The overall composition of a song is determined by the arrangement of the vocals and instruments, which can be manipulated in various ways to create different effects. For example, vocals can be harmonized with each other or with the instruments to create a sense of unity, while instruments can be played in unison or in counterpoint to create a sense of tension or complexity. The use of effects such as reverb or distortion can also dramatically alter the sound of a song.

Chapter 4

Methodology

4.1 Preparing Dataset

In order to use diffusion models for audio, the first step is to prepare the dataset. For this purpose, long audios ($\sim$10 hours) of each instrument, violin and guitar, were taken from the internet. These audios were then split into smaller chunks of 5 seconds each to facilitate processing.

Next, each of the shorter audio chunks was converted to its spectrogram representation. Spectrograms are a visual representation of the frequency content of a signal as it changes over time. In other words, they show how much of each frequency is present in the audio signal at different points in time.

To create a spectrogram, we use a mathematical technique called the Fourier transform to convert the audio signal from the time domain to the frequency domain. The resulting spectrum is then divided into small sections called "bins," and the magnitude of each bin is plotted over time to create a two-dimensional image. The resulting image is a representation of the audio signal that allows us to visualize its frequency content over time.

Once the audio chunks have been converted to spectrogram representations, they can be used to train diffusion models for audio. These models can be used to generate novel audio samples that are similar to the original audio data, but with added variations and modifications.

4.2 Diffusion Framework

In this section, we describe the network architecture, variance schedule, learning rate schedule and loss function used for training the diffusion model.

4.2.1 Network Architecture

The network consists of a type of two-dimensional encoder-decoder architecture. The encoder part of the network consists of 2D down blocks while the decoder part consists of 2D up blocks. Each down block consists of residual network layers and a downsampling layer whereas each up block consists of residual network layers and an upsampling layer Smith [2023]. Here, downsampling is done by using a 2D convolution layer and upsampling is achieved by interpolation and 2D convolution.

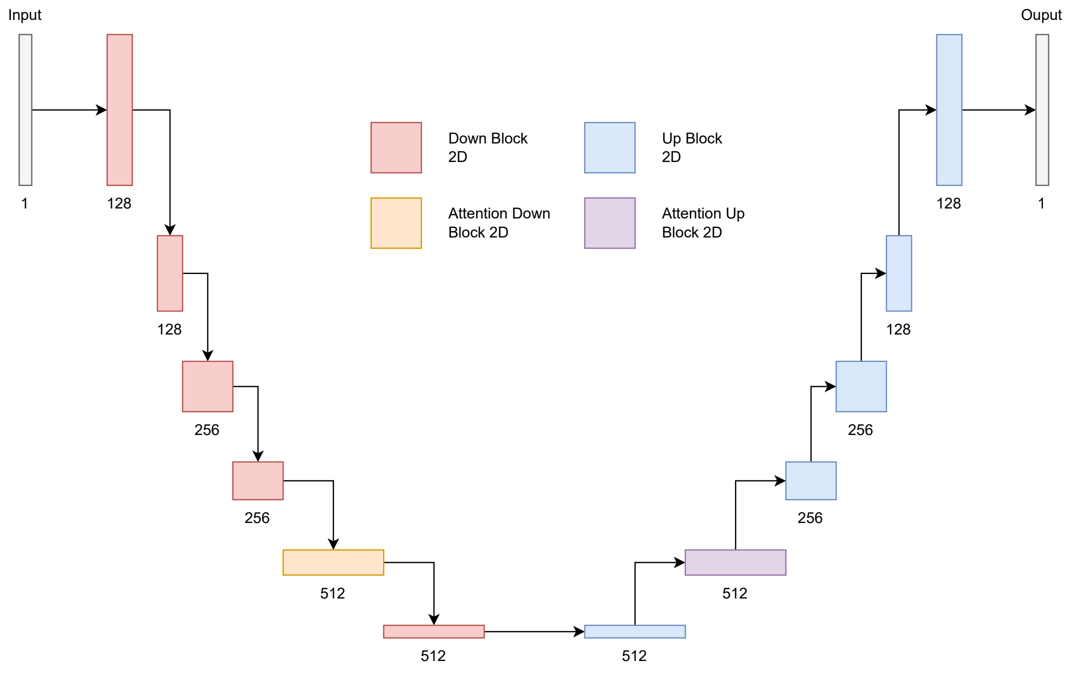

Figure 4.1: High-level U-Net architecture used for the diffusion model, showing the type of layers and the number of channels

4.2.2 Variance Schedule

The DDIM schedule Song and Ermon [2020] is used as the variance schedule to reduce the effect of added noise on the generated samples over the course of training. To achieve this, the noise level is reduced using a cosine annealing schedule. The cosine annealing schedule gradually reduces the standard deviation of the noise to zero over the course of the training. This allows the model to learn more effectively at the beginning of the training, when the noise level is high, and generate more accurate and diverse samples as the noise level decreases.

4.2.3 Learning Rate Schedule

The cosine learning rate schedule Vaswani et al. [2017] with warm-up is used which consists of two parts: a warmup period and a cosine annealing period. During the warmup period, the learning rate is gradually increased from a small value to the base learning rate. This helps to stabilize the training process and prevent the model from getting stuck in a poor local minimum at the beginning of training. The warmup period usually lasts for a few hundred to a few thousand iterations. After the warmup period, the learning rate is gradually reduced over time using a cosine annealing schedule. The cosine annealing schedule is a function that smoothly decreases the learning rate to a small value over a fixed number of iterations.

4.2.4 Loss Function

We use L1 loss for training which is also known as the mean absolute error (MAE). It is calculated by taking the absolute differences between the predicted and actual values, and then taking the mean of those differences. Mathematically,

$$\text{L1 loss} = \frac{1}{n} \times \sum |y_i - \hat{y}_i|$$

The L1 loss function is less sensitive to outliers than the L2 loss function, which is the mean squared error (MSE). This is because the absolute differences between the predicted and actual values are not squared, and therefore outliers do not have as much of an impact on the overall loss.

4.3 Proposed Work

4.3.1 Cross Diffusion

We first propose cross diffusion model. Here, we first train diffusion models independently for each of the instruments: guitar and violin. Let these models be called G_{guitar} and G_{violin} respectively.

Now, in the denoising process from diffusion model G, let the output from i^{th} denoising step be denoted as $G^t(Y_{t+1})$, where Y_{t+1} denotes the output from the previous denoising step.

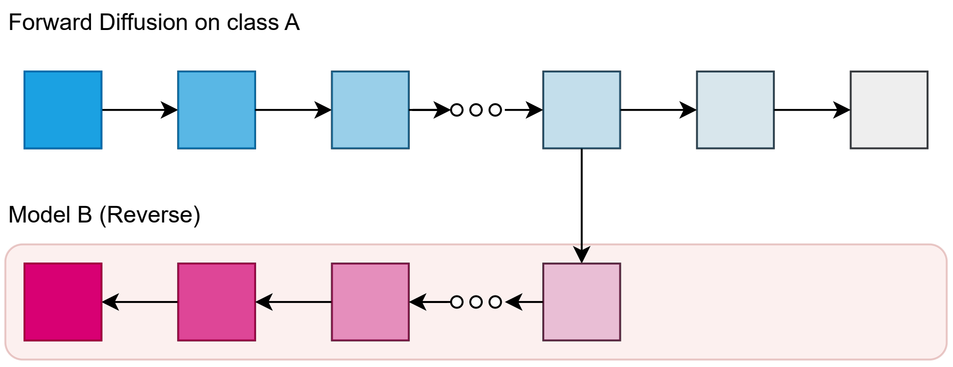

Figure 4.2: Data flow in Cross Diffusion where an instance of class A is diffused and an intermediate representation is denoised using model B

We hypothesize that denoising using a model G_i preserves salient features of the partially noised version. Based on this we propose that for applying style transfer from guitar to violin (could be any pair A to B in general), we can denoise a diffused version of the guitar sample using G_{violin}. Let us denote the diffused input X upto timestep t as $D_t(X)$, then mathematically:

$$X_B = G_B^0 \left(G_B^1 \left(\dots G_B^t \left(D_t(X_A) \right) \right) \right) \tag{4.1}$$

In our experiments, we observed that this method works well for style transfer from vocals to instruments as well, opening up a potential application for generating accompanying music given a vocal sound track.

Since, we train models for each class independently, this means that the models generated for the purpose of unconditional generation can be used for conditional generation. Moreover, we do not need to train diffusion model for the source class as is evident from eq. (4.1) which only uses the model of target class.

4.3.2 Double Diffusion

We now propose double diffusion model which is based on the same hypothesis as cross-diffusion model.

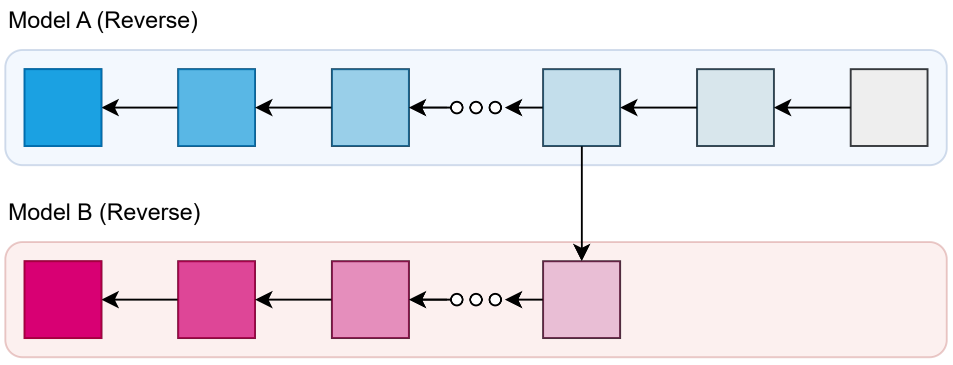

Figure 4.3: Data flow in Double Diffusion where model A generates a new output from noise and an intermediate representation also denoised using model B

Here, during the unconditional generation from a diffusion model trained on class A, we take an intermediate partially denoised version and denoise it through diffusion models corresponding to each of the class. This way, we have generated a pair of related audios.

$$X_A = G_A^0\left(G_A^1\left(\dots G_A^t\left(D_T(z)\right)\right)\right) \tag{4.2}$$

$$X_B = G_B^0\left(G_B^1\left(\dots G_B^t\left(Y_A^{t+1}\right)\right)\right) \tag{4.3}$$

Since, we have a pair of generated audios with same underlying pattern, this leads to a potential application of chorus generation.

In this case also, we train models for each class independently so the models generated for the purpose of unconditional generation can be used for conditional generation. However, we do require to train diffusion model for all the classes that we want to have output for as is evident from eqs. (4.2) and (4.3) which uses the model of both the classes, unlike cross diffusion model.

Chapter 5

Experiments and Results

5.1 Dataset and Evaluation Metric

For dataset, long audios of guitar and violin music were scrapped from the internet which were then split into segments of 5 seconds.

The Fréchet audio distance (FAD) Kilgour et al. [2019] is a measure of similarity between two audio signals, based on the concept of Fréchet distance Alt and Godau [1995], which is a metric for comparing curves or paths in multi-dimensional spaces. FAD is often used in audio processing applications such as music analysis, sound recognition, and speech recognition.

Technically, FAD is calculated as the minimum distance between two continuous paths in a multi-dimensional space, where each dimension represents a different acoustic feature of the audio signal.

To calculate the FAD between two audio signals, we first extract their feature vectors and align them in time. Then, we calculate the distance between each pair of feature vectors using a suitable distance metric, such as Euclidean distance. We then compute the Fréchet distance between the two feature vectors using an efficient algorithm such as the algorithm proposed by Alt and Godau [1995]. The resulting Fréchet distance is a measure of similarity between the two audio signals, with lower values indicating greater similarity.

FAD has several advantages over other distance metrics such as dynamic time warping (DTW) Keogh and Ratanamahatana [2005] and Euclidean distance.

FAD is more robust to noise and distortion in the audio signal and can handle variations in speed and tempo. Moreover, FAD is computationally efficient and can be computed in polynomial time.

5.2 Cross Diffusion & Double Diffusion

We trained two unconditioned diffusion models – one on guitar and another one on violin. Guitar to violin and violin to guitar style transfer was performed using cross diffusion. In both experiments, first a few samples from the source instrument were diffused to different degrees – no diffusion, 100, 200, 300, 400 steps out of 500 steps and complete 500 steps and then these were denoised using the diffusion model of the target instrument. Because the input to the target is partially denoised, so different denoising steps (out of total 50 steps) were tried.

The tables 5.1 and 5.2 contain the Frećhet audio distances. The rows and columns represent the number of forward diffusion steps on source audio (out of 500) and the starting denoising steps (out of 50) respectively.

	0	**10**	**20**	**30**	**40**
0	0.002	0.003	0.005	0.003	0.002
100	0.001	0.001	0.002	0.003	0.002
200	0.002	0.001	0.002	0.002	0.002
300	0.002	0.001	0.002	0.002	0.002
400	0.002	0.002	0.004	0.003	0.003
500	0.017	0.003	0.002	0.002	0.004

Table 5.1: Frećhet audio distances for guitar to violin cross-diffusion using VGGish model

	0	**10**	**20**	**30**	**40**
0	0.001	0.000	0.001	0.000	0.003
100	0.001	0.000	0.000	0.000	0.003
200	0.001	0.001	0.001	0.001	0.001
300	0.001	0.001	0.000	0.001	0.001
400	0.001	0.001	0.001	0.001	0.000
500	0.000	0.000	0.000	0.000	0.000

Table 5.2: Frećhet audio distances for violin to guitar cross-diffusion using VGGish model

From the tables, it is clearly evident that the generation quality is rather good as it lies well in the distribution of the corresponding instrument.

Following this, an end-to-end background music generation pipeline was made. This takes a vocal sound track as input and generates violin and guitar music corresponding to it. These tracks are cleaned by removing outlying high and low frequency components. The resulting tracks are normalized to keep amplitude levels the same for all the audio tracks post which point-wise addition of these tracks to vocal sound track is done, leading to the final song.

Having done analysis of how the performance of the model is affected by the number of forward diffusion steps and the denoising steps in case of cross-diffusion, we directly use 10 as the number of denoising split steps for double diffusion.

Specifically, for our experiments, we generate paired piece of guitar and violin music. These violin and guitar outputs are cleaned by filtering out very high and low frequency components. The perceptual quality for these pairs was found to be encouraging.

Chapter 6

Conclusions

In this work, we have proposed two novel diffusion models, cross-diffusion and double-diffusion, for audio generation tasks. We show that these models are memory and speed efficient, making them suitable for real-time applications. The cross-diffusion technique allows for conditional generation of high-quality audio without compromising on the output quality. We evaluate the effectiveness of this technique for instrument style transfer and generating background music for a given piece of lyrical vocal audio. The double-diffusion technique, on the other hand, is designed to produce an unconditional paired set of instrumental sounds for chorus generation.

The experimental results demonstrated that the proposed models are comparable in terms of perceptual quality of the generated audio when compared to the existing models while being significantly simple in terms of architecture and having lower memory and computational requirements. This suggests that diffusion models can be a promising approach for audio generation tasks.

We also analyzed the effect of different hyperparameters on the performance of the model, and show that the number of forward diffusion steps and the denoising steps have a very slight impact on the performance of the model. We find that the perceptual quality of the generated audio is encouraging for both cross-diffusion and double-diffusion models.

Overall, these results suggest that the proposed diffusion models are promising for various audio generation applications such as music production, sound design, and chorus generation. The ability to generate high-quality audio in real-time opens up new possibilities for interactive music and sound-based applications.

Appendix A

1. Deriving loss function for Diffusion Models

$$\begin{aligned}
-\log\left(p_\theta(x_0)\right) &\leq -\log\left(p_\theta(x_0)\right) + D_{KL}\left(q\left(x_{1:T}|x_0\right)||p_\theta(x_{1:T}|x_0)\right) \\
&\leq -\log\left(p_\theta(x_0)\right) + \log\left(\frac{q\left(x_{1:T}|x_0\right)}{p_\theta(x_{1:T}|x_0)}\right) \\
&\leq -\log\left(p_\theta(x_0)\right) + \log\left(\frac{q\left(x_{1:T}|x_0\right)}{\frac{p_\theta(x_0|x_{1:T})p_\theta(x_{1:T})}{p_\theta(x_0)}}\right) \\
&\leq -\log\left(p_\theta(x_0)\right) + \log\left(\frac{q\left(x_{1:T}|x_0\right)}{\frac{p_\theta(x_0,x_{1:T})}{p_\theta(x_0)}}\right) \\
&\leq -\log\left(p_\theta(x_0)\right) + \log\left(\frac{q\left(x_{1:T}|x_0\right)}{\frac{p_\theta(x_{0:T})}{p_\theta(x_0)}}\right) \\
&\leq -\log\left(p_\theta(x_0)\right) + \log\left(\frac{q\left(x_{1:T}|x_0\right)}{p_\theta(x_{0:T})}\right) + \log\left(p_\theta(x_0)\right) \\
&\leq \log\left(\frac{q\left(x_{1:T}|x_0\right)}{p_\theta(x_{0:T})}\right) \\
&\leq \log\left(\frac{\prod_{t=1}^{T} q(x_t|x_{t-1})}{p_\theta(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)}\right) \\
&\leq -\log\left(p_\theta(x_T)\right) + \log\left(\frac{\prod_{t=1}^{T} q(x_t|x_{t-1})}{\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)}\right)
\end{aligned}$$

$$
\begin{aligned}
-\log\left(p_\theta(x_0)\right) &\leq -\log\left(p_\theta(x_T)\right) + \sum_{t=1}^{T} \log\left(\frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)}\right) \\
&\leq -\log\left(p_\theta(x_T)\right) + \sum_{t=2}^{T} \log\left(\frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)}\right) + \log\left(\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right) \\
&\leq -\log\left(p_\theta(x_T)\right) + \sum_{t=2}^{T} \log\left(\frac{q(x_{t-1}|x_t, x_0)q(x_t|x_0)}{p_\theta(x_{t-1}|x_t)q(x_{t-1}|x_0)}\right) + \log\left(\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right) \\
&\leq -\log\left(p_\theta(x_T)\right) + \sum_{t=2}^{T} \log\left(\frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)}\right) + \sum_{t=2}^{T} \log\left(\frac{q(x_t|x_0)}{q(x_{t-1}|x_0)}\right) \\
&\quad + \log\left(\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right)
\end{aligned}
$$

The term $\log\left(\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right)$ is constant and can be ignored. The term $\sum_{t=2}^{T} \log\left(\frac{q(x_t|x_0)}{q(x_{t-1}|x_0)}\right)$ forms a telescopic series, leading to a modified loss function as follows:

$$
\begin{aligned}
\mathcal{L}_t &= \sum_{t=2}^{T} \log\left(\frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)}\right) - \log\left(p_\theta(x_0|x_1)\right) \\
&= \sum_{t=2}^{T} D_{KL}\left(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)\right) - \log\left(p_\theta(x_0|x_1)\right)
\end{aligned}
$$

Note that:

$$
\begin{aligned}
q(x_{t-1}|x_t, x_0) &\sim \mathcal{N}\left(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I\right) \\
p_\theta(x_{t-1}|x_t) &\sim \mathcal{N}\left(x_{t-1}; \mu_\theta(x_t, t), \beta I\right)
\end{aligned}
$$

Here, we are trying to estimate the distributions $\tilde{\mu}_t(x_t, x_0)$ as $\mu_\theta(x_t, t)$. We have,

$$\tilde{\mu}_t\left(x_t, x_0\right) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta}{\sqrt{1-\overline{\alpha_t}}}\varepsilon\right)$$
$$\tilde{\beta}_t = \frac{1-\overline{\alpha_{t-1}}}{1-\overline{\alpha_t}}\beta_t$$

The loss function can thus be expressed as follows since, the objective is to make the two distributions as close to each other as possible.

$$L_t = \frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha_t}}}\varepsilon\right) - \mu_\theta(x_t,t)\right\|^2$$

To simplify the expression, we can reparameterize $\mu_\theta(x_t,t)$ as:

$$\mu_\theta(x_t,t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha_t}}}\varepsilon_\theta(x_t,t)\right)$$

Hence, we have a simplified loss function, given as:

$$L_t = \frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\overline{\alpha_t})}\left\|\varepsilon - \varepsilon_\theta(x_t,t)\right\|^2$$

We can safely drop off the constant. This leads to the final loss function:

$$L_t = \left\|\varepsilon - \varepsilon_\theta(x_t,t)\right\|^2$$

And, the denoising step can be expressed mathematically as follows:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha_t}}} \varepsilon_\theta(x_t, t) \right) + \sqrt{\beta_t} \varepsilon$$

Appendix B

Work done in CP302

During this project, we delved into the theoretical framework of diffusion models and familiarized ourselves with the current state-of-the-art models in generative AI. We explored the capabilities of diffusion models in computer vision tasks by training on the Oxford flower dataset Nilsback and Zisserman [2006] and generating images. Through experimentation, we were able to generate high-quality images by leveraging the power of diffusion models.

Additionally, we looked into conditioned diffusion models and how they can be utilized to generate images after being trained on the MNIST-digit dataset LeCun et al. [1998]. This allowed us to explore the concept of conditioning in generative AI and how it can be used to manipulate the generated samples.

As we further explored the audio domain, we came across work on instrument and vocal separation using Wave-U-Net Stoller et al. [2018]. We were fascinated by the potential of these models to separate individual components of an audio track, allowing for more precise editing and manipulation.

Furthering our exploration of audio-related applications, we studied CycleGAN-VC Kaneko and Kameoka [2018], a technique for end-to-end style transfer on speech audio. We successfully applied this technique to Mann Ki Baat, a popular Indian radio program, allowing us to transform the speech style of the program while preserving the content.

Throughout this project, we have gained a deeper understanding of the theoretical underpinnings of diffusion models and their potential applications in computer vision and audio-related tasks. We have also explored various techniques for conditioning and style transfer, and the potential impact they could have in the field of generative AI. Our experimentation and exploration of these techniques have given us a valuable insight into the future potential of generative AI models.

The work done in CP302 helped us in understanding diffusion models and audio, which further accelerated our work in CP303.

References

Helmut Alt and Michael Godau. Computing the fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(01n02):75–91, 1995. 22

Jordan J Bird, Diego R Faria, Cristiano Premebida, Anikó Ekárt, and Pedro PS Ayrosa. Overcoming data scarcity in speaker identification: Dataset augmentation with synthetic mfccs via character-level rnn. In *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 146–151. IEEE, 2020. 3

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006. 1

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112 (518):859–877, 2017. 1

Jean-François Charles. A tutorial on spectral sound processing using max/msp and jitter. *Computer Music Journal*, 32(3):87–102, 2008. 4

Ayan Das. An introduction to Diffusion Probabilistic Models — ayandas.me. `https://ayandas.me/blog-tut/2021/12/04/diffusion-prob-models.html`, 2021. vii, 6

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper_files/paper/2021/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf`. 2, 8, 10

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016. 1

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. 1

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. 1

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002. 1

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006. 1

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 1, 2, 8, 9

Takuhiro Kaneko and Hirokazu Kameoka. Cyclegan-vc: Non-parallel voice conversion using cycle-consistent adversarial networks. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 2100–2104. IEEE, 2018. 32

Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7:358–386, 2005. 22

Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms. In *INTERSPEECH*, pages 2350–2354, 2019. 22

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1

Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. *Advances in neural information processing systems*, 28, 2015. 1

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324, 1998. 32

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521 (7553):436–444, 2015. 1

Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model, 2016. URL `https://arxiv.org/abs/1612.07837`. 3

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 1, 2

Maria-Elena Nilsback and Andrew Zisserman. A visual vocabulary for flower classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1447–1454, 2006. 32

Colin Raffel and Daniel PW Ellis. Intuitive analysis, creation and manipulation of midi data with pretty_midi. In *15th international society for music information retrieval conference late breaking and demo papers*, pages 84–93, 2014. 4

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. vii, 9, 10

Robert Dargavel Smith. Audio diffusion. `https://github.com/teticio/audio-diffusion`, 2023. 17

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015. 1, 2, 10

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33: 12438–12448, 2020. 18

Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. *CoRR*, abs/1806.03185, 2018. URL `http://arxiv.org/abs/1806.03185`. 32

Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016. 1

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. 9, 18

Shengyun Wei, Shun Zou, Feifan Liao, et al. A comparison on data augmentation methods based on deep learning for audio classification. In *Journal of Physics: Conference Series*, volume 1453, page 012085. IOP Publishing, 2020. 3

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017. 1

Chenshuang Zhang, Chaoning Zhang, Sheng Zheng, Mengchun Zhang, Maryam Qamar, Sung-Ho Bae, and In So Kweon. A survey on audio diffusion models: Text to speech synthesis and enhancement in generative ai. *arXiv preprint arXiv:2303.13336*, 2, 2023. 2