1.5em 0pt

PixMus - Video and Text Conditioned Background Music Generation using Latent Diffusion.

by

Tilak Sharma

May 2024

A dissertation submitted to the

Faculty of the Graduate School of

the University at Buffalo, The State University of New York

in partial fulfillment of the requirements for the

degree of

Master of Science

Department of Engineering and Applied Sciences

…

Acknowledgments

I would like to begin by expressing my deepest gratitude to Professor Nalini Ratha for his invaluable guidance throughout the past year. His mentorship provided me with the opportunities and encouragement needed to pursue the intriguing problems that interested me.

Special thanks are due to my guitar teacher, Abinandan Kar, who helped me pick up the nuances in the music side of things, enriching my understanding and appreciation of this art form.

Being a member of the Sound of AI community has been incredibly rewarding. I am grateful for the time shared and the insightful content created by Valerio Velardo, which has significantly contributed to my growth and development in this field.

I would also like to express my heartfelt gratitude to my friends, especially Arjun, Sanat, and Mahir, who have supported me and provided motivation throughout the completion of my thesis. This past year has been challenging in many ways—academically, personally, and emotionally. I could not have navigated this year without the incredible support from my friends and family, and my time on the guitar, which served as a crucial outlet for maintaining focus and balance. I am truly grateful for all the support I received, which was essential in helping me get through this year and complete my thesis.

Table of Contents

List of Tables

List of Figures

Abstract

In multimedia content creation, the synchronization of video with apt background music plays a pivotal role in enhancing the viewer's experience and engagement. This thesis presents a system designed to generate background music for videos using latent diffusion models. The focus is on developing a diffusion model for audio synthesis, specifically for music, that takes into account the features of the video content along with a text prompt. By analyzing visual elements and dynamics from videos, the system conditions the music generation process to produce background tracks that are coherent and contextually appropriate to the video's mood, tempo, themes, and prompt description.

A key component of this research involved the curation of a specialized background music dataset. Given the specific requirements of this project, existing datasets were insufficient as they did not adequately represent the diverse range of video contexts and the corresponding musical responses required for effective training. Therefore, we developed a dataset that includes a wide variety of video content paired with meticulously chosen music tracks. Each pair was annotated with metadata describing the video content, emotional tone, and musical characteristics, ensuring that the model could learn to produce highly relevant musical accompaniments based on both visual and textual inputs. This curated dataset not only fills a gap in available resources but also significantly enhances the model's ability to generate nuanced and context-sensitive background music.

Through empirical testing, this work assesses the system's effectiveness in creating quality background music, highlighting its potential to provide a customizable, efficient, and cost-effective tool for audio-visual content creation.

Chapter1

Introduction

This work explores the automation of background music generation for videos through the use of latent diffusion models, a novel approach aimed at bridging the gap between video content and its musical accompaniment. By analyzing and interpreting video features—such as schematics and thematic elements—the proposed system employs diffusion models to synthesize music that complements the visual narrative. This study is rooted in the confluence of artificial intelligence, audio engineering, and computer vision, aiming to simplify the content creation workflow, boost viewer engagement, and make a meaningful contribution to the rapidly expanding domain of AI within the creative industries. Through a comprehensive exploration of model architecture, conditioning strategies, and evaluative methodologies, this work aims to demonstrate the viability and potential impact of AI-driven music generation for video production.

1.1 Background and Motivation

Background music significantly influences video production, altering audience perceptions, deepening emotional resonance, and enhancing story engagement. The traditional process of integrating music with video involves manually selecting and synchronizing existing tracks or creating new compositions, a practice that is both resource-intensive and lacking in the flexibility needed to seamlessly match music with the ever-changing demands of video con-

tent.

Existing practices in generating background music for videos predominantly depend on the expertise of music composers and editors, who select and adjust the music to fit a video's atmosphere, story, and rhythm. This manual method requires considerable effort and resources, and its rigidity makes it difficult to tailor music to each video's specific needs without extensive human input. This results in a bottleneck for content production, particularly on platforms requiring a high volume of content.

Utilizing the potential of latent diffusion models, which are conditioned on video attributes, this study seeks to streamline the music generation process. By doing so, it lays a foundation for a scalable, efficient, and adaptable approach to improving video production. This strategy not only overcomes the shortcomings of traditional practices but also meets the increasing demand for customized, high-quality content creation tools, signaling a shift towards a new phase in combining music with video.

1.2 Existing Methods and Related Work

This section reviews the state-of-the-art in video-conditional music generation and related multi-modal learning tasks, highlighting the progress and limitations of existing datasets, methods, and models in the field.

1.2.1 Datasets for Multi-modal Learning

The development of multi-modal learning tasks has been significantly propelled by datasets encompassing various modalities, such as image-text, video-text, and video-audio pairings. However, the domain of video-conditional music generation still faces a dearth of tailored datasets. The HIMV dataset [1], despite its large scale with 200K video-music pairs from YouTube-8M [2], suffers from weak correspondence and poor quality in both music and video content. This highlights the necessity for datasets designed specifically for music generation

tasks. In response, the SymMV dataset [3] was introduced, featuring 1140 piano pieces in MIDI format paired with music videos and rich musical annotations, aiming to fill this gap by providing a resource tailored for music generation.

1.2.2 Advances in Music Generation Models

Traditionally, music generation research has focused on unconditional settings. Video-conditional music generation, however, has primarily concentrated on reconstructing music from silent instrument performance videos. More recent efforts have sought to generate music aligned with dancing or human activity videos, relying on rhythmic relations but requiring additional motion annotations, limiting their applicability to videos with diverse content. The CMT model [4] represents a step towards generating background music for general videos by utilizing video rhythmic features, yet it relies on rule-based video-music relationships without capturing semantic connections.

Recent innovations in sound generation include the AudioLM [5] model, which demonstrates the potential of combining state-of-the-art semantic and acoustic models to generate audio from prompts. Similarly, the AudioLDM [6] framework and the Tango [7] model represent significant advancements in text-to-audio generation by leveraging latent diffusion models and fine-tuned language models. In music generation, models like MusicLM [8] have shown promise in generating audio music directly from text captions, outperforming existing commercial software in several key metrics. Additionally, models such as Noise2Music [9] and MusicGen [10] introduce novel approaches to music generation, from utilizing large amounts of paired music and text data to implementing single-stage transformer models for enhanced control and quality.

1.2.3 Foundational Models in Audio Generation

WaveNet [11], introduced in 2016, represents a foundational model for waveform-level audio generation, using dilated convolutions to synthesize short clips of speech and music at

a 16kHz sampling rate. Jukebox [12], developed in 2020, improved upon this by generating 44kHz music using a hierarchy of quantized autoencoders and transformer upsamplers, capable of conditioning on lyrics, artists, and genres. In 2022, AudioLM [5] emerged, compressing waveforms into discrete tokens with a quantized autoencoder, followed by a cascade of transformer decoders to produce audio at 16kHz.

Another 2022 model, Musika [13], uses 1D convolutional autoencoders to compress spectrograms, subsequently employing a vocoder and a 2D GAN discriminator to regenerate high-fidelity 44kHz audio. Though limited in context length, its efficiency is notable due to the 1D convolution structure. Lastly, Riffusion [14] fine-tunes the Stable Diffusion [15] model on mel-spectrogram segments, using style transfer and textual conditioning to create cohesive audio sequences. Despite its brief 5-second context and slower speed due to its large 2D structure, it performs remarkably well.

1.2.4 Emerging Trends in Text-to-Music Models

A notable trend is the emergence of text-to-music models that offer various functionalities, including music inpainting, continuation, and generation conditioned on text prompts or audio fragments. Models like JEN-1 [16] focus on generating music that complements video content, indicating a growing interest in leveraging both video and textual inputs to inform the music generation process, acknowledging the importance of temporal information such as beats and emotions in creating a harmonious audio-visual experience.

This review underscores the dynamic and evolving nature of video-conditional music generation and sound generation research, highlighting the need for continued innovation in datasets, methodologies, and models to advance the field.

1.3 Challenges

In the domain of video-to-music systems, the task of automatically synthesizing background music that aligns with the dynamic nature of video content presents unique challenges, particularly in the context of waveform-level audio generation. This method, while offering a detailed and precise representation of sound, requires the generation of a vast number of values per second, making the production of lengthy, high-quality audio tracks a complex endeavor. The challenge is further compounded by the limitations of existing datasets tailored for video-conditional music generation and the constraints of symbolic music generation methods, such as MIDI, which may not capture the full richness and nuance of audio. Moreover, the scarcity of datasets specifically designed for this purpose hinders the development and training of models capable of producing diverse and copyright-free background music. Such music is crucial for independent content creators who seek to enhance their videos without the risk of copyright infringement. Addressing these challenges not only requires innovative approaches to audio synthesis and dataset creation but also underscores the importance of developing systems that can generate background music that is stylistically versatile, thereby supporting the creative freedom and needs of indie content creators.

Chapter2

Audio/Music Representation

In this chapter, we explore the multifaceted nature of music representation, which varies widely from traditional sheet music to modern digital formats. Music can be visually encoded in a musical score, communicated through protocols like MIDI for electronic instruments, or captured as acoustic sound waves in audio files. We categorize these into three main classes: sheet music, symbolic, and audio representations. By delving into these representations, we aim to lay a foundational understanding of how music is conveyed and perceived, setting the stage for a deeper examination of its basic properties and the technological methods used to analyze and generate music.

2.1 Representation types

Sheet Music Representations

Sheet music, also referred to as a musical score, is a visual representation of music, particularly in the Western classical tradition. It captures a musical composition using a formalized language of musical symbols and notation. This language is presented in a graphical-textual form that maps out instructions for performance. A musician interprets these instructions to create a live rendition of the piece.

The complexity of sheet music requires a musician to have a specialized form of mu-

sic notation literacy, which involves understanding the symbols and terms that dictate the playing of notes, rhythms, and other musical elements. In addition, musicians bring their personal touch to a piece through creative interpretation, influencing the music's flow by varying tempo, dynamics, and articulation. This process exemplifies the flexibility of sheet music; it provides the structural backbone of a piece while leaving ample room for individual interpretation and expression.

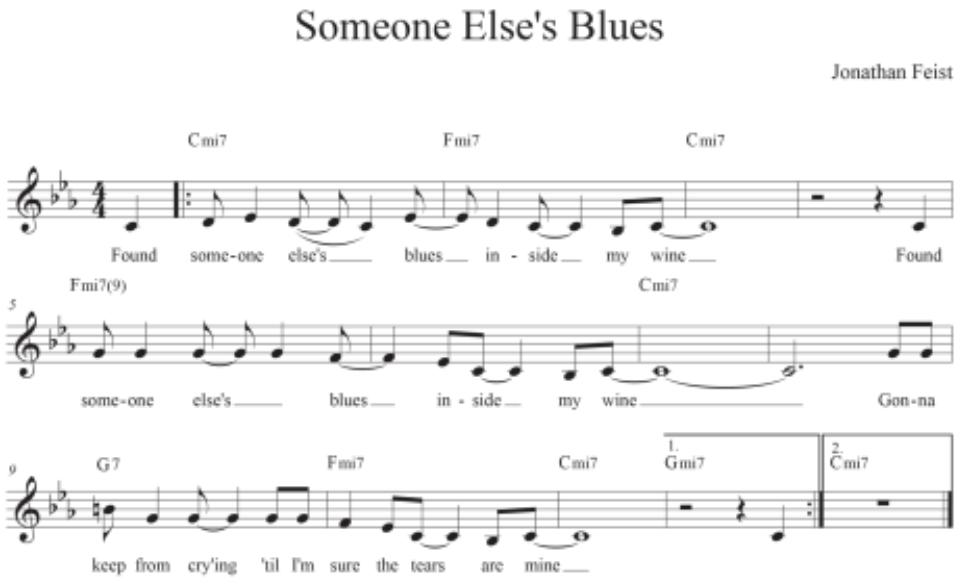


Figure 2.1: Sheet Music Representation

In exploring sheet music further, it is important to understand the notation system. Sheet music is structured around several key musical elements:

- **Notes and Pitches**: Notes are the primary symbols in sheet music, each representing a sound at a specific pitch and duration. The notation specifies how high or low a note sounds (its pitch) and how long it should be held (its duration).
- **Scales and Keys**: Sheet music indicates the key of a piece, which determines the scale that the piece predominantly uses. This affects the tonality and harmonic structure of the music.

- **Time Signatures and Rhythm**: The time signature is noted at the beginning of a piece and dictates the rhythmic structure, telling the performer how many beats are in each measure and what note value constitutes one beat.

- **Dynamics and Expressions**: Instructions for dynamics and expressive elements like crescendos, decrescendos, and fermatas are also included, guiding the intensity and emotional output of the performance.

2.1.1 Symbolic Representation (MIDI)

MIDI is a technical standard that describes a protocol, digital interface, and connectors, allowing a wide range of electronic musical instruments, computers, and other related devices to connect and communicate with one another. Symbolic representation, like MIDI, encodes musical information in a format that specifies the instructions for music performance rather than the sound itself. MIDI data contains information about note sequences, including their pitch, duration, velocity (which roughly corresponds to loudness), and timing, as well as control signals for parameters such as volume, vibrato, and panning.

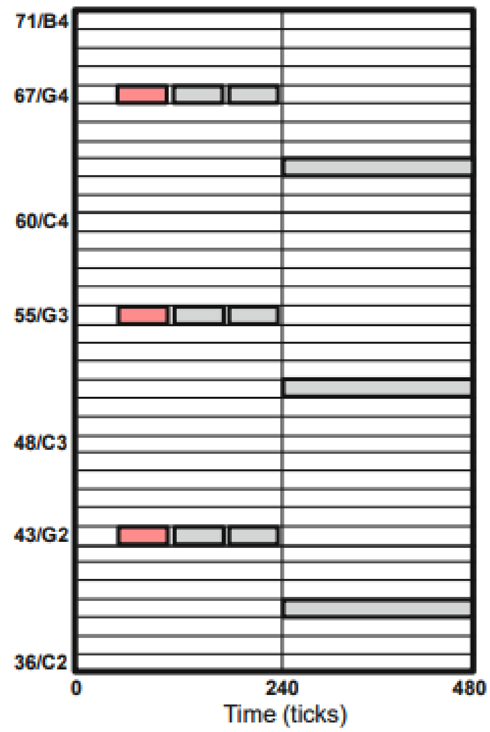
Symbolic representation is highly efficient for storing musical compositions, allowing for easy modification and interaction. For example, a MIDI sequence can be edited to change the instrument, tempo, or key without altering the underlying performance instructions. However, while MIDI excels in representing the structure and performance instructions of music, it lacks the ability to capture the timbral qualities and expressiveness of live performance. The sound quality of MIDI playback depends on the synthesis or sampling technology used to interpret the MIDI data, which can vary significantly between devices.

Although MIDI was not initially developed as a symbolic music format and faces limitations in its representational capabilities, its significance stems from widespread usage over the past three decades and the abundance of MIDI data available online.

The core MIDI messages are the note-on and note-off commands, which indicate the beginning and end of a note, respectively. These messages include several parameters: a MIDI

Time (Ticks)	Message	Channel	Note Number	Velocity
60	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	63	100
0	NOTE ON	2	51	100
0	NOTE ON	2	39	100
240	NOTE OFF	1	63	0
0	NOTE OFF	2	51	0
0	NOTE OFF	2	39	0

(a)



(b)

Figure 2.2: [17] Different Music Representations: (a) MIDI encoded and (b) Piano Roll

note number (ranging from 0 to 127) that specifies the pitch, a key velocity (also between 0 and 127) that controls sound intensity, a channel specification that determines the instrument, and a timestamp that dictates when the note should be played. For instance, MIDI note number 60 corresponds to C4, and the concert pitch A4 is MIDI note number 69. The key velocity affects the volume during note-on events and the decay during note-off events, though the exact interpretation depends on the specific instrument or synthesizer. MIDI channels, numbered from 0 to 15, allow multiple instruments to be played simultaneously, supporting polyphonic music. MIDI's design allows for the encoding of both musical and physical onset times and durations. Unlike absolute time units like microseconds, MIDI uses a system of clock pulses or ticks to subdivide a quarter note, with common resolutions such as 120 pulses per quarter note (PPQN). This system provides precise timing control over note events. Additionally, MIDI can include tempo messages that define the microseconds per quarter note, allowing for the calculation of beats per minute (BPM) and accommodating

both global and local tempo variations.

2.1.2 Digital Waveform Representation

In contrast to symbolic representation, digital waveform representation captures the actual sound waves produced by music. Audio waveforms are typically stored as digital audio files in formats such as WAV, MP3, or FLAC. These formats encode the amplitude of sound waves at discrete intervals, or samples, over time. The quality of a digital audio recording is determined by its bit depth and sampling rate. Bit depth refers to the number of bits used to encode the amplitude of each sample, affecting the dynamic range of the sound. The sampling rate, measured in Hertz (Hz), refers to the number of samples taken per second, affecting the frequency range that can be accurately reproduced.

Digital waveform representation can capture the full richness and complexity of sound, including timbral and expressive nuances that symbolic representations like MIDI cannot. This allows for a more accurate and detailed reproduction of live performances. However, digital audio files are typically much larger than symbolic representations and require more storage space and processing power to manipulate.

In the digital representation of audio, two fundamental properties—frequency and amplitude—play pivotal roles in determining the pitch and loudness of the sound, respectively. Frequency refers to the number of cycles (vibrations) per second of a sound wave, measured in Hertz (Hz), and directly correlates with the pitch of the sound: higher frequencies produce higher pitches, while lower frequencies result in lower pitches. Amplitude, on the other hand, measures the size of the vibrations and is perceived as loudness; larger amplitudes produce louder sounds.

When audio is digitized, a process known as sampling is employed to convert the continuous analog sound waves into a discrete set of samples that can be stored and processed digitally. This sampling process is characterized by a sampling rate—the number of samples taken per second. To digitize a signal, each point in time is measured, and the amplitude

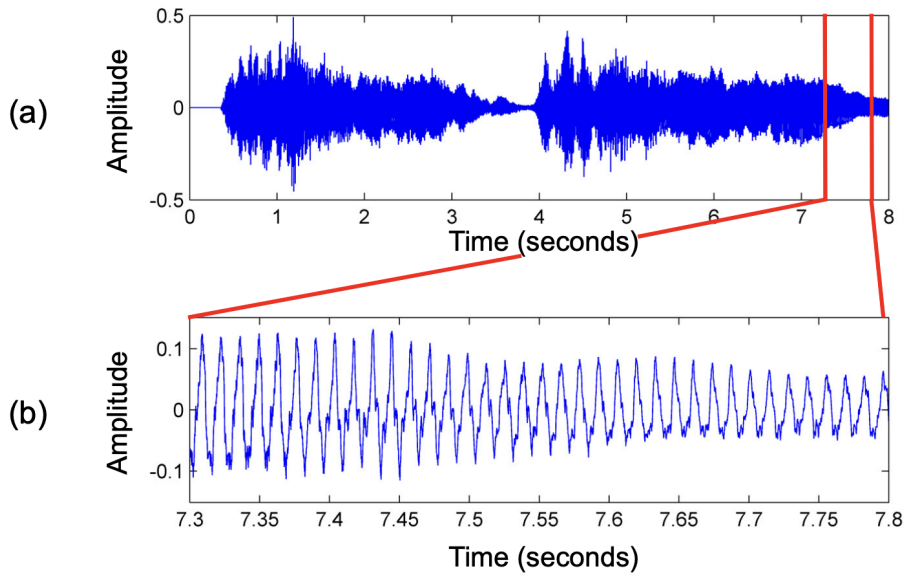


Figure 2.3: [17] (a) Waveform of the first eight seconds of a recording of the first five measures of Beethoven's Fifth. (b) Enlargement of the section between 7.3 and 7.8 seconds.

of the sound wave at that instant is stored as a digital value. This transformation from a continuous to a discrete signal requires precise timing; if the sampling rate is too low, some data is lost, which can degrade the audio quality.

The Nyquist frequency, which is half the sampling rate, serves as a critical threshold in this context. According to the Nyquist theorem, to accurately capture and reproduce a sound wave without loss of information, the sampling rate must be at least twice the highest frequency present in the sound. Frequencies above the Nyquist frequency cannot be correctly sampled, leading to artifacts and aliasing, where these excessive frequencies are misrepresented as lower-frequency sounds in digital audio. To mitigate such distortions, audio signals are often passed through a low-pass filter to remove frequencies above the Nyquist limit before sampling, ensuring the fidelity of the digital representation in capturing the true essence of the original sound.

Aliasing occurs when higher frequency components of the audio signal are sampled inadequately and thus appear falsely as lower frequencies. This misrepresentation not only affects the accuracy of the sound reproduction but can also introduce undesirable artifacts in the

audio playback. To avoid this, anti-aliasing filters are used before the sampling process to ensure that frequencies higher than half the sampling rate are attenuated, thus preserving the integrity and quality of the digital audio signal.

2.2 Fourier Transforms on Signals

Processing directly sampled audio or music is complex due to several factors. Audio signals, when sampled, convert from a continuous to a discrete form, leading to challenges such as high data volumes, maintaining dynamic range and precision, and managing the time-frequency complexity of the signals.

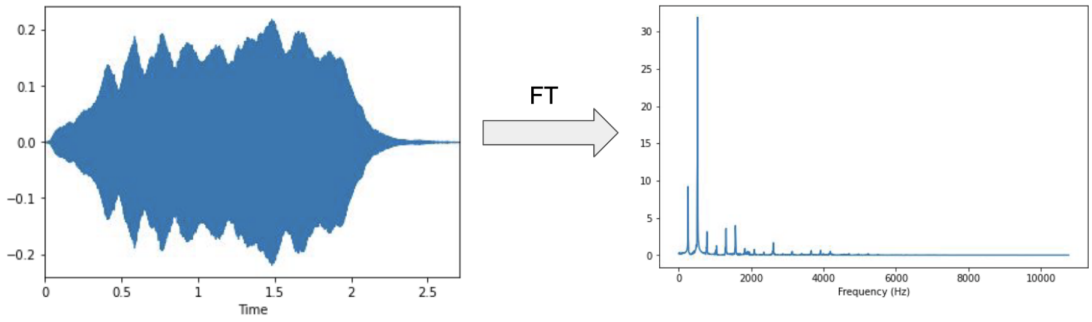


Figure 2.4: Time domain to frequency domain using Fourier Transform

Fourier Transforms play a crucial role in simplifying these complexities by transforming time-domain signals into their frequency-domain counterparts. This is essential for various audio processing tasks:

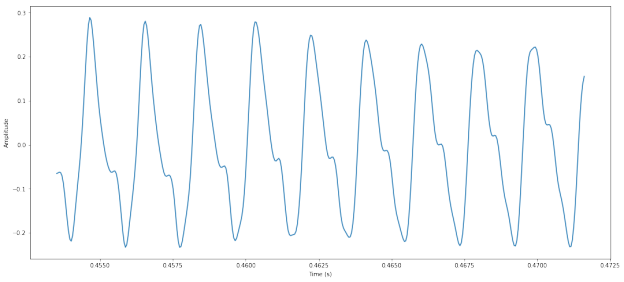
- **Spectral Analysis:** Decomposing signals into constituent frequencies to facilitate tasks like filtering and equalization.
- **Noise Reduction and Enhancement:** Identifying and filtering out unwanted noise while enhancing desired frequencies.
- **Compression:** Utilizing frequency domain characteristics to efficiently reduce data size without significantly impacting audio quality.

- **Feature Extraction:** Supporting advanced applications such as music recognition and audio classification through efficient feature extraction.

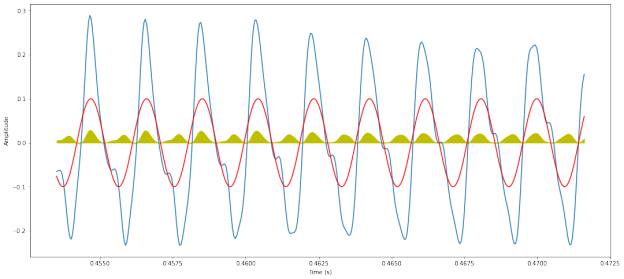
Fourier Transforms thus are indispensable for converting complex audio signals into a more manageable form for analysis and processing, making them integral to the field of digital audio processing.

2.2.1 Fourier Transform Intuition

The Fourier Transform process involves comparing a signal with sinusoidal waves of various frequencies. This comparison helps in determining the presence and strength of these frequencies in the signal. By calculating the integral of the product of the signal and a sinusoid, we obtain measures of magnitude and phase which indicate how much of a particular frequency is present in the signal and its phase shift relative to the start of the signal.



(a)



(b)

Figure 2.5: (a) Time domain signal snippet (b) Sin wave overlap over the signal snippet

The sinusoidal wave used for this comparison is generally represented by:

$$\sin(2\pi \cdot (ft - \phi)) \tag{2.1}$$

where f is the frequency of the sinusoid, t is time, and ϕ is the phase shift of the sinusoid.

To extract the frequency components from the signal $s(t)$, we calculate the following integrals for each frequency f, seeking the phase ϕ_f that maximizes the integral, thus giving us the phase and magnitude (d_f) of that frequency component:

$$\phi_f = \arg\max_{\phi \in [0,1]} \left(\int s(t) \cdot \sin(2\pi \cdot (ft - \phi))\, dt \right) \tag{2.2}$$

$$d_f = \max_{\phi \in [0,1]} \left(\int s(t) \cdot \sin(2\pi \cdot (ft - \phi))\, dt \right) \tag{2.3}$$

These equations allow us to determine the magnitude d_f and the phase shift ϕ_f for each frequency component, providing insight into the content of the signal at various frequencies. The magnitude d_f indicates the amplitude of the frequency f in the signal, and the phase ϕ_f indicates the shift of this frequency component within the signal, relative to its standard position.

2.2.2 Complex Numbers

Complex numbers are integral to Fourier Transforms because they simplify the computation and interpretation of spectral analyses. The motivation for using complex numbers in the context of Fourier transforms can be understood both algebraically and geometrically.

Algebraic Perspective: Complex numbers simplify the mathematical expressions involved in Fourier transforms. A complex number can be expressed in the form:

$$c = a + ib \tag{2.4}$$

where a and b are real numbers representing the real and imaginary parts, respectively,

and i is the square root of -1. This form is particularly useful in Fourier analysis because it allows for the encoding of both magnitude and phase information in a single expression.

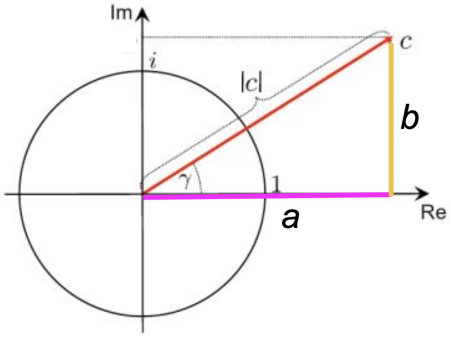


Figure 2.6: Complex numbers in Polar representation

Geometric Perspective: Geometrically, complex numbers can be represented as points or vectors in the complex plane, with the real part a along the horizontal axis and the imaginary part b along the vertical axis. The complex number is often represented in polar form as:

$$c = |c| \cdot (\cos(\gamma) + i \sin(\gamma)) \tag{2.5}$$

where $|c|$ is the magnitude of the complex number, and γ is the phase angle. The formulas for converting between rectangular (Cartesian) and polar forms are given by:

$$\gamma = \arctan\left(\frac{b}{a}\right), \quad |c| = \sqrt{a^2 + b^2}$$

$$a = |c| \cdot \cos(\gamma), \quad b = |c| \cdot \sin(\gamma)$$

Euler's Formula: Euler's formula provides a powerful link between exponential functions and trigonometric functions, which is particularly useful in the context of Fourier transforms. According to Euler's formula, we have:

$$e^{i\gamma} = \cos(\gamma) + i\sin(\gamma) \tag{2.6}$$

Therefore, the complex number c can also be expressed using Euler's formula as:

$$c = |c| \cdot e^{i\gamma} \tag{2.7}$$

This expression not only simplifies many mathematical manipulations but also provides an intuitive understanding of complex numbers as rotations and scalings in the complex plane.

2.2.3 Fourier Transforms with Complex Numbers

For a time-dependent function $g(t)$, the Fourier transform is formally defined as:

$$\hat{g}(f) = \int_{-\infty}^{\infty} g(t) \cdot e^{-i2\pi ft}\, dt \tag{2.8}$$

This transformation converts $g(t)$ from the time domain into its frequency components $\hat{g}(f)$, where f denotes the frequency. The kernel of the transform, $e^{-i2\pi ft}$, is a complex exponential, which encapsulates both the sinusoidal oscillation and its phase.

2.2.4 Inverse Fourier Transform

The original function $g(t)$ can be recovered from its Fourier transform $\hat{g}(f)$ using the inverse Fourier transform:

$$g(t) = \int_{-\infty}^{\infty} \hat{g}(f) \cdot e^{i2\pi ft}\, df \tag{2.9}$$

This equation allows for the reconstruction of the time domain signal from its frequency domain representation, effectively illustrating the symmetrical nature of Fourier transformations.

2.2.5 Discrete Fourier Transforms

While the continuous Fourier Transform is ideal for theoretical analysis, digital audio processing requires a discrete version due to the nature of digital systems which handle discrete data.

In digital signal processing, we deal with discrete signals obtained by sampling $g(t)$ at intervals T, leading to a sequence $g[n] = g(nT)$. We have:

$$\hat{g}(f) \approx \sum_{n=-\infty}^{\infty} g(nT) \cdot e^{-i2\pi fnT} \cdot T \tag{2.10}$$

For practical digital computation, we use the Discrete Fourier Transform (DFT), represented as:

$$\hat{x}(f) = \sum_{n=0}^{N-1} x[n] \cdot e^{-i2\pi fn} \tag{2.11}$$

where N is the total number of samples.

The DFT directly computes each term of the Fourier Transform, resulting in a complexity of $O(N^2)$, where N is the number of samples in the discrete signal. This quadratic complexity arises because each output frequency component requires a sum over all N input samples.

The Fast Fourier Transform (FFT) is an algorithm that dramatically reduces the computational burden associated with the DFT. The FFT efficiently computes the DFT of a sequence utilizing the symmetries in the periodic properties of the complex exponential function. The most common FFT algorithm is the Cooley-Tukey algorithm, which reduces the computational complexity from $O(N^2)$ to $O(N \log N)$.

This improvement is achieved by recursively breaking down the DFT of the sequence into smaller DFTs, leveraging the property that the DFT of a sequence is periodic and symmetrical. This makes FFT particularly advantageous for real-time signal processing where large data sets are common, and computational efficiency is critical.

2.2.6 Short-Time Fourier Transform (STFT)

The Short Time Fourier Transform (STFT) is a technique used to determine the sinusoidal frequency and phase content of local sections of a signal as it changes over time. Unlike the Fourier Transform, which provides the frequency content over the entire signal, STFT provides a time-frequency representation of the signal, which is essential for analyzing non-stationary signals where the frequency components vary over time.

The STFT is defined by the following equation:

$$S(m,k) = \sum_{n=0}^{N-1} x(n+mH) \cdot w(n) \cdot e^{-i2\pi \frac{kn}{N}} \tag{2.12}$$

where:

- $x(n)$ is the signal,
- $w(n)$ is the window function applied to the signal,
- N is the number of points in the FFT,
- m is the time index, and
- H is the hop size or the number of samples between successive frames.

The STFT applies a fixed-size window to the signal and slides this window across the signal, computing the Fourier Transform at each position. The window function $w(n)$ is typically a function like a Hamming window, Hanning window, or Gaussian window, which helps to mitigate the spectral leakage. STFT on a signal results in a spectogram as shown in 2.7

The Short Time Fourier Transform offers several advantages:

- **Time-Frequency Representation:** STFT provides a time-varying spectral representation that allows for the analysis of frequency components as they change over time, making it ideal for non-stationary signals.

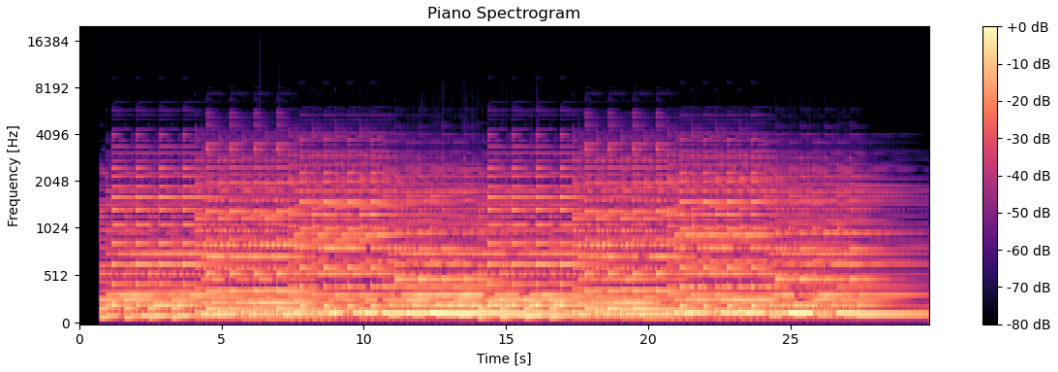


Figure 2.7: STFT on a piano piece

- **Flexibility:** The choice of window size N and the hop size H provides flexibility in analyzing the trade-off between time and frequency resolution. Smaller windows provide better time resolution, while larger windows provide better frequency resolution.

- **Real-time Analysis:** STFT is well-suited for real-time signal analysis because it can be implemented efficiently and provides the analysis incrementally as new data arrives.

The STFT, however, is not without its limitations. The primary challenge is the trade-off between time and frequency resolution, which is dictated by the Heisenberg uncertainty principle. This principle states that the product of the time and frequency resolutions is bounded, which implies that increasing the resolution in one domain results in a decrease in the other.

Chapter3

Diffusion Methods

3.1 Overview

Diffusion is a concept rooted in the natural sciences, particularly thermodynamics, and has novel applications in machine learning and artificial intelligence. The fundamental principle of diffusion in thermodynamics describes the process by which particles move from regions of higher concentration to regions of lower concentration, eventually reaching an equilibrium state. This natural phenomenon serves as an inspiration for diffusion models in machine learning, which are designed to generate complex data distributions through a controlled and reversible process.

The crux of diffusion methods in machine learning lies in their ability to model data generation as a gradual process of reversing diffusion. Initially, a data sample (in our case, the latent representation of audio) undergoes a forward diffusion process, where noise is added over a series of steps until it reaches a state nearly indistinguishable from Gaussian noise. This process effectively 'blurs' the original data, moving it towards a high-entropy state. The magic of diffusion models comes into play during the reverse diffusion process, where the aim is to learn how to remove the added noise, step by step, to recover the original data sample from its noised state.

A pivotal component in the architecture of diffusion models is the use of a U-Net, a type

of convolutional neural network known for its efficacy in tasks that require an understanding of detailed spatial hierarchies in data, such as image segmentation. In the context of diffusion models, the U-Net is trained to predict the noise that was added at each step of the forward diffusion process. By accurately predicting and subsequently removing this noise, the model gradually 'denoises' the data, reversing the diffusion process. Through this iterative procedure, the U-Net learns the complex, multi-scale structure of the data distribution, enabling the generation of high-fidelity samples.

Training the model involves minimizing the difference between the original noise added during the forward process and the noise predicted by the U-Net at each step, effectively teaching the model to reverse the diffusion process. This approach stands out for its ability to generate samples that are both diverse and closely aligned with the original data distribution, showcasing the power of diffusion methods in capturing and reproducing complex patterns and structures inherent in data, such as the nuanced textures of sound in music generation tasks.

3.2 Diffusion - DDPM

Diffusion models are a class of generative models adept at modeling complex data distributions in high-dimensional spaces, denoted as $p(\mathbf{x})$, where $\mathbf{x}$ represents data points within this space. Unlike likelihood-based models that attempt to learn this distribution directly which is both computationally demanding and inherently challenging, diffusion models adopt an alternative strategy. They aim to estimate the gradient of the log probability of the data, known as the score function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$.

The score function is critical as it indicates the direction and rate of increase in the probability density. In essence, it guides how to adjust $\mathbf{x}$ to increase its likelihood under the distribution $p(\mathbf{x})$. By learning the score function, diffusion models effectively learn the "shape" of the data distribution, enabling them to generate new data points that bear

similarity to the original data.

During the generation phase, diffusion models commence with a sample of random noise and iteratively refine this sample under the guidance of the learned score function. This process effectively 'walks back' the diffusion process—the gradual addition of noise to the data—by reversing the noise addition. This reversal is applied iteratively until the transformed noise yields a coherent sample resembling the data from the original distribution.

This methodology circumvents some of the challenges associated with direct modeling of $p(\mathbf{x})$ and permits the model to methodically sculpt the random noise into structured data. The process is steered by the intricacies of the data's distribution as captured in the score function, thereby empowering diffusion models to generate complex forms of data such as images and audio, which require a nuanced understanding of high-dimensional probability distributions.

Some of the key notations used:

- $\mathbf{x}_0$: Represents the original data point in the high-dimensional space, which we aim to model.
- $\mathbf{x}_t$: Denotes the data point at time step t during the diffusion process, where $t \in \{0, \ldots, T\}$ and T is the total number of diffusion steps.
- $\mathbf{x}_T$: Refers to the data point after the final diffusion step, which is typically a point sampled from a Gaussian distribution.
- $q(\mathbf{x}_t|\mathbf{x}_0)$: Is the forward diffusion process, describing the probability distribution of $\mathbf{x}_t$ given the original data point $\mathbf{x}_0$.
- $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$: Represents the reverse diffusion process, where we aim to learn the distribution of $\mathbf{x}_{t-1}$ given the subsequent time step $\mathbf{x}_t$.
- $\epsilon_\theta(\mathbf{x}_t, t)$: Is the noise prediction network parameterized by θ, which aims to predict the noise that has been added to the original data at time step t.

3.2.1 Forward Processes

Given a data point sampled from a real data distribution $x_0 \sim q(x)$, we define a *forward diffusion process* in which we add small amounts of Gaussian noise to the sample in T steps, producing a sequence of noisy samples $x_1, \ldots, x_T$. The step sizes are controlled by a variance schedule $\{\beta_t \in (0,1)\}_{t=1}^{T}$. The forward process is defined as:

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

and the overall diffusion process is:

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{3.2}$$

As the data sample x_0 undergoes this process, it gradually loses its distinguishable features as the step t becomes larger. Eventually, when $T \to \infty$, x_T is equivalent to an isotropic Gaussian distribution.

The forward diffusion process has a convenient property allowing us to sample the data at any step t in a closed form. This is achieved using the reparameterization trick, which provides a way to express the noisy data x_t directly in terms of the original data x_0 and independent Gaussian noise.

Define the coefficients $\alpha_t = 1 - \beta_t$ and the cumulative product $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. With these definitions, we can sample x_t as follows:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \tag{3.3}$$

where $\epsilon \sim \mathcal{N}(0, I)$ represents the Gaussian noise. This formulation allows us to combine the original data x_0 with the noise ϵ proportionally to how much the process has progressed at time t.

The distribution of x_t conditional on x_0 is given by a Gaussian with mean $\sqrt{\bar{\alpha}_t}x_0$ and

covariance matrix $(1 - \bar{\alpha}_t)I$:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)I) \tag{3.4}$$

It is noted that when we combine two independent Gaussian distributions with different variances, the resulting distribution is also Gaussian with a variance equal to the sum of the two variances.

3.2.2 Reverse Process

After completing the forward diffusion process where the original data x_0 has been transformed into a noisy version x_T, the reverse diffusion process aims to recover the original data from this noisy state. In the reverse process, we iteratively denoise the data by learning to reverse the noise that was added during the forward steps. The reverse process is formalized by a series of conditional probabilities:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{3.5}$$

Here, $\mu_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$ are the mean and covariance learned by the model with parameters θ, and are functions of the current noisy data x_t and the time step t. The reverse process effectively denoises the data step-by-step, using the learned parameters to approximate the reverse of the noise that was added at each forward step.

The iterative denoising can be thought of as a trajectory in reverse, starting from the noisy data and stepping backwards toward the data's original state. At each step, the model uses the conditional probability $p_\theta(x_{t-1}|x_t)$ to estimate the cleaner version of the current state. This process is repeated until the original data is recovered as x_0:

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{3.6}$$

The reverse process is learned by a neural network, which is trained to gradually denoise

the data. The training algorithm involves optimizing a loss function that encourages accurate prediction of the noise added during the forward diffusion process.

3.2.3 Training objective

Findings by [**Ho et al.**] (2020) suggest that diffusion models can be effectively trained using a simplified objective function used in the variational lower bound. The simplified objective function for the training of diffusion models is formulated as follows:

$$L_t = \mathbb{E}_{t,x_0,\epsilon}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|^2\right], \tag{3.7}$$

where $\epsilon \sim \mathcal{N}(0, I)$ represents the isotropic Gaussian noise, x_0 is a sample from the data distribution $q(x_0)$, and x_t is the data at timestep t corrupted by noise. The function $\epsilon_\theta(x_t, t)$ is the neural network parameterized by θ which predicts the noise added at timestep t.

3.2.4 Training Algorithm

The following algorithm describes the process for training the diffusion model:

Algorithm 1 Training the Diffusion Model

1: **repeat**
2: $x_0 \sim q(x_0)$
3: $t \sim \text{Uniform}(\{1, \ldots, T\})$
4: $\epsilon \sim \mathcal{N}(0, I)$
5: Take a gradient descent step on:
$\nabla_\theta \left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right)\right\|^2$
6: **until** convergence

3.2.5 Sampling Procedure

The sampling procedure for Denoising Diffusion Probabilistic Models (DDPMs) is methodically executed by initiating from a noise distribution $x_T \sim \mathcal{N}(0, I)$. We then sequentially and deterministically refine this distribution by applying the learned model T times in reverse. In

each iteration, the model employs estimated mean vectors $\mu_\theta(x_t)$ — or alternatively, inferred noise vectors $\epsilon_\theta(x_t)$ — for T distinct Gaussian distributions. This recursive refinement yields the sequence $x_{T-1}, x_{T-2}, \ldots, x_0$, where x_0 emerges as the desired output. It is noteworthy that the nature of this sampling sequence is inherently stochastic, incorporating additional noise at each iteration from Gaussian distributions.

3.2.6 Limitations

The success of sampling with Denoising Diffusion Probabilistic Models depends heavily on using a large number of steps, usually thousands, to produce good-quality samples. This is quite different from newer methods that allow for more adjustment in the sampling process. The fixed number of steps, known as T, set during the training and sampling stages of the model, can limit how adaptable and efficient the model is.

3.3 V-Diffusion

V-diffusion, also known as v-objective diffusion and inspired by Denoising Diffusion Implicit Models (DDIM), is a diffusion method characterized by a continuous variable $\sigma_t \in [0, 1]$. By defining a forward process that introduces noise over several steps and learning a corresponding reverse process, this method can generate high-fidelity samples from noisy data. In v-diffusion, x_{σ_t} denotes a data point from the data distribution when $\sigma_t = 0$, and represents Gaussian noise e when $\sigma_t = 1$. Unlike DDIM, where the model may be employed to predict the original data point x_0 or the noise e_t, v-diffusion estimates a velocity value v_{σ_t}, from which both x_0 and e_{σ_t} can be inferred.

3.3.1 Forward Process

The forward or the noising process is given a geometric interpretation with a weighting on a circle:

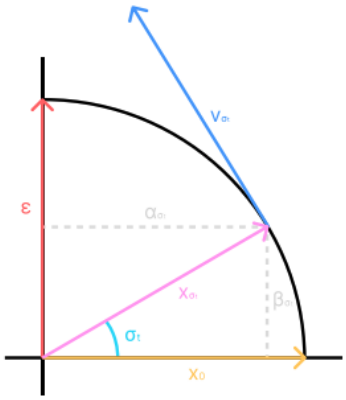


Figure 3.1: V-Diffusion

$$x_{\sigma_t} = \alpha_{\sigma_t} x_0 + \beta_{\sigma_t} \epsilon \tag{3.8}$$

The parameters α_{σ_t} and β_{σ_t} are determined as follows:

$$\alpha_{\sigma_t} = \cos(\phi_t), \quad \beta_{\sigma_t} = \sin(\phi_t), \quad \text{where} \quad \phi_t = \frac{\pi}{2}\sigma_t. \tag{3.9}$$

When $\sigma_t = 0$, then x_{σ_t} is just x_0, meaning no noise is added. Conversely, when $\sigma_t = 1$, then x_{σ_t} is purely noise $\epsilon \sim \mathcal{N}(0, I)$. The process of linearly increasing σ_t from 0 to 1 ensures a gradual transition from the original data x_0 to pure noise. By sampling σ_t across the interval [0,1], we tend to pick values where x_{σ_t} is more representative of x_0 than the pure noise ϵ, implying that the model will more often see data with a smaller amount of noise.

3.3.2 Reverse Process

In the denoising step of V-diffusion, we aim to estimate the original data point $\hat{x}_0$ from a noised observation x_{σ_t} at a given noise level σ_t. The model's velocity-estimating function $\hat{v}_{\sigma_t}$ is pivotal to this process, where $\hat{v}_{\sigma_t}$ is defined as the derivative of x_{σ_t} with respect to σ_t, capturing how the data point changes as noise is incrementally added or removed.

To recover $\hat{x}_0$, we use the following equations:

$$\hat{v}_{\sigma_t} = f_\theta(x_{\sigma_t}, \sigma_t) \tag{3.10}$$

$$\hat{x}_0 = \alpha_{\sigma_t} x_{\sigma_t} - \beta_{\sigma_t} \hat{v}_{\sigma_t} \tag{3.11}$$

$$\hat{\epsilon}_{\sigma_t} = \beta_{\sigma_t} x_{\sigma_t} + \alpha_{\sigma_t} \hat{v}_{\sigma_t} \tag{3.12}$$

$$x_{\sigma_{t-1}} = \alpha_{\sigma_{t-1}} \hat{x}_0 + \beta_{\sigma_{t-1}} \hat{\epsilon}_{\sigma_t} \tag{3.13}$$

Here, α_{σ_t} and β_{σ_t} are coefficients that modulate the influence of the data point and noise at each step, ensuring the noise is properly scaled as σ_t changes. The estimation of $\hat{x}_0$ is a crucial step as it represents the reverse engineering of the original data point from the noisy observation. These equations allow the model to iteratively refine the noise estimate and backtrack to the less noisy states until $\hat{x}_0$, an estimate of the clean data point, is obtained.

3.3.3 Training Algorithm

Algorithm 2 Training the V-Diffusion Model

1: **repeat**
2: $x_0 \sim q(x_0)$
3: $\sigma_t \sim \text{Uniform}([0, 1])$
4: $\epsilon \sim \mathcal{N}(0, I)$
5: Compute $\hat{v}_{\sigma_t}$ using the model's velocity-estimating function
6: Take a gradient descent step on:
$\nabla_\theta \left\| \hat{v}_{\sigma_t} - (\alpha_{\sigma_t} \epsilon - \beta_{\sigma_t} x_{\sigma_t}) \right\|^2$
7: **until** convergence

3.3.4 Sampling Procedure

The V-diffusion model's sampling process begins by sampling a noisy data point $x_T \sim \mathcal{N}(0, I)$. We then apply the trained model in a reverse manner for T steps. At each step t, the model predicts the velocity vector $\hat{v}_{\sigma_t}$, which is used to denoise and estimate the data point for the previous timestep:

Algorithm 3 Sampling from the V-Diffusion Model

$x_T \sim \mathcal{N}(0, I)$
for $t = T, \ldots, 1$ **do**
    Estimate $\hat{v}_{\sigma_t}$ using the model
    Compute α_{σ_t} and β_{σ_t} based on σ_t
    $x_{\sigma_{t-1}} = \alpha_{\sigma_{t-1}} x_0 + \beta_{\sigma_{t-1}} \hat{\epsilon}_{\sigma_t}$
end for
return x_0

Chapter4

Implementation Details

4.1 Dataset

4.1.1 Background Music Video Dataset

Given the objective of generating background music in response to video input, the scarcity of directly applicable datasets necessitates the creation of a specialized dataset for this task. While there are abundant datasets pairing text with music and video with text, the niche requirement of video-background music pairs is underrepresented. Recognizing this gap, our work introduces a curated video-background music dataset.

4.1.2 Data Collection

The development of background music tracks suitable for use with various video content presents unique challenges. Despite a wealth of available music datasets, the alignment of music with video themes remains a significant hurdle. To tackle this, we curated a specialized dataset, concentrating on specific types of YouTube videos known for their instrumental or software-produced background music. Our focus spanned a range of sources, including nature documentaries, vlogs, channels dedicated to copyright-free music, and movie scenes that heavily feature instrumental tracks.

4.1.3 Background Music Datasets

Music datasets serve as the foundational material for studies involving music generation, synchronization, and analysis. The primary datasets relevant to this research area are listed in Table 4.1. These datasets vary in structure, size, and the type of data they contain. For example, some focus on symbolic music representations, such as Musical Instrument Digital Interface (MIDI) files with accompanying labels or tags. Others consist of actual audio recordings (WAV files), sometimes paired with caption data or video content.

Dataset	Type	Size
LPD [18]	Music (MIDI)-Labels/Tags	172k
MuLaMCap [9]	Music-Captions	400k
SymMV [3]	Music (MIDI)-Captions-Video	1.1k
MusicBench [19]	Music (WAV)-Captions	52k
PixMus	**Music (WAV)-Captions-Video**	**53k**

Table 4.1: Comparison of Music Datasets.

The datasets summarized in Table 4.1 each have distinct characteristics and applications. For instance, LPD primarily consists of MIDI files with associated labels or tags, providing a symbolic representation of music. MuLaMCap, on the other hand, features music with corresponding captions, offering a different perspective on music data. SymMV includes video content with MIDI and caption data, emphasizing the integration of multimedia elements.

4.1.4 Challenges in Data Collection

One of the key challenges in creating a comprehensive dataset for background music is the common inclusion of vocal tracks or song lyrics. This contradicts the goal of obtaining instrumental or software-generated music tracks. Similarly, many datasets offer video content with accompanying music, but these tracks often lack a clear relationship with the video content, leading to inconsistencies in theme and tone.

To overcome these obstacles, we focused on specific categories of video content known to feature instrumental music. This included sports videos, nature documentaries, and

scenes from movies with instrumental backgrounds. Additionally, we avoided datasets that primarily offered thumbnails or other visual representations that did not align with our objective of creating a cohesive dataset of video-music pairs.

4.1.5 Our Dataset

Our dataset, comprises a total of 53,378 samples, with 22,482 videos and 30,896 images or thumbnails. This diverse collection accurately reflects the range of digital content found across online platforms and provides a rich resource for studying the interaction between video visuals and background music. This broad dataset scope supports a wide range of use cases and research applications, including the development and testing of music generation models.

Compared to other datasets like MusicCaps, which typically consist of around 5,000 samples, PixMus offers a significantly larger dataset, making it ideal for comprehensive studies on background music in videos. With its broad representation of video-music pairings, this dataset establishes a robust platform for research and experimentation in the domain of background music synchronization and generation.

4.1.6 Dataset Structure

The dataset comprises manually labeled data points, each categorized as either a video or a thumbnail. Videos were randomly sampled to extract a set of k frames, which were then arranged in a grid-like pattern. This arrangement facilitated the categorization process through a simple web interface, enhancing the scalability and efficiency of organizing the dataset for future expansions. Future automation of this process could involve checking for dynamic changes in the frames to distinguish videos, which exhibit motion, from static images representing thumbnails.

Each music-video pair was divided into multiple segments for detailed analysis and subsequent model training. Music was split into 10-second clips, while video was divided into

30-second clips. These segments were transformed into formats suitable for model consumption to create a coherent representation of audio and video elements.

The audio components were encoded into latent representations using a Variational Autoencoder (VAE) from the audio-LDM framework [6]. This process compresses audio into smaller, more manageable data representations while retaining essential information. To maintain continuity over time, consecutive 10-second latents were stacked to form 30-second sequences, offering a broader context and richer data for model training and analysis.

Video segments were processed into embeddings using the CLIP [20] model on the frames extracted at 1 fps, a pre-trained framework for transforming visual content into high-dimensional embeddings. This step allows for a comprehensive analysis of video features and enhances the integration of video and music data for various research applications.

4.2 System Architecture and Design

Our system architecture employs a denoising U-Net, central to the learning process of the reverse diffusion model. The U-Net structure is widely recognized for its remarkable pattern recognition capabilities in both image processing and large-scale audio models, due to its progressive compression and decompression mechanisms that allow for learning intricate patterns at multiple scales of granularity.

4.2.1 Incorporation of 3D Convolution

Taking inspiration from text-to-video techniques detailed in [21], our model incorporates 3D convolution within the U-Net's encoder and decoder. This approach is designed to handle sequential latent representations, essential for capturing temporal dynamics that ensure coherence in extended audio sequences.

The 3D convolutional layers process data in three dimensions: height, width, and depth (or time). This feature makes them ideal for working with sequential data, allowing the

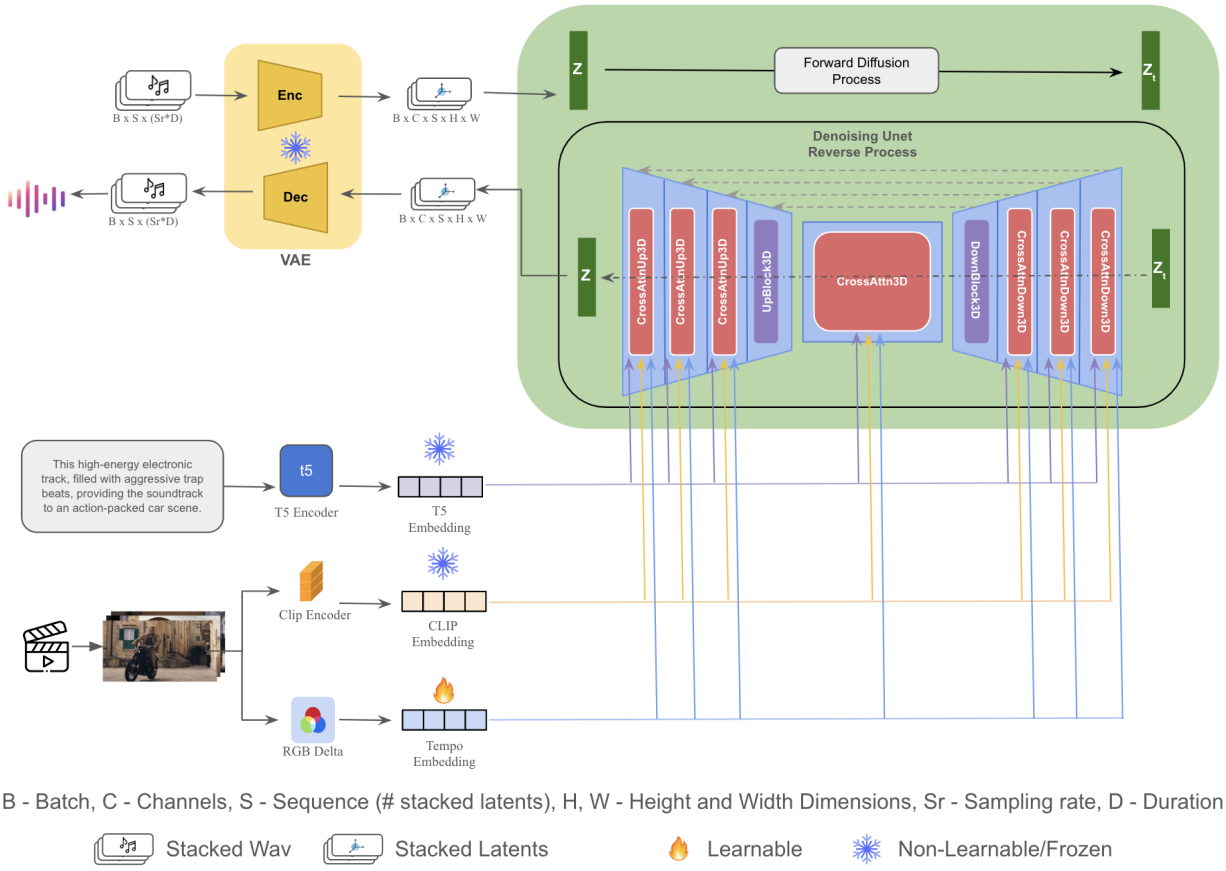


Figure 4.1: Architecture Overview: We use a frozen Variational Autoencoder (VAE) [6] to compress the input audio waveform into a latent representation. These latent representations are then stacked and fed into a U-Net-based diffusion model, which learns to denoise the latent representations conditioned on various signals, including text, semantic, and tempo embeddings. The final predicted latent is iteratively refined and then decoded by the VAE to generate the output audio waveform. **Training**: The training process involves noising the music latent using the forward process and then optimizing the U-Net to predict the noise at a particular x_{σ_t} along with the noise ϵ_t. **Inference**: The process starts with complete Gaussian noise and iteratively removes noise to generate x_0, guided by the conditioning signals.

network to understand patterns across time, a critical component for creating consistent audio narratives. By using 3D convolutions, the model can efficiently learn from spatial-temporal data without losing context as it compresses and decompresses information.

4.2.2 Architecture Overview

The architecture comprises three main components: the encoder, the decoder, and the mid-block. The encoder compresses the data through successive layers, while the decoder decompresses it to its original form, with each stage progressively capturing more detailed patterns. The mid-block, located between the encoder and decoder, focuses on cross-attention, enhancing the network's ability to contextualize data during processing without changing its dimensionality.

4.2.3 CrossAttentionDown3D and CrossAttentionUp3D Blocks

The encoder employs `CrossAttentionDown3D` blocks, which are transformer units designed to perform cross-attention with the conditioning signals while simultaneously downsampling the data. This approach merges the strengths of transformers' attention mechanisms with the efficiency of 3D convolutions. The cross-attention is between the conditioning features and the intermediate feature map, allowing the model to focus on specific aspects of the input data, enhancing pattern recognition and detail extraction during compression.

Similarly, the decoder utilizes `CrossAttentionUp3D` blocks, which perform cross-attention while upsampling the data through transpose convolutions. This process is crucial for maintaining spatial-temporal coherence while decompressing the data back to its original form. The use of cross-attention during both the downsampling and upsampling stages ensures that the contextual information is retained throughout the model, leading to smoother and more coherent outputs.

4.2.4 Residual Connections

To counter the issue of vanishing gradients in deep neural networks and enhance the stability of the training process, we incorporate residual connections within the encoder blocks and between corresponding layers of the encoder and decoder. These skip connections allow the model to bypass certain layers, ensuring that the gradients can flow more freely, reducing the risk of training stalls or instability. Residual connections also help in preserving low-level features, which is vital for maintaining high-quality output in extended audio sequences.

4.2.5 Handling Artifacts and Post-Processing

Although stacking latents offers a way to maintain continuity over time, it can lead to temporal artifacts, affecting the smoothness of the audio output. To address this, we employ post-processing techniques such as edge smoothing through averaging, blending transitions between audio chunks to reduce harsh edges and abrupt changes. This method effectively minimizes disruptive artifacts, though we acknowledge that more advanced methods could further reduce such inconsistencies, offering a path for future system refinement.

4.2.6 Impact of Design Choices

The design of our architecture, with its focus on 3D convolutions, cross-attention, self-attention, and residual connections, is guided by principles outlined in [19]. These design choices enable our model to generate extended, high-quality audio sequences that exhibit a high degree of temporal coherence with substantially minimized disruptive artifacts. This results in a smoother auditory experience, essential for creating background music tracks suitable for pairing with a wide variety of video content.

Chapter5

Experiments and Results

5.1 Data Preprocessing

5.1.1 Metadata Extraction and Enhancement

The preprocessing of the dataset commences with the extraction of metadata from YouTube, encompassing tags that describe each video and its corresponding musical components. These tags furnish basic information regarding the content.

To augment this information, an Automatic Tag Prediction Model [22] was utilized to generate additional tags from the audio content. This model aids in the identification of more specific audio features that the initial tags may overlook.

5.1.2 Generating Contextual Captions

The Mistral LLM [23] was employed to convert these tags into captions. Specific prompts were used to guide the caption generation process, ensuring that the captions accurately mirror the underlying content of the videos and their audio tracks. This methodological approach is instrumental as it produces more detailed descriptions, enhancing comprehension of the context. These enriched captions are crucial for the effective training of the model, providing the nuanced information required to generate music that aligns contextually with the video content. Subsequently, these captions were encoded using a T5 encoder [24].

5.1.3 Video Content Processing

The video data preprocessing involved extracting frames from each video, sampling at a rate of one frame per second to balance detail with processing efficiency. Following this, a CLIP Encoder [20] was used to extract embeddings from these frames. To synthesize a coherent representation of each video, these embeddings were aggregated to form a single embedding for the entire video clip. This aggregated embedding provides a consolidated view of the video content, which is crucial for aligning the generated music with the visual elements.

5.1.4 Audio Data Transformation

The audio data were segmented into 10-second chunks, which were then processed using the AudioLDM model's Variational Auto Encoder (VAE). This VAE first converts the audio into a Short Time Fourier Transform (STFT [2.2.6]) format. Subsequently, the encoder within the VAE encodes the audio into its latent representation. To effectively capture temporal dynamics, sequences of three consecutive latents were combined to form a single training data point. This combination was achieved by stacking three continuous latents to model it as a sequence in an additional dimension. Each latent representation of a 10-second audio segment has dimensions c, h, and w, specifically 8, 256, 16 respectively. When three such latents are stacked, they result in dimensions c, s, h, w where s represents the sequence length, culminating in a shape of 8, 3, 256, 16. This structure is denoted as (batch, channels, sequence_length, height, width), effectively preparing the audio data for training by modeling the sequential relationships within the audio.

5.2 Training

The training of the model is conducted until convergence, utilizing a linear learning rate decay schedule paired with the AdamW Optimizer. This optimization is performed on two

A100 GPUs, which are well-suited for handling the computational demands of the training process. The linear decay in the learning rate helps in stabilizing the training as it progresses towards convergence.

5.2.1 Performance Monitoring

Monitoring of the model's performance occurs every k epochs. During these checks, two primary metrics are assessed: the Frechet Audio Distance (FAD) [25] and the CLAP similarity score [26]. These metrics provide insights into the quality and similarity of the generated audio compared to the original dataset. In addition to these automated metrics, listening tests are conducted to ensure that the audio quality meets the subjective standards expected by human listeners.

5.2.2 Loss Function and Observations

The loss function used during the training fundamentally aims to minimize the mean squared error between the predicted and target noise in the latent dimension vector. Over the course of training, it is observed that the loss decreases very slowly, indicating a point of diminishing returns. However, despite the slow reduction in loss, the quality of the samples continues to improve. This improvement is evidenced by better scores in both FAD and CLAP metrics, justifying the continuation of training until almost complete saturation is achieved.

5.2.3 Classifier Guidance and Robustness

An essential part of our training regimen includes employing classifier guidance. This technique involves independently dropping each of the conditioning features 10 percent of the time. The purpose of this strategy is to prevent the model from becoming overly reliant on any single feature, thus enhancing its robustness and generalization capability during inference.

5.2.4 Classifier-Free Guidance and Sample Generation

Classifier-free guidance [27] is another technique used, where the model is trained to generate samples without any explicit class labels or conditioning. During inference, the outputs from the conditional and unconditional paths are interpolated to control the trade-off between sample diversity and fidelity to the conditioning information. This is mathematically expressed as:

$$x_{\text{guided}} = \gamma \cdot x_{\text{conditional}} + (1 - \gamma) \cdot x_{\text{unconditional}}, \tag{5.1}$$

where x_{guided} represents the final guided sample, $x_{\text{conditional}}$ is the sample with conditioning, $x_{\text{unconditional}}$ is the sample without conditioning, and γ is the guidance scale factor that is tuned to achieve the desired balance between diversity and fidelity.

5.2.5 Output Generation

For generating outputs, both unconditional and conditional samples are produced at each training step. Subsequently, a linear interpolation along the guidance scale is performed to merge these samples into the final output. This method ensures a balanced approach, allowing the model to produce high-quality audio that is both diverse and aligned with the conditioning data.

5.3 Evaluation Metrics

5.3.1 Frechet Audio Distance (FAD)

To assess the quality of the generated music, the Frechet Audio Distance (FAD) [25] is utilized. This metric employs the VGGish model to compare the distributions of the generated music against real music samples. A lower FAD score indicates a closer match to the real music distribution, signifying higher fidelity and, therefore, superior quality of the generated

background music.

While FAD provides a valuable quantitative measure of similarity between generated and real audio distributions, it has certain limitations:

- **Sensitivity to Noise:** FAD can be sensitive to background noise and audio artifacts, which may not necessarily affect the perceptual quality of the music from a listener's perspective.
- **Dependence on Feature Extraction Model:** The efficacy of FAD heavily relies on the feature extraction capabilities of the VGGish model. If VGGish does not capture all relevant musical characteristics effectively, the FAD scores may not fully reflect the true audio quality.
- **Lack of Perceptual Correlation:** FAD scores do not always correlate perfectly with human perception, meaning that a low FAD score does not guarantee that listeners will judge the music as high-quality or appropriate for its intended use.

These limitations highlight the importance of using FAD alongside other subjective and objective measures to provide a more comprehensive evaluation of music quality.

5.3.2 CLAP (Contrastive Language-Audio Pretraining)

The CLAP metric [26], which involves a pre-trained neural network on (audio, text) pairs, projects audio and corresponding textual descriptions into a shared representation space. The cosine similarity between these projections is measured to evaluate the relevance of the generated audio to the actual ground truth. This metric is crucial for ensuring that the generated background music aligns well with its intended narrative context, maintaining thematic consistency.

5.3.3 Low-Level Music/Audio Features

In addition to high-level similarity metrics, the analysis of low-level music/audio features plays a critical role in evaluating the generated music. These features include the spectral centroid, chroma, and spectral contrast, each corresponding to specific musical traits that influence the perception of the background music:

- **Spectral Centroid:** This feature correlates with the perceived brightness of sound, affecting the mood and energy conveyed by the background music. A higher spectral centroid typically indicates a brighter, more lively sound.
- **Chroma:** It captures the intensity of different pitch classes in the music, facilitating the identification of harmonic and melodic structures within the background music.
- **Spectral Contrast:** Measures the contrast between the peaks and valleys in the sound spectrum, relating to the timbral texture and richness of the music.

For each of these features, both the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE) are calculated over 30-second durations for each audio sample. These metrics are averaged to provide a comprehensive measure of the model's ability to replicate the dynamics and characteristics of the target audio. This detailed analysis offers insight into the precision with which our model generates audio that aligns with specific acoustic properties, ensuring that the synthesized music matches the desired mood and thematic elements of the background music.

5.3.4 Comprehensive Assessment

Together, these metrics furnish a multifaceted understanding of both the subjective and objective quality of the generated music. They facilitate a thorough assessment of the music's suitability for its intended use, allowing for targeted improvements and refinements in the model's performance.

5.4 Results and Analysis

Following the training phase, the generation capabilities of both our model, PixMus, and AudioLdm [18] were evaluated using a standardized test set for music generation. The generation process was uniformly applied to both models, involving 100 steps with a classifier-free guidance (CFG) scale of 3.5, to ensure a fair and direct comparison of their performance.

5.4.1 Comparative Analysis

Upon generating music with the specified parameters, the outputs from both PixMus and AudioLdm [18] were subjected to a comparative analysis. This evaluation focused on assessing the quality and coherence of the generated music, considering the complex interplay of melody, harmony, and rhythm that characterizes compelling musical compositions.

5.4.2 Quantitative Outcomes

The evaluation involved diverse metrics to thoroughly compare PixMus and AudioLdm. Notably, AudioLdm2-music, with a parameter count of 350M Unet Params, registered a higher Fréchet Audio Distance (FAD) of 2.87, suggesting a potential disparity from the target distribution of audio features compared to PixMus, which achieved a significantly lower FAD of 0.66 with its 405M parameters. This indicates that PixMus may generate audio samples that are closer to the target distribution, potentially perceived as higher in quality or naturalness.

In terms of CLAP Score, PixMus achieved a score of 0.732, surpassing the 0.575 score of AudioLdm2-music, reflecting PixMus's capacity to generate more coherent and well-formed audio. Additionally, the Root Mean Square Error (RMSE) for the Spectral Centroid was slightly lower for PixMus at 0.269 compared to 0.275 for AudioLdm2-music, indicating a slight advantage for PixMus in maintaining the central tendency of the spectral distribution, an important aspect of timbral quality.

5.4.3 Spectral Features Comparison

When examining Spectral Contrast, both models exhibited comparable RMSE values, with PixMus slightly edging out at 0.1463 against AudioLdm2-music's 0.1458. This near-parity highlights the models' similar effectiveness in capturing the amplitude differences between peaks and valleys in the sound spectrum. The Chroma RMSE scores, reflecting the models' accuracy in capturing harmonic content, were lower for PixMus at 0.390 compared to 0.414 for AudioLdm2-music, pointing to PixMus's slightly superior ability to model the chromatic features of music, a crucial aspect of harmony.

5.4.4 Spectrograms and Visual Analysis

In 5.2, spectrograms of one of the test samples are presented, offering a high-level comparison between the Short-Time Fourier Transform (STFT) outputs of the ground truth, our PixMus model, and AudioLDM. These spectrograms indicate that PixMus closely mirrors the ground truth, more so than AudioLDM, particularly in terms of energy distribution throughout the frequency and time domains.

In **Figure 5.2**, we present the spectrograms for one of our test samples, offering a high-level view of the comparison between the Short-Time Fourier Transform (STFT) outputs of the ground truth, our PixMus model, and AudioLDM. The spectrograms indicate that PixMus closely mirrors the ground truth, more so than AudioLDM, particularly in terms of energy distribution throughout the frequency and time domains.

5.4.5 Summary

In summary, the results underscore PixMus's robustness in generating high-fidelity audio, evidenced by its overall superior performance across evaluated metrics. These results suggest that PixMus is more adept at creating audio that is not only qualitatively superior but also quantitatively more aligned with the intrinsic characteristics of natural musical compositions.

Further analysis and discussion of these results will be essential for understanding the specific strengths and weaknesses of each model, informing future developments and enhancements aimed at generating musically coherent and aesthetically pleasing compositions.

Metrics	**audioldm2-music**	**PixMus**
Params	350M Unet Params	**405M**
Generation Steps	100	**100**
FAD $\downarrow$	2.87	**0.66**
CLAP Score $\uparrow$	0.575	**0.732**
Spectral Centroid RMSE $\downarrow$	0.275	**0.269**
Spectral Contrast RMSE $\downarrow$	0.1458	**0.1463**
Chroma RMSE $\downarrow$	0.414	**0.390**

Table 5.1: Evaluation on Test Data consisting of 481 samples

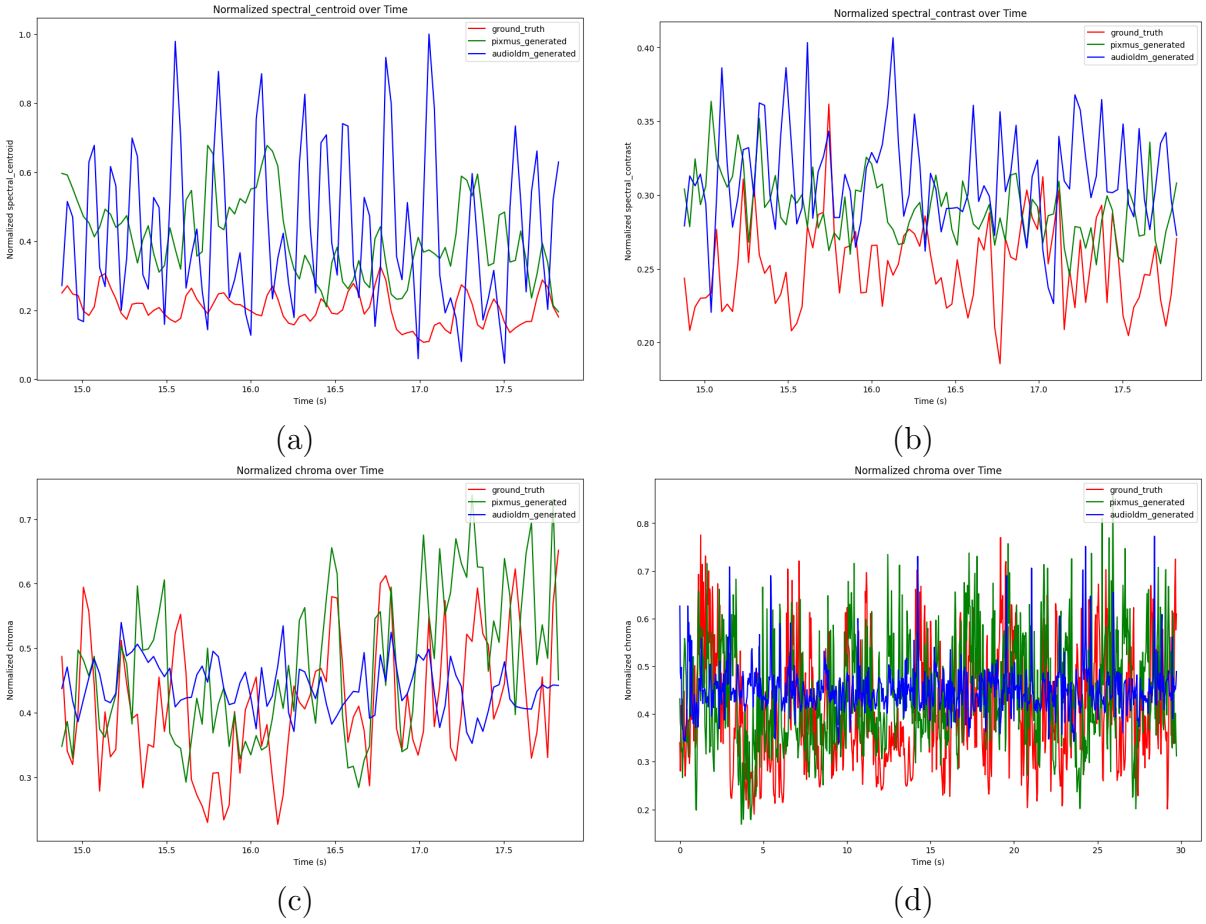


Figure 5.1: Analysis of Low Level Features for a specific data point reveals the following: (a) The Spectral Centroid, (b) Spectral Contrast, and (c) Chroma Vector are depicted for the central 3-second segment (for better visualization), whereas (d) Full Sequence Chroma represents the complete 30-second duration. It's evident that the AudioLdm [6] outcomes tend to cluster around a central point and exhibit oscillations, leading to rather generic creations. Conversely, the patterns generated by our model demonstrate an attempt to more accurately mirror the ground truth patterns.

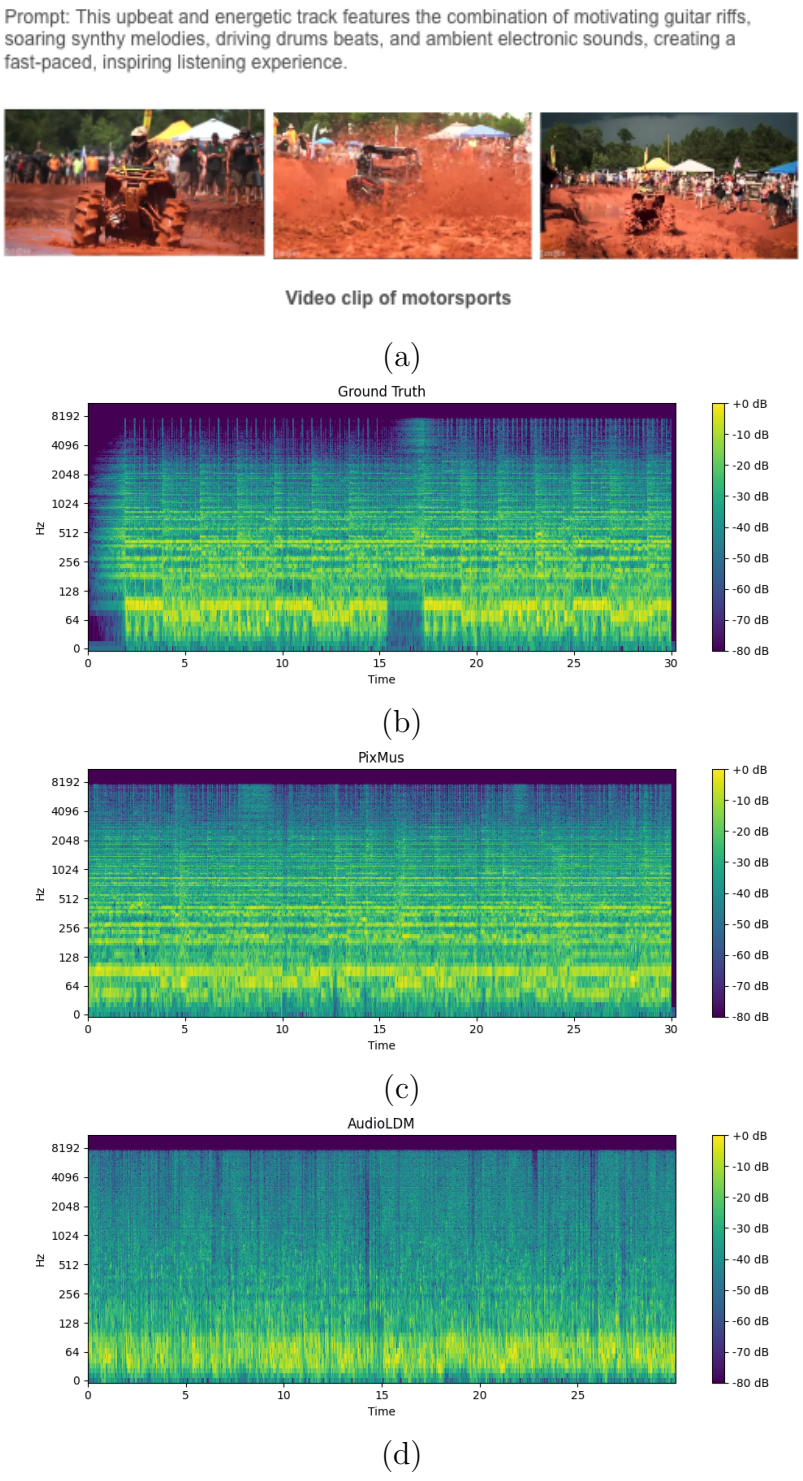


Figure 5.2: Sample video of adrenaline motorsports and the prompt used to generate audio. (a) Video frames; (b) Ground truth STFT; (c) PixMus Generated audio STFT; and (d) AudioLDM Generated STFT. Visual similarity score between GT and PixMus = 0.970 and GT and AudioLDM = 0.944.

Chapter6

Conclusion

In conclusion, the proposed novel approach demonstrates the successful application of a diffusion model conditioned on video content to synthesize background music that resonates with the visual narrative. Our model, PixMus, leverages latent representations to generate music aligned with the emotional and thematic elements of videos, showcasing the versatility of diffusion models in audio-visual content creation. The empirical results, as depicted in Table 2, establish PixMus as a formidable approach, yielding lower Frechet Audio Distance (FAD) and higher CLAP scores compared to the baseline model, AudioLdm2-music [18]. This performance reflects PixMus's nuanced understanding of music generation, evidencing its ability to create more natural and coherent musical pieces. Notably, PixMus's advantage in terms of lower RMSE values for spectral features further validates the model's sophistication in capturing essential audio characteristics. Remarkably, these accomplishments were achieved despite the constraint of a relatively small dataset, underscoring the model's efficiency and setting the stage for future advancements.

Chapter7

Future Work

Looking forward, the potential for PixMus to achieve even greater heights is evident. Expanding training with larger and more diverse datasets could significantly enhance the model's capabilities, propelling it towards production-quality standards suitable for widespread application. Future research could also focus on sourcing higher-quality datasets and experimenting with upsamplers for enhanced audio generation quality. The current generation and inference times are not yet practical; therefore, ongoing improvements in the diffusion model space could be adapted to enhance efficiency. Since the model is limited to generating fixed-size outputs, exploring a hybrid approach combining diffusion with autoregressive transformers presents a compelling direction. For example, diffusion could be used to train quantizers for improved audio/music representation, which could then be modeled as an autoregressive task to produce outputs of variable lengths and ensure seamless continuity. Additionally, our contribution to the community through the creation and future release of a novel dataset containing over 53,000 samples will catalyze further innovation in multimedia content creation. This availability is expected to spur advancements in the development of generative models capable of producing contextually rich and emotionally compelling music for videos.

Bibliography

[1] Sungeun Hong, Woobin Im, and Hyun S. Yang. *Content-Based Video-Music Retrieval Using Soft Intra-Modal Structure Constraint*. 2017. arXiv: 1704.06761 [cs.CV].

[2] Sami Abu-El-Haija et al. *YouTube-8M: A Large-Scale Video Classification Benchmark*. 2016. arXiv: 1609.08675 [cs.CV].

[3] Le Zhuo et al. *Video Background Music Generation: Dataset, Method and Evaluation*. 2023. arXiv: 2211.11248 [cs.CV].

[4] Shangzhe Di et al. "Video Background Music Generation with Controllable Music Transformer". In: *Proceedings of the 29th ACM International Conference on Multimedia*. MM '21. ACM, Oct. 2021. DOI: 10.1145/3474085.3475195. URL: http://dx.doi.org/10.1145/3474085.3475195.

[5] Zalán Borsos et al. *AudioLM: a Language Modeling Approach to Audio Generation*. 2023. arXiv: 2209.03143 [cs.SD].

[6] Haohe Liu et al. *AudioLDM: Text-to-Audio Generation with Latent Diffusion Models*. 2023. arXiv: 2301.12503 [cs.SD].

[7] Deepanway Ghosal et al. *Text-to-Audio Generation using Instruction-Tuned LLM and Latent Diffusion Model*. 2023. arXiv: 2304.13731 [eess.AS].

[8] Andrea Agostinelli et al. *MusicLM: Generating Music From Text*. 2023. arXiv: 2301.11325 [cs.SD].

[9] Qingqing Huang et al. *Noise2Music: Text-conditioned Music Generation with Diffusion Models*. 2023. arXiv: 2302.03917 [cs.SD].

[10] Jade Copet et al. *Simple and Controllable Music Generation*. 2024. arXiv: 2306.05284 [cs.SD].

[11] Aaron van den Oord et al. *WaveNet: A Generative Model for Raw Audio*. 2016. arXiv: 1609.03499 [cs.SD].

[12] Prafulla Dhariwal et al. *Jukebox: A Generative Model for Music*. 2020. arXiv: 2005.00341 [eess.AS].

[13] Marco Pasini and Jan Schlüter. *Musika! Fast Infinite Waveform Music Generation*. 2022. arXiv: 2208.08706 [cs.SD].

[14] Seth* Forsgren and Hayk* Martiros. "Riffusion - Stable diffusion for real-time music generation". In: (2022). URL: https://riffusion.com/about.

[15] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2022. arXiv: 2112.10752 [cs.CV].

[16] Peike Li et al. *JEN-1: Text-Guided Universal Music Generation with Omnidirectional Diffusion Models*. 2023. arXiv: 2308.04729 [cs.SD].

[17] M. Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer International Publishing, 2016. ISBN: 9783319357652. URL: https://books.google.com/books?id=Wl8ivgAACAAJ.

[18] Hao-Wen Dong et al. *MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment*. 2017. arXiv: 1709.06298 [eess.AS].

[19] Jan Melechovsky et al. *Mustango: Toward Controllable Text-to-Music Generation*. 2024. arXiv: 2311.08355 [eess.AS].

[20] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV].

[21] Omer Bar-Tal et al. *Lumiere: A Space-Time Diffusion Model for Video Generation*. 2024. arXiv: 2401.12945 [cs.CV].

[22] Minz Won et al. *Evaluation of CNN-based Automatic Music Tagging Models*. 2020. arXiv: 2006.00751 [eess.AS].

[23] Albert Q. Jiang et al. *Mistral 7B*. 2023. arXiv: 2310.06825 [cs.CL].

[24] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

[25] Kevin Kilgour et al. *Fréchet Audio Distance: A Metric for Evaluating Music Enhancement Algorithms*. 2019. arXiv: 1812.08466 [eess.AS].

[26] Yusong Wu et al. *Large-scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword-to-Caption Augmentation*. 2024. arXiv: 2211.06687 [cs.SD].

[27] Jonathan Ho and Tim Salimans. *Classifier-Free Diffusion Guidance*. 2022. arXiv: 2207.12598 [cs.LG].