Ainur: Enhancing Vocal Quality through Lyrics-Audio Embeddings in Multimodal Deep Music Generation

Thesis by
Giuseppe Concialdi

In Partial Fulfillment of the Requirements for the
Degree of
Master of Science in Data Science and Engineering

POLITECNICO DI TORINO
Turin, Italy

2023
Defended 15/06/2023

To Love, the immutable engine propelling life, uniting family, friends, and lovers. Without whom, this work would not be.

ACKNOWLEDGMENT

I am profoundly grateful to my supervisors, Dr. Eliana Pastor and Dr. Alkis Koudounas, for their continuous support and guidance. Our regular meetings and their insightful suggestions were instrumental in the completion of this thesis.

A special thanks to Dr. Koudounas for his thorough revision and invaluable advice on the thesis layout, which significantly enhanced the quality of my work.

Lastly, I wish to acknowledge the crucial computational resources provided by HPC@POLITO (http://www.hpc.polito.it) which were indispensable to this research.

To all those mentioned and to those who have contributed silently in the background, I extend my heartfelt gratitude. This thesis stands as a testament to your support, guidance, and faith in my abilities.

GC

TABLE OF CONTENTS

CHAPTER **PAGE**

CHAPTER PAGE

CHAPTER PAGE

TABLE OF CONTENTS (continued)

CHAPTER **PAGE**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (continued)

FIGURE **PAGE**

LIST OF ABBREVIATIONS

AI	Artificial intelligence
AR	Autoregressive
CM	Consistency models
CLASP	Contrastive Lyrics-Audio Spectrogram Pre-training
CLAP	Contrastive Language-Audio Pre-training
CLIP	Contrastive Language-Image Pre-training
CNN	Convolutional neural network
CPC	Contrastive predictive coding
CQT	Costant-Q transform
DAG	Directed acyclic graph
DDIM	Denoising diffusion implicit model
DDPM	Denoising diffusion probabilistic model
DLVM	Deep latent variable model
GAN	Generative adversarial network
GRU	Gated recurrent unit
LDM	Latent diffusion model
LSTM	Long-short term memory

LIST OF ABBREVIATIONS (continued)

MFCC	Mel-frequency cepstral coefficient
MLP	Multilayer Perceptron
MSE	Mean squared error
NADE	Neural autoregressive distribution estimation
NAR	Non-autoregressive
NF	Normalizing flows
NLP	Natural language processing
NN	Neural network
ODE	Ordinary differential equation
RNN	Recurrent neural network
SOTA	State-of-the-art
STFT	Short-time Fourier transform
TTS	Text-to-speech
VAE	Variational autoencoder

SUMMARY

As an emerging research field, deep music generation faces significant challenges, such as handling high-dimensionality of audio data, computational resource requirements, and quality concerns, particularly with generated vocals. This study aims to address these concerns by introducing Ainur, an innovative deep learning model designed specifically to enhance the quality of generated vocals.

We investigate the effectiveness of various deep learning techniques and multimodal input conditioning strategies to improve vocal generation. Additionally, the utility of transfer learning and pre-trained models is examined, along with the impact of multimodal input strategies on the quality and diversity of the produced music. Ainur employs a hierarchical diffusion model and a latent diffusion prior for handling high-dimensional data and uses Contrastive Lyrics-Audio Spectrogram Pre-training (CLASP) embeddings for multimodal data fusion. Our findings reveal Ainur's capability to produce high-quality and varied music, substantiating the use of our proposed novel evaluation metrics.

The study also acknowledges the importance of ethical considerations and limitations inherent to deep music generation. Recognizing the potential implications of AI-generated music on creative integrity, and the potential misuse of such technology, we emphasize the need for responsible use. This work significantly contributes to the deep music generation field, establishing novel methodologies, offering robust tools, and providing directions for future research, while promoting collaboration and transparency through the open-source nature of Ainur.

CHAPTER 1

INTRODUCTION

Music is an art form that has always been associated with human creativity, emotion, and expression. However, with the advent of computer technology, the idea of generating music through machines became a reality. *Computer-generated* or *synthetic music* refers to music created automatically by computer programs or algorithms. The potential applications of computer-generated music range from music production to education and entertainment. In music production, it can be used as a tool for generating new ideas and exploring different styles and genres. In education, it can be used to teach music theory and composition and provide students with opportunities to experiment with different musical ideas. In entertainment, it can be used to create original soundtracks for films, video games, and other media. Furthermore, the development of these systems has the potential to democratize music production, allowing aspiring musicians and composers to generate high-quality music without the need for expensive studio equipment or formal training. The development of deep learning architectures has been particularly transformative for the field of music generation. These architectures are designed to learn from large volumes of data, such as audio recordings or sheet music, and use this data to generate new musical compositions that are technically proficient and stylistically consistent. In this work, we will delve into the intricacies of music generation focusing on modern deep learning architectures and emphasizing the role of multimodal conditioning in guiding and improving the quality of the synthesis.

1.1 Research Questions

In this research, we investigate the domain of deep music generation by employing the latest state-of-the-art architectures for generation and exploring multimodal input conditioning to steer the generation process. Music generation has received less attention than image and text generation and is still an underdeveloped field with several challenges that limit its scalability and reproducibility. These issues primarily arise from the lack of readily available copyright-free data and the high dimensionality of audio data, making audio generation computationally intensive and requiring high-performance computing resources. Despite these challenges, the potential benefits of developing successful music generation models are enormous, and this research aims to contribute to the advancement of the field by investigating various deep learning techniques and multimodal input conditioning strategies to generate high-quality musical compositions.

This thesis focuses on enhancing the quality of the vocals generated in the field of deep music generation. While prior research has successfully generated high-quality music by conditioning it on text descriptions, the quality of the vocals produced has been inadequate and often unintelligible. To address this issue, this thesis aims to answer the following research questions:

1. What deep learning techniques can be employed to improve the quality of vocals generated in the field of deep music generation?
2. How can multimodal input conditioning strategies be leveraged to generate vocals that are coherent and consistent with the overall theme and mood of the generated music?

3. Can the use of transfer learning or pre-trained models effectively improve the quality of vocals generated in deep music generation, and if so, what are the best approaches for their implementation?

4. How do different multimodal input conditioning strategies, such as combining text, image, and symbolic musical representations, impact the quality and diversity of the generated music?

The research questions listed above will guide the investigation, and extensive experimentation and ablation will be conducted to provide empirical evidence of the results. The findings of this study are expected to provide valuable insights into the challenges and opportunities in the underdeveloped field of music generation, thereby contributing to the advancement of the field.

1.2 Contribution

Our research makes significant strides in the field of deep music generation. Let us delve into each major contribution, as each one sets a precedent, encouraging future work in this compelling area of research.

- **CLASP model and embeddings**: One of the linchpins of our work is the adaption and innovative use of the Contrastive Lyrics-Audio Spectrogram Pre-training (CLASP) model and embeddings for music generation. In the quest to provide a richer, more holistic approach to music generation, we harness CLASPs unique ability to integrate lyrics and audio spectrograms to build a comprehensive multimodal representation of the music.

By intertwining such diverse modalities, our approach has the potential to fundamentally change how deep learning perceives and interprets music, thus opening up exciting avenues for future research.

- **Hierarchical diffusion model**: Breaking away from conventional architectures, our study introduces a novel hierarchical diffusion model, an extension of a diffusion autoencoder architecture. The purpose of this model is twofold: to infuse a higher level of quality into the generated music, and to inject an extra layer of sophistication into the music generation process. With this innovative approach, we aim to facilitate deeper exploration into music generation, augmenting the richness of the music while keeping the computational complexity in check.
- **Single GPU inference and training**: In an era dominated by extensive computational needs, we champion the cause of accessibility with Ainur. Ainur, our proposed model, is specially designed to run on single, consumer-grade GPUs. This strategic decision brings state-of-the-art music generation within the reach of many, while simultaneously mitigating the need for massive computational resources. As we tread the path of democratizing advanced music generation techniques, we believe Ainur's contributions will resonate within the research community and beyond, thus encouraging further advancements in the field.
- **Lyrics-to-music generation**: A key highlight of our work is the birth of Ainur, a deep learning model specifically tailored to address the challenges of lyrics-to-music generation. Through Ainur, we direct attention towards an underexplored domain within deep music

generation, and set the stage for further exploration in this direction. This step could potentially act as a catalyst for future advancements, creating a ripple effect in the field of music generation.

C3 and FAD evaluation metrics: Finally, our research gives rise to two new evaluation metrics - the C3 evaluation metric and the FAD evaluation with the YAMNet model. These robust methods promise to pave the way for objective, standardized, and uniform assessment of music generated by deep learning models. We believe the introduction of these evaluation mechanisms will have profound implications for the field, providing a common benchmark against which future models can be evaluated and compared.

The multitude of contributions from this research hopes to inspire further advancements in the field of deep music generation, introducing new methodologies, establishing benchmarks, and making high-quality music generation a practical reality.

1.3 Open Source

This research project aims to make all its outputs, including pre-trained models and user interfaces for downstream tasks, publicly available on a GitHub repository. By doing so, we seek to promote openness and collaboration in the field of deep music generation, allowing other researchers, practitioners, and users to access the tools developed in this study.

The pre-trained models and the user interface are designed to cater to the needs of researchers and practitioners who are interested in improving and advancing the state-of-the-art in music generation. Additionally, users can query the model for generating musical compositions, and they will hold the intellectual property rights to the resulting outputs.

The model and its components are not intended for commercial use, and there are no profit-making intentions behind this application. Instead, the primary goal is to contribute to the academic community and make a positive impact on the field of deep music generation. By providing access to the model and its outputs, this research hopes to inspire further innovation and foster collaboration in the field.

1.4 Structure of the Thesis

This thesis is arranged according to the structure delineated in Figure 1. We begin in Chapter 2, which supplies the necessary background information and elucidates key topics that form the bedrock of the approaches and techniques used throughout this thesis.

Next, in Chapter 3, we conduct a detailed exploration of the significant literature that has influenced the development of this research and advanced the field of deep music generation. This chapter considers not only the works directly related to raw music generation but also pioneering studies from the realms of text-to-speech, video generation, and symbolic music generation. These areas have generated key insights that can be fruitfully applied to the domain of music generation. The literature review within this chapter has been arranged chronologically to underscore the progression of influential works in deep music generation from the advent of the deep learning era, grouped by the corresponding generative model family.

The main body of the thesis commences with Chapter 4, which introduces Ainur, a novel model designed for multimodal conditional music generation. In this chapter, we provide a thorough examination of the model's architecture, its use of multimodal input conditioning, and the complete workflow of the model.

We then move to Chapter 5, which outlines the comprehensive evaluation procedure employed to assess the outputs generated by Ainur. The outcomes of the rigorous experimentation phase and the consequential ablation studies are then presented in Chapter 6.

Chapter 7 follows next, where we discuss potential enhancements and innovative ideas that emerged during the development of Ainur, providing a roadmap for future research directions.

Lastly, we conclude the thesis with Chapter 8, summarizing our key findings, reflecting on the implications of our research, and pointing toward future prospects in the field of deep music generation.

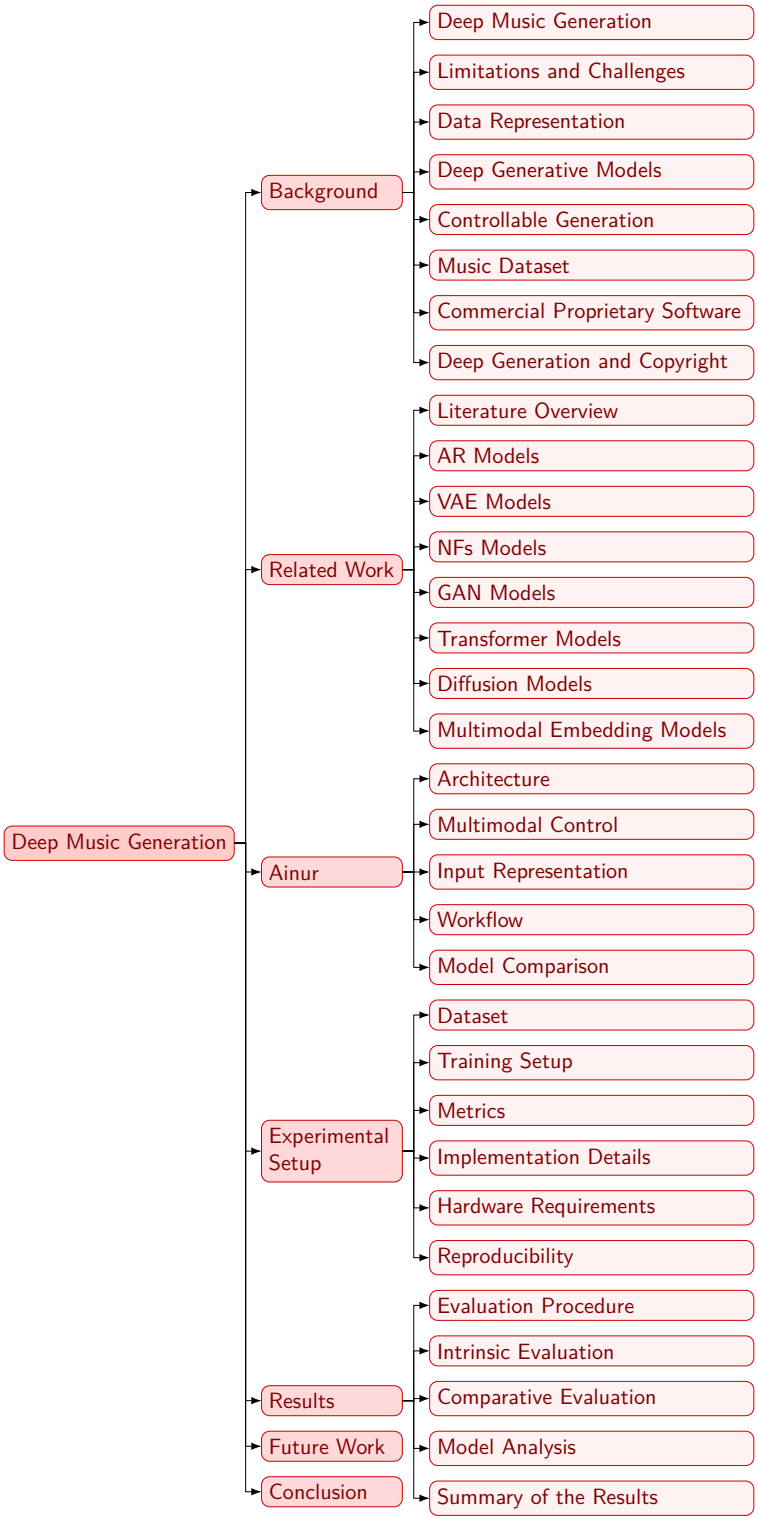

Figure 1: Structure of the thesis.

CHAPTER 2

BACKGROUND

2.1 Deep Music Generation

Deep music generation refers to the task of creating new and unseen samples of music through deep learning architectures. Combining deep learning techniques with multimodal conditioning has enabled the incorporation of various types of data, such as images, text, and other perceptual input, to guide and improve the quality of the generated music. The multimodal-guided generation has opened up a new world of creative possibilities, allowing for the production of music that is not just artificially generated but also personalized and tailored to specific contexts and themes.

2.1.1 Timeline

Synthetic music has come a long way since its inception in the mid-twentieth century. The development of computer technology in the 1950s and 1960s paved the way for experiments with automated music composition, leading to the emergence of computer-generated music in the following decades. Since then, computer-generated music has undergone significant advancements and has become a thriving field of research, incorporating various techniques and applications.

Early Years. The origins of synthetic music can be traced back to the early experiments with automated music composition in the 1950s and 1960s. Despite many unpublished or

unpopular projects carried out in 1955 and 1956 [23], computer-generated music debuted in 1957 with the creation of a 17-second song called "The Silver Scale" [6]. The composition was the creation of Newman Guttman, and it was brought to life by a software known as Music I. This program, specifically engineered for sound synthesis, was the brainchild of Max Mathews, a pioneering figure in computer music, during his tenure at Bell Laboratories. During the same year, the Illiac Suite composition [33] was released. It consisted of four pieces of music that were generated by an electronic computer called the Illiac I. The Illiac I was one of the earliest electronic computers developed by the University of Illinois at Urbana-Champaign. The Illiac Suite was composed using several sets of rules and probabilities to determine which notes to play and when to play them, based on the principles of Markov chains. In the following years, other researchers experimented with automated music composition using different methods, such as algorithmic composition and rule-based systems [23].

Deep Learning Era. While the concept of creating music using machines has existed for several decades, it was only with the development of modern deep learning architectures that the generation of highly complex and nuanced musical compositions became possible. These architectures have allowed for the creation of music that can rival the quality of human-generated music. The impressive victory of the AlexNet architecture [55] at the 2012 ImageNet large-scale visual recognition challenge marked the renaissance of deep learning. Several deep learning architectures have been employed for the task of music generation or, more in general, for audio synthesis.

2.2 Limitations and Challenges

Deep music generation faces several unique challenges and limitations that distinguish it from other fields of deep learning, such as image generation and natural language synthesis. Unlike images and text, raw audio data is highly dimensional, and downsampling or compression of audio data can lead to a loss of quality, making the training of deep generative models for music generation incredibly challenging. For instance, a medium-sized 256×256 RGB image has approximately the same dimensionality as only two seconds of stereo audio sampled at 48,000 kHz [91]. Moreover, music is inherently a human form of artistic expression, and the lack of a coherent framework for music generation can result in unnatural and incoherent music creation. Additionally, music requires a structure and coherence that follows the rules of music theory. To address these challenges, researchers have proposed various techniques such as incorporating music theory into the training process, conditioning the model on musical features, and incorporating randomness into the training process to encourage creativity and originality [44].

2.2.1 Originality

Despite high-level similarities among songs belonging to the same genre, each song possesses individual features that make it unique. The *originality* properties of music arise from the creative work of composers and artists and include the ability to create new musical structures and patterns that have not been previously heard, as well as the capacity to produce music that is emotionally evocative and aesthetically pleasing. Deep music generation tries to abstract the process of composition by training over hours of song data and trying to replicate the recurring

patterns that underlie the musical structure. Theoretically, sampling from the distribution of a fully pre-trained generative model should always produce unique results different from the samples used during the training. However, in practice, many tests are conducted in order to ensure that the samples are fairly different and do not plagiarise in any way the existing and copyrighted works [1] (more about copyright in Section 2.8).

2.2.2 Variability

In addition to originality, variability is another important property that deep music generation systems must possess. Variability refers to the ability of the system to produce music that is diverse and interesting, with a wide range of musical elements and styles.

- From a *content* perspective, this means that the system should be able to generate music that is melodically and harmonically varied, with a mix of simple and complex patterns and rhythms. Additionally, the system should be able to generate diverse music in terms of its instrumentation, timbre, and texture.
- From a *temporal* perspective, variability refers to the ability of the system to produce music that changes over time, with a sense of progression and development. This requires the system to be able to generate musical phrases and motifs that evolve and develop over time with a sense of structure and coherence.

Achieving these variability properties is challenging, as it requires the system to be able to learn and model a wide range of musical patterns and structures and to be able to generate novel and interesting variations of these patterns.

2.2.3 Controllability

Controllability is a critical property for deep music generation systems, enabling users to guide and direct the creative output of the system according to their preferences and requirements. This requires the system to be able to respond to a wide range of external inputs and constraints, including user-defined criteria such as key, tempo, and genre, as well as other contextual factors such as the emotional content of the music. In order to achieve controllability, deep music generation systems must incorporate sophisticated algorithms and models that are able to take into account these external factors and generate music that is both musically coherent and aligned with the user's preferences.

2.2.4 Coherence

Coherence in music refers to the way in which different musical elements, such as melody, harmony, and rhythm, work together to create a sense of unity and cohesion. Achieving coherence in music generation is a formidable challenge, particularly in the case of multitrack audio with vocals, where the generation process must synchronize diverse audio modalities and harmonize them in a compelling manner. This poses a substantial obstacle to generative models, which lack access to all modalities of the data, and require human intervention to optimize coherence in music composition. The attainment of coherence is highly valued in music generation and demands a deep understanding of musical structure and form, as well as the capacity to incorporate creative and improvisational elements in the music generation process.

2.2.5 Discussion

Despite the considerable progress made in deep music generation research, there remain several challenges and limitations that need to be addressed. One of the primary limitations is the difficulty in achieving a balance between originality and coherence. While originality is important for generating novel and interesting music, it can lead to a lack of coherence if not properly balanced with other properties. Additionally, there is often a trade-off between variability and coherence in deep music generation, as generating highly variable music can make it difficult to maintain a consistent sense of musical structure and coherence.

Another challenge is the difficulty in evaluating the results in deep music generation [101]. While there has been significant progress in developing objective metrics that are useful to compare different generative models, there remains a need for more sophisticated evaluation metrics that are highly correlated with the perception of audio and that can replace the subjective evaluation that is time-consuming and prone to errors [49; 34; 109; 61].

Furthermore, deep music generation systems are still limited by the quality of the data they are trained on, which can affect the system's ability to generate high-quality music. This is particularly true in the case of audio with complex audio modalities, such as vocals, where the system's ability to generate coherent and musically satisfying music is often limited by the quality and diversity of the training data. The lack of a reference dataset and large datasets for the training makes it difficult for practitioners and researchers to study and develop techniques useful to overcome the challenges posed by this task [1].

In conclusion, while deep music generation has made significant progress in recent years, there remain several challenges and limitations that need to be addressed in order to achieve truly remarkable results. Overcoming these limitations will require continued research and development in the field, as well as the incorporation of new techniques and approaches from related fields, such as music theory, audio engineering, and cognitive science.

2.3 Data Representation

Audio data representations in a computer are crucial for various applications, including music production, speech recognition, and acoustic signal processing. Audio data is represented by the computer in the form of digital signals: audio is encoded by sampling and quantizing the continuous information provided by sensors that are able to detect the air vibrations that produce sound.

There are various types of audio data, each with its unique characteristics. The most common types of audio data are speech, music, and environmental sounds. Speech is characterized by its high-frequency content and narrow dynamic range. The human voice is a complex sound, with different frequencies occurring at different levels of intensity. In contrast, music is characterized by a wide range of frequencies and a high dynamic range. The dynamic range of music refers to the difference between the quietest and loudest sounds in a piece of music. Environmental sounds are characterized by their variability and unpredictability. These sounds can include natural sounds, such as wind and water, as well as man-made sounds, such as traffic and machinery.

The different nature of audio data requires different processing techniques. For example, speech recognition systems need to be able to distinguish between different phonemes and recognize the patterns of speech. In contrast, music production requires techniques such as equalization and compression to adjust the frequency balance and dynamic range of the audio. Environmental sound analysis requires techniques such as feature extraction and pattern recognition to identify and classify different sounds.

2.3.1 Raw Waveform

The raw waveform of a song is a time-domain representation of the audio signal that captures the instantaneous amplitude of the signal at each point in time. It is a 1-dimensional continuous representation of a signal, and it is characterized by the *sampling frequency* and the *bit depth*.

- The **sampling frequency** determines the quality and fidelity of the resulting audio signal. The sampling frequency, also known as the sampling rate or the Nyquist frequency, represents the number of samples per second that are taken to represent an analog signal digitally. Typically, the sampling frequency for raw waveform data is at least twice the highest frequency in the analog signal, according to the Nyquist-Shannon sampling theorem. The most common sampling frequency for audio is 44.1 kHz, which is the standard for audio CDs. Other common sampling frequencies include 48 kHz, 96 kHz, and 192 kHz, which are used in high-quality audio applications such as studio recording and audio mastering.
- The **bit depth** is a fundamental parameter that defines the precision and dynamic range of the audio signal. Bit depth refers to the number of bits used to represent each sample

of the digital audio signal. The higher the bit depth, the greater the dynamic range and precision of the audio signal. In raw audio data, the most common bit depth is 16 bits, which allows for a dynamic range of 96 dB. However, higher bit depths, such as 24 bits and 32 bits, are becoming increasingly popular in professional audio applications, as they provide greater dynamic range and accuracy.

This representation provides a highly detailed and accurate representation of the audio signal, allowing for high-quality and high-fidelity playback. However, generating audio directly as a raw waveform is a challenging task, as it requires generating a vast number of samples, each of which must be precisely controlled to create a coherent and realistic audio signal. Furthermore, the high dimensionality of the raw waveform representation makes it difficult to store and process large audio files efficiently. Generating long audio samples is a computationally intensive task that requires significant computing resources and specialized machine learning architectures.

2.3.2 2D Representation

The spectrogram representation of audio data is a widely used alternative to the raw waveform representation, which captures the spectral content of the audio signal over time. Unlike raw waveform data, spectrograms have a lower dimensionality, making them easier to process and store. However, the lower dimensionality of spectrograms comes at the cost of losing some of the fine-grained details of the audio signal, such as phase information.

Spectrograms can be computed using various mathematical techniques that involve analyzing the spectral content of the audio signal over time $x(t)$. One common method for computing spectrograms is the short-time Fourier transform (STFT), which involves dividing the audio

signal into small overlapping time windows and computing the Fourier transform of each window.

$$\mathrm{STFT}\{x(t)\}(\tau, \omega) = \int_{-\infty}^{+\infty} x(t)\omega(t-\tau)e^{-i\omega t}dt \tag{2.1}$$

This results in a time-varying frequency representation of the audio signal, where the amplitude of each frequency component at each point in time is represented as a color or grayscale value in the spectrogram. Other approaches to computing spectrograms include the Mel scale, which uses a non-linear frequency scale that approximates the human auditory system, and the constant Q transform (CQT), which uses logarithmically spaced frequency bins to capture the spectral content of the audio signal more accurately [69].

These different spectrogram computation methods can be used to improve the task of audio generation by providing a more compact and interpretable representation of the audio signal, enabling more efficient processing and analysis, and facilitating the design of more effective machine learning models for audio synthesis and processing.

Amplitude Spectrogram. An amplitude spectrogram is a type of spectrogram that represents the amplitude of each frequency component of an audio signal over time. Unlike a power spectrogram, which represents the square of the amplitude of each frequency component, an amplitude spectrogram represents the actual amplitude of each frequency component, providing a more direct and interpretable representation of the audio signal. The amplitude spectrogram $s(t, \omega)$ is computed using STFT over the waveform representation $x(t)$, and the amplitude

of each frequency component is then computed by taking the absolute value of the Fourier transform.

$$s(\tau, \omega) = |STFT(\tau, \omega)|^2 \tag{2.2}$$

The resulting spectrogram is represented as a matrix, where the rows represent frequency bins, the columns represent time windows, and the value at each element represents the amplitude of the corresponding frequency component at the corresponding time window.

When the STFT representation is processed by only considering its absolute value, the corresponding phase component of the spectrogram is lost. This loss of phase information in the representation renders it impossible to reverse the transformation. However, it is worth noting that the phase component is complex and difficult to generate, while the amplitude spectrogram is relatively easier to model. To overcome the challenge of the missing phase component, one can utilize a neural vocoder to estimate the discarded phase [94; 87; 53] or use phase approximation algorithms like Griffin-Lim [10].

Mel Spectrogram. The Mel spectrogram is based on the Mel scale, which is a non-linear frequency scale that approximates the human auditory system's perception of frequency. The mapping between frequency units f to Mel units m is the following:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{2.3}$$

The Mel scale is divided into a set of equally spaced frequency bins, and the power spectral density of the audio signal is then mapped onto these bins using a series of overlapping triangular

filters [68]. The resulting Mel spectrogram provides a more perceptually meaningful representation of the audio signal than a traditional spectrogram, as it emphasizes the frequency ranges that are most relevant for human hearing. The resulting Mel spectrogram is represented as a matrix, where the rows represent the Mel frequency bins, the columns represent time windows, and the value at each element represents the power spectral density of the corresponding Mel frequency bin at the corresponding time window.

Costant-Q Transform. The CQT is based on the frequency-domain transform that approximates the human cochlea's frequency response, which is more closely related to the logarithmic frequency scale than the linear frequency scale [113]. Unlike the traditional STFT, which employs a constant-width frequency filter bank, the CQT uses a variable-width filter bank. The variable-width filters ensure that each frequency bin in the CQT has a constant quality factor (Q-factor), which is defined as the ratio of the center frequency to the bandwidth of the filter. This results in a more efficient representation of the audio signal, with reduced redundancy and increased energy concentration, particularly for harmonic sounds. The CQT representation is also robust to changes in pitch and timbre, making it an ideal representation for tasks such as audio synthesis, transcription, and manipulation.

Rainbowgrams. Rainbowgrams are a type of CQT spectrogram that utilizes a color mapping scheme to visually represent the energy distribution of the audio signal in the time-frequency domain [69]. Rainbowgrams provide a more intuitive and aesthetically pleasing representation of the audio signal, allowing for better visualization of the harmonic and inharmonic structures

of the sound. The color mapping scheme used in the rainbowgrams typically ranges from blue for low energy to red for high energy, with green, yellow, and orange shades in between.

2.3.3 Latent

Latent audio data is a compressed form of representation that captures the underlying structure and features of the audio signal in a lower-dimensional space. This type of representation is generated using a neural network trained on a large dataset of audio signals. Latent representations have the advantage of being highly informative and containing more meaningful and abstract features than raw waveforms or spectrograms, as they can capture complex temporal and spectral patterns of the audio signal. This property of the latent representation makes it highly useful in tasks such as audio generation, music transcription, and speech recognition. However, the use of latent representations in audio analysis and processing requires careful consideration of the choice of network architecture, training data, and optimization procedures to ensure that the resulting representations are both informative and stable. Furthermore, the interpretation of latent features is often challenging due to their abstract nature [69].

Latent audio representations can be seen as audio embeddings similar to text embeddings used in natural language processing (NLP). They are both forms of compressed representations that capture the underlying structure and features of the signal or text in a lower-dimensional space. One key difference is that text embeddings operate on discrete symbols (words) with a clear semantic structure, whereas audio signals are continuous and do not have an inherent semantic structure. As a result, the representation of audio signals needs to be learned in a different way, for example, through a CNN or an autoencoder. Another difference is the nature

of the features captured by each representation. Text embeddings capture the semantic meaning of words, while latent audio representations capture the acoustic and temporal structure of the audio signal. This difference arises because the types of features that are informative for each domain are different.

One of the key advantages of latent representations is their ability to embed different modalities into a unified space, enabling a multi-faceted representation of the data. A notable example of such multimodal embeddings is the Contrastive Language-Image Pre-training (CLIP) framework [80], which learns a joint text-image embedding space through a contrastive learning objective. By aligning the text and image embeddings in the same space, CLIP enables cross-modal retrieval and manipulation of the two modalities. In the generative field, CLIP embeddings have been employed in various text-to-image tasks and have demonstrated impressive results. For instance, StyleCLIP [77] combines CLIP embeddings with styleGAN [48] to generate images with fine-grained control over their visual attributes. This approach has been shown to outperform state-of-the-art image generation methods on several benchmark datasets. The ability of CLIP embeddings to bridge the gap between text and image modalities has also been utilized in natural language processing tasks such as image captioning, visual question answering, and image retrieval.

2.3.4 Symbolic

Symbolic representation is a specific type of audio representation that is tailored for music. It utilizes knowledge of music theory and encodes audio data into a compact form that is easy to manipulate. The most commonly used form of symbolic music representation is MIDI, which

includes information about the timing, pitch, and duration of individual notes in a musical composition. Because MIDI files are highly compact, they are widely used in music generation tasks. However, a major limitation of symbolic representation is that it can lack the fine-grained details and nuances that are present in real musical compositions, which can lead to a less natural or artificial sound.

Additionally, the MIDI representation is not suitable for vocal representation since it is only used to encode synthetic sounds or synthesized voices. Therefore, it is necessary to use additional techniques to encode the vocals in music generation tasks. Despite these limitations, the low dimensionality and efficient representation of MIDI has made it a popular choice for music generation tasks.

2.3.5 Acoustic Features

Acoustic feature audio representation is a type of audio representation that leverages domain-specific knowledge to extract high-level features from audio signals. These features include mel-frequency cepstral coefficients (MFCCs), spectral centroid, and zero-crossing rate, among others. One of the main advantages of this representation is its ability to capture the timbral and rhythmic aspects of audio, making it useful in applications such as speech recognition and music genre classification. However, acoustic feature representations are not well-suited for audio generation due to their low-level nature and lack of information about higher-level musical concepts such as harmony and melody. Additionally, the accuracy of feature extraction can be influenced by environmental factors (such as the location where the audio is recorded) and noise, making the representation susceptible to variations.

2.4 Deep Generative Models

Deep generative models form a specific group within machine learning algorithms, whose main task is to comprehend the underlying pattern within a dataset and subsequently generate new data that mirrors the characteristics of the initial input. These models, usually built on neural networks, are employed in various tasks, including image synthesis, text generation, and music composition. These models are primarily divided into three types: autoregressive, non-autoregressive, and latent variable models. Autoregressive models predict each data point based on its predecessors, learning the sequence's dependencies. Non-autoregressive models, in contrast, can predict all data points simultaneously, allowing faster operations. Finally, latent variable models first learn a low-dimensional representation or *latent space* of the input data. This representation, encapsulating essential characteristics of the input, is then used to generate new, similar samples.

2.4.1 Autogressive Models

Autoregressive (AR) generative models are based on the chain rule of probability, where the probability of a variable that can be decomposed as $\mathbf{x} = x_1, ..., x_n$ is defined as follows:

$$p(\mathbf{x}) = p(x_1, ..., x_n) = \prod_{i=1}^{n} p(x_i|\mathbf{x}_{<i}) \tag{2.4}$$

where the vector $\mathbf{x}_{<i} = \{x_1, ..., x_{i-1}\}$ and we define $p(x_1|x_{<1}) = p(x_1)$ as the initial state distribution. In this way, the product rule allows for factorizing the joint distribution in an ordered fashion into several conditional distributions. However, this is not the only possible

sorting: the same result can also be achieved by factorizing the joint distribution backward $p(\mathbf{x}) = \prod_{i=n}^{1} p(x_i|\mathbf{x}_{>i})$. This is called anticausal direction, and it is usually harder to learn [67]. Equation 2.4 is general and does not make any assumption of conditional independence. The complexity of computing each conditional probability increases exponentially with the number of variables n that constitute the joint density. It is possible to represent an autoregressive

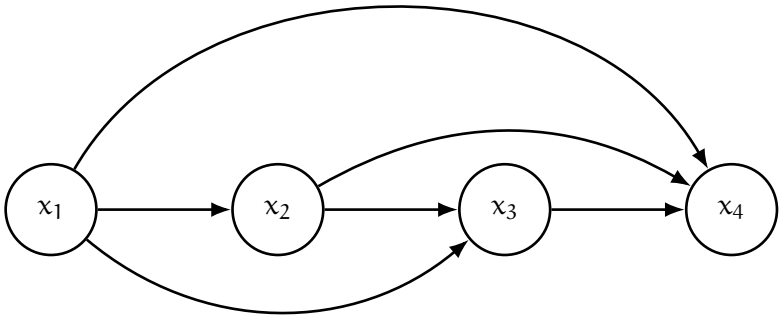

Figure 2: Fully connected DAG of an autoregressive model with four variables.

model with a Directed Acyclic Graph (DAG), as shown in Figure 2. Each arrow represents a conditional relationship between the source and the destination vertex. It is clear to see that the number of edges grows exponentially with the cardinality of the nodes. Hence, modelling all conditional distributions $p(x_i|\mathbf{x}_{<i})$ is unfeasible [103]. Several techniques employ a single model with shared parameters to overcome this issue. These models are often parameterized

with neural networks: there are many ways to achieve the same result leading to different model implementations.

2.4.1.1 Markov Model

A straightforward way to deal with the intractability of the factorization of the joint density is to make a first-order *Markov's assumption*. Therefore, we suppose that each variable in the sequence is conditionally dependent only on its predecessor and conditionally independent from all other variables $p(x_i|\mathbf{x}_{<i}) = p(x_i|x_{i-1})$. This significantly restricting assumption makes it challenging to model long-range relationship patterns. We can alleviate this assumption by allowing longer dependencies. Higher-order Markov's assumptions enhance the expressivity of the model, but the trade-off between model complexity and capability makes these simple models unsuitable for most use cases. An example of first-order Markov's model is shown

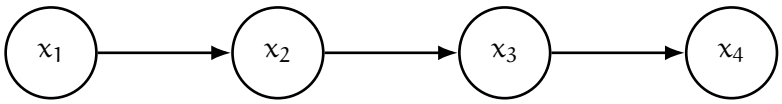

Figure 3: First order Markov's autoregressive model with four variables.

in Figure 3. For this simple model, the joint probability density can be easily factorized as

$$p(\mathbf{x}) = p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3) \tag{2.5}$$

2.4.1.2 Recurrent Neural Network

The problem of short-range memory can be solved by employing an internal state z_i that includes the information of the past states. In a Recurrent Neural Network (RNN), the internal state, denoted as z_i, is determined by previous observations, represented by $x_{<i}$. This means that z_i is a function that systematically depends on these past observations. This approach allows to model long-range dependencies and does not rely on any Markov assumptions. However, RNNs are sequential models and are usually very slow. Figure 4 show the representation

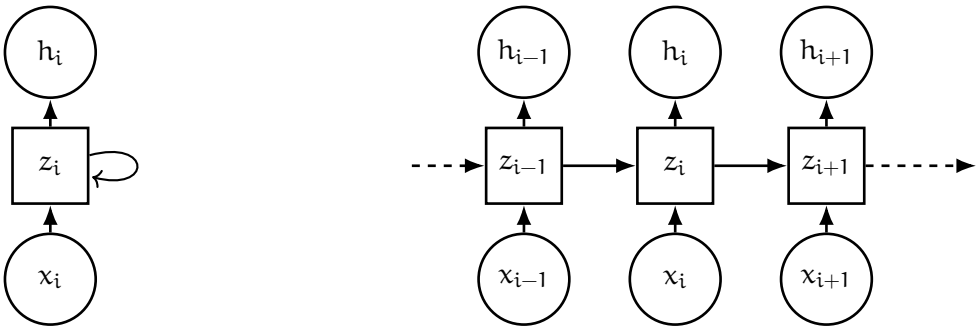

Figure 4: RNN architecture. Recurrent cell representation (left) and unfolded version of the recurrent cell (right).

of the recurrent cell of an RNN. It is possible to notice that when RNNs are deep during the forward propagation, they tend to forget their initial states due to the vanishing gradients problem. This issue can be overcome using Long-Short Term Memory (LSTM) cells [59].

2.4.1.3 Convolutional Neural Network

Instead of using RNNs to model long-range dependencies, it is possible to employ Convolutional Neural Networks (CNNs) [32]. This architecture is particularly suitable for autoregressive models because the parameters can be easily shared, and it is parallelizable, therefore, faster than RNNs. Normal convolution cannot be applied for autoregressive models as long as it

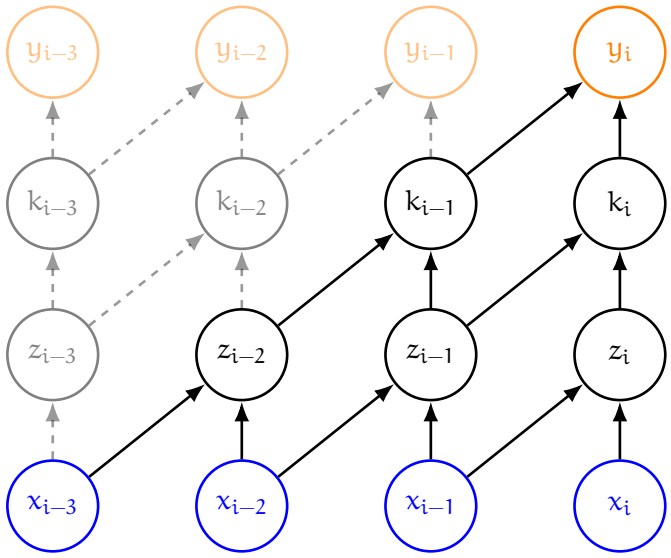

Figure 5: Causal convolution with no dilation. Input variables are shown in blue ⓘ, hidden variables are represented in black ⓗ and output variables in orange ⓞ.

is not causal. Causal convolutions (Figure 5), a specific type of convolution for temporal data, ensure that the model does not infringe on the sequence in which we model the data. This means the prediction $p(x_t|\mathbf{x}< t)$ made by the model at timestep t must not rely on any future

timesteps such as $xt+1, x_{t+2}, ..., x_n$. In the context of images, the equivalent of causal convolution is a masked convolution. This is executed by creating a mask tensor and performing an element-wise multiplication of this mask with the convolution kernel prior to its application [71]. Additionally, it is possible to increase the field of view by performing *dilated convolution*. Dilated (atrous[1]) convolution was first introduced by [114]. It is a type of convolution that inflate the kernel by inserting holes between the kernel elements. An additional parameter r (dilation rate) indicates how much the kernel is widened. There are usually $r-1$ spaces inserted between kernel elements.

2.4.1.4 Transformer

The transformer is a model architecture that overcomes the recurrence and relies entirely on an attention mechanism to draw global dependencies between input and output [107]. Self-attention is an attention mechanism relating different positions of a single sequence to compute a representation of the sequence. The Transformer has an encoder-decoder structure: the discrete input $(x_1, x_2, ..., x_n)$ is mapped into a continuous latent space $\boldsymbol{z} = (z_1, z_2, ..., z_n)$. Given $\boldsymbol{z}$, he decoder is responsible for creating an output sequence, represented as $(y_1, y_2..., y_m)$, in which each symbol is generated one at a time. With each progressive step, the model behaves auto-regressively, utilizing the symbols that were previously generated as supplementary input for the generation of the subsequent symbol. A query, a set of key-value pairs, and an output, all of which are vectors, can be mapped to one another by an attention function. The result is

---

[1]Derived from the French term "*à trous*" meaning "with holes".

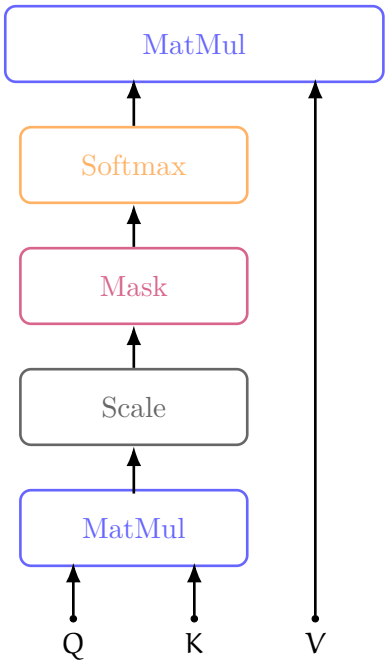

Figure 6: Functional block diagram for the scaled dot-product attention.

calculated as a weighted sum of the values, with the weights assigned to each value determined by how well the query matches the key in question. Given the matrix of queries, Q, and the matrices of keys and values K, V, the calculation for the scaled dot-product attention is as follows:

$$\text{Attention}(\mathsf{Q}, \mathsf{K}, \mathsf{V}) = \text{softmax}\left(\frac{\mathsf{Q}\mathsf{K}^{\mathsf{T}}}{\sqrt{\mathsf{d}_{\mathsf{k}}}}\right)\mathsf{V} \tag{2.6}$$

In this formula, d_{k} represents the dimensionality of the queries and keys. A schematic representation of the scaled-dot product attention is shown in Figure 6. Moreover, there exists a mechanism known as multi-head attention. This concept enables the model to simultaneously

pay attention to varying sets of information, each from different representational spaces, at distinct positions. The vanilla implementation of the Transformers is designed to work with textual data: it employs several self-attention layers in both the encoder and the decoder.

2.4.2 Non-Autoregressive Models

While AR models provide accurate estimates of the target distribution, sampling from them requires several sequential steps. Non-autoregressive (NAR) models [9; 56; 119] are a type of sequence generation model that can generate output sequences in parallel without depending on previously generated tokens. NAR models can be faster to sample from than AR models because they generate each token independently, allowing for parallel computation. Additionally, AR models typically require careful training techniques, such as teacher-forcing, to ensure that the model generates high-quality output sequences. These training techniques can slow down the training process and make the model less efficient. NAR models, on the other hand, do not require teacher-forcing, and can be trained more efficiently, leading to faster training times and potentially better performance [110].

2.4.3 Deep Latent Variable Models

Deep latent variable models (DLVMs) are based on the idea that a generative process can be seen as an abstraction of reality and a consequent concretization of the abstracted idea into a tangible output [103]. The abstraction process presumes that it is possible to idealize a concept in a low-dimensional manifold, and then, by sampling in this space, the compressed representation of the concept is expanded into the original high-dimensional space. In particular, the low-dimensional space is referred to as latent space, and the latent variables $\boldsymbol{z}$ are called

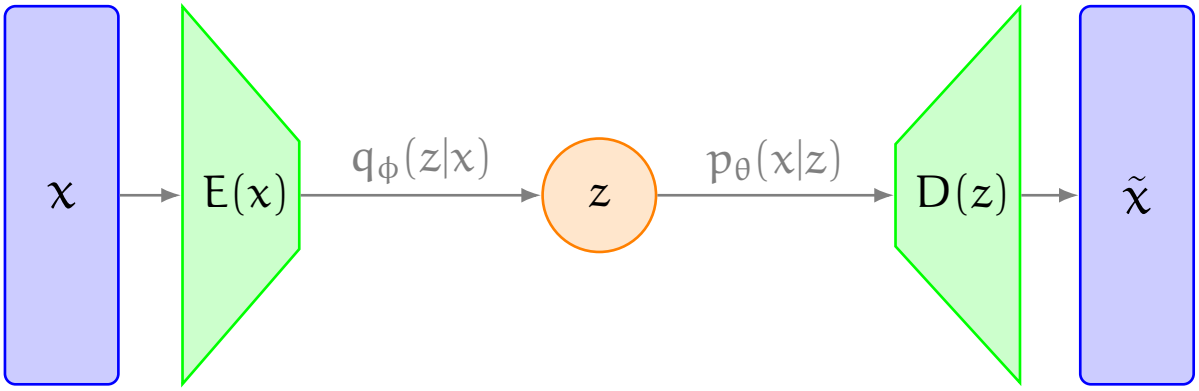

Figure 7: VAE architecture. The input x and target $\tilde{x}$ data are shown in blue, the compressed latent variable z in orange and the encoder/decoder architecture in green. The input data is compressed by the encoder to a compact latent representation. The latent is then decoded $\tilde{x}$ to be as close as possible to original input distribution x.

hidden factors in data. The idea behind latent variable models is that we introduce the latent variables are denoted as $\boldsymbol{z}$ and the joint distribution is factorized in the following manner: $p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$. However, during training, we only have access to $\boldsymbol{x}$. As a result, the marginal likelihood function is expressed as:

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z} \tag{2.7}$$

2.4.3.1 Variational Autoencoder

Variational autoencoders (VAEs) [51] employ variation inference to compute a lower bound for $p(\boldsymbol{x})$ and optimizing the lower bound instead of directly optimizing the likelihood (Equation 2.7) which is intractable [67]. If we take into account a family of Gaussian variational distributions,

parameterized by $\phi = \mu, \sigma^2 : q_\phi(z)\phi$, the logarithm of the marginal distribution can be approximated in the following manner:

$$\log p(x) \geq \mathbb{E}z \sim q_\phi(z)[\log p(x|z)] - \mathbb{E}z \sim q\phi(z)[\log q_\phi(z) - \log p(z)] \tag{2.8}$$

Nonetheless, implementing variational inference for every data point is not efficient. A more practical approach involves considering an *amortized* variational posterior, that is, $q_\phi(z|x)$ in place of $q_\phi(z)$ for each x. In such a case, we obtain:

$$\log p(x) \geq \mathbb{E}_{z\sim q_\phi(z|x)}[\log p(x|z)] - \mathbb{E}_{z\sim q_\phi(z|x)}[\log q_\phi(z|x) - \log p(z)] \tag{2.9}$$

An amortized variational posterior refers to a strategy for learning a function to approximate the posterior distribution instead of separately estimating the posterior for each datapoint. By employing this approach, we establish a model reminiscent of an auto-encoder, featuring both a stochastic encoder and a stochastic decoder. The lower bound of the log-likelihood function is called the evidence lower bound (ELBO): the first term in (Equation 2.9) accounts for the *reconstruction error* when z is decoded into x while the second term represents the Kullback-Leibler (KL) divergence between $q_\phi(z|x)$ and the difference between the ELBO and the true log-likelihood is denoted by $p(z|x)$.

Both the stochastic decoder and encoder are parameterized by neural networks (NNs). The schematic in Figure 7 summarizes the main features of the VAE architecture. The *reparameterization trick* is employed in the encoder $q_\phi(z|x)$ for reducing the variance of the gradient.

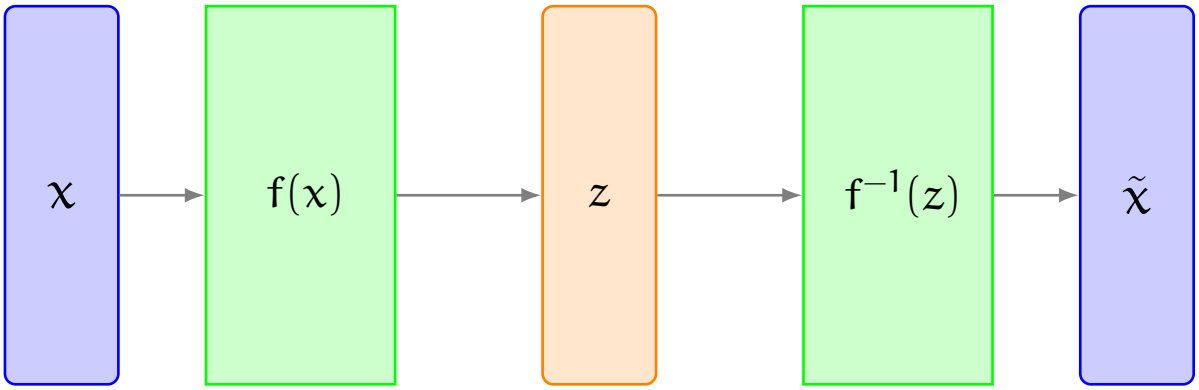

Figure 8: NF architecture. The transformation $f(x)$ and the inverse $f^{-1}(z)$ represents the composition of all the functions used for the mapping. The latent z has the same dimensionality of the input data.

Due to the flexibility in the architecture of the encoder and decoder, as well as in the selection of the prior, VAEs represent a highly potent class of models.

2.4.3.2 Normalizing Flows

The decoding of the prior in VAEs is not trivial, and it usually results in blurry samples. Normalizing flows (NFs) circumvent this issue using invertible transformations. Whenever a variable is mapped in another space through a function, it is necessary to take into account the spatial reshaping due to the transformation:

$$p_X(\mathbf{x}) = p_Z(f^{-1}(\mathbf{x}))|\mathbf{J}_{f^{-1}}(\mathbf{x})| \tag{2.10}$$

$$= p_Z(f^{-1}(\mathbf{x}))|\mathbf{J}_f(z)|^{-1} \tag{2.11}$$

Where the term $\left|\mathbf{J}_{f^{-1}}(\mathbf{x})\right|$ is the determinant of Jacobian of the inverse function $f^{-1}(\mathbf{x})$ and accounts for the morphing of the manifold inducted by the transformation. (Equation 2.11) shows a property of the Jacobian operator that allows switching between the Jacobian of the inverse function with the inverse of the Jacobian of the transformation.

Following this path, it is possible to use the idea of the change of variable to learn complex transformation that maps the data to an arbitrary space and then use the inverse transformation to bring the latent back to the data-space, as shown in Figure 8. The problem with this reasoning is that inverting an arbitrary transformation is computationally demanding, and it does not scale up well with large amounts of training data. The main idea of NFs is to chain several simple, computationally efficient, and invertible transformations to get a complex mapping:

$$p_X(\mathbf{x};\theta) = p_Z(f_\theta^{-1}(\mathbf{x}))\prod_{m=1}^{M}\left|\mathbf{J}_{f_\theta^m}(z_m)\right|^{-1} \tag{2.12}$$

Where θ represents the parameters used in the NN for learning the transformation, and M is the number of composed functions used. The main problem with NFs is that we have to choose a NN which is invertible and for which the determinant of the Jacobian matrix is easy to compute [103]. In practice, this restriction makes it difficult to deal with NF models, which are usually replaced by novel denoising diffusion probabilistic models (DDPMs). Diffusion models are similar to NFs with the difference that the chain of arbitrary function is replaced with the composition of isotropic Gaussian functions (more about DDPM in Section 2.4.3.4).

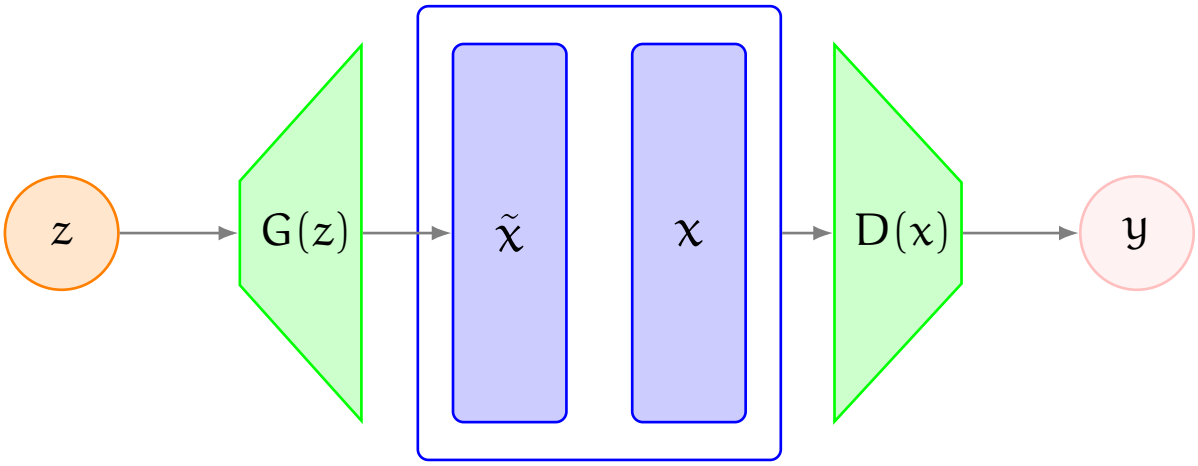

Figure 9: GAN architecture. During training, starting from the noise z, the generator produces a sample $\tilde{x}$. Then, the discriminator receives both samples from the data distribution x and the estimated one $\tilde{x}$, and tries to predict whether the sample is real or synthetic.

2.4.3.3 Generative Adversarial Network

Another approach to generation is to model the target distribution implicitly. Generative adversarial networks (GANs) [28] perform an adversarial training procedure that allows the model to learn the underlying distribution of the data. The model is composed by a generator $G(z)$ and a discriminator $D(x)$, the training requires two steps:

1. The *generator* takes Gaussian noise as input and generates samples of the target density distribution.

2. The *discriminator* may receive the real input data or the generated data, and it tries to classify the input as real or fake.

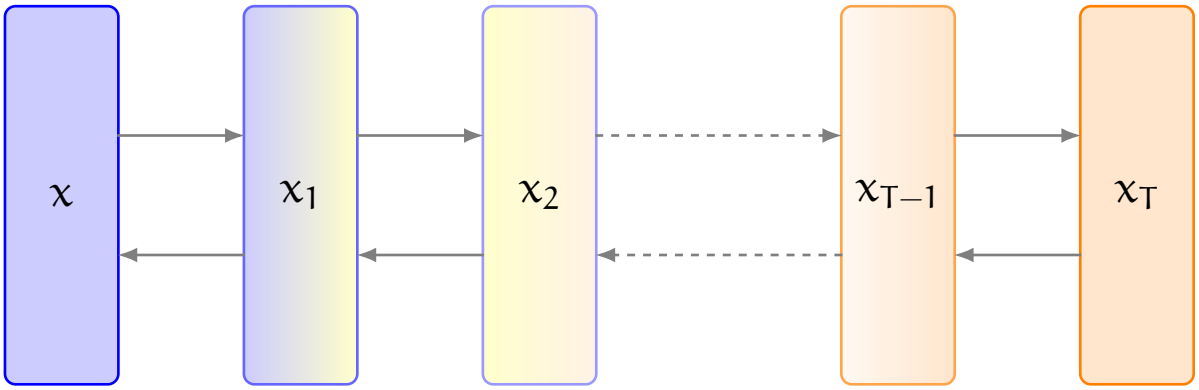

Figure 10: DDPM architecture. In the forward stage, the input data is gradually corrupted into Gaussian noise. In the backward stage, it is possible to recover the original input by starting from pure noise and removing it step by step. All the intermediate representations have the same dimensionality as the input.

This procedure is backed up by the following adversarial objective:

$$\min_{G} \max_{D} \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\big[\log D(\boldsymbol{x})\big] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\big[\log(1 - D(G(\boldsymbol{z})))\big] \tag{2.13}$$

In practice, we play a min-max game in which we are trying to improve the generation by fooling the discriminator into believing that the generated samples are coming from the real data distribution. When the objective converges, the generator is able to produce samples that, from the discriminator standpoint, are indistinguishable from the original ones. The training procedure is reported in Figure 9, where y is usually a binary label that indicates whether the input sample is predicted to be real (1) or fake (0).

2.4.3.4 Diffusion Model

Denoising diffusion probabilistic models (DDPMs) [35] possess state-of-the-art performances in many tasks. The key idea of DDPMs is shown in Figure 10: the diffusion process increasingly corrupts the input data into noise by modeling it as a first-order Markov chain. The diffusion process foresees two stages:

1. *Forward*: during this stage, each intermediate representation is augmented with some level of noise with respect to the previous step. After several steps, the distribution of the representation is the same as an isotropic Gaussian.
2. *Backward*: starting from pure Gaussian noise, it is possible to generate unseen samples from the input distribution by removing some amount of noise that the network is able to learn during the training phase.

The forward diffusion procedure, comprising T steps, can be expressed as follows:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{2.14}$$

Here, β_t serves as the variance schedule, acting as a tuning parameter to regulate the amount of noise introduced at each stage. Given that the forward process involves the composition of normal distributions, it is viable to sample from any given time step in a closed form:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{2.15}$$

In this equation, α_t is given by $1-\beta_t$ and $\bar{\alpha}_t$ is the product of all α_i for $i = 1$ to t. The backward stage defines the generative process of the model. Going backward from noise, it is possible to approximate $q(x_{t-1}|x_t)$ with a Normal distribution if β_t is small in each diffusion step. The denoising process is parameterized with a neural network that has the following objective:

$$\mathbb{E}_{t\sim[1,T],x_0,\epsilon_t}\left[\|\epsilon_t - \epsilon_\theta(x_t, t)\|^2\right] \tag{2.16}$$

Where ϵ_t is the noise at timestep t. In practice, the NN is trained to understand how much noise is injected at timestep t with a simple mean squared error (MSE) loss. Therefore, to perform the training of the model, it suffices to sample a random timestep t and some noise ϵ from a standard Normal distribution and then perform the gradient descent step on the loss in (Equation 2.16) until convergence. Generally, the U-Net architecture [89] is employed to model $\epsilon_\theta(x_t, t)$ injecting positional embeddings in the U-Net layers to acquire the temporal information.

Over the years, several authors have proposed approaches to improve vanilla DDPMs. One of the major drawbacks of these models is that during the sampling, many steps (~ 1000) are needed to produce high-quality results. [70] suggest an improved noise scheduler that outperforms the linear scheduling, and [96] introduces denoising diffusion implicit models (DDIMs) that generalize the Markovian process and are able to generate a sample with an order of magnitude fewer steps (~100). Distillation techniques [65] make it possible to use a pre-trained DDPM and distill the diffusion, making it possible to achieve results comparable to the start-

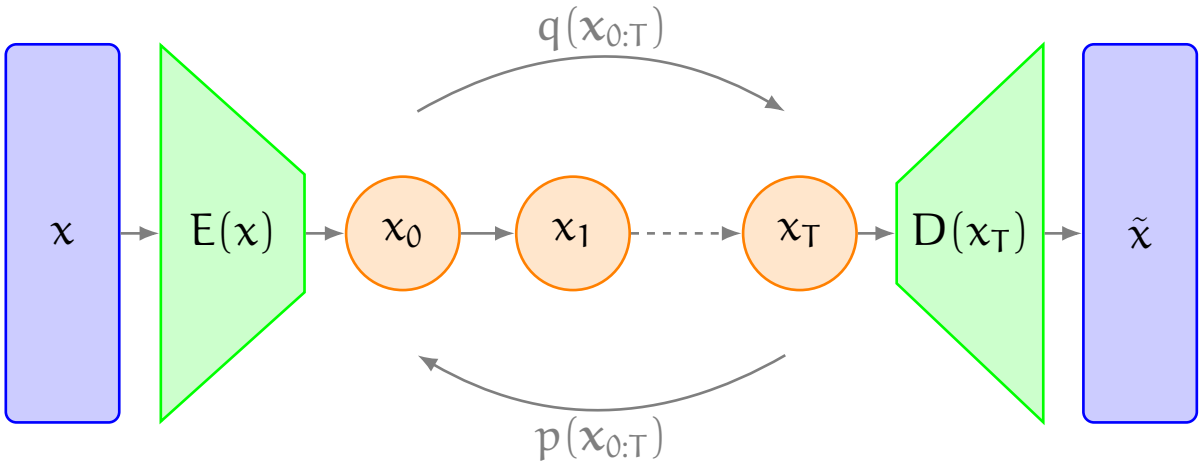

Figure 11: Latent diffusion model architecture. The diffusion process is performed in the latent space of a pre-trained autoencoder. The cross-attention operation during the diffusion process allows for conditional generation.

ing DDPM but with dozens of steps. Finally, consistency models (CMs) [97] allow to distill pre-trained DDPMs or to train consistency models from scratch that are able to perform the sampling in a single backward step from noise to data[1].

2.4.4 Hybrid Models

Different combinations of deep generative models can be used, typically to leverage the strengths of each model for improved performance. Some popular examples of hybrid models include replacing the prior in the VAE architecture with autoregressive models and performing adversarial training with non-GAN models. One of the state-of-the-art (SOTA) architectures for image generation is the latent diffusion model (LDM) [88], which merges VAE, DDPMs, and

[1] Additional steps improve the quality of the generation.

transformers. A schematic visualization of an LDM is presented in Figure 11: in the latent space of an autoencoder, a DDPM performs the diffusion process on the compressed latent variable. In the U-Net layers that parameterize the diffusion, several attention heads perform the cross attention between the input and some embedded conditioning signal, making it possible to perform guided diffusion with crossmodal input sources.

2.4.5 Transfer Learning

Transfer learning has become a popular technique in the field of machine learning and artificial intelligence due to its ability to improve the performance of models by leveraging knowledge gained from previously learned tasks. It refers to the practice of utilizing pre-trained models as a foundation for a new task rather than starting the training process from scratch. Typically, the pre-trained model is trained on a sizable dataset that shares similarities with the new task, allowing it to learn features that are useful for the new task [82].

Transfer learning can be especially useful when the new dataset is small, as training a model from scratch on a small dataset may lead to overfitting, a scenario where the model excels at the training data but struggles with new, unseen data. By using a pre-trained model, the model has already learned to recognize and extract useful features, which can reduce the risk of overfitting. Transfer learning also allows for faster training times, as the pre-trained model has already learned a large amount of knowledge, reducing the number of iterations required for convergence. Another advantage of transfer learning is that it can be used to improve the performance of models in domains where large amounts of labeled data are not readily available, such as audio generation or speech recognition. By using a pre-trained model that has been

trained on a similar domain, the model can learn from the pre-trained features and improve its performance on the new domain.

Transfer learning has been widely applied in the field of audio generation, specifically in the domains of speech, music, and sound effects.

- In *speech generation*, transfer learning has been used to improve speech recognition and speech synthesis [66; 95]. For example, pre-trained models for speech recognition can be fine-tuned to a specific speaker or language with a smaller amount of training data, resulting in better accuracy and efficiency. Similarly, pre-trained models for speech synthesis can be used as a starting point for generating new speech samples and then fine-tuned on a specific task, such as emotion recognition or voice conversion.
- In the domain of *music generation*, transfer learning has been applied to various tasks, including melody and harmony generation, instrument separation, and music style transfer [43; 116; 61; 40]. For instance, pre-trained models for music generation can be fine-tuned to generate new melodies or harmonies that follow a certain style or mood. Transfer learning can also be used to separate individual instruments from a mixed audio signal, which is useful for tasks such as music remixing and audio source separation. Moreover, pre-trained models can be used to transfer the style of one piece of music to another, allowing for the creation of new music that preserves certain characteristics of the original piece.
- Finally, transfer learning has been used in the field of *sound effects generation*, which involves synthesizing sounds that correspond to specific events or actions [20; 102]. Pre-

trained models can be fine-tuned to generate new sound effects that match the characteristics of a given scene or environment.

Overall, transfer learning has proven to be a useful technique for improving the accuracy and efficiency of audio generation tasks in a variety of domains by leveraging pre-existing knowledge from related tasks or domains.

2.5 Controllable Generation

Controllability in deep generative models is a crucial aspect for their successful implementation in real-world applications. In the field of deep music generation, controllability is particularly important in order to produce high-quality and diverse musical compositions that cater to the user's preferences. The main attributes of conditional deep music generation are as follows:

- **Controllability**: enables the generation of music constrained to a specific class or set of attributes. With the aid of controllability, it is possible to steer the generation process towards a specific music genre, tempo, or key, among others.
- **User experience**: the user is able to create its own outputs by inputting specific attributes, thus empowering the user to have control over the generated outputs.
- **Compositionality**: a critical attribute of controllability in deep generative models. The ability to generate novel concept combinations is essential in music generation as it provides the opportunity to generate unique and distinct compositions.
- **Multimodal conditioning**: controlling the generation with cross-modal inputs may improve the quality and the speed of the generation. By inputting additional information

such as text, images, or videos, it is possible to provide more information to the model and guide the generation process more efficiently.

2.5.1 Text-to-Music

Text-to-music generation is a task that involves generating a musical composition given textual input. In this task, the model is trained to learn the correlation between the input text and its corresponding musical representation. The input text can be a description of a musical genre, the name of artists, or any other textual information that can be semantically associated with musical elements. The generated music can be in any genre, style, or tempo, depending on the training data and the model's architecture. The task of converting text into music is demanding owing to the complexity and diversity of music and language, as well as the subjective nature of music.

2.5.2 Lyrics-to-Music

Lyrics-to-music generation is a specific subtask of text-to-music that involves generating music with vocals based on lyrics as input conditioning. This task is particularly challenging as lyrics information serves as both textual and temporal conditioning. The temporal aspect of lyrics links the words in the lyrics to specific time intervals in the music data. Consequently, the lyrics' content has a significant impact on controlling the music generation process. To address this challenge, some models, such as Jukebox [13], have attempted to generate music and vocals conditioned on lyrics information using an autoregressive approach.

2.5.3 Music-to-Music

Music-to-music generation is the task of generating novel musical compositions that are inspired by a given musical piece or a set of pieces. In this context, the input and output are both musical data, and the generative model is tasked with learning the underlying patterns and structures of the input music and generating novel pieces that share similar qualities. This task is challenging due to the complexity and diversity of musical compositions and the need to capture and reproduce the intricate relationships between musical elements such as melody, harmony, rhythm, and instrumentation. The success of music-to-music generation depends on the ability of the model to learn and generalize from the input music and generate compositions that are both diverse and coherent.

2.5.4 Symbolic Conditioning

Symbolic conditioning is a subtask of music-to-music generation that leverages symbolic musical representation to generate new music compositions. This conditioning method is different from audio conditioning since it utilizes symbolic data in the form of MIDI files, which encode music as a sequence of notes with temporal and timbre information. The symbolic conditioning approach provides several advantages, including compact representation, easy manipulation, and control over the generated music's melody and structure. Symbolic conditioning models have been successfully applied to generate music compositions that emulate a specific style, genre, or composer. However, this method is not without limitations. The symbolic conditioning approach has difficulty capturing the complex timbre and subtle nuances of the music.

Therefore, while symbolic conditioning is a promising subtask of music-to-music generation, it has room for improvement in terms of generating more realistic and natural-sounding music.

2.5.5 Image-to-Music

Image-to-music generation is a challenging task in the field of deep generative models that aims to generate music given an image as input. This task belongs to the realm of multimodal generation, where the generation process starts from a modality that is very different from the audio domain. This approach offers many possibilities to create new and innovative musical compositions, leveraging the rich variety of features embedded in images. For instance, it is possible to generate music that conveys feelings expressed in the input image, such as sadness, joy, or excitement.

2.6 Music Dataset

Music datasets play a crucial role in the development and evaluation of machine learning models for music generation tasks. However, creating large and diverse music datasets can be a challenging task due to various reasons. One of the main challenges is the availability of the data. The vast majority of music recordings are owned by record labels and music publishers, making it difficult for researchers to access them. This limitation has resulted in the creation of smaller datasets or even the need for researchers to create their own datasets by manually curating and annotating music recordings. Such datasets are often limited in size and scope, hindering the ability of researchers to develop and evaluate robust music generation models.

Another challenge in creating music datasets is the scarcity of audio data. While there are a plethora of music recordings available, they often come in different formats and qualities,

making it difficult to curate a standardized dataset. Additionally, the cost of acquiring high-quality recordings is often prohibitive, making it difficult for researchers to collect large amounts of data. This scarcity of audio data can result in overfitting of the models to the available data, leading to poor generalization and limited applicability.

Furthermore, the use of copyrighted material in music datasets can create legal challenges for researchers. While some datasets are made available under a permissive license, others may require the researcher to obtain permission from the copyright holders before using the data. This can be a time-consuming and costly process, resulting in delays in research progress. Additionally, the use of copyrighted material can limit the distribution of the datasets, as some researchers may not have the necessary permissions to use the data.

Lastly, the lack of a reference dataset for comparison poses a challenge to the evaluation of music generation models. While there are several publicly available music datasets, there is no standardized reference dataset for evaluating the performance of different music generation models. This can make it difficult to compare the performance of different models and hinder progress in the development of better music generation techniques.

2.6.1 Raw Music Dataset

Several freely available raw waveform music datasets can be used in machine learning research. These datasets are particularly valuable because they offer direct access to the audio data that can be fed into a machine learning model. However, it should be noted that storing raw waveform data can be space-consuming, and as a result, some of these datasets may contain

relatively small amounts of data or be excessively large and difficult to manage. Some examples of the most popular raw music datasets are:

- **Free Music Archive** [11] is a publicly available dataset that can be easily accessed and used for various tasks related to Music Information Retrieval. It includes a wide variety of music genres and contains a large collection of audio files, making it a valuable resource for research and experimentation.
- **GTZAN** dataset [99] consists of 1000 audio tracks, each 30 seconds in length, and is used for evaluating music genre classification models. There are 10 different genres included in the dataset, with 100 tracks per genre. All tracks are in mono 16-bit WAV format and sampled at 22050Hz.
- **MagnaTagATune** dataset [57] comprises more than 25,000 music clips, each with a duration of 29 seconds. These clips are sourced from a diverse collection of over 5,000 songs, 445 albums, and 230 artists. The dataset spans a wide variety of music genres, such as classical, jazz, rock, and pop. Uniquely, each music clip in the dataset carries binary annotations for over 180 different tags. These tags denote the presence or absence of certain features like vocals, various instruments, and specific musical genres. Human players participating in the online game TagATune generated these annotations. Players listened to audio clips during the game and assigned descriptive tags accordingly.

2.6.2 Symbolic Music Dataset

Symbolic music datasets are a popular alternative to raw waveform data because they contain symbolic representations of musical information such as pitch, timing, and duration. They can be processed by music information retrieval algorithms, music transcription, and music generation models. These datasets are highly popular and widely available online due to their ability to facilitate many research areas related to music technology. Symbolic music datasets cover a major portion of the available music dataset online, thanks to their lighter and more manageable format that can be easily processed and analyzed. Here are some examples of freely available symbolic music datasets:

- The **Lakh MIDI** [81] dataset contains over 176,581 unique MIDI files from a variety of sources, including the web, user contributions, and digitized music scores. This dataset is considered one of the largest symbolic music datasets available and is suitable for various research tasks, including music information retrieval and music generation.
- **Maestro** [30] is a dataset of MIDI performances that includes over 172 hours of solo piano music performed by various artists. It covers a wide range of musical styles. It includes several MIDI features, such as velocity, timing, and expression, that can be used in research related to expressive music performance analysis and modeling.
- **NES-MDB** [16] is a dataset that contains over 36,000 MIDI files of music from classic Nintendo Entertainment System (NES) games. This dataset has been used in research related to video game music analysis and machine learning models for music generation. It can be used to study the unique features of 8-bit video game music.

- **NSynth** [21] is a dataset of over 300,000 unique sounds generated by a neural network trained on the Google Magenta platform. This dataset contains a wide range of sounds, including musical instruments, human voices, and environmental sounds. It can be used in research related to sound synthesis, timbre analysis, and machine learning models for music generation.

2.6.3 Music Metadata Dataset

Metadata music datasets are a valuable resource for music generation research, as they provide information about various characteristics of music, such as artist, genre, tempo, and mood. These datasets do not contain raw music files but rather provide a wealth of information about the music that can be used to train machine learning models. The metadata can be used to generate new music based on certain features or characteristics or to classify and organize large music collections. Additionally, metadata can be combined with raw music datasets to enhance the performance of machine learning models by providing additional contextual information. Some examples of freely available metadata music datasets are:

- The **Million Song Dataset** [4], which contains the audio features for one million songs. The metadata includes information such as artist, release year, tempo, and key, as well as acoustic features extracted from the audio signal such as loudness and timbre.
- **AudioSet** [26] is a large-scale dataset of annotated audio events that includes over 2 million 10-second sound clips from a wide range of sources. The dataset includes annotations for various sound events, such as music, speech, and ambient sounds. It can be

used to train machine learning models for several tasks, such as sound event detection and classification.

- **MusicCaps** [1] is a recent dataset of 5,521 10-second music clips from AudioSet, each with a free-text caption and an aspect list describing features such as genre, mood, and instrumentation. The MusicCaps dataset has been conceived with the aim of using it as an evaluation dataset rather than a training dataset. The dataset is solely dedicated to music and offers high-quality annotations supplied by experts.

2.7 Commercial Proprietary Software

The landscape of AI-generated music has expanded beyond the realm of academic research, with a rising number of commercial platforms providing AI-driven music generation services. These platforms are gaining traction as they cater to the growing demand for customizable, royalty-free music that can be used in various creative projects, such as online videos, video games, or other artistic endeavors. One of the main characteristics of these commercial platforms is their focus on generating instrumental music, with only a few options for vocal tracks or music with vocals.

Notably, the algorithms and training data used by these commercial services are proprietary and undisclosed, which makes it challenging to comprehend the inner workings of these models, including their training methodologies and data sources. Despite this, their increasing popularity is a testament to the burgeoning interest in AI-generated music and its potential to revolutionize the creative landscape.

The following platforms are among the prominent commercial online services that offer synthetic music generation capabilities:

- AIVA[1]
- Amper[2]
- Soundful[3]
- Ecrett[4]
- SoundRaw[5]
- Boomy[6]
- Amadeus Code[7]
- Mubert[8] [3]

---

[1] AIVA website (aiva.ai).

[2] Amper website (ampermusic.com).

[3] Soundful website (soundful.com).

[4] Ecrett website (ecrettmusic.com).

[5] SoundRaw website (soundraw.io).

[6] Boomy website (boomy.com).

[7] Amadeus Code website (amadeuscode.com).

[8] Mubert website (mubert.com).

These platforms showcase how AI-generated music can enhance the creative process and unlock new opportunities for creators. Their success and continued development serve as an inspiration for the broader research community, motivating further investigation into AI-driven music generation techniques.

As more research is published in the open-source domain, we can expect to witness even greater advancements in AI-generated music, leading to more sophisticated models and higher-quality output. This synergy between commercial platforms and academic research will continue to drive innovation and expand the possibilities for AI-driven music generation. By embracing this collaboration, both creators and consumers will benefit from the ongoing evolution of AI-generated music, ultimately enabling a richer and more diverse creative landscape.

2.8 Deep Generation and Copyright

Music generation is a relatively underdeveloped domain that lacks readily available audio data sources. Unlike other fields, such as natural language processing and computer vision, the task of gathering a complete and high-quality music dataset with metadata and customized audio quality attributes is still very challenging. As a result, the lack of reference data creates many obstacles for researchers and practitioners in the field of music generation, leaving them dependent on the research of companies and eminent universities that have the resources to access huge computational power.

Gathering a web-crawled dataset requires terabytes of storage and a large cluster of servers that are capable of downloading music at a high-speed rate. Additionally, audio data is inherently high-dimensional, which makes the training of machine learning models challenging on

its own. The aforementioned issues have given rise to a discernible imbalance in the research landscape over the last decade in the domain of music generation. This predicament creates an uneven playing field, with sizable corporations and research institutions enjoying a marked advantage over their smaller counterparts, ultimately constraining progress in the field of music generation research.

Moreover, the nature of music data as an intellectual property protected by copyright laws further slows down research progress in music generation. In contrast to natural language processing and computer vision, where publicly available reference datasets are often provided, the lack of publicly available music datasets hinders the ability of researchers to train and evaluate models. To avoid copyright infringement, researchers must obtain explicit permission from the copyright holders, which is a time-consuming and often costly process. This limitation makes it more difficult for researchers to advance the state of the art in music generation.

2.8.1 Dealing with Protected Data

The field of deep music generation relies heavily on the availability of large datasets to obtain concrete research results. However, the use of private web-crawled datasets that are not readable or accessible has become a common practice in recent works [13; 63; 1; 3; 92]. It is possible that these datasets contain copyrighted material, and obtaining permission from every intellectual property holder is not feasible given the large number of samples required for training a music model.

One possible option for obtaining large amounts of royalty-free music data is to purchase it from online platforms that specialize in selling such content. While these platforms can

provide high-quality music samples that are free from copyright restrictions, they often come at a high cost, which can be a barrier for smaller labs or individual researchers. Additionally, the metadata associated with these datasets is often limited, which can make it difficult to utilize the data in research fully. For example, some platforms may not provide information on the commercial success of a particular song or the associated artist, which can be crucial in analyzing the characteristics of a successful piece of music. Despite these limitations, online platforms that sell royalty-free music remain a viable option for researchers who require large amounts of high-quality music data for their projects.

2.8.2 Research and Fair Use

When it comes to using copyrighted material for research purposes in the field of deep music generation, it is crucial to acknowledge that employing copyrighted material for research purposes is generally considered lawful under the fair use doctrine. Fair use is a legal provision within copyright law that permits restricted utilization of copyrighted material without obtaining explicit permission from the copyright holder [12; 24]. While the specifics of fair use can vary depending on the circumstances, using copyrighted material for research purposes is often considered a valid use under this doctrine. However, it is still important for researchers to be aware of the potential legal issues that could arise and to take steps to minimize any risks involved in using copyrighted material in their work. Furthermore, any commercial use of the generated content would require obtaining appropriate licensing and permissions.

2.8.3 Limitations

Despite the significant advancements in the field of deep music generation, there remain certain limitations and risks associated with utilizing machine learning models for music generation. Specifically, the use of such models may lead to unwanted or unexpected behaviors, including the creation of music that is inappropriate or offensive, the infringement of intellectual property rights, and potential legal liabilities for individuals or organizations involved in the creation or distribution of such music. As such, it is crucial that users of these technologies are aware of the potential risks and effects associated with their use and take appropriate steps to mitigate any potential legal or ethical concerns. Furthermore, it is important that developers of deep music generation models consider these limitations and work towards developing more ethical and responsible approaches to the development and deployment of these technologies. Any input for conditioning the generation of the music and the resulting output should be attributed to the user. Additionally, the user should own the rights to the generated output. It is crucial to have clear guidelines and agreements in place regarding ownership and attribution to ensure that the use of deep music generation technology is both legal and ethical.

CHAPTER 3

RELATED WORK

3.1 Literature Overview

Numerous works from various domains have significantly shaped the field of deep music generation, each contributing innovations and advancements. Many novel architectures offer insights and research directions that can be directly applied to music generation tasks. Additionally, related fields such as text-to-speech and audio generation have also played a crucial role in deep music generation research, fostering the development of new models and techniques.

In this thesis, the Related Work chapter explores a selection of influential models that have pushed the boundaries of the state-of-the-art in music generation. While this review is not exhaustive, it offers a detailed examination of key models that have contributed to raw music generation and those that utilize 2D representations of audio data. It is important to note that this review does not cover works focused on symbolic music generation.

The Related Work chapter delves into the milestones achieved in deep music generation and their transformative impact on the field. It provides an extensive overview of various models, their underlying architectures, and training mechanisms. The chapter highlights the models' contributions to raw music generation, which involves creating music from scratch without relying on pre-existing musical structures. Furthermore, the review investigates influential models that employ 2D representations of audio data, enabling the synthesis of more sophisticated and

nuanced music. Ultimately, this chapter offers a thorough review of the related work in deep music generation, laying the foundation for the remainder of the thesis.

3.2 AR Models

In the early stages of autoregressive modeling for deep music generation, several notable models emerged that laid the foundation for more advanced architectures. This section of the Related Work aims to review the key contributions of the most influential work on early autoregressive modeling, leaving aside the Transformer models, which are examined in Section 3.6. Particular emphasis will be put on autoregressive models that are able to directly generate raw waveform audio due to their importance for this thesis.

3.2.1 Image and Raw Music Generation

Uria et al. [104] proposed Neural Autoregressive Distribution Estimation (NADE) as a deep generative model for representing the joint distribution of high-dimensional data. NADE is based on the idea of modeling the probability distribution of a data point by decomposing it into a product of conditional distributions, each of which is represented by a neural network. In the context of music generation, each conditional distribution models the probability of a specific note or event occurring given the previous ones. Mathematically, the joint distribution modeled by NADE can be represented as follows:

$$p(\mathbf{x}) = \prod_{i=1}^{D} p(x_i|\mathbf{x}_{<i}), \tag{3.1}$$

where $\boldsymbol{\mathrm{x}}$ is a data point, D is the dimensionality of the data, and $\boldsymbol{\mathrm{x}}_{<\mathrm{i}}$ represents the vector of previous components. NADE's ability to capture complex dependencies in the data made it a useful early model for music generation, but its feedforward architecture limited its capacity to model long-term dependencies.

The PixelRNN model, proposed by Oord et al. [72], addressed the limitations of NADE by introducing RNNs into the autoregressive framework. PixelRNN aimed to model images by predicting the intensity value of a pixel given the previous pixels in a row-by-row manner. While originally designed for images, PixelRNN's architecture could be adapted for music generation by modeling the generation process as a sequence of discrete events. The model's RNN-based architecture allowed it to capture longer-term dependencies more effectively than feedforward models like NADE, thus improving the quality of generated music. The core component of PixelRNN is the LSTM layer, which allows the model to learn complex patterns in sequences without suffering from the vanishing or exploding gradient problems typically encountered in traditional RNNs. By utilizing LSTMs, PixelRNN is capable of modeling longer and more complex musical structures compared to its predecessors.

WaveNet, a groundbreaking work by Oord et al. [71], is a model for generating raw audio that has had a significant impact on the field of music generation. Unlike previous models that relied on a discrete representation of musical events, WaveNet directly models the raw audio waveform, allowing it to generate high-quality audio with a wide range of musical structures and timbres. WaveNet employs a stack of dilated causal convolutional layers, which can effectively

model long-range dependencies in the audio signal while maintaining a manageable number of parameters. The model's autoregressive nature is expressed as:

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t|\mathbf{x}_{<t}), \tag{3.2}$$

where T is the length of the audio signal, and x_t represents the value of the waveform at time t. WaveNet's ability to model long-range dependencies, combined with its direct generation of raw audio, has enabled it to produce state-of-the-art results in various music generation tasks, making it a pivotal model for this thesis.

WaveNet introduced a novel architecture for generating raw audio that has become a cornerstone in the field of music generation. The structure of the model is composed of a series of dilated causal convolutional layers meticulously designed to efficiently grasp long-range dependencies present within the audio signal. Dilated convolutions [114] is a key feature of WaveNet, as they allow the model to increase its receptive field exponentially with depth while maintaining a relatively small number of parameters. This property is crucial for modeling the long-range structure in music, which often spans several seconds or more. Another crucial aspect of WaveNet's architecture is the use of causal convolutions, which ensures that the model respects the temporal order of the data. This is achieved by masking the future context in the convolution operation, making sure the output at each time step only depends on the current and previous inputs. This causal structure is essential for autoregressive models, as it preserves the generative process's sequential nature.

SampleRNN, introduced by Mehri et al. [64], is an unconditional end-to-end neural audio generation model that builds on the autoregressive framework. It combines the strengths of RNNs and feedforward models to generate raw audio waveforms efficiently. SampleRNN is a hierarchical model that processes and generates audio at different time scales, allowing it to capture both local and long-range dependencies in the audio signal. The architecture of SampleRNN consists of multiple tiers, with each tier corresponding to a specific temporal resolution. The higher-level tiers capture long-term dependencies and global structures in the audio, while the lower-level tiers model local and fine-grained dependencies. This design allows SampleRNN to effectively represent the complex structures and patterns in music that span across various temporal extents. Each tier in the SampleRNN architecture consists of a combination of RNNs and autoregressive models. The RNNs are responsible for capturing dependencies across coarser time scales, while the autoregressive models refine the generated audio by modeling the dependencies at finer time scales. The use of autoregressive models, such as the 1D PixelRNN, within the lower tiers of the hierarchy enables SampleRNN to generate high-quality, detailed audio waveforms. The final output of the SampleRNN is a raw audio waveform generated at the highest temporal resolution.

Compared to NADE, SampleRNN is more capable of modeling long-range dependencies due to its hierarchical structure and the use of RNNs. The use of multiple time scales allows SampleRNN to efficiently capture complex structures in music that span different temporal extents. While NADE relies on a feedforward architecture, which can limit its capacity to model long-term dependencies, SampleRNN's recurrent structure provides a more suitable ar-

chitecture for capturing the dynamics of music. Compared to PixelRNN, SampleRNN shares similarities in its use of RNNs for capturing long-range dependencies. However, the hierarchical structure of SampleRNN differentiates it from PixelRNN, which employs a single level of temporal resolution. This hierarchical approach allows SampleRNN to efficiently generate audio by modeling dependencies at different time scales, providing a more scalable architecture for audio generation. In contrast to WaveNet, SampleRNN uses a combination of recurrent and autoregressive models instead of dilated causal convolutions. While WaveNet's architecture is designed to capture long-range dependencies with a relatively small number of parameters, SampleRNN achieves this through its hierarchical structure. Both models have demonstrated success in generating high-quality audio, but their architectural differences result in distinct trade-offs in terms of computational efficiency and model complexity.

3.2.2 Text-to-Speech Synthesis

Several autoregressive models have primarily focused on the task of text-to-speech (TTS) synthesis. However, their architectures and the techniques they introduced have also had an impact on the field of deep music generation as they expand the possibilities of autoregressive modeling.

Ping et al. [78] presents a fully convolutional TTS system that employs a sequence-to-sequence architecture with attention mechanisms. The encoder consists of several convolutional layers followed by a multi-head self-attention mechanism, transforming the input text into a high-level representation. The decoder, also composed of convolutional layers and self-attention, generates a mel spectrogram conditioned on the encoder's output. A separately

trained WaveNet vocoder then converts the mel spectrogram into raw audio waveforms. This fully convolutional architecture and the use of self-attention mechanisms can be adapted to music generation tasks, providing a flexible framework for modeling complex relationships between input and output sequences in an autoregressive setting.

Tacotron [108] uses an encoder-decoder architecture with attention mechanisms for end-to-end speech synthesis. The encoder maps the input text to a high-level representation, which is then passed to an attention-based decoder that generates a mel spectrogram. The mel spectrogram is converted to a raw audio waveform using a separate WaveNet vocoder. This attention-based encoder-decoder architecture can be adapted for music generation tasks, as it provides a framework for learning complex relationships between input and output sequences.

Tacotron 2 [94] refines the original Tacotron architecture and incorporates a WaveNet-based vocoder for generating raw audio directly from the predicted mel spectrogram. Improvements include a modified attention mechanism, a more advanced pre-net for the decoder, and the use of convolutional layers in the encoder. These enhancements result in higher-quality synthesized speech and a more stable training process. The integration of a WaveNet vocoder in the Tacotron 2 pipeline further highlights the versatility of autoregressive models in generating high-quality audio across various domains, including music generation.

3.2.3 Recent Advances in Audio Synthesis

WaveRNN [46] combines the strengths of autoregressive modeling and RNNs to generate raw audio waveforms efficiently. WaveRNN employs a Gated Recurrent Unit (GRU) with a dual softmax layer to output two probability distributions at each time step, one for the

most significant bits and one for the least significant bits of the audio sample. This approach allows for a more efficient sampling process compared to models like WaveNet, which requires a large number of output channels. WaveRNN leverages a compact architecture, making it more computationally efficient and easier to deploy on resource-constrained devices while still maintaining high-quality audio generation capabilities.

MelNet [106] generates audio in the frequency domain by modeling the time-frequency structure of spectrograms. MelNet utilizes a 2D autoregressive architecture with both horizontal and vertical dependencies, allowing it to capture both temporal and spectral structures in the audio. This approach differs from other autoregressive models like WaveNet and SampleRNN, which operate directly on raw audio waveforms. By working in the frequency domain, MelNet can more effectively learn the underlying patterns and structures in the audio data, potentially leading to improved generative capabilities and a broader range of applications.

Seq-U-Net [98] introduces a model inspired by the WaveNet architecture, using a one-dimensional causal U-Net for efficient sequence modeling. Seq-U-Net employs local and global skip connections that allow it to capture long-range dependencies while reducing computational complexity compared to WaveNet. This model demonstrates the potential for combining autoregressive modeling with U-Net architectures to improve efficiency in audio synthesis tasks. Furthermore, Seq-U-Net incorporates dilated convolutions, enabling it to handle a large receptive field without a significant increase in the number of parameters. This results in enhanced computational efficiency, making Seq-U-Net suitable for real-time applications. By embracing these innovations in autoregressive modeling and audio synthesis, the field of deep music gener-

ation can continue to evolve and create new possibilities for generating high-quality and diverse musical outputs.

3.3 VAE Models

VAEs have emerged as a powerful technique in the field of deep music generation due to their unique ability to work in low-dimensional spaces and their fast inference capabilities. By encoding high-dimensional input data into a low-dimensional latent space, VAEs can effectively compress the data, allowing for efficient manipulation of the latent representations. This compression property is particularly valuable in the music and audio generation domain, where the input data can be very high-dimensional and complex.

Furthermore, VAEs excel in representation learning [117; 7], enabling the discovery of meaningful and interpretable structures within the data. Through the optimization of a lower bound on the data likelihood, VAEs learn to encode and decode the data in a way that captures the underlying distribution. This enables the generation of new samples with similar characteristics to the training data, making VAEs particularly well-suited for generating diverse and coherent musical structures. In the following sections, we discuss various VAE models and their impact on deep music generation.

3.3.1 Vector-Quantized VAE

Vector-quantized VAEs have proven to be especially suited for music and audio generation tasks, as they allow for the effective compression of high-dimensional data and enable the generation of diverse and high-quality samples. In the work by Oord et al. [73], the authors propose VQ-VAE, a VAE model that employs vector quantization in the bottleneck layer to

learn discrete latent representations. The VQ-VAE model consists of an encoder that maps the input data to a continuous latent space, followed by a quantization step that discretizes the continuous representation into a set of discrete latent codes. The decoder then reconstructs the original data from the discrete codes. VQ-VAE has shown promise in generating diverse and coherent samples, making it suitable for deep music generation tasks.

The VQ-VAE-2 model [84] builds upon the original VQ-VAE framework by employing a hierarchical architecture. This hierarchical structure enables the model to capture multi-scale dependencies in the input data more effectively. The VQ-VAE-2 consists of multiple levels of VQ-VAE models, each capturing information at different resolutions. This hierarchical approach allows the model to create high-quality results while still maintaining the advantages of discrete latent representations. Although the paper primarily focuses on image generation, the hierarchical architecture can be applied to music generation, capturing various levels of abstraction in the audio data.

In the study by Hadjeres and Crestel [29], the authors present VQ-CPC, a model that combines the vector quantization technique from VQ-VAE with contrastive predictive coding (CPC) to learn meaningful representations for template-based music generation. The VQ-CPC model consists of an encoder and a discrete autoregressive model that predicts future latent codes based on the current context. The encoder maps the input audio to a set of discrete latent codes, while the autoregressive model learns to predict future codes, thus enabling the generation of coherent musical structures. This combination of vector quantization and contrastive predictive coding offers a powerful approach for deep music generation tasks.

Jukebox [13], a generative model for music, has significantly impacted the deep music generation field due to its impressive capabilities in generating high-quality and coherent music samples across a wide range of genres and styles. Developed by OpenAI, Jukebox is based on a hierarchical VQ-VAE architecture, which enables the model to effectively capture long-range dependencies and multi-scale structures present in music. The Jukebox model consists of three separate VQ-VAEs, each operating at a different level of the hierarchical architecture. The first level downsamples the raw audio data, while the second level focuses on capturing mid-level structures, such as musical phrases and local patterns. The third level captures the highest-level abstractions, such as long-range dependencies and global structures in the music. These hierarchical levels work together to enable the generation of music samples with a high degree of coherence and quality.

A key aspect of the Jukebox model is its ability to condition the generation process on various metadata, such as genre, artist, or even lyrics. This conditioning information is passed through an embedding layer and combined with the latent codes at each level of the hierarchy. This allows the model to generate music that is not only coherent but also customized according to specific user preferences. To generate new music samples, Jukebox utilizes a top-down approach, first generating high-level latent codes from the topmost VQ-VAE, then progressively refining the generated audio through each lower level of the hierarchy. This approach allows for the efficient generation of music samples while maintaining the high quality and diversity characteristic of the VQ-VAE models.

3.3.2 Non-Vector-Quantized VAE

One notable model in the domain of non-vector-quantized VAEs for music and audio generation is RAVE [8]. RAVE is a VAE-based model that focuses on generating high-quality audio samples efficiently. The model uses a customized architecture that leverages the strengths of VAEs while addressing some of the limitations associated with generating high-quality audio.

In RAVE, the authors employ a convolutional VAE architecture with a masked autoencoder for distribution estimation (MADE) [27] as the prior distribution for the latent codes. The MADE prior allows the model to capture complex dependencies between latent dimensions, leading to a more expressive latent space. This results in improved sample quality while still maintaining the fast inference capabilities characteristic of VAE models. The architecture of RAVE includes a hierarchical structure with several levels of encoding and decoding. At each level, the model employs dilated convolutions, which allow it to capture long-range dependencies in the input audio data. One key aspect of RAVE is its ability to generate audio samples in parallel, which significantly speeds up the generation process. This is achieved by employing a parallel WaveGAN [110] architecture in the final decoder stage, allowing for the efficient generation of high-quality audio samples.

3.4 NFs Models

Normalizing flows (NFs) transform a simple distribution into a more complex one through a series of invertible mappings. They provide several benefits for deep music and audio generation, including exact likelihood computation, tractable inference, and efficient sampling. In this

section, we discuss influential normalizing flow models that have shaped the field of music and audio generation.

RealNVP [15] serves as a foundational work in normalizing flows, introducing a series of invertible transformations called affine coupling layers. These layers enable the efficient computation of both forward and inverse mappings, paving the way for subsequent advancements in the field.

Glow [50] extends the concepts of RealNVP by incorporating a new type of invertible transformation, the invertible 1×1 convolution. This addition further enhances the efficiency and expressiveness of normalizing flows, allowing Glow to learn more intricate audio data distributions.

Latent Normalizing Flows for Discrete Sequences [119] explores the application of normalizing flows to discrete data, such as symbolic music representations. The authors propose a method that uses normalizing flows in continuous latent space to learn latent-variable models of discrete sequences. This approach enables the modeling of complex dependencies between discrete variables while preserving the benefits of normalizing flows, such as efficient sampling and exact likelihood computation.

Blow [93], a normalizing flow model specifically designed for raw-audio voice conversion tasks, features a single-scale architecture that simplifies the overall structure and enables more efficient training and inference. Blow employs hyperconditioning to condition the flow of auxiliary information, such as speaker identity or linguistic content, making it well-suited for audio applications.

Argmax Flows [38], a recent work in the field, develops a framework for learning categorical distributions using normalizing flows. The authors propose two methods: Argmax Flows, which models the categorical distribution through a continuous relaxation, and Multinomial Diffusion, which extends probabilistic diffusion models to categorical variables. These methods allow for the modeling of complex categorical distributions while retaining the advantages of normalizing flows, such as tractable inference and efficient sampling.

3.5 GAN Models

Generative Adversarial Networks (GANs) have demonstrated remarkable success in various domains, including music and audio generation. In this subsection, we will discuss three influential GAN models that have impacted the field of deep music generation.

WaveGAN [17] is an early attempt to apply GANs to raw waveform synthesis. The model consists of a generator and a discriminator network trained in an adversarial fashion. WaveGAN is capable of generating high-quality audio samples from random noise and has been used for various tasks, including speech, music, and sound effect generation. Conditional WaveGAN (cWaveGAN) [58] extends WaveGAN by conditioning the generation process on additional input, such as a class label or metadata. This conditioning allows cWaveGAN to generate more diverse and contextually relevant audio samples.

GANSynth [19] is another notable model that leverages GANs for audio synthesis. Unlike WaveGAN, which operates directly on raw audio waveforms, GANSynth employs a spectral representation of audio, specifically, the log magnitude of the STFT. This representation enables the model to generate high-quality audio samples with improved frequency-domain control.

GANSynth utilizes a progressive growing training scheme, which allows it to generate samples at various resolutions and achieve faster training convergence compared to traditional GAN models.

MelGAN [56] is a GAN model specifically designed for conditional waveform synthesis. It generates high-quality audio waveforms from mel-spectrogram features. MelGAN's generator consists of a stack of transposed convolutional layers followed by residual blocks, while the discriminator uses a multi-scale architecture, enabling the model to capture both local and global features of the generated audio. One of the main advantages of MelGAN is its highly efficient inference, which is significantly faster than real-time on standard hardware.

Parallel WaveGAN [110] builds upon the success of WaveGAN and improves its architecture further. It employs a multi-resolution spectrogram loss during training, which helps the model generate high-fidelity waveforms. The generator architecture in Parallel WaveGAN is based on the WaveNet [71] model, while the discriminator utilizes a multi-scale design similar to MelGAN. By leveraging the strengths of both MelGAN and WaveNet, Parallel WaveGAN achieves superior audio quality and faster inference compared to previous models.

StyleGAN [47; 48] is a groundbreaking GAN architecture that introduces a novel style-based generator design. Although originally proposed for image synthesis, StyleGAN has been adapted to generate high-quality audio samples as well. The key innovation in StyleGAN is the introduction of adaptive instance normalization (AdaIN) layers, which allow the model to control the style and content of the generated samples separately.

HiFi-GAN [52] is a generative adversarial network specifically designed for efficient and high-fidelity speech synthesis. It builds upon the successes of MelGAN and Parallel WaveGAN, incorporating lessons learned from these models to achieve superior audio quality. The generator architecture of HiFi-GAN is based on hierarchical multi-scale residual networks, while the discriminator uses a multi-scale design. During training, the model employs a combination of adversarial, feature-matching, and perceptual losses to guide the generation process. HiFi-GAN++ [2] extends HiFi-GAN to address bandwidth extension and speech enhancement tasks. The main contribution of HiFi-GAN++ is a novel generator architecture that incorporates additional modules: SpectralUnet for spectral preprocessing, WaveUNet as a convolutional encoder-decoder network, and SpectralMaskNet for learnable spectral masking.

Lastly, Musika [75] introduces a rapid music generation system that can be trained efficiently on extensive music datasets and enables faster-than-real-time generation of arbitrary-length music on consumer-grade hardware. The system employs adversarial autoencoders to learn a compact, invertible representation of spectrograms, and a GAN is subsequently trained on this representation for specific music domains. A latent coordinate system allows for the parallel generation of long music sequences, while a global context vector ensures stylistic coherence throughout the generated music. Musika supports both unconditional and conditional generation and accommodates various conditioning signals, such as note density and tempo information.

3.6 Transformer Models

In recent years, Transformer models have revolutionized deep learning by delivering powerful sequence-to-sequence modeling capabilities with efficient attention mechanisms [107]. Their prowess in capturing long-range dependencies and generating coherent sequences has enabled their use in various domains, including music and audio generation. In this section, we delve into some groundbreaking Transformer-based models that have significantly advanced deep music generation.

A notable model designed explicitly for music generation tasks is the Music Transformer [39]. By building on the original Transformer architecture and incorporating the concept of relative positional encoding, it captures the structure of music more effectively. The Music Transformer surpasses the limitations of earlier models, such as RNNs, by generating coherent and expressive music with a longer-term structure. The model's impressive capabilities include creating complex pieces of music, featuring polyphonic compositions with multiple voices and intricate rhythmic patterns.

Shuffle-Exchange Networks [25] offer an innovative approach to sequence processing. By combining the powerful attention mechanism of Transformers with greater computational efficiency, the Shuffle-Exchange Network (SEN) optimizes input sequence rearrangements to perform sequence-to-sequence tasks in $O(n \log n)$ time, compared to the $O(n^2)$ complexity of a standard Transformer.

3.6.1 Applications in Audio Generation

FastSpeech [87] is a text-to-speech model that builds upon the Transformer architecture. Instead of using an autoregressive model, FastSpeech employs a non-autoregressive approach, which enables faster and more robust speech synthesis. The key innovation of FastSpeech is the introduction of a duration predictor, which predicts the length of the generated mel-spectrogram frames. This allows the model to generate speech with fine-grained control over prosody and naturalness.

FastSpeech 2 [86] is an extension of the original FastSpeech model, which further improves the quality and controllability of the synthesized speech. FastSpeech 2 introduces a variance adaptor, which predicts the pitch, energy, and duration of the generated speech, allowing for better control over prosody and expressiveness. This model also employs a knowledge distillation approach to transfer the autoregressive teacher model's knowledge to the non-autoregressive student model, leading to improved performance.

AudioGen [54] is a model for generating audio guided by text, addressing the difficulty of producing high-quality audio samples based on descriptive text captions. The model is comprised of two primary phases: the initial phase encodes raw audio into a discrete series of tokens by employing a neural audio compression model. This audio representation generates high-quality audio samples while maintaining a compact representation. The subsequent stage incorporates the use of an autoregressive Transformer-decoder language model. This model functions by working on the discrete audio tokens that have been derived from the initial phase, conditioning these tokens based on the provided textual inputs. The text is represented

using a separate pre-trained text encoder model, specifically T5 [82], enabling the model to generalize to text concepts that may not be present in the current text-audio datasets.

Compared to existing text-to-audio works, AudioGen generates samples with better objective and subjective metrics, resulting in more natural-sounding audio compositions not seen in the training data. The authors also demonstrate the model's ability to extend to audio continuation, both conditionally and unconditionally. This innovative approach to textually guided audio generation has the potential to impact deep music generation tasks by enabling the creation of complex and diverse audio samples conditioned on textual descriptions.

3.7 Diffusion Models

Diffusion models have rapidly evolved into a highly promising technique in deep generative modeling, establishing themselves as the state-of-the-art approach in various tasks [14]. Although primarily popular in the realm of image generation, the architectural innovations introduced by diffusion models can be readily adapted and applied to audio and music generation fields.

A critical development in diffusion models involves fusing diffusion-based techniques with transformer architectures. This hybrid approach combines the strengths of both paradigms, offering a flexible and powerful framework for generative modeling. These state-of-the-art models deliver high-quality results while maintaining a high level of control over the generative process, making them particularly attractive for applications in music generation.

The success of diffusion models in image generation has opened the door for their application to audio generation. These models have already demonstrated their potential by producing

high-quality results, surpassing the performance of other architectures in the field. Additionally, diffusion models for audio generation boast faster generation speeds compared to autoregressive approaches, addressing one of the longstanding challenges in the field.

In this master's thesis, we use diffusion models as the foundation for our implementation. We will conduct a thorough investigation of various diffusion model architectures specifically designed for audio generation, examining their distinct characteristics and evaluating their impact on the deep music generation domain. Throughout our analysis, we will assess the strengths and weaknesses of these models and explore their potential to drive future innovations in the field of deep music generation.

3.7.1 Image Generation

With the introduction of improved Denoising Diffusion Probabilistic Models (iDDPM) [70], the authors significantly reduced the number of diffusion steps while enhancing the sample quality. This achievement is attributed to several modifications to the original DDPM architecture. The authors focus on learning the reverse process variances by altering the learning objective of the original DDPM. Furthermore, they propose a new cosine noise scheduler that yields improved results compared to the original linear noise scheduler [35].

Cascaded Diffusion Models (CDMs) [36] enable the generation of high-fidelity images in a sequential manner. A CDM includes multiple diffusion models operating sequentially, each responsible for generating images with progressively higher resolution. The pipeline initiates with a base diffusion model that generates the coarsest, or lowest-resolution image. This image then undergoes one or more stages of super-resolution diffusion models. These models upscale

the image and incrementally introduce finer details. The efficacy of this cascading sequence is heavily reliant on conditioning augmentation. To this end, the authors introduce a novel technique for enhancing the lower-resolution inputs that condition the super-resolution models. The key innovation in CDMs is the conditioning augmentation for super-resolution models, which is crucial for achieving high sample fidelity.

Denoising Diffusion Implicit Models (DDIMs) [96], in comparison to their counterparts, are viewed as a superior and more productive variant of iterative implicit probabilistic models that follow the same training methodology as DDPMs. The generative process in DDPMs is framed as the inverse of a particular Markovian diffusion process. To enhance DDPMs, the researchers introduced a variety of non-Markovian diffusion processes which maintain the same training target. These non-Markovian processes might represent deterministic generative processes. This leads to the creation of implicit models that can produce superior quality samples at an impressively increased speed.

Diffusion Autoencoders (DAEs) [79] utilize DDPMs for representation learning, aiming to achieve an interpretable and decodable image representation via autoencoding. This strategy includes using a trainable encoder to recognize high-level semantics and employing a DDPM as the decoder to manage remaining stochastic variations. The DAE transforms images into a dual-component latent code. The first component holds semantic importance and exhibits linearity, while the second component captures stochastic, or random, details, thereby enabling accurate reconstructions. This method accommodates advanced applications like altering attributes in

actual images, amplifies the efficiency of noise reduction, and simplifies numerous subsequent tasks, such as few-shot conditional sampling.

Latent diffusion models (LDMs) [88] were developed to maintain the quality and adaptability of Denoising Diffusion Probabilistic Models (DDPMs) while addressing the constraints of limited computational resources. By applying DDPMs within the latent space of pre-trained autoencoders, LDMs strike an optimal balance between reducing complexity and preserving detail, significantly enhancing visual fidelity. Incorporating cross-attention layers into the LDM architecture allows for generating various conditioning inputs, such as text or bounding boxes, enabling high-resolution synthesis through a convolutional approach. LDMs demonstrate superior scalability when dealing with high-dimensional data, resulting in more accurate and detailed reconstructions. Moreover, LDMs feature a versatile conditioning mechanism based on cross-attention, which supports multi-modal training and various applications, including class-conditional, text-to-image, and layout-to-image models.

Distillation. The distillation process for diffusion models aims to reduce the number of timesteps required during generation without compromising the quality of the generated samples. This process is particularly important for improving the efficiency of diffusion models in both the training and inference phases.

Salimans et al. [90] introduce a novel parameterization for diffusion models, which enhances their stability when using fewer sampling steps. In addition, the authors propose a method for distilling a trained deterministic diffusion sampler that requires fewer sampling steps. This process, called progressive distillation, is applied iteratively, halving the number of required

sampling steps at each iteration while maintaining high sample quality. The proposed approach significantly reduces the sampling time of diffusion models for both unconditional and class-conditional image generation. The progressive distillation procedure is also efficient in terms of computational resources, as its total runtime is comparable to that of training the original model.

Meng et al. [65] address the computational expense of classifier-free guided diffusion models during inference. The authors propose a two-stage distillation approach to improve the sampling efficiency of these models. In the first stage, a single student model is introduced to match the combined output of the teacher's conditional and unconditional models. In the second stage, the student model is progressively distilled into a model that requires fewer sampling steps, using the method introduced by Salimans and Ho in [90]. The distilled model is capable of handling various guidance strengths and efficiently balancing sample quality and diversity. The proposed distillation framework is applicable to both pixel-space and latent-space diffusion models trained on autoencoders. The distilled models are demonstrated to generate high-quality results using as few as 2-4 denoising steps in text-guided image editing and inpainting tasks.

Song et al. [97] introduced a novel family of generative models called Consistency Models (CMs) designed to overcome the slow sampling speed of diffusion models, which has limited their potential for real-time applications. Consistency models achieve high sample quality without adversarial training and are capable of fast one-step generation. Additionally, they enable a few-step sampling for trading to compute resources for improved sample quality, and

they support zero-shot data editing tasks, such as image inpainting, colorization, and super-resolution, without explicit training.

Consistency models can be trained either as a method to distill pre-trained diffusion models or as standalone generative models. These models build upon the probability flow (PF) ordinary differential equation (ODE) found in continuous-time diffusion models, and they learn to map any point at any timestep to the trajectory's starting point. This self-consistency property enables the generation of data samples through a single network evaluation.

The authors propose two training methods for consistency models, both based on enforcing the self-consistency property. The first method employs numerical ODE solvers and a pre-trained diffusion model to generate pairs of adjacent points on a PF ODE trajectory. By minimizing the difference between model outputs for these pairs, a diffusion model can be effectively distilled into a consistency model for high-quality, one-step sample generation. The second method, however, does not require a pre-trained diffusion model and trains a consistency model independently. This positions consistency models as a separate family of generative models.

3.7.2 Audio Generation

Diffusion models have unlocked new potential in generating high-fidelity audio samples, rivaling traditional generative methods. However, their iterative nature often leads to slow sampling speeds, limiting their use in real-time applications. As a result, recent research has focused on improving the efficiency of diffusion models for audio generation by reducing inference time, thus expanding their potential use cases.

WaveGrad [9], an early application of diffusion models in audio generation, is a conditional model for waveform generation that estimates data density gradients. WaveGrad, which expands on previous developments in score matching and diffusion probabilistic models, begins with a Gaussian white noise signal that is progressively refined using a gradient-based sampler conditioned on the mel-spectrogram. This method provides a balanced compromise between the speed of inference and the quality of the sample, achieved by adjusting the count of refinement steps. Consequently, this effectively mediates the quality disparity between non-autoregressive and autoregressive models in the realm of audio. Notably, WaveGrad can generate high-quality audio samples in as few as six iterations, expanding the possibilities for applications in audio generation.

DiffWave [53] is a flexible diffusion probabilistic model designed for both conditional and unconditional waveform generation. It effectively converts white noise signals into structured waveforms through a constant-step Markov chain during the synthesis process. By adopting a feed-forward and bidirectional dilated convolution architecture inspired by WaveNet, DiffWave accomplishes high-quality speech synthesis while considerably cutting down on synthesis time. Its versatility allows for the creation of high-quality audio signals in both conditional and unconditional waveform generation tasks, demonstrating superior performance to WaveGAN and WaveNet in terms of audio quality and sample diversity.

FastDiff [42] is a rapid conditional diffusion model, specifically designed with speech synthesis in mind. It leverages time-aware, location-dependent convolutions, featuring a range of receptive field designs to effectively model long-duration temporal dependencies under adapt-

able circumstances. In addition, FastDiff incorporates a noise schedule predictor, enabling a reduction in sampling steps whilst maintaining high-quality generation. When deployed in an end-to-end text-to-speech synthesis system known as FastDiff-TTS, the model is able to generate superior speech waveforms, negating the need for intermediary features such as Mel-spectrograms. FastDiff sets a new standard in speech quality and offers a sampling speed that's 58 times quicker than real-time on a V100 GPU, marking the first time diffusion models have been practically viable for speech synthesis deployment.

Diffsound [111], an innovative text-to-audio generation model, comprises four main components: a text encoder, a VQ-VAE, a decoder, and a vocoder. The authors focus on designing an effective non-autoregressive decoder based on a discrete diffusion model. Diffsound predicts all mel-spectrogram tokens in one step and refines them in the next, ultimately obtaining the best-predicted results after several steps. By using contextual information from all tokens and revising any token in each step, Diffsound effectively avoids the unidirectional bias and accumulated prediction error problems associated with traditional autoregressive decoders. Experiments show that Diffsound not only achieves better text-to-sound generation results but also exhibits a generation speed five times faster than the autoregressive decoder.

Msanii [62] is a novel diffusion-based model for efficiently synthesizing long-context, high-fidelity music. Combining the expressiveness of mel spectrograms with a novel U-Net architecture and diffusion models, Msanii can synthesize high-quality music samples at a high sample rate without relying on concatenative synthesis, cascading architectures, or compression techniques. This approach represents a significant advance in the music synthesis field, as it gener-

ates long samples of high-quality music. Additionally, Msanii can be used to solve other audio tasks, such as audio inpainting and style transfer, without requiring retraining. This represents the pioneering effort to effectively utilize a diffusion-based model for generating extended music samples at high sample rates.

Riffusion [63], created as a hobby project, fine-tunes the Stable Diffusion AI model to generate images of spectrograms, which are then converted to audio clips. To generate infinite AI-generated jams, the creators pick one initial image and generate variations of it using image-to-image generation with different seeds and prompts. They create initial images that are an exact number of measures and then smoothly interpolate between prompts and seeds in the latent space of the model to produce smooth transitions between clips. By interpolating the latent space, the in-between points still sound like plausible clips, offering more interesting results than interpolating the raw audio. This approach allows for diverse and unique riffs and motifs to emerge during the interpolation process.

Moûsai [91; 92] is a text-conditional cascading diffusion model developed for text-conditional music generation. It employs a bespoke two-stage cascading diffusion technique that compresses the audio waveform using an innovative diffusion autoencoder. It then learns to create condensed latent representations conditioned on the text embedding produced by a pre-trained language model. The model utilizes an efficient 1D U-Net architecture for both cascade stages, which facilitates real-time audio generation on a single consumer-grade GPU. Moûsai allows for the generation of long-context 48kHz stereo music that extends beyond a minute based on context, while still maintaining a reasonable inference speed.

ERNIE-Music [118] can generate text-to-waveform music that can receive arbitrary texts using diffusion models. It incorporates free-form textual prompts as conditions to guide the waveform generation process. To address the lack of large parallel text-to-music datasets, the model collects data from the Internet using a comment voting mechanism. ERNNIE-Music applies conditional diffusion models to process musical waveforms and studies the impact of different text formats on learning text-music relevance. The generated music samples rival related works in terms of diversity, quality, and text-music relevance.

In parallel with Moûsai and ERNIE-Music, Noise2Music [41] introduces a novel approach presenting an innovative text-driven music generation technique using a series of cascading diffusion models. This architecture involves a generator model responsible for creating an intermediate representation based on text prompts and a cascader model that produces high-quality audio using the intermediate representation and the text prompts when applicable. Two alternative intermediate representations are explored: a log-mel spectrogram and a lower-fidelity 3.2kHz waveform. The diffusion models utilize 1D U-Nets to learn noise vectors and incorporate pre-trained language models to derive text embeddings. The proposed method successfully generates 30-second, high-quality, 24kHz music samples that capture essential text prompt characteristics, such as genre, tempo, instruments, mood, and era, while also encompassing fine-grained semantic details. Furthermore, Noise2Music contributes to developing the MuLaMCap dataset, comprising 400K music-text pairs, which is anticipated to benefit future research in music captioning, retrieval, and generation.

3.8 Multimodal Embedding Models

Multimodal embeddings have given rise to an array of models that capitalize on the fusion of different modalities, creating novel samples by encoding cross-modal information into a compact representation. This results in a more expressive and holistic representation of data. In this section, we delve deeper into the state-of-the-art models that employ multimodal embeddings for guided generation, focusing on CLIP, AudioLDM, and MusicLM.

CLIP (Contrastive Language-Image Pretraining) [80] represents a breakthrough in learning transferable visual representations, as it takes advantage of the multimodal nature of data. The model is built upon a dual-encoder architecture comprising a vision transformer [74] and a transformer-based language model [107]. CLIP's primary goal is to jointly learn a shared embedding space for images and text, which allows for the alignment of semantic content across both modalities. This alignment enables the generation of images that are semantically consistent with the given text prompts, paving the way for techniques such as StyleCLIP [77] and DALLE-2 [83]. These methods demonstrate the potential of CLIP's multimodal embeddings in guiding generative models for various image generation and manipulation tasks.

Moving to the audio domain, AudioLDM [60] is a state-of-the-art model for text-to-audio generation that generates high-quality audio from textual descriptions. The architecture of AudioLDM revolves around a mel-spectrogram-based VAE to learn continuous latent representations of audio signals. A latent model conditioned on contrastive language-audio pretraining (CLAP) embeddings is employed to enable the generation of semantically coherent audio based on the given textual prompt. The key factor contributing to AudioLDM's success is its use of the

audio-text-aligned embedding space (CLAP embeddings). This allows the model to be trained with audio data alone, resulting in a high-quality and computationally efficient text-to-audio generation system.

Lastly, MusicLM [1] stands as a state-of-the-art model for producing high-quality music based on textual descriptions. This model employs a hierarchical sequence-to-sequence architecture, building upon AudioLM's [5] multi-stage autoregressive modeling and extending it with text conditioning. A crucial component of MusicLM is the integration of SoundStream audio embeddings [115] and MuLan embeddings, which are joint music-text embeddings that enable the generation of music consistent with a text prompt without the need for captions during training. Originating from the MuLan model [40], MuLan embeddings project both music and its corresponding text description into a shared embedding space, enabling MusicLM to generate music based on the text input during inference. These embeddings share similarities with CLIP embeddings in that they both utilize multimodal information for generating content; however, while CLIP focuses on images and text, MuLan embeddings encompass music and text.

The use of multimodal embeddings, such as CLIP, CLAP, and MuLan, demonstrates the effectiveness of leveraging cross-modal information for various generation tasks, including image, audio, and music generation. These models exemplify how the fusion of modalities can lead to more expressive and versatile generative models, paving the way for future research in the realm of multimodal data representation and generation.

CHAPTER 4

AINUR

In this chapter, we delve into Ainur[1], an innovative model designed to enhance the quality of vocals in raw-generated music. Ainur leverages a hierarchical diffusion model architecture combined with Contrastive Lyrics-Audio Spectrogram Pre-training (CLASP) embeddings to guide the prior generation and improve the vocal quality. Utilizing both textual descriptions and lyrics as conditioning inputs, Ainur stands out with its remarkable capabilities in text-to-music and lyrics-to-music generation tasks, offering performance levels that are on par with the current state-of-the-art models in the field. Ainur's most distinguishing feature is its ability to achieve these impressive results with near real-time inference speeds. This propels it leagues ahead of other leading models like Jukebox [13] and MusicLM [1], outpacing them by orders of magnitude in terms of efficiency. This balance of speed and quality makes Ainur a notable contribution to the field of music generation.

The textual descriptions encompass essential metadata of a song, such as genre, similar artists, and progression sequence within a piece (e.g., part 2 of 4). On the other hand, the lyrics consist of text fragments aligned with the words uttered during specific time spans in the song. Thanks to the contrastive lyrics-audio pre-training, the lyrics' representations are

[1]The name "Ainur" is inspired by J.R.R. Tolkien's *The Silmarillion*, in which the Ainur are divine spirits responsible for creating the world through their music. This creation story is known as the "Ainulindalë," meaning "The Music of the Ainur" in Tolkien's Elvish language.

harmoniously aligned with the audio, enabling Ainur to generate music from music input with coherent lyrics.

Remarkably, Ainur can generate up to 22 seconds of high-quality 48,000 kHz stereo audio in almost real-time, offering an excellent balance between quality and generation time. To put this into perspective, the MusicLM model takes a whopping 153.1 seconds just to generate 5 seconds of audio. In stark contrast, Ainur comfortably creates 22 seconds of music in a mere 5.8 seconds, not only performing faster but also delivering comparable, if not superior, audio quality. This impressive performance underscores Ainur's potential for real-time, high-quality music generation. In the following sections, we will comprehensively explore Ainur's architecture, shedding light on its modules and workflow to provide an in-depth understanding of the model's capabilities and inner workings.

4.1 Architecture

The Ainur architecture, as depicted in Figure 12, draws inspiration from state-of-the-art models and novel architectures in both image and audio generation fields. The motivation for enhancing vocal generation and addressing the challenging task of lyrics-to-audio generation comes from Jukebox [13]. To the best of our knowledge, Jukebox is the only work that tackles this specific task, which presents unique challenges compared to the text-to-audio task.

A key contribution of Ainur is the incorporation of CLASP embeddings during training to guide the generation process. These embeddings are conceptually related to CLIP [80] embeddings and serve as an adaptation of CLAP embeddings [60] in the context of lyrics and music data. CLASP embeddings extend the potential of contrastive pre-training approaches to

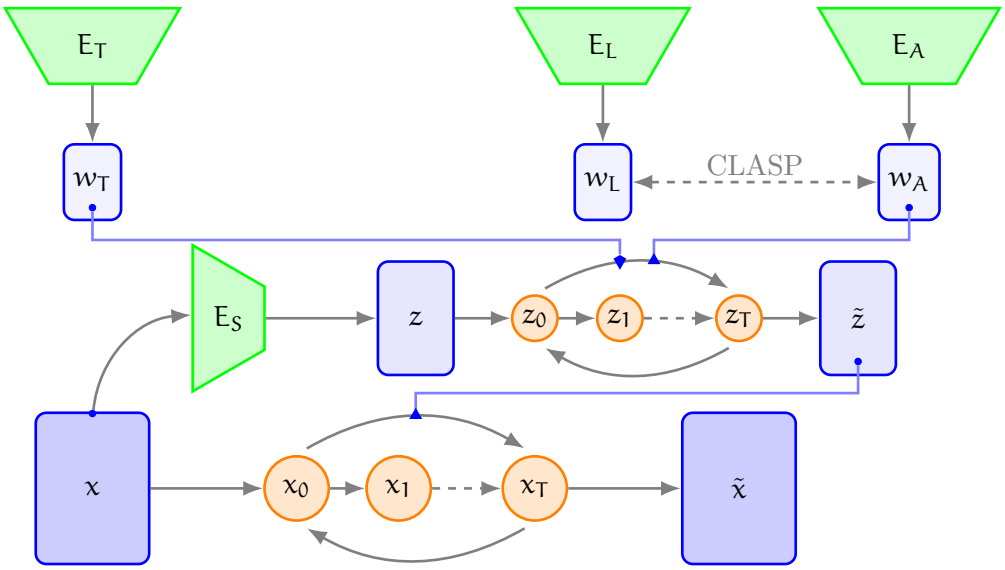

Figure 12: An illustrated overview of Ainur's architecture, showcasing its three hierarchical layers. From top to bottom: (1) input encoders and CLASP embeddings for textual and audio data; (2) a diffusion prior module guided with text embeddings and audio CLASP embeddings; and (3) a diffusion autoencoder conditioned on the generated prior for synthesizing the output. The blue kite symbol ♦ represents the cross-attention operation; blue triangles ▲ signify the conditioning of the diffusion process via latent injection.

align multimodal representations, resulting in versatile and robust representations capable of conveying coherent and structured information from both modalities.

Ainur's hierarchical structure and its strategy of leveraging different modalities and representations to guide the diffusion process are inspired by the DALLE 2 paper [83]. Like DALLE 2, Ainur performs contrastive pre-training to align representations and maximize their similarity. It then carries out diffusion prior training, followed by training a decoder to produce the output image. In Ainur, the final stage decoder is replaced by a diffusion autoencoder guided by the prior generated in the second stage.

Furthermore, Ainur's model and architecture are significantly influenced by the Moûsai model [92] for music generation. We rely on several libraries and pre-trained models provided by the authors in the ArchiSound repository[1] [91]. Like Moûsai, Ainur employs latent diffusion [88] and diffusion autoencoder [79] to model and learn the underlying structure of audio data in a more compact space while using cross attention to guide the generation with multimodal inputs. Unlike Moûsai, which uses a frozen pre-trained encoder for encoding textual descriptions, Ainur performs contrastive pre-training on lyrics-audio samples to align the encoders, producing similar latent representations. In terms of audio representation, while Moûsai uses amplitude spectrogram, Ainur adopts Mel-spectrogram representations. For the textual description embedding, Ainur utilizes the same pre-trained T5 transformer [82] employed by the Moûsai model.

4.1.1 Three-Stage Architecture

Ainur architecture employs a three-stage hierarchical model, which loosely resembles the structure of a hierarchical Variational Autoencoder (VAE) [105]. This hierarchical approach allows Ainur to effectively manage and generate the high-dimensional audio signal by relying on a series of increasingly higher-dimensional representations.

At each stage of the hierarchy, Ainur focuses on different levels of abstraction and granularity in the data, allowing the model to capture both global and local structures within the audio

---

[1] The Archinet GitHub account hosts a collection of invaluable libraries that have significantly contributed to the development of this project. Throughout our work, we have extensively utilized the repositories *audio-diffusion-pytorch* and *a-unet*, as well as the pre-trained models available in the *archisound* repository.

signal. The hierarchical framework provides a strong foundation for Ainur's architecture, as it is known for its ability to encode smaller priors and then decode them in a sequential and hierarchical manner.

1. The first stage of Ainur's hierarchical model involves generating a high-level, compact representation of the input data using pre-trained encoders. This stage focuses on capturing the global structure and contextual information present in the data.
2. In the second stage, the model utilizes these compact representations to generate intermediate representations with higher dimensions. These intermediate representations provide a richer and more detailed description of the data, enabling the model to incorporate more specific information about the content and structure of the generated music.
3. Finally, in the third stage, the model uses the intermediate representations to guide the generation of the high-dimensional audio signal. At this point, the model is capable of capturing the fine-grained details and nuances of the audio signal, resulting in a high-quality and coherent output.

The hierarchical design of Ainur offers several advantages, including improved interpretability, scalability, and modularity. By dividing the generative process into distinct stages, Ainur can focus on different aspects of the data at each level of the hierarchy, allowing the model to generate high-quality and coherent audio signals while maintaining computational efficiency. Furthermore, this hierarchical structure enables the architecture to be easily adapted or ex-

tended for different downstream tasks and performance improvements, providing a versatile framework for audio generation.

4.1.2 Hierarchical Model

As depicted in Figure 12, Ainur is designed with a sequential hierarchical structure that progresses from low-dimensional embeddings. These embeddings guide the latent prior generation through cross-attention and latent injection, which is then incorporated into the layers of the U-Net responsible for modeling the reverse diffusion process within the diffusion autoencoder. In the following sections, we will delve deeper into each architecture layer, providing a comprehensive understanding of the individual components. We will specifically highlight the multimodal nature of the inputs and discuss how the resulting output representations effectively convey the relevant information.

4.1.2.1 Lyrics-Audio Pre-Training

Lyrics and audio data inherently possess distinct characteristics, leading to notably different digital representations. On the one hand, lyrics manifest as a form of natural language that evolves over time, typically lacking explicit temporal information. They are often organized into sentences and paragraphs that correspond with the temporal progression of a song. In the case of synced-lyrics, however, temporal information is encoded as metadata or natural language, providing a richer representation of the song's structure [13].

On the other hand, audio data, specifically music, is the result of quantizing and sampling continuous information. This data is usually represented as high-quality raw audio or lossless compressed data (such as MP3). Music data exhibits a much higher dimensionality than natu-

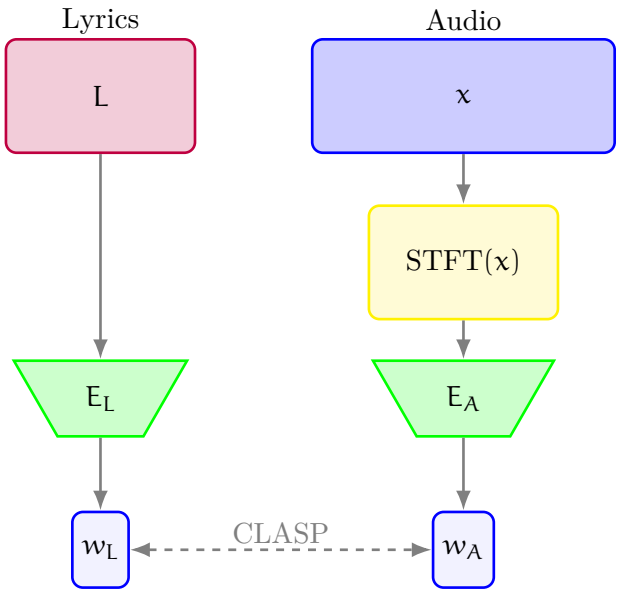

Figure 13: Close-up of the CLASP process. Audio data is first transformed into Mel-spectrograms, and then both spectrograms and lyrics are separately encoded into embeddings. These embeddings are compared, and the encoders are optimized to generate embeddings that maximize the relative similarity between the two distinct representations.

ral language and is inherently non-discrete. This discrepancy makes it challenging to align the representations of text and audio directly. However, by leveraging the low-dimensional representations of encoded text and audio, these two modalities can acquire a more similar nature, enabling the optimization of encoders to produce closely related latent representations.

In order to effectively manage both audio and text inputs, we rely on a multimodal model that is proficient at handling these different types of data simultaneously. CLIP [80] is a method that applies contrastive learning between images and natural language in order to enhance the correspondence between information related to an image and the image itself. This is achieved

by optimizing the encoders to produce (text, image) tuple embeddings with the highest scaled pairwise cosine similarity when compared to all other non-corresponding pairs of text and images within the same data batch. Following this concept, CLAP [60] aligns natural language and audio by first transforming the audio into a 2D representation using the STFT and then applying the same idea as CLIP.

Building on this foundation, CLASP embeddings are representations of lyrics and audio spectrograms that work to enforce the similarity between (lyrics, audio) tuples. Input lyrics, which are synced with the audio, embed temporal information in the form of natural language. These lyrics are preprocessed and cropped to align with the corresponding audio fragment. Consequently, CLASP embeddings inherently encode the temporal information associated with lyrics data as well as the ordinal information of the sequence of textual tokens embedded in the transformer encoder through positional embeddings [107].

As depicted in Figure 13, the spectrogram encoder in CLASP is a simple vision transformer [18] and a causal language model to get the text features. Rather than training from scratch, CLASP leverages transfer learning by using the CLIP checkpoint as a baseline for training. When fine-tuned on the data in the form (lyrics, spectrograms), CLASP effectively generates coherent vocals-lyrics embeddings from both audio and lyrics, illustrating its versatility and effectiveness in handling such complex data representations.

4.1.2.2 Diffusion Prior

CLASP embeddings play a crucial role in guiding the generation of coherent lyrics in music. In addition to providing structural guidance for the content of the generated music, natural

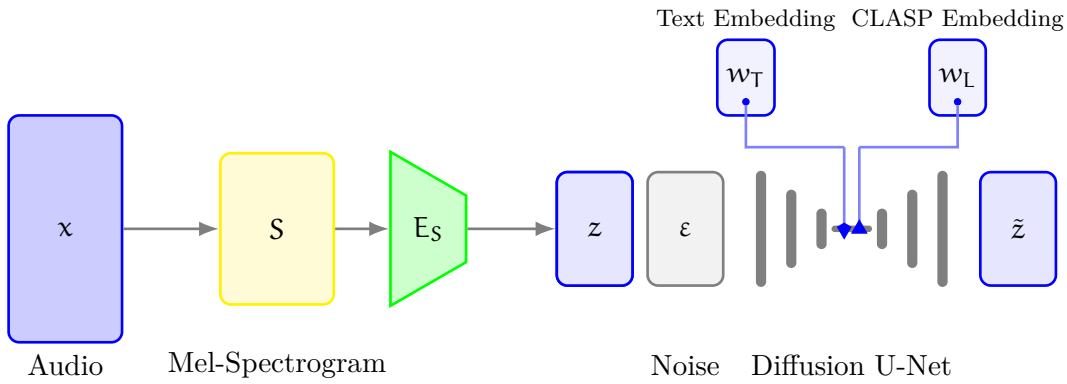

Figure 14: Close-up of the prior diffusion process. The audio x is first transformed into Mel-spectrograms and then encoded into a latent variable z. The diffusion process is guided by the textual description and the CLASP embedding through cross-attention and latent injection operations during the reverse diffusion stage.

language descriptions are utilized to supply extra information that helps steer the synthesis of music. This textual input is similar to that used in text-to-music generation, where it can express the genre or style of the song, the groups and artists that the generated piece resembles, and the temporal descriptors of the song's progression. A pre-trained, frozen T5 encoder is employed to encode the text description, and cross-attention is performed with the resulting embedding to guide the generation toward a specific style and part of the track. Additionally, CLASP embeddings are injected into the layers of the diffusion U-Net to provide structured multimodal information of audio and lyrics to condition the generation of music.

Following the approach in [88], cross-attention is not performed in the high-dimensional audio space but rather in the latent space of an autoencoder, as depicted in the close-up view in Figure 14. This prior represents a compressed version of the original audio sample. To

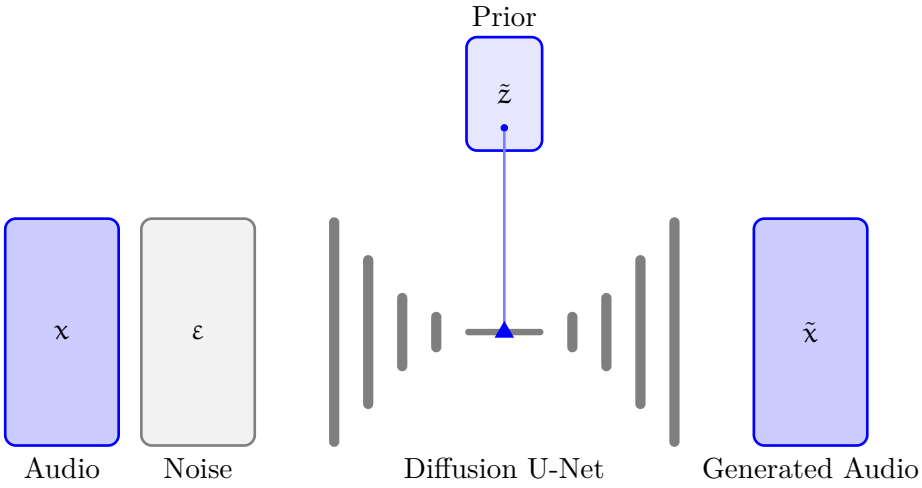

Figure 15: Close-up of the diffusion autoencoder. By incorporating the previously generated latent variable $\tilde{x}$ into the U-Net architecture, the decoding process is able to reverse the noise and generate a new audio sample $\tilde{x}$.

achieve this, the audio sample is first transformed into a frequency-time representation using the STFT and then mapped onto the Mel scale for a more perceptually accurate representation. Subsequently, the spectrogram is further compressed using a simple 1D U-Net encoder [91]. The resulting latent serves as the central element in the latent diffusion process with cross-attention and latent injection guidance.

Learning to generate this low-dimensional prior is essential for real-time generation of high-quality audio and effectively allows the guidance of the generation process using the attention mechanism. This would be infeasible in a high-dimensional environment, thus making the compression and guidance of the latent space a pivotal aspect of the Ainur architecture.

4.1.2.3 Diffusion Autoencoder

The final layer of the Ainur architecture features a diffusion autoencoder capable of generating high-quality 48,000 kHz stereo audio (Figure 15). The ability to directly model such a high-dimensional signal is attributed to two key aspects of this architecture:

1. The U-Net is utilized for modeling the reverse diffusion process, which has a unique architecture that performs 1D convolution operations. This enables the network to significantly speed up the downsampling and upsampling operations, resulting in a considerable overall acceleration of the process.
2. The prior, generated in the previous stage, is injected into the layers of the U-Net and concatenated with the noise of the reverse denoising process. This enables the network to function as an autoencoder, with the generation process being aided instead of relying solely on pure noise for generation.

These features allow the Diffusion Autoencoder (DAE) to handle high-dimensional data while maintaining close-to-real-time generation performance. Injecting the prior during the denoising process enables the model to act as both a decoder and a vocoder [92].

Ainur employs a pre-trained DAE from the ArchiSound [91] library. This model has 86 million parameters and can compress the input signal into a latent representation with a 32x compression rate. Consequently, the prior size becomes manageable, making it feasible to effectively incorporate useful information while not being overly compressed, complicating the diffusion process, as suggested in [88]. In [92], the authors discovered that satisfactory results

in audio latent diffusion can be achieved with compression factors up to 64x. As such, we determined that a 32x compression factor offers a good balance between reducing data dimensionality and retaining a reasonably uncompressed data representation.

4.1.3 Encoders

Ainur utilizes a variety of encoders to create compact and more manageable representations of data. Each encoder serves a unique purpose within the Ainur ecosystem. The CLASP representation learning forms a crucial component of Ainur, and the encoders employed play a vital role throughout the entire architecture. In this section, we will delve into the details of the encoder architectures used and discuss how these encoders contribute significantly to Ainur's overall functionality.

4.1.3.1 Text Transformer

CLASP, which is fine-tuned on CLIP, employs the same text encoder architecture as presented in the original CLIP paper. This text encoder is a powerful masked self-attention Transformer designed to handle natural language processing tasks effectively [107]. The architecture of the Transformer model consists of several layers that are identical in structure, each incorporating a multi-head self-attention mechanism, which is then followed by feed-forward networks that are fully connected and position-aware. This multi-head self-attention mechanism enables the model to focus on different parts of the input text, making it highly efficient at capturing long-range dependencies and contextual information.

To process input text sequences in CLIP, the text encoder first converts the text into token embeddings. These token embeddings are then combined with positional embeddings, allowing

the model to incorporate the position of each token within the sequence. The combined embeddings are fed into the Transformer layers, which learn to encode the input text into a high-level representation that captures both semantic and syntactic information.

The masked self-attention mechanism in the text encoder permits the model to concentrate on relevant parts of the input text while disregarding the masked sections. This ability helps the model to learn word dependencies more effectively and generate coherent, contextually accurate representations. By using the same text encoder architecture from CLIP, CLASP can build upon the powerful natural language processing capabilities of the Transformer model.

4.1.3.2 Vision Transformer

CLASP also leverages the same vision encoder as used in CLIP, which is based on the Vision Transformer (ViT) architecture [18]. Specifically, CLASP employs the ViT-B/32 variant, which is designed to process images effectively while maintaining a manageable model size.

The ViT architecture adapts the original Transformer model for image processing tasks. Instead of directly processing raw pixel data, the ViT first divides the input image into non-overlapping fixed-size patches. Each patch is then linearly embedded into a flat vector, forming a sequence of patch embeddings. These embeddings are combined with positional embeddings to account for the spatial location of each patch within the image. This sequence of patch embeddings is then processed by the Transformer layers, which are identical to those used in the text encoder.

The ViT-B/32 variant of the Vision Transformer, in particular, uses a patch size of 32×32 pixels, resulting in a balance between the granularity of the input representation and compu-

tational efficiency. The multi-head self-attention mechanism in the vision encoder enables the model to focus on different parts of the input image, capturing both local and global contextual information.

By using the ViT-B/32 architecture for the vision encoder, CLASP takes advantage of the powerful image processing capabilities of the Vision Transformer, allowing it to effectively learn high-level representations of image data that can be aligned with textual information.

4.1.3.3 T5

Ainur employs a T5 Transformer as a frozen pre-trained encoder for handling textual description input. The T5, or Text-to-Text Transfer Transformer, is a state-of-the-art model specifically designed for natural language processing tasks [82]. It follows the encoder-decoder architecture and is based on the original Transformer model introduced by [107].

In Ainur, only the encoder part of the T5 is utilized. The T5 encoder is composed of a series of identical layers. Each of these layers houses a multi-head self-attention mechanism which is then succeeded by feed-forward networks that are fully connected and consider the position of each element. Like other Transformers, the T5 model also incorporates positional embeddings to account for the position of each token in the sequence. The multi-head self-attention mechanism enables the model to focus selectively on different parts of the input text, capturing both local and long-range dependencies effectively.

The T5 model has been pre-trained on a wide range of natural language processing tasks using a text-to-text transfer learning approach, which allows it to generate powerful and contextually rich text representations. By using a pre-trained T5 encoder and keeping it frozen during

training, Ainur leverages the extensive knowledge and understanding of language that the T5 model has already acquired. This enables Ainur to effectively process textual descriptions and incorporate them into the overall generative process.

4.1.3.4 Spectrogram Encoder

Ainur employs a straightforward 1D U-Net encoder, which is a component of the ArchiSound DAE[1], to process the Mel-spectrograms of the input audio. This encoder is designed to efficiently compress the audio representation while retaining the vital information needed for the generative process.

The input for this encoder is an 80-channel Mel-spectrogram created by applying the STFT to the audio signal. The utilization of 80 channels is a strategic choice, as it allows for a comprehensive representation of the audio signal. This level of granularity helps maintain a healthy balance between capturing essential details and ensuring computational feasibility, hence preserving the overall quality of the audio data. In Ainur, the STFT is conducted with a window size of 1024 samples and a hop length of 256 samples between consecutive frames. This approach provides a detailed frequency analysis while preserving a suitable time resolution. The window function spans 1024 samples, and the audio signal has a sampling rate of 48 kHz.

Following the STFT, the spectrogram is transformed into a Mel-spectrogram by mapping the frequency bins to 80 Mel-frequency bands. This conversion yields a perceptually meaningful representation of the audio signal, as it more accurately reflects the human auditory system's response to various frequencies.

The encoder aims to produce a compressed version of the input Mel-spectrogram while maintaining the crucial information required for the generative process. The encoder output is a compressed input representation, downsampling the original data by a factor of 2 to produce 32 channels. This downsampling enables the model to generate a more compact and manageable representation of the audio data while still retaining the essential information needed for the subsequent generative stages.

4.1.4 Diffusion

Ainur's music generation process leverages diffusion models, specifically incorporating two diffusion models utilizing the v-diffusion objective [90; 92]. These models are at the core of Ainur's ability to generate rich and diverse musical content. In this section, we will discuss the diffusion models employed in Ainur, with a particular focus on the latent diffusion model's classifier-free guidance [37] that enables controllable generation.

Diffusion models have emerged as a powerful paradigm for generative modeling, capable of producing high-quality samples while maintaining computational efficiency. In Ainur, the v-diffusion objective is the foundation for both diffusion models, offering a robust and flexible framework for learning and generating complex musical structures. The v-diffusion (Equation 4.1) objective builds upon previous work in denoising diffusion models and extends their capabilities, allowing for more efficient and versatile generation processes. The v-objective is formulated as follows:

$$\mathcal{L} = \mathbb{E}_{t\sim[0,1],\sigma_t}\left[||\widehat{\boldsymbol{v}}_{\sigma_t} - \boldsymbol{v}_{\sigma_t}||_2^2\right] \tag{4.1}$$

The objective in Ainur's diffusion models deviates from the traditional diffusion equation, as expressed in (Equation 2.16). In this case, $\widehat{\boldsymbol{\nu}}_{\sigma_t}$ and $\boldsymbol{\nu}_{\sigma_t}$ represent the generated and original velocities, respectively. These velocities are defined as the derivative $\frac{\partial \mathbf{x}_{\sigma_t}}{\partial \sigma_t}$, where σ_t denotes the noise level. The velocity term captures the extent to which a data point changes due to a small alteration in the noise level σ_t.

The first diffusion model used in Ainur is the diffusion autoencoder, which is responsible for encoding the input data into compact latent representations and subsequently decoding them to generate the final audio signals.

The second diffusion model, the latent diffusion model, generates the high-level latent representations that guide the autoencoder during the music generation process. In order to enable controllable generation, the latent diffusion model employs classifier-free guidance [37]. This approach allows for manipulating the latent representations in a meaningful and intuitive way, enabling users to exert precise control over the generated music's attributes and characteristics.

By combining these two diffusion models and leveraging the v-diffusion objective, Ainur is able to generate high-quality and diverse musical content that can be precisely controlled by the user. The hierarchical structure of the models and the incorporation of classifier-free guidance ensure that Ainur remains both efficient and effective in generating a wide range of musical styles and genres.

4.1.5 Modularity

Ainur's architecture combines various components and integrates both trainable and pre-trained modules into a cohesive, unified structure. While we have thoroughly examined each

component of Ainur, it is important to note that the architecture is entirely modular and customizable, allowing for using different or more advanced modules. The selection of modules for Ainur has been primarily driven by three main objectives:

1. **Transfer Learning**: To achieve impressive results with limited computational resources, we emphasize utilizing transfer learning. This approach allows us to leverage pre-trained models as baselines for Ainur's architecture and to expedite the development process.
2. **Reliability**: Some pre-trained components of Ainur, such as the T5 text encoder, were chosen not only for their excellent performance but also for the maturity and reliability of the underlying research. By doing so, we avoid using immature architectures that might lead to undocumented issues during the model's development.
3. **State-of-the-art**: Ainur incorporates architectures that have been established as state-of-the-art in their respective tasks. As a result, Ainur can produce high-quality results by building upon the accomplishments of these cutting-edge models.

Despite these guiding principles, each module in Ainur's architecture can be altered or updated with more advanced or alternative modules to perform different downstream tasks or enhance its overall performance.

4.2 Multimodal Control

Ainur boasts a flexible conditioning mechanism, allowing for a wide range of control over the generated output. The model can accept various inputs, such as textual descriptions of the desired song, lyrics to be woven into the music, or audio signals that condition the generation

process. Each input signal is encoded and harnessed to influence the generation of the prior representation.

Thanks to the CLASP embeddings, Ainur can seamlessly handle different types of input signals, including lyrics and audio. These representations are specifically designed to be aligned, which enables Ainur to treat them consistently. As a result, Ainur can adeptly perform lyrics-to-music and music-to-music generation tasks. Moreover, when certain inputs are omitted, the model's task changes accordingly. For instance, omitting lyrics or audio signals leads to a standard text-to-music generation task, while removing both the text description and lyrics or audio signals turns Ainur into an unconditional music generation model. This level of adaptability is unparalleled in the music generation field, making Ainur the first of its kind to significantly enhance user experience in generating music.

In the next section, we will explore the various conditional signals employed to guide music generation in Ainur, shedding light on how these signals contribute to creating unique and engaging musical outputs.

4.2.1 Lyrics

Ainur's training process incorporates lyrics information in the form of natural language, typically structured into sentences and paragraphs. To effectively manage the temporal aspect of lyrics as they evolve throughout a song, Ainur uses synced lyrics during the training phase. Synced lyrics embed the temporal information directly within the text by including the start time for each sentence in the song. This time information is essential for aligning the lyrics with the corresponding music timestamps. Once the lyrics and audio are aligned, the temporal

information is removed, leaving only the natural language content of the lyrics. Here's an example of synced lyrics used during the training:

«[00:45.18]I got your hey oh, now listen what I say oh
[00:54.61]When will I know that I really can't go
[00:57.69]To the well once more - time to decide on.
[01:00.49]Well it's killing me, when will I really see, all that I need to look inside.
[01:05.05]Come to believe that I better not leave before I get my chance to ride,
[01:09.29]Well it's killing me, what do I really need - all that I need to look inside.
[01:13.92]Hey oh... listen what I say oh»[1]

During the inference phase, only the lyrics information is needed, without any temporal annotations or time information. This is achievable due to the positional embeddings inherent in the transformer encoders used in Ainur. Consequently, the ordering of the vocals is already embedded within the textual representation. Ainur generates audio that aligns with the specific lyrics provided as input during inference.

---

[1]Lyrics from Snow (Hey Oh) - Red Hot Chili Peppers.

4.2.2 Text Descriptors

Ainur utilizes a textual description to guide the generation process, which is created during training by merging the metadata associated with each song. The textual description includes the following attributes:

- Name of the **artist** or **group** performing the song
- Musical **genres** and **styles** associated with the song
- Progression **sequence number** within the song

The artist's name helps the model learn to generate music that resembles the style of a specific author or group. The genre of the song is the most crucial attribute for guiding the generation toward the desired music style. Modeling the entire distribution of possible music is inherently challenging; therefore, this information is essential to steer the diffusion model toward the right data modality representing a particular genre.

The sequence number serves as an indicator, enabling the model to learn which part of the song is being played and, implicitly, the total length of the song. This information acts as a form of positional embedding, helping the model understand the structure and sequencing of a song's various parts.

To ensure robustness, the attribute positions within the textual description are randomized, and some attributes may be omitted during different epochs. Furthermore, the items are concatenated using various symbols and delimiters, which are randomly chosen based on a custom distribution that reflects how these delimiters are typically used for separating words.

The resulting text descriptors provide a flexible and comprehensive representation of the songs for the model to learn from. Some examples of text descriptors for the songs include:

```
Red Hot Chili Peppers,Alternative Rock-8 of 10

Alternative Rock R.E.M. 4 of 13

Steam Powered Giraffe,Comic 7 of 19
```

4.2.3 Audio

Ainur's music generation can be guided by various inputs, including song fragments, rhythms, or motifs. The input audio should share the same characteristics as the generated audio, which means it should have a maximum length of 21 seconds, be an uncompressed raw high-quality audio file, sampled at 48,000 kHz, and contain two channels (stereo). Providing an audio fragment as input will strongly influence Ainur's generated music to resemble the style of the conditioning song.

Furthermore, Ainur can seamlessly accept either lyrics or audio as input signals, thanks to the CLASP embedding representation. These embeddings can effectively convey the information from the input song, and due to the contrastive pre-training, they also carry structured and consistent information about the vocals present in the generated audio. This flexibility allows Ainur to adapt its music generation process based on various input types, enhancing the user experience and producing high-quality results.

4.2.4 Image

Ainur possesses a unique capability for image-to-music generation, even without being specifically fine-tuned on images. This inherent ability stems from the CLASP embeddings, which are derived from CLIP, a model trained on (text, image) pairs. As a result, Ainur can carry out image-to-music generation tasks natively without any modifications.

Although CLASP embeddings do not directly align audio embeddings, they instead encode audio as a 3- (identical) channels Mel-spectrogram, which is subsequently aligned with the textual representation. This characteristic allows Ainur to effectively perform image-to-music generation by leveraging the transfer learning benefits gained from CLIP's training process. In this way, Ainur can generate audio based on images, opening up new possibilities for creative applications and user experiences in the domain of audio generation.

4.3 Input Representation

Ainur's innovative approach to handling diverse input data is facilitated by the use of embeddings. As a common method for representing complex data in machine learning systems, embeddings are incredibly effective in managing data with varying characteristics under a unified representation. Additionally, these embeddings offer a compressed representation, making them easier to handle due to their inherently reduced dimensionality. Consequently, operations such as cross-attention and reverse denoising can be executed more efficiently.

To process each input, Ainur utilizes specialized encoders that have been specifically trained for their respective data encodings. These representations are crucial for guiding the generation of priors within a low-dimensional latent space, ultimately contributing to Ainur's ability to

create unique musical outputs. In the following discussion, we will delve into the characteristics of these multimodal data embeddings, particularly emphasizing their dimensionality. By examining these features, we can gain a deeper understanding of Ainur's capabilities in generating music content.

4.3.1 Text Embeddings

Ainur's text embeddings are generated using a pre-trained T5 encoder, which has been frozen to maintain its original state during the training process. This approach produces embeddings that consist of 768 features, effectively capturing the rich information contained within the textual input. Additionally, the maximum length of these embeddings is limited to 64 tokens, ensuring that the model's processing capacity is not overwhelmed. The utilization of the T5 encoder not only allows for an efficient and compact representation of textual data but also contributes to Ainur's impressive performance in music generation tasks. By employing such powerful text embeddings, Ainur is better equipped to handle the complexities of various input signals and generate high-quality musical outputs that are guided by the encoded information.

4.3.2 Lyrics Embeddings

Ainur's lyrics embeddings are generated using a straightforward text transformer that serves as a text encoder. These embeddings consist of 512 features and incorporate positional embeddings to encode the ordinal information of the tokens present in the lyrics. The resulting representation is not only compact and easy to manage but also effectively captures the essence of the natural language and, implicitly, the temporal dimension of the lyrics. By utilizing this efficient and informative representation, Ainur is able to understand better and process the lyri-

TABLE I: MUSIC GENERATION TASKS.

Text Descriptor	Lyrics	Audio	Image	Task
✓	✓			lyrics-to-music
✓		✓		audio-to-music
✓			✓	image-to-music
✓				text-to-music
				uncoditional music generation

cal content, which in turn contributes to the generation of musically coherent and expressive outputs that align with the provided lyrics.

4.3.3 Audio Embeddings

In Ainur, audio embeddings are derived by converting the audio signal into a Mel-spectrogram and then encoding it using a vision transformer, which functions as an image encoder. This process results in embeddings with 512 features, allowing for a consistent representation that can be easily compared with other embeddings within the CLASP framework. By utilizing this innovative approach, Ainur effectively translates the rich and intricate information contained in the audio signal into a compact and coherent format. This, in turn, enables the model to understand better and manipulate the musical characteristics, ultimately generating captivating and high-quality music that aligns with the provided inputs.

4.4 Workflow

Ainur's hierarchical architecture relies on a series of sequential operations, carefully structured to optimize the music generation process. While the training and inference workflows

differ regarding input data, both workflows share a top-down approach that starts with low-dimensional embeddings and progresses to the audio space domain. The inputs required for each stage are as follows:

1. **Training**:
    - Music
    - Text descriptors
    - Synced-lyrics
2. **Inference**:
    - Text descriptors
    - Lyrics
    - Audio
    - Image

It is worth noting that all the inputs used during inference are optional, and various combinations of them enable Ainur to tackle different tasks in an agnostic manner. The relationships between inputs and resulting tasks can be found in Table I[1]. Interestingly, when no input is provided, Ainur serves as a model for unconditional music generation. In the following sections, we will delve into the specific training and inference workflows Ainur uses at each stage.

---

[1]Relationship between the input conditioning data and the corresponding tasks performed during Ainur's inference stage.

4.4.1 Training

The high-level overview of Ainur's training workflow is illustrated in Figure Figure 12. The three-stage architecture is trained sequentially in a sandwich-like manner, involving the following steps:

1. **Contrastive pre-training**: First, the lyrics-audio contrastive pre-training occurs at the top of the architecture. Both audio and lyrics inputs are transformed into 512-dimensional embeddings by their respective encoders. In line with the CLASP framework, the encoders are jointly optimized to produce representations that maximize the similarity between matched lyrics-audio pairs and minimize the similarity between each representation and all other non-matching counterparts in the batch.

2. **Diffusion autoencoder**: Next, the diffusion autoencoder learns to compress and recover the input audio signal by optimizing an encoder that generates a latent representation. This latent representation is then injected into the denoising U-Net during the high-dimensional audio signal decoding phase, along with the noise. The input audio signal is initially transformed and mapped into a Mel-spectrogram and then encoded into a latent representation. Simultaneously, the raw waveform audio is corrupted by noise during the forward diffusion process and denoised during the reverse diffusion process, together with the injected latent spectrogram representation. The spectrogram encoder and decoding U-Net are trained to minimize the v-diffusion objective between the reconstructed audio signal and the original one.

3. **Diffusion prior**: Lastly, the middle layer is trained to reproduce and learn the underlying distribution of the latent used to guide the generation of the diffusion decoder. Due to its low dimensionality, this latent representation is particularly suitable for conditioning through cross-attention with multimodal inputs. To direct the generation, cross attention is computed using the text embeddings along with the CLASP embeddings injected into the diffusion U-Net. Similarly to the diffusion autoencoder, the v-objective between the generated latent and input latent is minimized, and classifier-free guidance with a probability of 0.1 is used. The embedding scale factor, which determines the impact of the multimodal guidance, will be an essential hyperparameter during the model's evaluation.

In Ainur, we used a pre-trained DAE, making it possible to skip the second stage of training and focus solely on the contrastive pre-training and prior generation.

4.4.2 Inference

Ainur's inference process proceeds sequentially from top to bottom, as illustrated in Figure Figure 16. The speed of inference is influenced by two crucial factors:

- **Number of diffusion steps**: The number of denoising steps required to transition from noise to music. This hyperparameter directly impacts the inference time and can be adjusted to balance quality and speed.
- **Embedding scale and conditioning inputs**: The weight of the conditioning signals used for guiding the generation may affect the convergence of reverse diffusion. Furthermore, the presence of particularly significant input signals can steer the generation more quickly toward a modality that facilitates faster inference time.

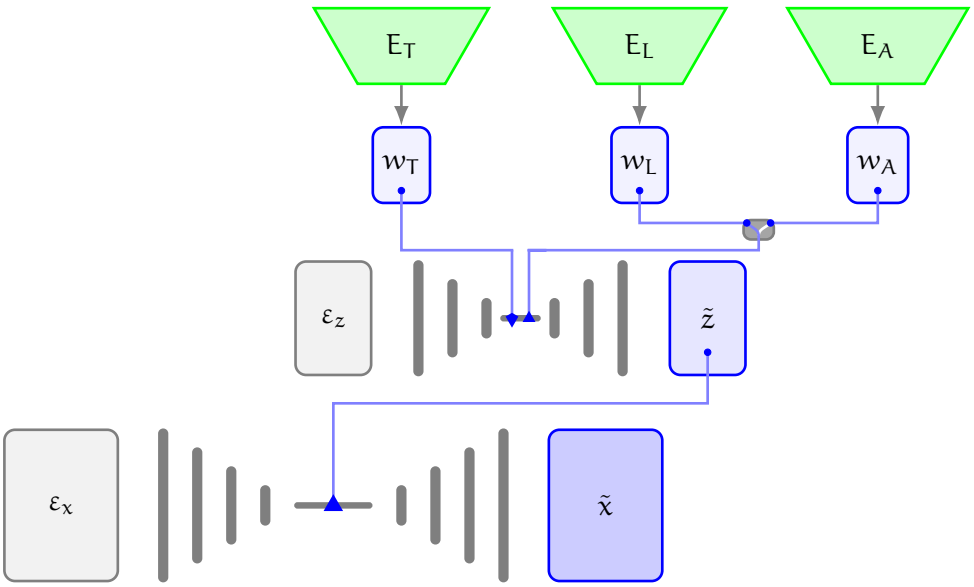

Figure 16: Ainur inference workflow. The process begins at the top with inputs. (1): text and lyrics are used by default, but inference can also be conducted using an audio input and text description. The input embeddings are used to guide the (2) prior generation from noise ε_z to reconstructed latent $\tilde{z}$, utilizing cross-attention for the text descriptors and latent injection for the lyrics/audio inputs during the diffusion process. In the bottom layer (3), noise ε_x is decoded into the generated audio $\tilde{x}$, conditioned on the generated latent. The switch is used to select the task: lyrics-to-music (default) or audio-to-music.

The inference path depends on the input selected for Ainur. By default, Ainur performs the lyrics-to-music generation task, and the outputs of the CLASP encoders (E_L, E_A) are injected during the prior diffusion along with the cross-attention with textual embeddings of the text descriptor encoder (E_T).

At this stage, all inputs are encoded into a compact representation. Unlike the training phase, no other inputs are used besides these embeddings. Starting from Gaussian noise ε_z, the model generates a latent representation $\tilde{z}$ guided by the encoded input via cross-attention.

TABLE II: STATE-OF-THE-ART AUDIO GENERATION MODELS.

Model	Year	#Parameters [M]	Sample Rate [kHz]	Channels
Ainur	2023	910	**48**	**2**
AudioGen [54]	2022	285	16	1
AudioLDM-S [60]	2023	181	16	1
AudioLDM-L [60]	2023	739	16	1
Jukebox [13]	2020	1000	44.1	1
Moûsai [92]	2022	1060	**48**	**2**
MusicLM [1]	2023	1890	16	1
Riffusion [63]	2022	890	44.1	1

This latent diffusion process is relatively fast compared to the diffusion performed in the audio space, and the generated prior includes all the information needed to generate the song in the high-dimensional space according to the conditioning inputs.

Finally, at the bottom layer, high-dimensional Gaussian noise ε_x is fed into the decoding U-Net along with the injected latent that guides the noise decoding. After a series of denoising steps, the generated audio becomes a completely new and original sample with style resembling the music distribution used for training the model.

4.5 Model Comparison

The following analysis centers on a comparative evaluation of state-of-the-art models in music and audio generation. As shown in Table II[1], we consider key metrics such as the

[1]Comparison of state-of-the-art models in music and audio generation. The table presents the model's publication year, total parameters, the audio's sample rate, and the number of channels indicating whether the audio is mono or stereo.

model's publication year, total parameters, the audio's sample rate, and the number of channels to discern whether the audio is generated in mono or stereo.

Our model, Ainur, stands out in its capacity to generate high-quality stereo audio at a 48 kHz sample rate, akin to the Moûsai model. This characteristic is crucial in producing high fidelity audio, which is a key determinant of audio quality. The higher the sample rate, the higher the frequency response and, thus, the better the sound quality. Moreover, stereo audio generation, as opposed to mono, provides a more immersive listening experience, reproducing the spatial location of sound sources for the listener, a crucial feature for music generation.

In terms of the number of parameters, Ainur holds 910 million parameters, which is less than some models such as MusicLM, Moûsai, and Jukebox. However, the number of parameters is not always directly proportional to model performance, and an increased number of parameters can sometimes lead to overfitting or increased computational costs. As such, Ainur strikes a balance between model complexity and performance.

Notably, Ainur, AudioLDM and MusicLM models were published in 2023, marking them as the most recent developments in the field. This indicates their potential to leverage the latest advancements and techniques in AI for music generation.

In conclusion, the forthcoming results in Chapter 6 will substantiate the model's potential, signifying a substantial advancement in the realm of music and audio generation. It exhibits a promising combination of high-quality audio generation in terms of sample rate and channel number, while maintaining a balanced number of parameters, which underscores its efficiency and state-of-the-art design.

CHAPTER 5

EXPERIMENTAL SETUP

This chapter serves as a comprehensive exposition of the various facets involved in the development and evaluation of the Ainur system. This section systematically elucidates the different components, such as the dataset, training setup, metrics, implementation details, hardware requirements, and reproducibility considerations. By delving into the minutiae of the research process, this chapter aims to offer readers a thorough understanding of the robust methodologies employed in the study of Ainur. Overall, this chapter sets the stage for understanding the results presented later on.

5.1 Dataset

In order to effectively train and evaluate Ainur, we meticulously assembled a dataset comprising over 31,000 music tracks, which amounts to approximately 2k hours of high-quality 48,000 kHz stereo music (more in Section 5.4.2). In addition to the raw waveform data encoded as `.flac` files, the dataset encompasses metadata information about authors and genres, which serve to create the text descriptors. Time-synced lyrics embedded within the songs are a critical component for training Ainur. By extensively crawling the web, we aimed to create a diverse and comprehensive dataset that encompasses a wide range of music styles, mirroring the contemporary landscape of digital music catalogs. To achieve this goal, the dataset spans a broad

time frame, with most songs released between the 2000s and 2020s and several selections dating as far back as the 1960s. Predominantly, the music and lyrics in the dataset are in English.

Possessing a rich and diverse dataset fosters the model's ability to capture the distribution of contemporary music more closely. This characteristic comes at the expense of a greater demand for the model to comprehend various music modalities, which can differ significantly from one another. Consequently, the learning process becomes more challenging and time-consuming than focusing on a single genre or style. Nevertheless, we were determined to develop a machine learning model capable of excelling across various music styles without overfitting to a specific genre and maintaining its capacity for generalization.

Despite the dataset's relatively modest size, the music samples used for training are segmented into approximately 20-second-long pieces, thereby increasing the number of training samples exposed to the model during each epoch. This approach is also applied to the lyrics input, which is aligned with the audio samples. The songs are cropped using overlapping 20-second windows, and the text descriptor contains information regarding the window's position within the song as well as the overall song length.

5.1.1 Evaluation Dataset

To accurately evaluate Ainur's performance on an unseen dataset, we reserved approximately 1,000 samples specifically for validation and testing purposes. The evaluation dataset, which was randomly sampled from the original dataset, contains both raw music data and the necessary song metadata, including lyrics, to assess Ainur's performance effectively.

A small portion of the evaluation dataset is dedicated to monitoring the model's progress throughout the training process. Several validations are conducted at regular intervals during training, utilizing a validation set that contains around 10 hours of music data. This data provides a rough estimate of the Fréchet Audio Distance (FAD) [49] (discussed further in Section 5.3), which serves as a metric for overseeing the training process. For the final testing of the model, we employed a larger dataset, consisting of 50 hours of music, to obtain a more accurate estimation of the FAD, thus allowing for a more precise evaluation.

5.2 Training Setup

5.2.1 Pre-Processing

The song metadata undergoes a pre-processing stage before being fed into the model. First, lyrics and text are extracted from the metadata and normalized into Unicode, and any incorrectly formatted time references are eliminated. Next, both the audio and the lyrics are randomly cropped into a window of approximately 22 seconds (2^{20} samples at 48,000 kHz) and synchronized within the same window. As the temporal alignment is sentence-based, it is possible that the lyrics may not perfectly correspond with the audio window, resulting in a potential discrepancy of one sentence more or less than the corresponding audio segment. During the alignment process, the temporal information is removed from the lyrics, leaving only natural language.

To assemble the text descriptors, group, genre, and sequence information is combined using random shuffling and random delimiters, which follow a custom distribution. If multiple genres or artists are present, they are concatenated in the text descriptors. The audio remains

unaltered and unscaled, as we opted not to normalize it to enable the model to learn different loudness levels and pitches characteristic of specific genres, thus facilitating multimodal learning. Any silent audio samples are discarded from the song dataset.

5.2.2 Loss Functions

Ainur utilizes three distinct loss functions, each corresponding to a stage within the training architecture. During the contrastive pre-training phase, Ainur optimizes the cosine similarity between the audio and lyrics encoded representation within a contrastive learning framework. The cosine similarity serves as the optimization objective for this stage. In both the prior and autoencoding stages, the v-objective optimization loss is employed to fine-tune the U-Net during the reverse diffusion process.

5.2.3 Training Procedure

We used batch sizes of 16 for CLASP training and batch sizes of 8 for prior training. CLASP training was conducted over more than 330 epochs, totaling approximately 120k iterations, and took over 120 hours on a single Nvidia V100 GPU . Ainur's prior was trained for 730 epochs and with 1M+ iterations for approximately 720 GPU hours. To stabilize the training process, a tanh bottleneck was implemented in both diffusion U-Nets to constrain values between -1 and 1, which enhances diffusibility. The training procedure inherently augments the training samples by performing random cropping of 22-second audio segments.

We employed the AdamW optimizer for training Ainur, with a learning rate of 10^{-4}, weight decay 10^{-3}, $\beta_1 = 0.95$, $\beta_2 = 0.999$, and $\epsilon = 10^{-6}$. Additionally, we utilized stochastic moving average (SWA) with a learning rate of 10^{-4}, along with an exponential moving average (EMA)

that has a β value of 0.999. During the prior stage training, we compensated for the lack of computational resources to handle large batch sizes by using gradient accumulation with a batch scheduler of $0:4, 600:2$. This scheduler progressively reduces the number of accumulation batches from 4 at epoch 0 to 2 beyond epoch 600. We also clipped gradients larger than 5 to prevent exploding gradients during training.

5.2.4 Hyperparameter Tuning

Our efforts in hyperparameter tuning were limited, as we did not conduct extensive grid or random searches. Instead, we opted for standard hyperparameter values known to work well with models similar to Ainur, as well as widely used values in the training of various diffusion models. While we acknowledge that Ainur could be further optimized, our primary focus in this research was not on extensive hyperparameter tuning. We leave the thorough optimization of Ainur and related hierarchical diffusion models to future work.

5.2.5 Validation Strategy

We monitored Ainur's training progress by conducting frequent validation steps every 10 epochs. During these validations, we employed a hold-out method with a small validation set to compute the FAD metric using the VGGish model [32]. This offered a general indication of the quality of the generated audio. The validation involved only 50 diffusion steps and an embedding scale of 7, providing a quick and rough estimate of the model's performance. During testing, we utilized several additional metrics to obtain a more comprehensive evaluation of the model using a larger reference test set of audio samples.

5.3 Metrics

In this section, we discuss the various metrics employed to evaluate Ainur's ability to generate high-quality music and vocals. Our primary focus is on the FAD [49], a widely used metric for assessing the quality of generated audio. Additionally, we will explore the coherence of the multimodal CLASP embeddings and their contribution to Ainur's generation capabilities through the use of our proposed C3 metric. By examining these metrics, we aim to provide a comprehensive understanding of Ainur's performance and its effectiveness in generating realistic and coherent music.

5.3.1 Fréchet Audio Distance

The evaluation of Ainur's performance in generating high-quality audio and vocals primarily relies on the FAD, a well-established metric for assessing the quality of generated audio. To obtain a comprehensive understanding of the model's capabilities, we employ multiple embedding models that offer different perspectives on the quality of the generated music [1]. In particular, we utilize the VGGish [32] and YAMNet [100] models to assess the musical quality, while the TRILL [95] model is employed to evaluate the vocal quality.

The Fréchet Audio Distance (FAD) is a metric used for evaluating the quality of generated audio, particularly in the context of generative models. It is based on the Fréchet Inception Distance (FID) [32], a popular metric for assessing the quality of generated images. FAD is designed to capture the similarity between two sets of audio samples, such as real audio samples from a dataset and generated audio samples from a model.

To compute the FAD, we performed the following steps:

1. **Feature extraction**: Both sets of audio samples (real and generated) are first passed through a pre-trained audio embedding model, such as VGGish, YAMNet, or TRILL. This step transforms the raw audio data into high-level feature representations, which are then used for comparison.
2. **Compute statistics**: For each set of feature representations, the mean and covariance are calculated. These statistics represent the central tendency and dispersion of the feature distributions, respectively.
3. **Compute Fréchet distance**: The Fréchet distance between the two Gaussian distributions defined by the means and covariances of the real and generated feature sets is computed. This distance measures the dissimilarity between the two distributions and is used as the FAD score.

A lower FAD score indicates that the generated audio is more similar to the real audio, and thus, the generative model's performance is better. The FAD metric is useful for evaluating generative models as it provides a quantitative measure of the audio quality and enables a comparison between different models or configurations.

Our approach to utilizing different encoding models stems from the notion that these models, due to their varying training data, are expected to measure distinct aspects of audio quality. While the VGGish and YAMNet models are trained on audio data, the TRILL model is specifically designed for speech data. As a result, the combination of these models allows us to capture both speech and non-speech aspects of the generated audio, providing a more thorough evaluation of Ainur's performance.

5.3.2 CLASP Cycle Consistency

In this work, we introduce the CLASP Cycle Consistency (CCC or C3) metric, a novel evaluation metric designed to assess the coherence between generated audio samples and their corresponding lyrics CLASP embeddings that guide the generation process. The C3 metric is inspired by the MuLan Cycle Consistency (MCC) metric [1], which evaluates the alignment of generated audio and the corresponding MuLan text embeddings. As CLASP embeddings provide a multimodal representation of lyrics and audio, the C3 metric is well-suited to measure the consistency between the two modalities in the context of our generative model, Ainur.

To compute the C3 metric, we follow these steps:

- **Generated audio embeddings**: First, the generated audio samples are encoded into CLASP embeddings using the pre-trained audio CLASP encoder. This step transforms the generated audio into high-level feature representations that can be compared with the corresponding lyrics embeddings.
- **Reference lyrics embeddings**: Next, we encode the guiding lyrics using the pre-trained lyrics CLASP encoder. This produces a set of high-level feature representations that capture the semantics of the input lyrics.
- **Cosine similarity**: Finally, we compute the cosine similarity between the generated audio embeddings and the reference lyrics embeddings. The cosine similarity measures the angle between the two embedding vectors and serves as a proxy for the coherence between the generated audio and guiding lyrics.

The C3 metric ranges between $[0, 1]$, with values closer to 1 indicating a higher similarity between the generated audio and the guiding lyrics. By computing the C3 metric on our test set, we can assess the coherence between the audio and lyrics and analyze the influence of the CLASP embeddings in guiding the generation of music. This evaluation provides valuable insights into the performance of our model and the effectiveness of the CLASP embeddings in achieving our desired generative outcomes.

5.4 Implementation Details

5.4.1 Software Frameworks

Ainur is a Python-based implementation using version 3.8.8 and leverages various popular libraries for machine learning and numerical data manipulation. The architecture of Ainur is built using PyTorch [76], while PyTorch Lightning [22] facilitates the training process. Thanks to the pre-trained CLIP model [80] for the CLASP encoders and the Mel-spectrogram-based diffusion autoencoder from the ArchiSound library [91], Ainur's training has been significantly accelerated. Furthermore, libraries provided by the ArchiNet[1] organization have played a crucial role in the development of Ainur's architecture.

Operating in the latent space of the diffusion autoencoder, Ainur's prior model takes in 32 input channels stemming from the Mel-spectrogram encoder. The diffusion U-Net then downsamples the input signal to 512 times the temporal dimension, using a sequence of downsampling factors per layer: $[1, 2, 2, 2, 2, 2]$. The U-Net consists of seven layers, with a progressively

[1] ArchiNet GitHub repository (github.com/archinetai).

increasing number of channels $[128, 256, 512, 512, 1024, 1024]$ and a corresponding number of repeating items per layer $[2, 2, 2, 4, 8, 8]$. Ainur employs self-attention in the last four layers, with 12 attention heads and 64 attention features per attention item. Cross-attention with the text descriptors occurs at all stages to guide music generation according to the chosen modalities. CLASP embeddings are injected into the first layer of the diffusion U-Net, enhancing the noise with structured embeddings that facilitate vocals generation.

5.4.2 Data Handling and Storage

The training, validation, and test datasets for Ainur were compiled over the course of a week through web scraping techniques. The music scraping procedure was conducted using Spotify's APIs[1], and the usage of this music complies with fair use regulations, as it is strictly for research purposes and devoid of any commercial intentions. This process involved gathering raw music samples accompanied by song metadata and synchronized lyrics. The acquired data was stored on BeeGFS [31], a high-performance parallel file system specifically designed for optimized storage with a throughput higher than 5 GB/s (Infiniband EDR 100Gb/s). This data was retained on the file system solely for the duration of Ainur's training and promptly deleted upon completion.

TABLE III: HARDWARE SPECIFICATIONS.

Architecture	Cluster Linux Infiniband-EDR MIMD Distributed Shared-Memory
Node Interconnect	Infiniband EDR 100 Gb/s
Service Network Gigabit	Ethernet 1 Gb/s
CPU Model	2x Intel Xeon Scalable Processors Gold 6130 2.10 GHz 16 cores
GPU Node	24x nVidia Tesla V100 SXM2 - 32 GB - 5120 cuda cores
Performance	90 TFLOPS
Computing Cores	1824
Number of Nodes	57
Total RAM Memory	22 TB DDR4 REGISTERED ECC
OS	CentOS 7.6 - OpenHPC 1.3.8.1
Scheduler	SLURM 18.08

5.5 Hardware Requirements

5.5.1 Computing Resources

The training of Ainur was carried out on high-performance computing (HPC) SLURM [112] cluster, graciously provided by HPC@POLITO[1] at the Polytechnic University of Turin. This powerful HPC infrastructure featured a Linux Infiniband-EDR MIMD Distributed Shared-Memory architecture, Infiniband EDR 100 Gb/s node interconnect, and Gigabit Ethernet 1 Gb/s service network. The cluster's processing capabilities were supported by 57 nodes, each equipped with 2x Intel Xeon Scalable Processors Gold 6130, operating at 2.10 GHz and 16 cores. Furthermore, the nodes housed 24x nVidia Tesla V100 SXM2 GPUs, boasting 32 GB

[1]Spotify web API documentation website (developer.spotify.com/documentation/web-api).

[1]HPC@POLITO website (hpc.polito.it).

of memory and 5120 CUDA cores, resulting in a total of 90 TFLOPS in performance. The HPC cluster contained 1824 computing cores and 22 TB of DDR4 REGISTERED ECC RAM memory, all managed by the CentOS 7.6 - OpenHPC 1.3.8.1 operating system and the SLURM 18.08 scheduler (Table III[1]).

Despite the impressive capabilities of the HPC cluster, we opted to train Ainur on a single GPU. This decision was made to enhance reproducibility and ensure that users with access to only a single consumer-grade GPU could still deploy and fine-tune the model effectively.

The training of CLASP employed a single non-parallelized job with 16 CPUs and 6400MB of primary memory for each CPU. The training required approximately 1 week for a total of approximately 120k iterations of finetuning with batch sizes of 64. On the other hand, training Ainur prior was a more intensive task, taking about a month. This was accomplished on a single GPU V100, running over a million iterations.

In our experimentation, we explored the possibility of scaling Ainur's model by parallelizing the training process across multiple GPUs. We found that Ainur could be trained seamlessly in a distributed environment. Thanks to the PyTorch Lightning framework, the model's training scaled gracefully with the addition of more GPUs, which in turn reduced the overall training time proportionally to the number of devices utilized. This parallel training approach employed a distributed data-parallel (DDP) strategy, synchronizing gradients across all devices to efficiently compute the backpropagation process.

---

[1]Specifications of the SLURM servers cluster used for training Ainur, detailing key hardware components and performance metrics.

5.5.2 Hardware Limitations and Challenges

Training Ainur on a single consumer GPU presented some challenges, mainly due to the time it took to achieve acceptable quality results and the memory limitations imposed by GPU devices. This required us to use smaller batch sizes. In the original CLIP paper [80], the authors were able to use much larger mini-batches of 1,712 text-image tuples, which proved to be beneficial for the contrastive learning framework. However, given the increased dimensionality of audio representations and our limited computational resources, we could only manage batches of 64 lyrics-audio (spectrograms) tuples and small mini-batches of 8 data points for the prior training.

We partially addressed this issue by using batch gradient accumulation and mixed-precision training while ensuring stability during training by applying gradient clipping with a maximum value of 5. In the future, training Ainur could greatly benefit from using more GPUs to speed up the process and more primary memory to handle larger batch sizes. During inference, the model could leverage additional GPU memory to perform more diffusion steps and generate higher-quality samples. Moreover, implementing distributed parallelization strategies would result in an overall training speedup, further reducing the time required for training.

5.6 Reproducibility

Ensuring the reproducibility of results is a crucial aspect of scientific research. To promote transparency and facilitate the replication of our work, we have made every effort to provide all the necessary content, model, and training details for the Ainur project.

The complete codebase required for creating Ainur can be found in the Ainur GitHub repository[1]. We have provided model weights and checkpoints within the repository, allowing users to fine-tune Ainur or deploy it for their own research purposes. It is important to note that we do not take responsibility for the improper use of Ainur. This software is intended for research purposes only and not for commercial use. We strongly discourage and condemn any misuse of our research.

Ainur is released under the MIT License, which is a permissive free software license. This means that users are free to use, modify, and distribute the software, provided that they include the original copyright notice and license text in any copy of the software or substantial portions of it. The MIT License does not impose any restrictions on the use of the software, whether for research or commercial purposes. However, it does not provide any warranty or liability protection; users employ the software at their own risk. By adhering to the terms of the MIT License, we aim to foster a collaborative and open research environment that encourages further advancements in the field.

Reproducibility is of paramount importance when it comes to training Ainur, and we have taken several steps to ensure that other researchers can reproduce our results or build upon our work. While the original training data used for Ainur will not be made available due to copyright constraints, and all data will be erased after the training process, we have taken

---

[1] Ainur project GitHub repository (github.com/Gio99c/ainur).

measures to ensure that the pre-trained model's weights do not explicitly contain any form of copyrighted material that can be directly employed or extracted.

To facilitate the reproducibility of our work, we will provide pre-trained models, checkpoints, and other intermediate artifacts that can be used to reproduce our results or build upon our work. We will also include comprehensive documentation and tutorials on how to use and deploy Ainur, ensuring that researchers can effectively utilize the model for their own purposes.

In our experiments, we took great care in handling random seeds and initialization to ensure that others can reproduce our results with the same level of randomness. By providing details on the specific random seeds used and the initialization procedures, we enable researchers to obtain the same results under the same experimental conditions.

To further enhance reproducibility, we provide all the necessary dependencies and software requirements needed to run our code. This includes specific libraries, frameworks, or tools and their corresponding versions. In addition, we aim to provide a Docker container that replicates the environment used for training and deploying Ainur, making it easier for others to set up the necessary environment.

Despite our efforts, there may be some limitations or potential issues when attempting to reproduce our work. It is essential for researchers to be aware of these challenges and take appropriate measures to address or mitigate them. For instance, differences in hardware or software configurations may lead to variations in results. In such cases, researchers should consult the provided documentation and seek guidance from the Ainur community to help resolve any discrepancies.

CHAPTER 6

RESULTS

In this chapter of the thesis, we dive deep into the performance analysis of our novel deep learning model, Ainur. Our primary aim here is to understand and illustrate the efficacy of Ainur in in the task of generating music from text, presenting a comprehensive overview of the findings of our research.

We begin by laying out the precise steps of the evaluation procedure that were followed to ensure the accuracy and relevance of our results. This includes a clear explanation of the metrics used for measuring the model's performance, the testing data and process, and the rationale behind our approach.

Once the evaluation methodology is clearly established, we shift our focus to an intrinsic evaluation of Ainur, diving into the model's own characteristics and analyzing the impact of each component on the final results. This involves *ablation studies*, where we progressively remove or alter features to assess their contribution to the overall performance.

As we move further into the chapter, we broaden our lens to consider Ainur in comparison to other state-of-the-art models in the field of text-to-music generation. This comparative analysis provides a benchmarking of Ainur against other models, facilitating a clearer understanding of its strengths and potential areas of improvement.

Finally, we wrap up the chapter with a discussion and commentary on the results obtained. This includes an analysis of how different model parameters influence the output, and the

implications of these findings for future work in this area. Our goal is to provide not only raw data but also meaningful interpretations, connecting our results back to our initial research objectives, and paving the way for further advancements in this domain.

6.1 Evaluation Procedure

The evaluation is twofold: an *intrinsic evaluation* aimed at assessing Ainur's capabilities and the influence of different parameters on the generated results, and a *comparative evaluation*, where Ainur is benchmarked against other state-of-the-art diffusion models.

In both types of evaluation, the Fréchet Audio Distance (FAD) is used as a primary metric for assessing the quality of the music generated. For a more nuanced understanding, FAD is computed with several different audio embedding models, each providing unique perspectives. Specifically, VGGish and YAMNet, trained on AudioSet, facilitate an evaluation of the audio quality, while Trill, trained on speech data, offers insights into the quality of generated vocals.

Our testing dataset comprises 456 songs, segmented into 22-second clips, each aligned with its respective synced-lyrics. This dataset, which Ainur has not been previously trained on, contains raw music data enhanced with metadata such as lyrics, artist, style, genre, and song sequence information.

The CLASP Cycle Consistency (C3) metric is employed where applicable, particularly when evaluating the coherence of generated audio in relation to the input lyrics information. However, as not all models are capable of lyrics-to-music generation, the use of this metric is primarily limited to instances where Ainur employs lyrics CLASP embeddings as an input conditioning mechanism.

TABLE IV: INTRINSIC QUALITY EVALUATION.

Model	CLASP	Steps	FAD VGGish ↓	FAD YAMNet ↓	FAD Trill ↓	C3 ↑
best	Lyrics	20	10.51	19.91	0.597	**0.29681**
best	Audio	20	10.53	21.10	0.625	
best	None	20	10.20	**19.58**	**0.588**	
best	Lyrics	50	8.38	20.70	0.659	0.29412
best	Audio	50	8.19	20.61	0.663	
best	None	50	8.40	20.86	0.636	
best	Lyrics	100	8.60	21.70	0.695	0.29412
best	Audio	100	8.29	21.70	0.697	
best	None	100	8.65	21.90	0.679	
last	Lyrics	20	7.89	24.63	0.817	0.29413
last	Audio	20	8.16	25.59	0.760	
last	None	20	7.86	27.01	0.791	
last	Lyrics	50	7.48	25.46	0.826	0.29413
last	Audio	50	7.69	25.90	0.791	
last	None	50	**7.37**	26.57	0.815	
last	Lyrics	100	7.58	26.45	0.848	0.29413
last	Audio	100	7.74	26.43	0.815	
last	None	100	7.45	25.42	0.829	

Finally, we place substantial emphasis on the inference time of each model. We regard this metric as critical for enabling music generation technology to be applicable in real-world scenarios, with a focus on ensuring democratized access even in the absence of substantial computational resources. All tests are conducted using a single Nvidia A100 GPU to ensure results that are relatable to consumer-grade equipment. The inference times for each tested model are reported to offer a clear comparison.

6.2 Intrinsic Evaluation

Our intrinsic evaluation of Ainur required us to delve into the interplay of various hyper-parameters and their subsequent effects on the model's performance and output quality. The distinctive versions of Ainur tested in this evaluation were formulated based on an array of combinations from three fundamental parameters:

1. **Training Epochs:** We tracked the performance of two distinct versions of Ainur, which we refer to as BEST and LAST. The BEST model, as the name suggests, achieved the lowest loss during training, peaking at a performance of 581 epochs with a loss score of 0.135. The LAST model, on the other hand, represents the model's state at the final epoch of training. This version ran through 732 epochs, ending with an epoch loss of 0.145. The comparison of results from these two models allows us to comprehend how variations in training loss can influence the quality of the generated samples.

2. **CLASP Conditioning:** We explored three different conditioning mechanisms, each manipulating how CLASP embeddings were integrated into Ainur's latent layer. We evaluated the effect of the CLASP embeddings resulting from both Lyrics and Audio input data. Furthermore, to gauge the overall impact of the CLASP conditioning mechanism, we also tested a version with no conditioning on CLASP embeddings, which we refer to as None conditioning strategy. For cases where the Lyrics input mechanism was employed, we further calculated the C3 metric on the generated outputs to assess the coherence between the input signal and the resulting music.

3. **Number of Diffusion Steps:** A key focus of our investigation was to understand the performance and inference time of Ainur in relation to the number of diffusion steps executed, both in latent diffusion and decoding diffusion. We discovered that Ainur is capable of generating high-quality music even with a small number of 20 diffusion steps. The music quality was subsequently evaluated with diffusion steps set at 20, 50, and 100. Each level of diffusion steps was also associated with a measure of inference time, providing valuable insights into the trade-off between quality and computational demands.

Table IV[1] captures the detailed results of our intrinsic evaluation study. All these experiments were conducted using an embedding scale factor of 7.0. This value determines the influence of text descriptor conditioning on the results, and it was found to yield the best performance. These results are organized based on three primary factors: the number of training epochs, the CLASP conditioning mechanism utilized, and the number of diffusion steps employed during both latent and decoding diffusion stages.

Across the board, the BEST model offers the best performance. Remarkably, even with just 20 diffusion steps, it yields the best values for FAD YAMNet, FAD Trill, and C3 metrics. These results indicate the BEST model's high caliber in terms of both audio and vocal quality.

---

[1]Evaluation of the samples generated by the Ainur model. The Fréchet Audio Distance metric is employed to assess the quality of the audio and vocals, while CLASP Cycle Consistency (C3) measures the coherence to the lyrics, applicable only to Ainur models utilizing CLASP lyrics conditioning. The models are differentiated based on the number of training epochs (marked as LAST and BEST), the CLASP conditioning mechanism (categorized as LYRICS, AUDIO, and NONE) and the number of diffusion steps (denoted by values 20, 50, and 100).

Notably, this model also shows the highest C3 similarity amongst all tested models, implying a superior coherence between the conditioning lyrics input and the generated music.

Following, the BEST model with 50 diffusion steps reveals very close metric values to the 20-step model, while showcasing a lower value for FAD VGGish. Both FAD VGGish and FAD YAMNet gauge the quality of the generated audio. Therefore, comparing and averaging these two results provides a more robust and consistent understanding of the output quality. As a result, we regard this model as the most stable and well-rounded.

The LAST model displayed better evaluations in terms of FAD VGGish, but unfortunately, it worsened significantly in other metrics. We postulate that the BEST model, with its lower training loss, yields better output quality – even with fewer training epochs compared to the LAST model. Moreover, we observe similar C3 scores across all models using LYRICS conditioning.

We observed that simply increasing the number of diffusion steps does not necessarily translate to improved metric values. This implies the model swiftly converges towards optimal results. We suggest that the CLASP conditioning mechanism could be the driving factor behind this phenomenon, demonstrating its influence even with a minimal number of diffusion steps.

Taking all these insights into account, we designate the **BEST model** with **50 diffusion steps** as **Ainur**: the model that best balances all the evaluation metrics.

From this point forward, the BEST model will be used as the reference for Ainur. We also want to draw attention to the time Ainur requires for inference when generating music samples.

TABLE V: INTRINSIC INFERENCE TIME BENCHMARK.

Diffusion Steps	CLASP Conditioning	Inference Time ↓ [mean ± std. dev.]
20	Lyrics	**5.75 s ± 33 ms**
20	Audio	5.92 s ± 36 ms
20	None	5.89 s ± 26 ms
50	Lyrics	**14.5 s ± 24 ms**
50	Audio	14.7 s ± 92 ms
50	None	14.7 s ± 103 ms
100	Lyrics	**28.8 s ± 59 ms**
100	Audio	29.3 s ± 162 ms
100	None	29.3 s ± 120 ms

In Table V[1], we have compiled the results of Ainur's performance benchmarks with an increasing number of diffusion steps and various conditioning mechanisms.

It is noticeable that the Lyrics conditioning mechanism consistently delivers a marginally faster inference time for generation. We suggest that this could be because the lyrics CLASP embedding acts as a guiding force, speeding up the diffusion process in the latent space. Additionally, as seen in Table IV, there is not a significant difference in output quality among the various conditioning mechanisms. Given these findings, we have chosen to make the Lyrics CLASP conditioning the default input conditioning mechanism for Ainur during inference.

[1]Inference time benchmark for the Ainur model. Results are presented according to varying numbers of diffusion steps (20, 50, and 100) and the type of CLASP conditioning mechanism employed (Lyrics, Audio, and None). Each inference time reported is the average of 7 test runs, with each run executing one loop. The reported time is presented as the mean plus/minus the standard deviation across these runs.

The inference times we observed tend to scale linearly with the number of diffusion steps. It is impressive to see that Ainur can generate high-quality music with as few as 20 diffusion steps. While 50 diffusion steps lead to more consistent results, Ainur also can be switched to *fast inference mode* that uses 20 diffusion steps, significantly reducing the inference time without compromising the quality of the generated samples to a significant degree.

6.3 Model Analysis

The lyrics CLASP conditioning mechanism of Ainur represents a unique feature, offering a way to direct music generation according to specific lyrics. The impact of this mechanism on both the quality of the generated music and the model's performance has been evaluated to fully understand its role within the Ainur model.

Comparing different CLASP conditioning mechanisms, it is evident that the lyrics conditioning offers a notable advantage in terms of inference time (see Table V). On average, models employing lyrics CLASP conditioning consistently exhibited faster generation times, regardless of the number of diffusion steps employed. This suggests that lyrics embeddings might provide stronger guidance during the diffusion process in the latent space, accelerating the generation of music samples.

From a quality perspective, the lyrics CLASP conditioning mechanism proves its worth when observing the evaluation metrics in the intrinsic evaluation study (see Table IV). While the differences between the three conditioning mechanisms (Lyrics, Audio, None) are not particularly stark, the Lyrics conditioning shows competitive results in all FAD metrics. More importantly, the C3 metric, employed to gauge the coherence between input lyrics and output

music, confirms the effectiveness of the lyrics CLASP conditioning. A statistical significance paired bootstrap test, which can be found in Appendix B, further supports these findings, confirming the impactful and significant contribution of the CLASP embedding conditioning technique.

Generally, the lyrics CLASP conditioning mechanism demonstrates its value, contributing to faster generation times and upholding high-quality standards in the produced music. The coherence between the input lyrics and the generated music, as measured by the C3 metric, underscores the success of the lyrics CLASP conditioning in ensuring relevance between the input and the output. Therefore, the lyrics CLASP conditioning mechanism emerges as a vital feature for the Ainur model.

6.4 Comparative Evaluation

Following the same modus operandi of the intrinsic evaluation, we decided to juxtapose Ainur's performance with that of other state-of-the-art music generation models, specifically those specializing in the task of converting text to music. We handpicked these models based on their demonstrated prowess in the field and their accessibility, as the models and pre-trained checkpoints are readily available online. The models featured in this comparative evaluation are:

- **AudioLDM** [60]: a deep generative model that leverages the power of language models for audio synthesis. AudioLDM treats audio synthesis akin to a language modeling task, functioning within a discrete representation space. It uses a progression of discrete audio units (tokens), from a rough to a refined level for the generation process. This method en-

ables AudioLDM to maintain high-quality and long-term consistency for periods spanning dozens of seconds. The model can create realistic audio from corpora consisting solely of audio, whether it's speech or piano music, without the need for any annotations.

- **Jukebox** [13]: a generative model that produces music in various genres in the raw audio domain. It uses a VQ-VAE to compress raw audio to discrete tokens, and then uses a transformer model to generate music autoregressively. The model is capable of generating high-quality music with long-term coherence, but it may display noticeable artifacts.
- **MusicLM**[1] [1]: a model has been designed to create high-quality music based on textual descriptions. This model interprets conditional music generation as a multi-level sequence-to-sequence modeling task, demonstrating the capability to generate music at 24 kHz that preserves its continuity over an extended period. Furthermore, it exhibits the versatility to be conditioned on both textual descriptions and a melody, allowing it to modify tunes, whether whistled or hummed, to align with the style conveyed by a text caption.
- **Riffusion** [63]: a generative model that uses Stable Diffusion to generate images of spectrograms from text prompts. These spectrograms can then be converted into audio clips. The model can generate infinite variations of a prompt by varying the seed, and supports image-to-image conditioning and interpolation in the latent space of the model for smooth transitions between different prompts.

---

[1]While the exact version of the MusicLM model is not publicly accessible, a comparable implementation used in our study is available on GitHub. This alternate version employs CLAP embeddings as opposed to the MuLAN embedding used in the original MusicLM. The open-source project can be found on the user zhvng's `open-musiclm` GitHub repository (github.com/zhvng/open-musiclm).

TABLE VI: COMPARATIVE EVALUATION.

Model	Sample Length [s]	Inference [s] ↓	FAD VGGish ↓	FAD YAMNet ↓	FAD Trill ↓
Ainur	22	14.5	8.38	20.70	0.659
Ainur (fast)	22	5.8	10.51	19.91	0.597
AudioLDM	22	**2.2**	15.50	784.17	0.521
Jukebox	1[2]	538.1	20.41	178.10	1.586
MusicLM	5	153.1	15.00	61.58	**0.471**
Riffusion	5	6.9	**5.24**	**15.96**	0.696

The results of our comparative evaluation are displayed in Table VI[1]. What stands out immediately is that AudioLDM tops the list in terms of speed, able to generate 22 seconds of music in a scant 2.2 seconds. Following closely behind are Riffusion and Ainur in terms of inference time, while MusicLM and Jukebox trail behind significantly, falling short of real-time generation capabilities. It is noteworthy that Jukebox takes a staggering 9 minutes to generate a single second of music, highlighting its substantial computational requirements. Yet, in its fast inference mode, Ainur secures the second spot for generation speed.

Despite its impressive speed, AudioLDM disappointingly underperforms in sample quality. With a staggering FAD YAMNet score of 784.17, it suggests poor audio quality, even though

[1]Comparative evaluation of samples generated by various text-to-music models. Metrics used include inference time in seconds and Fréchet Audio Distance (FAD) to assess the quality of generated music. Note that certain models, due to their computational constraints or design, were unable to match the sample length of the music generated by Ainur, resulting in shorter sample durations.

[2]Jukebox, the oldest model in our comparison, carries substantial computational demands and requires extended inference times. Despite its autoregressive functionality allowing it to generate songs up to 22 seconds long, the sheer amount of time it takes to generate even a single second of music is substantial. On a typical consumer-grade GPU, we were only able to generate one second of music. Consequently, the results we present may not fully encapsulate the true capabilities of Jukebox.

its FAD VGGish score is reasonably low. This vividly illustrates the pitfalls of relying solely on one metric to evaluate the quality of generated music, and highlights the inherent limitations of using only objective metrics. These metrics, though useful, may not align with human perception or capture all facets of generated music.

Riffusion stands out with the best FAD VGGish and FAD YAMNet scores, while MusicLM excels in producing quality vocals, resulting in the lowest FAD Trill score of 0.471. Ainur exhibits strong performance, producing results on par with these leading models, but with the added advantage of having a significantly shorter inference time and being capable of generating longer samples in a fraction of the time. This combination of attributes establishes Ainur as a competitive player in the text-to-music generation domain, delivering near real-time generation with quality comparable to the industry's best. The convergence of these factors makes Ainur the most balanced choice for music generation, optimally blending speed and quality.

6.5 Summary of the Results

In the course of our comprehensive assessment of Ainur, we undertook an intrinsic evaluation, scrutinizing the impact of various parameters and settings on the model's performance, followed by a comparative evaluation against some of the top text-to-music generation models. Our findings yield meaningful insights into the performance of Ainur and how it measures up against the state-of-the-art in music generation.

From our intrinsic evaluation, it was evident that the BEST Ainur model, defined as the one which achieved the lowest training loss, offered the most balanced performance across various metrics. This model was able to generate high-quality music with impressive audio and vocal

qualities, even with a limited number of diffusion steps. The inference time was also very reasonable, which is vital for practical, real-world application.

Notably, the variation in the number of diffusion steps used during latent and decoding stages had a surprisingly marginal impact on the quality of the generated music. This suggests that Ainur converges quickly towards optimal results, potentially due to the influence of the CLASP conditioning mechanism, which assists in the generation even with a limited number of diffusion steps.

In the comparative evaluation, Ainur emerged as a compelling model for music generation, demonstrating performance on par with leading models such as Riffusion and MusicLM, while outpacing these models in terms of inference time. Particularly noteworthy was Ainur's superiority to Jukebox in both quality of results and computational efficiency.

The findings also underlined the criticality of leveraging multiple evaluation metrics. This was highlighted by the results from AudioLDM, which despite demonstrating incredible speed, failed in generating high-quality samples, as indicated by its high FAD YAMNet score.

These results position Ainur as a significant contender in the realm of music generation. It stands out as a model that does not just match its competitors in the quality of generated music, but also excels in delivering this performance within a practical timeframe, which is a decisive factor for real-world deployment.

The implications of these findings are far-reaching. They underscore the importance of achieving an optimal balance between quality, computational efficiency, and generation speed in music generation models. They also provide a roadmap for future work, underscoring the

potential benefits of leveraging effective conditioning mechanisms like CLASP and optimizing the number of diffusion steps to achieve high-quality music generation without undue computational demands. For further insight and quantitative visualizations of these results, please refer to Appendix A.

CHAPTER 7

FUTURE WORK

This chapter revisits the key findings from the previous chapters, providing a brief synopsis while also outlining the areas of potential improvement and exploration for the Ainur model. The Ainur model has proven to be a significant advancement in the field of music generation AI, and this chapter will serve as a roadmap for future research and development, pushing the boundaries of what has been achieved so far.

Ainur has convincingly demonstrated its knack for crafting high-quality music, spanning a broad array of genres and styles. A key to its success lies in its unique hierarchical structure and the incorporation of a latent diffusion prior. These innovative design elements empower Ainur to model high-dimensional audio data and master the art of generating music within a compact, low-dimensional space. This process culminates in creating high-fidelity stereo music, courtesy of the diffusion decoding mechanism that restores the audio to its original high-dimensional form.

Another notable feature of Ainur is its adept use of CLASP embeddings. These embeddings have showcased their potential in encoding multimodal information, providing a sturdy launching pad for the music generation process. Coupled with the guiding influence of the text descriptor's cross-attention embeddings, Ainur has proven its adaptability to various real-world applications and a range of downstream tasks. This conditionality ability to steer music gener-

ation based on the user's multimodal input is a crucial aspect of Ainur. It takes Ainur beyond being just an unconditional music generator, making it a valuable tool in the hands of its users.

While Ainur represents a commendable step towards generating music that's virtually indistinguishable from compositions crafted by human hands, it's clear that there is room for improvement. In pursuing this higher standard, we identify several areas for potential enhancement that we believe will further unlock Ainur's capabilities in the future.

1. **Improvements in model architecture**:

    - Improvements in the Ainur model's architecture are potential avenues for future exploration. One aspect that could benefit from further enhancement is the CLASP embeddings. In this project, we utilized pre-trained CLASP embeddings, but fine-tuning them more specifically on lyrics-audio tuples could potentially enhance the model's performance. This would result in a closer relationship between the lyrics and the corresponding music, which is crucial for creating coherent musical pieces.
    - Moreover, integrating different types of training optimization procedures could lead to better results. For instance, exploring various loss functions for the diffusion model, such as perceptual loss, may be beneficial [92]. This loss function could be better suited for the characteristics of mel-spectrogram representations, thus potentially improving the quality of generated music.
    - Further, the diffusion model employed in Ainur could be replaced or distilled with novel consistency models [97]. These models have the capability to perform the diffusion process in a significantly fewer number of steps, thereby improving the speed

of the model without compromising on the quality. Additionally, the introduction of different forms of noise during the diffusion process could be considered. For instance, replacing Gaussian noise with *pink* noise [23], which exhibits a power intensity that is perceptually more relevant and correlates with the mel-spectrogram representation, might yield better results.

- Additionally, an exploration of varied input conditioning strategies could potentially enhance multimodal feature learning capabilities. The impact of different methods for integrating multimodal information, such as using solely cross-attention in a unified embedding space or latent injection alone, warrants further investigation.
- Finally, there is potential to leverage the hierarchical design of Ainur further. By incorporating additional levels, we could potentially further enhance the quality of the music generated. This expanded structure might also provide an opportunity to exploit various sequential stages of latent diffusion, akin to the hierarchical approach used in Hierarchical VAE [105]. This could open up new avenues for refining the performance of our model.

2. **Expansion of training data**:

- The capacity of the Ainur model can be further improved by utilizing a larger and more diverse dataset. The current dataset, while sufficient for our study, is relatively small. With a more comprehensive dataset, the model's understanding of various musical genres, languages, and cultural music forms can be enhanced.

- Moreover, there is scope for extending the CLASP framework to languages other than English, which would significantly broaden the model's applicability and reach.

3. **Enhancements in evaluation metrics**:

   - We are keen on conducting a comprehensive subjective evaluation of Ainur's performance. While objective evaluations provide critical insights, they have their inherent limitations, particularly in the field of audio generation. It is standard practice in our domain to involve human participants in subjective evaluations, asking them to score the perceived quality of the generated audio samples. This Mean Opinion Score (MOS) [85], gathered via a Likert scale, offers valuable feedback from the end user's perspective. Therefore, we're mapping out plans to conduct such human-centric evaluations to refine further and assess the capabilities of Ainur.
   - We also see an opportunity for introducing additional metrics to evaluate the model's performance. The current evaluation methods have their limitations, and integrating multi-modal evaluation techniques could present a more holistic view of the model's capabilities. This, in turn, would inform better strategies for model improvement and refinement.

4. **Scalability and efficiency**:

   - The scalability and efficiency of the Ainur model present areas for further improvement. By optimizing the model for distributed environments, the training time could be reduced substantially, thereby increasing the model's efficiency. Techniques for

improving the scalability of the model could be developed and explored in future research.

- In our pursuit of heightened efficiency, we're eager to delve into the potentials of consistency models [97] as a substitute for our current diffusion model. By their nature, consistency models could offer significant reductions in both training and inference time without compromising the quality of Ainur's musical output. Additionally, they present promising opportunities for better memory efficiency, allowing for larger batch sizes during training and potentially leading to improved model performance. All in all, this exploration constitutes a promising avenue for refining the architectural design of Ainur and optimizing its algorithmic efficiency.
- On a similar note, we aim to increase the model's efficiency in terms of energy consumption. With the growing concerns around the environmental impact of machine learning models, it's imperative to develop more energy-efficient models. This could involve exploring energy-efficient hardware or implementing software solutions to optimize energy usage.
- Finally, we plan to explore hardware improvements that could further accelerate the training process. This could include utilizing more powerful GPUs or exploring other emerging hardware technologies.

5. **Ethical and social considerations**:

- The deployment of music generation AI like Ainur also brings forth several ethical and social considerations. Issues related to copyright, authorship, and cultural appropriation are some of the potential challenges that must be addressed. Future work in this area should not only be focused on improving the technical aspects of the model but also on developing guidelines to navigate these ethical and social challenges.

CHAPTER 8

CONCLUSION

In this concluding chapter, we encapsulate the journey of our research, providing a succinct overview of the problem we set out to solve, the means we employed to tackle it, and the outcomes we achieved.

8.1 Summary of Research

Our investigation was prompted by the need to improve the quality of vocals generated in the field of deep music generation. Despite the significant progress made in the realm of image and text generation, music generation remains relatively underexplored and riddled with challenges, particularly those related to data availability, high-dimensionality of audio data, and the computational resources required. The main questions that shaped our exploration were:

1. Which deep learning techniques can enhance the quality of generated vocals?
2. How can multimodal input conditioning strategies be used to create vocals congruent with the theme and mood of the generated music?
3. Can transfer learning or pre-trained models ameliorate the quality of generated vocals, and what would be the optimal approach for their application?

4. How do different multimodal input conditioning strategies, like amalgamating text, image, and symbolic musical representations, influence the quality and diversity of the music produced?

Addressing these research questions, we conducted an extensive empirical study, employing various deep learning techniques and multimodal input conditioning strategies.

Our primary methodological approach was centered around developing Ainur, a deep learning model that leverages the power of a hierarchical structure and a latent diffusion prior. This was supplemented with the utilization of Contrastive Lyrics-Audio Spectrogram Pre-training (CLASP) embeddings, which enabled us to fuse multimodal information effectively.

The results of our research have been promising. Ainur demonstrated its ability to generate high-quality raw music spanning various genres and styles. This was made possible by the model's capacity to handle high-dimensional audio data, a result of the hierarchical architecture it employs, and the latent diffusion prior, which facilitates music generation in a compressed, low-dimensional space. The incorporation of CLASP embeddings was instrumental in enabling Ainur to encode multimodal information, initiating the music generation process.

While Ainur does not completely resolve the challenges of the lyrics-to-music generation task, given the vocals remain somewhat unintelligible, its results nonetheless suggest that the multimodal integration of lyrics-audio embeddings is a promising approach for enhancing the quality and coherence of the generated music.

In light of these findings, we have made strides in addressing our research questions. We have successfully employed deep learning techniques and multimodal input conditioning strategies to

improve the quality of vocals in music generation. We have also demonstrated the effectiveness of transfer learning through the use of pre-trained models like CLASP embeddings. Finally, our study has provided insights into the impact of multimodal input conditioning strategies on the quality and diversity of generated music.

As a final note, it is important to mention that all the products of this research, including pre-trained models and user interfaces, are made publicly available on a GitHub repository. This initiative fosters openness and collaboration in the field of deep music generation, empowering other researchers, practitioners, and users to build upon our work.

8.2 Discussion of the Results

As we move towards the conclusion of this research project, it is crucial to reflect upon and synthesize the wealth of insights we have gleaned from our exploration and evaluation of Ainur, a state-of-the-art model for text-to-music generation. It is important to tie these findings back to our initial research questions, understand their broader implications, and trace how they contribute to the field of deep music generation.

One of the primary areas of focus was the application of deep learning techniques to enhance the quality of vocals. The results from our intrinsic evaluation provide a clear affirmation for this approach. The BEST model distinguished itself through a balanced performance, producing vocals that were both high in quality and coherent with the input lyrics. These findings speak volumes to the effectiveness of deep learning techniques, such as the CLASP conditioning mechanism, in refining the vocal component of music generation, addressing our first research question.

The second research question pertained to the role of multimodal input conditioning strategies in shaping the theme and mood of the generated music. Here, the Lyrics CLASP conditioning emerged as a significant player. Our evaluation demonstrated that embedding the strength of text descriptors had a profound impact on the generation process, guiding the diffusion in the latent space and accelerating the process to produce more coherent outputs.

Addressing our third research question, we examined the impact of transfer learning and the application of pre-trained models on the quality of generated vocals. In our exploration, Ainur serves as an exemplary testament to the effectiveness of this approach. The model is founded on a series of pre-trained components: the CLASP is based on pre-trained CLIP, and the base DAE of Ainur is also pre-trained. Furthermore, the text encoder is a frozen pre-trained T5 transformer. By extensively and successfully integrating pre-trained models and transfer learning, Ainur not only accelerates its training process but also significantly reduces its environmental impact, underscoring the potent benefits of these methods in the realm of deep music generation.

In terms of our fourth research question - the effect of diverse multimodal input conditioning strategies on the quality and diversity of generated music - our evaluation threw light on several significant insights. The intrinsic evaluation demonstrated that even with varying conditioning mechanisms, the quality of results did not fluctuate dramatically, suggesting that the model could maintain a steady performance across different multimodal inputs. However, it is essential to note that the Lyrics conditioning did stand out, yielding slightly better inference time.

The evaluation of Ainur presented a profound understanding of the model's capabilities and demonstrated its promising potential in the field of text-to-music generation. As outlined in the intrinsic evaluation, Ainur showcases both high-quality generation performance and a distinct capability to balance various factors such as quality, coherence, and inference time.

A remarkable highlight from the intrinsic evaluation was the impressive performance of the BEST model. By reaching its peak at 581 training epochs, this model achieved a loss of 0.135 and exhibited high performance even with just 20 diffusion steps. What sets this model apart is its ability to provide high audio quality and vocal fidelity, whilst ensuring a strong correlation between input lyrics and the music generated. This balance is a testament to the efficacy of the model's architecture and the CLASP conditioning mechanism.

The varying number of diffusion steps tested revealed that the model performs optimally at 50 steps. Despite increasing the number of steps further to 100, there were no significant improvements to the quality of the music generated. This suggests that the model quickly converges to producing high-quality results and indicates that the CLASP conditioning mechanism can influence the generation even with a relatively small number of diffusion steps.

The comparative evaluation further consolidated Ainur's standing in the realm of music generation. It displayed strong results, competing with state-of-the-art models while outperforming them in terms of inference time. Remarkably, Ainur managed to generate high-quality music in near-real-time, a feat unmatched by many of the tested models.

However, the evaluation also illuminated areas for potential improvement. For instance, the FAD VGGish and FAD YAMNet metrics for Ainur were not the lowest among the models tested. Future work could, therefore, aim to enhance the model's performance in these areas.

These results hold significant implications for Ainur and the broader field of music generation. Firstly, Ainur's ability to rapidly generate high-quality music, while ensuring coherence with the input lyrics, makes it a powerful tool for various applications, from aiding composers to providing a unique user-driven music experience. Secondly, the evaluations have shown that Ainur's novel architecture and the use of the CLASP conditioning mechanism could serve as a blueprint for future models in the domain. Finally, the findings also highlight the importance of a balanced evaluation framework, considering not only objective quality metrics but also coherence and inference time.

The implications of these findings extend far beyond the realm of Ainur. They pave the way for a better understanding of how deep learning techniques and multimodal input conditioning strategies can be harnessed to improve music generation. This research also underlines the importance of a balanced model, one that can negotiate the delicate act of producing high-quality output while ensuring efficient generation times.

8.3 Key Contributions

This research has made notable strides in the arena of deep music generation, introducing innovative methodologies and expanding the horizons of the field. Each contribution, outlined below, marks a step forward in the quest for high-quality music generation and provides a robust foundation for future investigations.

- **CLASP model and embeddings:** The cornerstone of our work is the adaptation and application of the Contrastive LyricsAudio Spectrogram Pre-training (CLASP) model and embeddings for multimodal conditioning. CLASP, a specialized adaptation of CLIP for the music domain, efficiently processes lyrics as natural language and audio spectrograms to create a comprehensive multimodal representation of the data. This novel approach breaks new ground in the field, allowing for the synthesis of diverse data modalities to enhance the music generation process.

- **Hierarchical diffusion model:** We have introduced a novel extension of a diffusion autoencoder architecture into a hierarchical diffusion model. This innovative model uses conditional information to guide the generation process in the latent space and then sequentially cascades the reconstructed information to yield high-dimensional, high-quality output. This pioneering approach enriches the quality of the generated music, bringing an added layer of sophistication to the music generation process.

- **Single GPU inference and training**: Ainur's implementation signifies a major step towards making advanced music generation techniques more accessible. Specifically designed to be trained and utilized on single, consumer-grade GPUs, our model mitigates the necessity for colossal computational resources often associated with such tasks. This design choice not only makes the model practically deployable, but also ensures its accessibility to a wider research community. The goal is to foster further research and improvements in this area, making state-of-the-art music generation models not just a theoretical possibility, but a practical reality. Despite these constraints, Ainur still deliv-

ers impressively low inference times and high-quality outputs, reinforcing its value as a robust and efficient tool for music generation.

- **Lyrics-to-music generation:** Our research has birthed Ainur, one of the few existing deep learning models that specifically target improving the quality of vocals in the field of deep music generation. By focusing on the challenging task of lyrics-to-music generation, Ainur is a significant contribution to the field. We anticipate that our work with Ainur will inspire further exploration in this direction, catalyzing advancements in the field.
- **C3 and FAD evaluation metrics:** Our research introduces the C3 evaluation metric and the FAD evaluation with the YAMNet model serving as an embedding model reference. These objective metrics correlate with human perception, providing a solid base for evaluating deep music generation models. Introducing these evaluation methods establishes a common platform for comparison and assessment, promoting uniformity and accuracy in the evaluation process.

In summary, our research paves the way for further advancements in the field of deep music generation, contributing innovative approaches, practical tools, and standardized evaluation methods. We hope that these contributions will stimulate ongoing innovation and exploration in this exciting domain.

8.4 Limitations

Despite the accomplishments and breakthroughs achieved by this research, it is critical to openly acknowledge and address its limitations and the ethical considerations inherent to deep music generation and the use of the Ainur model.

Firstly, Ainur's performance, like any deep learning model, depends on the quality and diversity of the data it is trained on. While we have endeavored to provide as varied and extensive a dataset as possible, inherent biases in the data could potentially limit the diversity of the music Ainur can generate. Furthermore, the quality of the generated audio is limited by the resolution of the audio data in the training set.

Secondly, the scalability and efficiency of Ainur, though considerably improved, still pose a challenge. The computational requirements for training such a model are substantial, necessitating high-performance hardware, which might not be readily available to all researchers. Additionally, Ainur requires a considerable amount of time to generate music, which might limit its use in real-time applications.

From an ethical perspective, the possibility of generating indistinguishable music from human-composed music raises important questions. On the one hand, Ainur can be a valuable tool for music creation, enabling artists to explore new musical ideas or aiding in music education. However, on the other hand, it is crucial to ensure that such technology is not used to infringe on the rights of artists by plagiarizing their work or devaluing their creative efforts by flooding the market with AI-generated music. The responsibility to use Ainur ethically rests with the end users, and proper guidelines and safeguards must be in place to prevent misuse.

Lastly, the open-source nature of Ainur, while promoting collaboration and transparency in research, also opens the door for potential misuse. In the wrong hands, this technology could be used unethically or maliciously. Therefore, it is imperative that the use of Ainur is governed by a comprehensive set of ethical guidelines.

In summary, while this research has made significant strides in the field of deep music generation, there is still a clear path ahead filled with challenges and opportunities. By recognizing and addressing these limitations and ethical considerations, we can ensure that our journey forward is not only scientifically rigorous but also ethically sound.

APPENDICES

Appendix A

AUXILIARY RESULTS

In light of the results discussed in Table IV, we supplement these findings with additional plots to better illustrate the intrinsic comparison between different iterations of the Ainur model. Our evaluation of the quality of generated music and vocals was grounded in the FAD metric, using different reference embedding models, namely VGGish, Trill, and YAMNet. To streamline the understanding of these results, we introduce a metric called the normalized FAD (NFAD), which offers a straightforward and insightful gauge of the overall quality of the produced music.

NFAD is a normalized average of the ensemble of reference embedding models used. It yields a value between 0 and 1, with a lower NFAD indicating superior overall quality of the generated music. Given m configurations $\mathbf{cj}$ (defined by model type, conditioning mechanism, number of diffusion steps) and n reference embedding models used to compute the $\mathsf{FAD}_{\mathsf{E}_i}$ (e.g., E_0 = VGGish), we compute the NFAD for a given music sample $\boldsymbol{x}$ as follows:

$$\mathsf{NFAD}^{\mathbf{c}_j} = \frac{1}{n}\sum_{i=0}^{n}\left[\frac{\mathsf{FAD}^{\mathbf{c}_j}_{\mathsf{E}_i}(\boldsymbol{x})}{\max\limits_{\mathbf{c}^*_j \in \mathbf{c}}(\mathsf{FAD}^{\mathbf{c}^*_j}_{\mathsf{E}_i}(\boldsymbol{x}))}\right] \tag{A.1}$$

In this equation, $\mathsf{NFAD}^{\mathbf{c}_j}$ represents the value of the normalized FAD with the configuration of parameters $\mathbf{cj}$, and $\mathsf{FAD}^{\mathbf{c}_j}_{\mathsf{E}_i}$ represents the value of the FAD metric using the reference embedding model E_i and the configuration of parameters $\mathbf{c}_j$. To put it concisely, NFAD is a normalized and

Appendix A (continued)

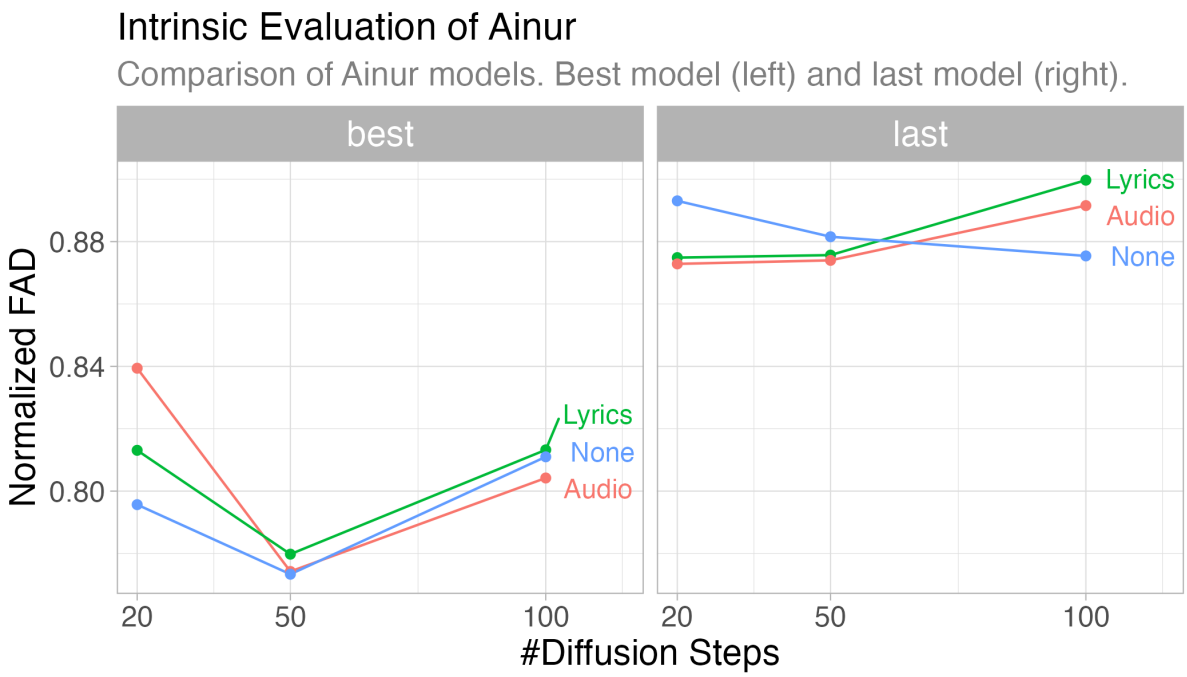

Figure 17: Intrinsic quality evaluation of Ainur.

averaged version of the FAD across different embedding models and configurations, offering a user-friendly, comprehensive indicator of overall music quality.

Figure 17 and Figure 18 present the intrinsic evaluation measures NFAD and C3, respectively. In Figure 17, we compare the BEST (model that achieved the lowest loss during training) and the LAST (model that underwent the maximum number of training epochs) in terms of the computed NFAD, as described in Equation A.1. The models are grouped according to the type of conditioning mechanism employed (lyrics, audio, or no CLASP conditioning), and the results are represented for an escalating number of diffusion steps (20, 50, and 100). The figure

Appendix A (continued)

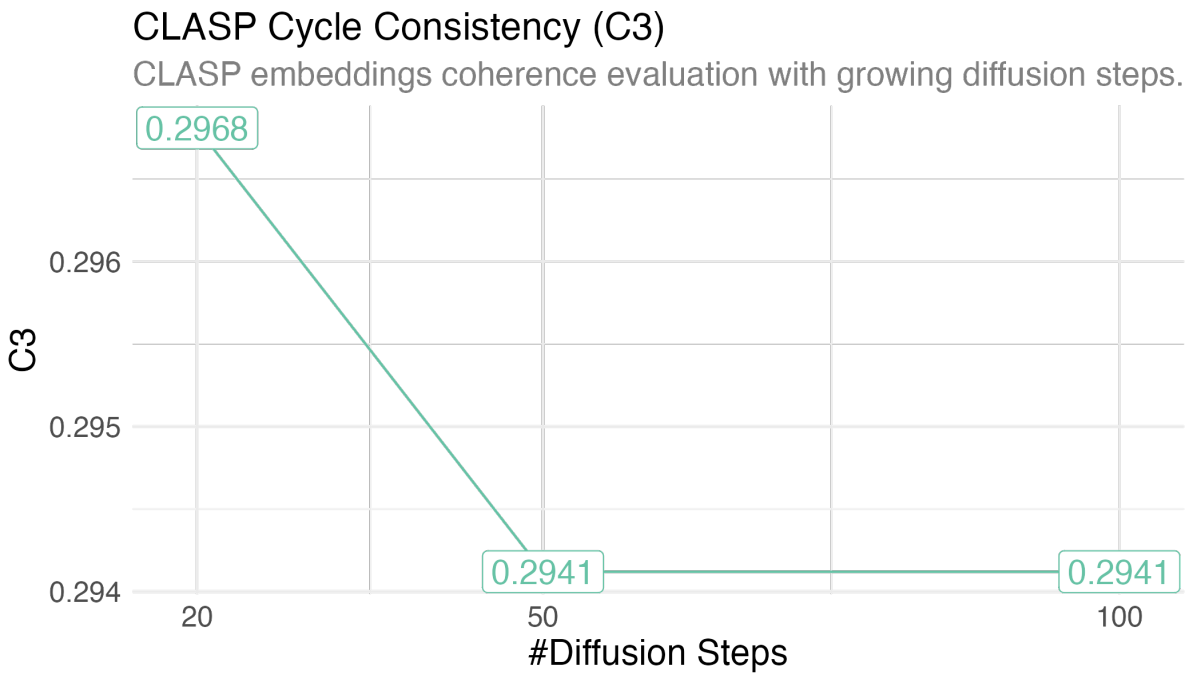

Figure 18: Intrinsic coherence evaluation of Ainur.

reveals that the BEST model consistently outperforms the LAST model in each configuration, as evidenced by a lower NFAD. Of all configurations, the BEST model with 50 diffusion steps demonstrates the highest performance. Intriguingly, an increase in the number of diffusion steps does not correlate directly with an enhancement in generated music quality.

Figure 18 concentrates on the model's performance concerning the coherence of the generated music, as measured by the C3 metric. The model scrutinized here employs the lyrics CLASP conditioning mechanism. The C3 value is computed between the text embeddings of conditioning lyrics and the audio embeddings of the music generated using this conditioning

Appendix A (continued)

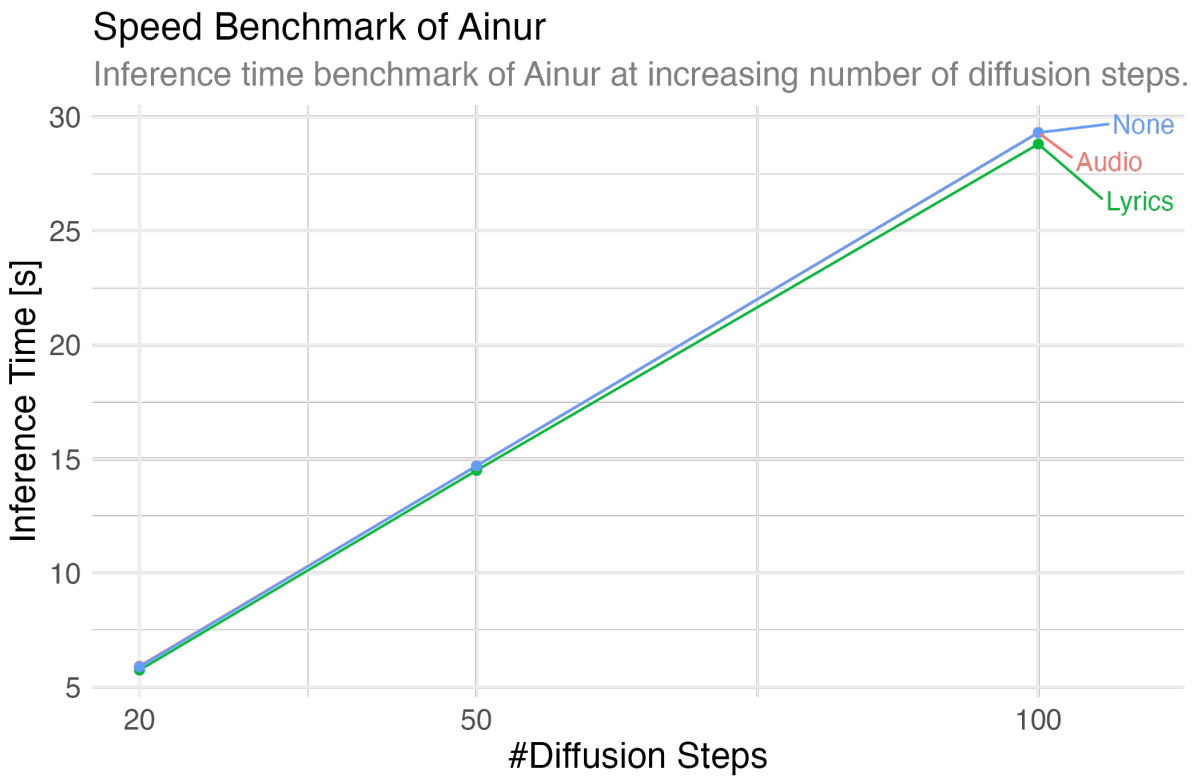

Figure 19: Intrinsic inference time benchmark of Ainur.

mechanism. This visualization demonstrates that the highest similarity is achieved with just 20 diffusion steps, contradicting the assumption that increasing the number of diffusion steps leads to improved coherence in the results.

In the subsequent analysis, we focused on gauging the inference speed needed for various Ainur models to generate a single music sample. Using the data provided in Table V, Figure 19 and Figure 20 provide an overview and a more detailed account, respectively, of the intrinsic inference time benchmark. Figure 19 displays the time in seconds required for Ainur to generate

Appendix A (continued)

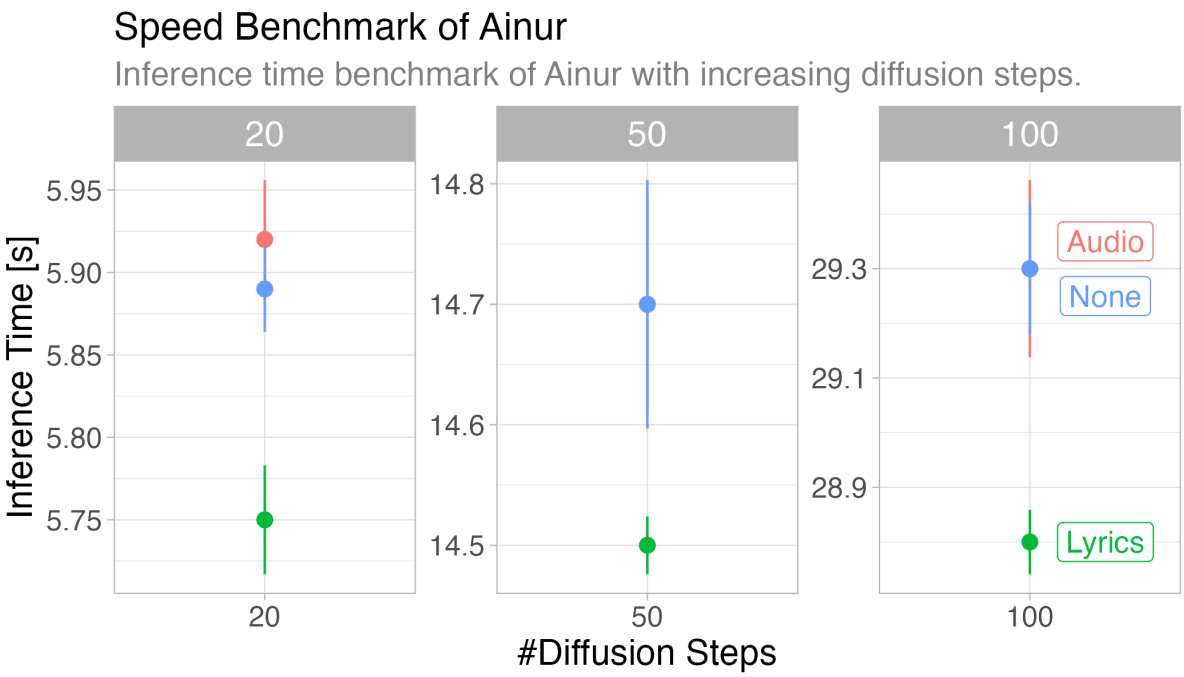

Figure 20: Detailed intrinsic inference time benchmark of Ainur.

music. The results are grouped by different conditioning mechanisms and an increasing number of diffusion steps. As can be anticipated, the time required for generating a music sample increases linearly with the number of diffusion steps.

For a more granular analysis, Figure 20 presents the inference time using different time scales for each group of diffusion steps. The CLASP lyrics conditioning mechanism consistently results in a shorter inference time, suggesting that this approach facilitates faster model convergence, and thus, more time-efficient sample production.

Appendix A (continued)

Based on the aforementioned results, we have distinguished two optimal configurations of the Ainur model. These are:

- **Ainur**:
  - Utilizes the BEST model variant
  - Incorporates 50 diffusion steps
  - Distinguished for generating samples of superior quality
- **Ainur (FAST)**:
  - Employs the BEST model variant
  - Incorporates 20 diffusion steps
  - Noted for its exceptional efficiency in terms of inference time

Next, we undertook a comparative analysis of the inference time and quality of our proposed Ainur models against several contemporary leaders in the field of text-to-music generation. This analysis is predicated on the data highlighted in Table VI, from which we formulated the plots exhibited in Figure 21 and Figure 22.

Interestingly, Ainur's efficiency becomes particularly apparent when considering the generation of music samples of 22 seconds in duration. Many leading models in the field are challenged in generating music of this length in a timely manner. For instance, OpenAI's Jukebox requires approximately 9 minutes to produce just a single second of music. Similarly, Google's MusicLM demands in excess of 3 minutes to generate a mere 5 seconds of audio. As illustrated in

Appendix A (continued)

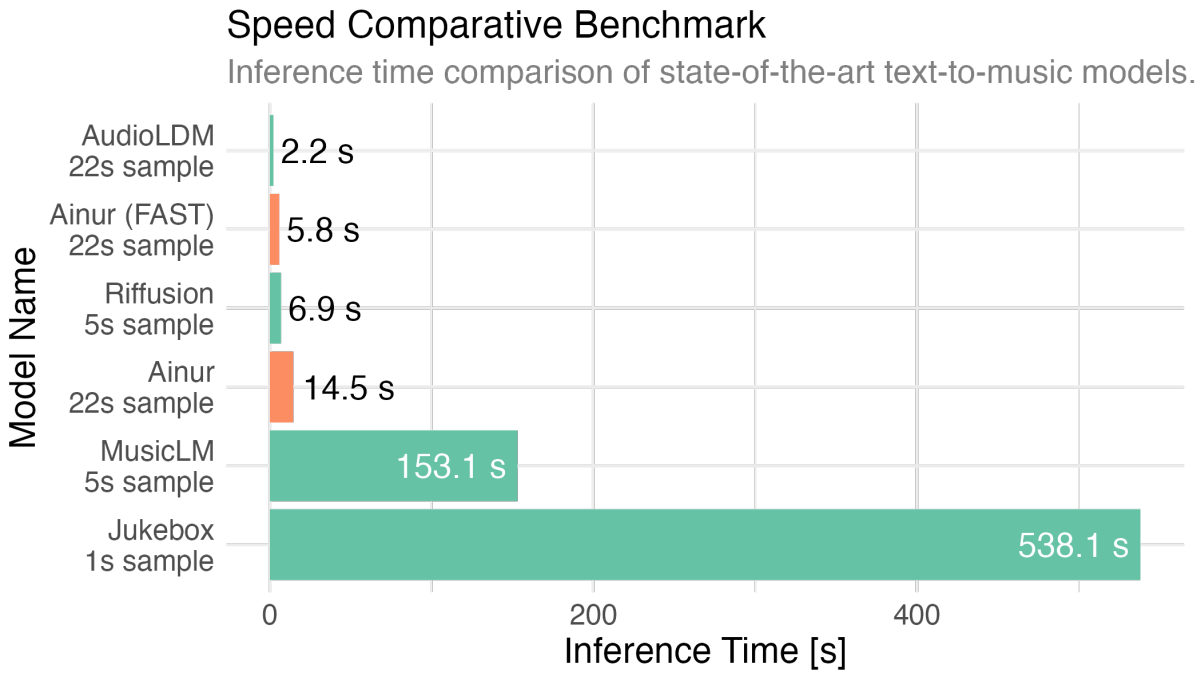

Figure 21: Inference time comparative benchmark.

Figure 21, when it comes to speed, Ainur (FAST) is only outpaced by AudioLDM. Although Riffusion is faster than Ainur, its capabilities are limited to generating songs of only 5 seconds in length.

When evaluating audio quality, Figure 22 demonstrates that Riffusion is currently unsurpassed. However, both the Ainur and Ainur (FAST) models outperform all other contenders. This remarkable balance of speed and quality positions Ainur as a leading choice in the current landscape of state-of-the-art models for text-to-music generation.

Appendix A (continued)

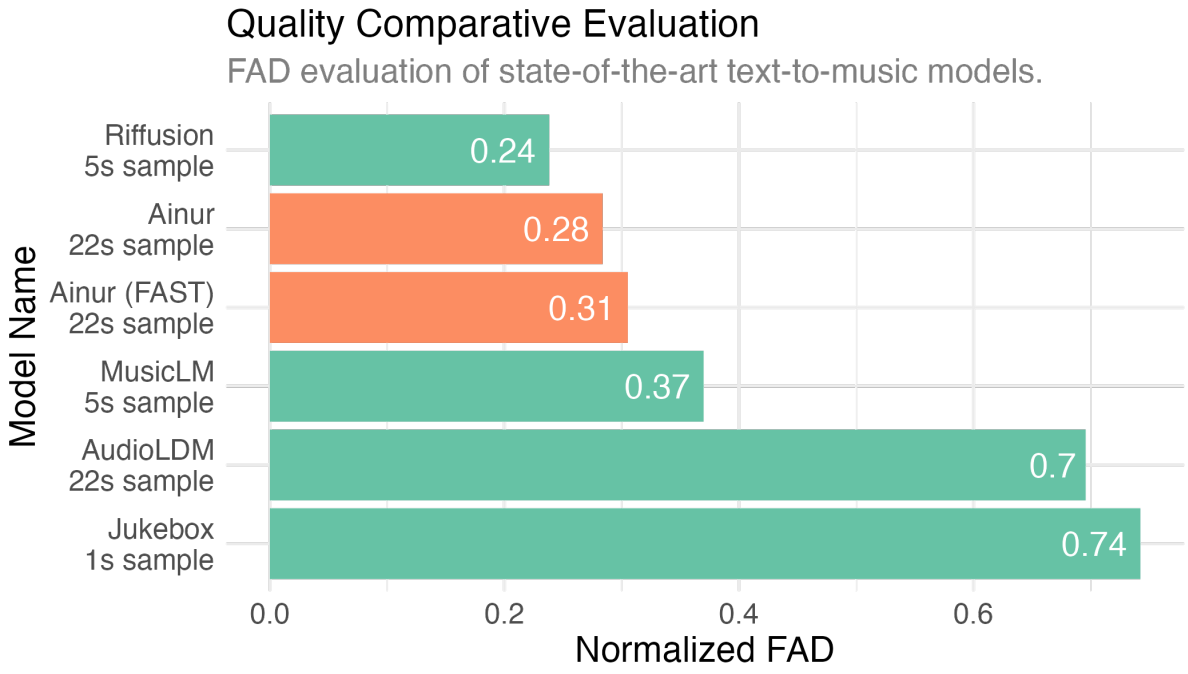

Figure 22: Quality comparative evaluation.

Appendix B

CLASP STATISTICAL SIGNIFICANCE

The paired bootstrap test [45] is a robust statistical procedure designed to measure the significance of differences between two models. The test is based on the concept of resampling and provides an empirical estimate of the sampling distribution of a statistic, notably when the theoretical distribution is unknown or hard to derive.

In a paired bootstrap test, the test statistic of interest is calculated on a large number of bootstrap samples, which are created by randomly sampling pairs of observations (with replacement) from the original dataset. This process, performed numerous times, allows us to form an approximation of the sampling distribution of the statistic and calculate confidence intervals or p-values.

The basic mechanism of the paired bootstrap test can be expressed as follows:

1. We draw a random sample of paired observations (x_i, y_i) with replacement from our original data of size n.
2. We calculate the statistic (e.g., mean, median) of interest on this bootstrap sample and record it.
3. We repeat this process B times to form a bootstrap distribution of our statistic of interest.
4. From this distribution, we calculate the p-value or confidence interval.

Appendix B (continued)

Mathematically, the p-value is defined as the proportion of times the observed test statistic in the bootstrap sample exceeds the test statistic calculated from the original sample:

$$\text{p-value} = \frac{1}{B}\sum_{i=1}^{B} \mathbb{1}\big(\delta(x^{(i)}) - \delta(x) \geq 0\big) \tag{B.1}$$

where $\delta(x)$ represents the difference in the observed test statistic of the two models and $\delta(x^{(i)})$ is the difference in the bootstrap test statistic, $\mathbb{1}(\cdot)$ is the indicator function, and B is the number of bootstrap samples.

In a paired bootstrap test, the test statistic of interest is calculated on a large number of bootstrap samples, which are created by randomly sampling pairs of observations (with replacement) from the original dataset. This process, performed numerous times, allows us to form an approximation of the sampling distribution of the statistic and calculate confidence intervals or p-values.

In the case of a paired bootstrap test, the p-value is computed differently than the conventional procedure. The paired bootstrap p-value is calculated using the following formula:

$$\text{p-value} = \frac{1}{B}\sum_{i=1}^{B} \mathbb{1}\big(\delta(x^{(i)}) \geq 2\delta(x)\big) \tag{B.2}$$

In this study, we focus on the hypothesis test defined as follows:

H_0: Model A (Ainur model with lyrics CLASP conditioning) is not better than Model B (Ainur model with no CLASP conditioning)

Appendix B (continued)

To evaluate the quality of the generated samples, we have used the FAD Trill metric, and we have computed the differences (or deltas) of the FAD Trill between the two models over the evaluation dataset.

Following the paired bootstrap methodology, we performed the bootstrap process $B = 100$ times to obtain a p-value. For this hypothesis test, we considered a significance level of 0.05, corresponding to 95% confidence. The resulting p-value from our bootstrap test was **0.0314**.

Given that this p-value is less than our significance level of 0.05, we have sufficient evidence to reject the null hypothesis. Therefore, we can confidently conclude that Model A (Ainur model with lyrics CLASP conditioning) is indeed superior to Model B (Ainur model with no CLASP conditioning).

CITED LITERATURE

1. Agostinelli, A., Denk, T. I., Borsos, Z., Engel, J., Verzetti, M., Caillon, A., Huang, Q., Jansen, A., Roberts, A., Tagliasacchi, M., Sharifi, M., Zeghidour, N., and Frank, C.: MusicLM: Generating Music From Text, January 2023. arXiv:2301.11325 [cs, eess].

2. Andreev, P., Alanov, A., Ivanov, O., and Vetrov, D.: HiFi++: a Unified Framework for Bandwidth Extension and Speech Enhancement, September 2022. arXiv:2203.13086 [cs, eess] version: 2.

3. Belikov, I.: Mubert - Thousands of Staff-Picked Royalty-Free Music Tracks for Streaming, Videos, Podcasts, Commercial Use and Online Content, January 2023.

4. Bertin-Mahieux, T., Ellis, D. P. W., Whitman, B., and Lamere, P.: The Million Song Dataset. pages 591–596, 2011.

5. Borsos, Z., Marinier, R., Vincent, D., Kharitonov, E., Pietquin, O., Sharifi, M., Teboul, O., Grangier, D., Tagliasacchi, M., and Zeghidour, N.: AudioLM: a Language Modeling Approach to Audio Generation, September 2022. arXiv:2209.03143 [cs, eess].

6. Briot, J.-P., Hadjeres, G., and Pachet, F.-D.: Deep Learning Techniques for Music Generation – A Survey, August 2019. arXiv:1709.01620 [cs] version: 4.

7. Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., and Lerchner, A.: Understanding disentangling in β-VAE, April 2018. arXiv:1804.03599 [cs, stat].

8. Caillon, A. and Esling, P.: RAVE: A variational autoencoder for fast and high-quality neural audio synthesis, December 2021. arXiv:2111.05011 [cs, eess].

9. Chen, N., Zhang, Y., Zen, H., Weiss, R. J., Norouzi, M., and Chan, W.: WaveGrad: Estimating Gradients for Waveform Generation, October 2020. arXiv:2009.00713 [cs, eess, stat].

CITED LITERATURE (continued)

10. Decorsière, R., Søndergaard, P. L., MacDonald, E. N., and Dau, T.: Inversion of Auditory Spectrograms, Traditional Spectrograms, and Other Envelope Representations. IEEE/ACM Transactions on Audio, Speech, and Language Processing , 23(1):46–56, January 2015. Conference Name: IEEE/ACM Transactions on Audio, Speech, and Language Processing.

11. Defferrard, M., Benzi, K., Vandergheynst, P., and Bresson, X.: FMA: A Dataset For Music Analysis, September 2017. arXiv:1612.01840 [cs].

12. Deltorn, J.-M.: Deep Creations: Intellectual Property and the Automata. Frontiers in Digital Humanities , 4, February 2017.

13. Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., and Sutskever, I.: Jukebox: A Generative Model for Music, April 2020. arXiv:2005.00341 [cs, eess, stat].

14. Dhariwal, P. and Nichol, A.: Diffusion Models Beat GANs on Image Synthesis, June 2021. arXiv:2105.05233 [cs, stat].

15. Dinh, L., Sohl-Dickstein, J., and Bengio, S.: Density estimation using Real NVP, February 2017. arXiv:1605.08803 [cs, stat].

16. Donahue, C., Mao, H. H., and McAuley, J.: The NES Music Database: A multi-instrumental dataset with expressive performance attributes, June 2018. arXiv:1806.04278 [cs, eess] version: 1.

17. Donahue, C., McAuley, J., and Puckette, M.: Adversarial Audio Synthesis, February 2019. arXiv:1802.04208 [cs] version: 3.

18. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N.: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, June 2021. arXiv:2010.11929 [cs] version: 2.

19. Engel, J., Agrawal, K. K., Chen, S., Gulrajani, I., Donahue, C., and Roberts, A.: GANSynth: Adversarial Neural Audio Synthesis, April 2019. arXiv:1902.08710 [cs, eess, stat] version: 2.

20. Engel, J., Hantrakul, L., Gu, C., and Roberts, A.: DDSP: Differentiable Digital Signal Processing, January 2020. arXiv:2001.04643 [cs, eess, stat] version: 1.

CITED LITERATURE (continued)

21. Engel, J., Resnick, C., Roberts, A., Dieleman, S., Eck, D., Simonyan, K., and Norouzi, M.: Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders, April 2017. arXiv:1704.01279 [cs].

22. Falcon, W. and team, T. P. L.: PyTorch Lightning, April 2023.

23. Fernandez, J. D. and Vico, F.: AI Methods in Algorithmic Composition: A Comprehensive Survey. Journal of Artificial Intelligence Research , 48:513–582, November 2013. arXiv:1402.0585 [cs].

24. Franceschelli, G. and Musolesi, M.: Copyright in generative deep learning. Data & Policy , 4:e17, 2022.

25. Freivalds, K., Ozoli, E., and ostaks, A.: Neural Shuffle-Exchange Networks – Sequence Processing in O(n log n) Time, October 2019. arXiv:1907.07897 [cs] version: 3.

26. Gemmeke, J. F., Ellis, D. P. W., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., and Ritter, M.: Audio Set: An ontology and human-labeled dataset for audio events. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) , pages 776–780, New Orleans, LA, March 2017. IEEE.

27. Germain, M., Gregor, K., Murray, I., and Larochelle, H.: MADE: Masked Autoencoder for Distribution Estimation, June 2015. arXiv:1502.03509 [cs, stat].

28. Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y.: Generative Adversarial Networks, June 2014. arXiv:1406.2661 [cs, stat].

29. Hadjeres, G. and Crestel, L.: Vector Quantized Contrastive Predictive Coding for Template-based Music Generation, April 2020. arXiv:2004.10120 [cs, eess] version: 1.

30. Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C.-Z. A., Dieleman, S., Elsen, E., Engel, J., and Eck, D.: Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset, January 2019. arXiv:1810.12247 [cs, eess, stat].

31. Heichler, J.: Introduction to BeeGFS. November 2014.

CITED LITERATURE (continued)

32. Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., Slaney, M., Weiss, R. J., and Wilson, K.: CNN Architectures for Large-Scale Audio Classification, January 2017. arXiv:1609.09430 [cs, stat].

33. Hiller, J. and Isaacson, L. M.: Musical Composition with a High-Speed Digital Computer. Journal of the Audio Engineering Society , 6(3):154–160, July 1958. Publisher: Audio Engineering Society.

34. Hilmkil, A., Thomé, C., and Arpteg, A.: Perceiving Music Quality with GANs, April 2021. arXiv:2006.06287 [cs, eess] version: 2.

35. Ho, J., Jain, A., and Abbeel, P.: Denoising Diffusion Probabilistic Models, December 2020. arXiv:2006.11239 [cs, stat] version: 2.

36. Ho, J., Saharia, C., Chan, W., Fleet, D. J., Norouzi, M., and Salimans, T.: Cascaded Diffusion Models for High Fidelity Image Generation, December 2021. arXiv:2106.15282 [cs].

37. Ho, J. and Salimans, T.: Classifier-Free Diffusion Guidance, July 2022. arXiv:2207.12598 [cs].

38. Hoogeboom, E., Nielsen, D., Jaini, P., Forré, P., and Welling, M.: Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions, October 2021. arXiv:2102.05379 [cs, stat].

39. Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Shazeer, N., Simon, I., Hawthorne, C., Dai, A. M., Hoffman, M. D., Dinculescu, M., and Eck, D.: Music Transformer, December 2018. arXiv:1809.04281 [cs, eess, stat].

40. Huang, Q., Jansen, A., Lee, J., Ganti, R., Li, J. Y., and Ellis, D. P. W.: MuLan: A Joint Embedding of Music Audio and Natural Language, August 2022. arXiv:2208.12415 [cs, eess, stat].

41. Huang, Q., Park, D. S., Wang, T., Denk, T. I., Ly, A., Chen, N., Zhang, Z., Zhang, Z., Yu, J., Frank, C., Engel, J., Le, Q. V., Chan, W., Chen, Z., and Han, W.: Noise2Music: Text-conditioned Music Generation with Diffusion Models, March 2023. arXiv:2302.03917 [cs, eess].

CITED LITERATURE (continued)

42. Huang, R., Lam, M. W. Y., Wang, J., Su, D., Yu, D., Ren, Y., and Zhao, Z.: FastDiff: A Fast Conditional Diffusion Model for High-Quality Speech Synthesis, April 2022. arXiv:2204.09934 [cs, eess] version: 1.

43. Hung, H.-T., Wang, C.-Y., Yang, Y.-H., and Wang, H.-M.: Improving Automatic Jazz Melody Generation by Transfer Learning Techniques, August 2019. arXiv:1908.09484 [cs, eess] version: 1.

44. Ji, S., Luo, J., and Yang, X.: A Comprehensive Survey on Deep Music Generation: Multi-level Representations, Algorithms, Evaluations, and Future Directions, November 2020. arXiv:2011.06801 [cs, eess].

45. Jurafsky, D. and Martin, J. H.: Speech and language processing (3rd (draft) ed.), 2019.

46. Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., Stimberg, F., Oord, A. v. d., Dieleman, S., and Kavukcuoglu, K.: Efficient Neural Audio Synthesis, June 2018. arXiv:1802.08435 [cs, eess] version: 2.

47. Karras, T., Laine, S., and Aila, T.: A Style-Based Generator Architecture for Generative Adversarial Networks, March 2019. arXiv:1812.04948 [cs, stat].

48. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T.: Analyzing and Improving the Image Quality of StyleGAN. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) , pages 8107–8116, Seattle, WA, USA, June 2020. IEEE.

49. Kilgour, K., Zuluaga, M., Roblek, D., and Sharifi, M.: Fr\'echet Audio Distance: A Metric for Evaluating Music Enhancement Algorithms, January 2019. arXiv:1812.08466 [cs, eess].

50. Kingma, D. P. and Dhariwal, P.: Glow: Generative Flow with Invertible 1x1 Convolutions, July 2018. arXiv:1807.03039 [cs, stat].

51. Kingma, D. P. and Welling, M.: Auto-Encoding Variational Bayes, December 2022. arXiv:1312.6114 [cs, stat].

52. Kong, J., Kim, J., and Bae, J.: HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis, October 2020. arXiv:2010.05646 [cs, eess] version: 2.

CITED LITERATURE (continued)

53. Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B.: DiffWave: A Versatile Diffusion Model for Audio Synthesis, March 2021. arXiv:2009.09761 [cs, eess, stat] version: 3.

54. Kreuk, F., Synnaeve, G., Polyak, A., Singer, U., Défossez, A., Copet, J., Parikh, D., Taigman, Y., and Adi, Y.: AudioGen: Textually Guided Audio Generation, September 2022. arXiv:2209.15352 [cs, eess].

55. Krizhevsky, A., Sutskever, I., and Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems , volume 25. Curran Associates, Inc., 2012.

56. Kumar, K., Kumar, R., de Boissiere, T., Gestin, L., Teoh, W. Z., Sotelo, J., de Brebisson, A., Bengio, Y., and Courville, A.: MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis, December 2019. arXiv:1910.06711 [cs, eess] version: 3.

57. Law, E., West, K., and Mandel, M.: EVALUATION OF ALGORITHMS USING GAMES: THE CASE OF MUSIC TAGGING. Oral Session , 2009.

58. Lee, C. Y., Toffy, A., Jung, G. J., and Han, W.-J.: Conditional WaveGAN, September 2018. arXiv:1809.10636 [cs] version: 1.

59. Lipton, Z. C., Berkowitz, J., and Elkan, C.: A Critical Review of Recurrent Neural Networks for Sequence Learning, October 2015. arXiv:1506.00019 [cs] version: 4.

60. Liu, H., Chen, Z., Yuan, Y., Mei, X., Liu, X., Mandic, D., Wang, W., and Plumbley, M. D.: AudioLDM: Text-to-Audio Generation with Latent Diffusion Models, January 2023. arXiv:2301.12503 [cs, eess].

61. Lu, W.-T., Wu, M.-H., Chiu, Y.-M., and Su, L.: Actions Speak Louder than Listening: Evaluating Music Style Transfer based on Editing Experience. In Proceedings of the 29th ACM International Conference on Multimedia , pages 3936–3944, October 2021. arXiv:2110.12855 [cs, eess].

62. Maina, K.: Msanii: High Fidelity Music Synthesis on a Shoestring Budget, January 2023. arXiv:2301.06468 [cs, eess] version: 1.

63. Martiros, S. F. a. H.: Riffusion, December 2022.

CITED LITERATURE (continued)

64. Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., Courville, A., and Bengio, Y.: SampleRNN: An Unconditional End-to-End Neural Audio Generation Model, February 2017. arXiv:1612.07837 [cs] version: 2.

65. Meng, C., Rombach, R., Gao, R., Kingma, D. P., Ermon, S., Ho, J., and Salimans, T.: On Distillation of Guided Diffusion Models, November 2022. arXiv:2210.03142 [cs].

66. Michelashvili, M. and Wolf, L.: Hierarchical Timbre-Painting and Articulation Generation, September 2020. arXiv:2008.13095 [cs, eess] version: 2.

67. Murphy, K. P.: Probabilistic machine learning: an introduction . Adaptive computation and machine learning series. Cambridge, Massachusetts, The MIT Press, 2022.

68. Nagarajan, S., Nettimi, S. S. S., Kumar, L. S., Nath, M. K., and Kanhe, A.: Speech emotion recognition using cepstral features extracted with novel triangular filter banks based on bark and ERB frequency scales. Digital Signal Processing , 104:102763, September 2020.

69. Natsiou, A. and O'Leary, S.: Audio representations for deep learning in sound synthesis: A review, January 2022. arXiv:2201.02490 [cs, eess].

70. Nichol, A. and Dhariwal, P.: Improved Denoising Diffusion Probabilistic Models, February 2021. arXiv:2102.09672 [cs, stat].

71. Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K.: WaveNet: A Generative Model for Raw Audio, September 2016. arXiv:1609.03499 [cs] version: 2.

72. Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K.: Pixel Recurrent Neural Networks, August 2016. arXiv:1601.06759 [cs].

73. Oord, A. v. d., Vinyals, O., and Kavukcuoglu, K.: Neural Discrete Representation Learning, May 2018. arXiv:1711.00937 [cs].

74. Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, Shazeer, N., Ku, A., and Tran, D.: Image Transformer, June 2018. arXiv:1802.05751 [cs].

75. Pasini, M. and Schlüter, J.: Musika! Fast Infinite Waveform Music Generation, August 2022. arXiv:2208.08706 [cs, eess] version: 1.

CITED LITERATURE (continued)

76. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems , volume 32. Curran Associates, Inc., 2019.

77. Patashnik, O., Wu, Z., Shechtman, E., Cohen-Or, D., and Lischinski, D.: StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery. In 2021 IEEE/CVF International Conference on Computer Vision (ICCV) , pages 2065–2074, Montreal, QC, Canada, October 2021. IEEE.

78. Ping, W., Peng, K., Gibiansky, A., Arik, S. O., Kannan, A., Narang, S., Raiman, J., and Miller, J.: Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning, February 2018. arXiv:1710.07654 [cs, eess].

79. Preechakul, K., Chatthee, N., Wizadwongsa, S., and Suwajanakorn, S.: Diffusion Autoencoders: Toward a Meaningful and Decodable Representation. In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) , pages 10609–10619, June 2022. ISSN: 2575-7075.

80. Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I.: Learning Transferable Visual Models From Natural Language Supervision, February 2021. arXiv:2103.00020 [cs].

81. Raffel, C.: Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching. 2016.

82. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J.: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, July 2020. arXiv:1910.10683 [cs, stat].

83. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M.: Hierarchical Text-Conditional Image Generation with CLIP Latents, April 2022. arXiv:2204.06125 [cs].

84. Razavi, A., Oord, A. v. d., and Vinyals, O.: Generating Diverse High-Fidelity Images with VQ-VAE-2, June 2019. arXiv:1906.00446 [cs, stat].

CITED LITERATURE (continued)

85. Rec, I.: P. 800: Methods for subjective determination of transmission quality. International Telecommunication Union, Geneva , 22, 1996.

86. Ren, Y., Hu, C., Tan, X., Qin, T., Zhao, S., Zhao, Z., and Liu, T.-Y.: FastSpeech 2: Fast and High-Quality End-to-End Text to Speech, August 2022. arXiv:2006.04558 [cs, eess] version: 8.

87. Ren, Y., Ruan, Y., Tan, X., Qin, T., Zhao, S., Zhao, Z., and Liu, T.-Y.: FastSpeech: Fast, Robust and Controllable Text to Speech, November 2019. arXiv:1905.09263 [cs, eess] version: 5.

88. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B.: High-Resolution Image Synthesis with Latent Diffusion Models, April 2022. arXiv:2112.10752 [cs].

89. Ronneberger, O., Fischer, P., and Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation, May 2015. arXiv:1505.04597 [cs].

90. Salimans, T. and Ho, J.: Progressive Distillation for Fast Sampling of Diffusion Models, June 2022. arXiv:2202.00512 [cs, stat].

91. Schneider, F.: ArchiSound: Audio Generation with Diffusion. January 2023.

92. Schneider, F., Jin, Z., and Schölkopf, B.: Mo\^usai: Text-to-Music Generation with Long-Context Latent Diffusion, January 2023. arXiv:2301.11757 [cs, eess].

93. Serrà, J., Pascual, S., and Segura, C.: Blow: a single-scale hyperconditioned flow for non-parallel raw-audio voice conversion, September 2019. arXiv:1906.00794 [cs, eess, stat] version: 2.

94. Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerry-Ryan, R. J., Saurous, R. A., Agiomyrgiannakis, Y., and Wu, Y.: Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions, February 2018. arXiv:1712.05884 [cs] version: 2.

95. Shor, J., Jansen, A., Maor, R., Lang, O., Tuval, O., Quitry, F. d. C., Tagliasacchi, M., Shavitt, I., Emanuel, D., and Haviv, Y.: Towards Learning a Universal Non-Semantic Representation of Speech. In Interspeech 2020 , pages 140–144, October 2020. arXiv:2002.12764 [cs, eess, stat].

CITED LITERATURE (continued)

96. Song, J., Meng, C., and Ermon, S.: Denoising Diffusion Implicit Models, October 2022. arXiv:2010.02502 [cs] version: 4.

97. Song, Y., Dhariwal, P., Chen, M., and Sutskever, I.: Consistency Models, March 2023. arXiv:2303.01469 [cs, stat] version: 1.

98. Stoller, D., Tian, M., Ewert, S., and Dixon, S.: Seq-U-Net: A One-Dimensional Causal U-Net for Efficient Sequence Modelling, November 2019. arXiv:1911.06393 [cs, eess, stat] version: 1.

99. Sturm, B. L.: The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use. Journal of New Music Research , 43(2):147–172, April 2014. arXiv:1306.1461 [cs].

100. Tensorflow: YAMNet, 2020.

101. Theis, L., Oord, A. v. d., and Bethge, M.: A note on the evaluation of generative models, November 2015. arXiv:1511.01844 [cs, stat] version: 1.

102. Tokui, N.: Towards democratizing music production with AI-Design of Variational Autoencoder-based Rhythm Generator as a DAW plugin, April 2020. arXiv:2004.01525 [cs, eess] version: 1.

103. Tomczak, J. M.: Deep Generative Modeling . Cham, Springer International Publishing, 2022.

104. Uria, B., Côté, M.-A., Gregor, K., Murray, I., and Larochelle, H.: Neural Autoregressive Distribution Estimation, May 2016. arXiv:1605.02226 [cs].

105. Vahdat, A. and Kautz, J.: NVAE: A Deep Hierarchical Variational Autoencoder, January 2021. arXiv:2007.03898 [cs, stat].

106. Vasquez, S. and Lewis, M.: MelNet: A Generative Model for Audio in the Frequency Domain, June 2019. arXiv:1906.01083 [cs, eess, stat] version: 1.

107. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I.: Attention Is All You Need, December 2017. arXiv:1706.03762 [cs].

CITED LITERATURE (continued)

108. Wang, Y., Skerry-Ryan, R. J., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., Le, Q., Agiomyrgiannakis, Y., Clark, R., and Saurous, R. A.: Tacotron: Towards End-to-End Speech Synthesis, April 2017. arXiv:1703.10135 [cs] version: 2.

109. Watcharasupat, K. N. and Lerch, A.: Evaluation of Latent Space Disentanglement in the Presence of Interdependent Attributes, October 2021. arXiv:2110.05587 [cs, eess, math] version: 1.

110. Yamamoto, R., Song, E., and Kim, J.-M.: Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram, February 2020. arXiv:1910.11480 [cs, eess] version: 2.

111. Yang, D., Yu, J., Wang, H., Wang, W., Weng, C., Zou, Y., and Yu, D.: Diffsound: Discrete Diffusion Model for Text-to-sound Generation, July 2022. arXiv:2207.09983 [cs, eess].

112. Yoo, A. B., Jette, M. A., and Grondona, M.: Slurm: Simple linux utility for resource management. In Job Scheduling Strategies for Parallel Processing: 9th International Workshop, JSSPP 2003, Seattle, WA, USA, June 24, 2003. Revised Paper 9 , pages 44–60. Springer, 2003.

113. Youngberg, J. and Boll, S.: Constant-Q signal analysis and synthesis. In ICASSP '78. IEEE International Conference on Acoustics, Speech, and Signal Processing , volume 3, pages 375–378, Tulsa, Oklahoma, 1978. Institute of Electrical and Electronics Engineers.

114. Yu, F. and Koltun, V.: Multi-Scale Context Aggregation by Dilated Convolutions, April 2016. arXiv:1511.07122 [cs].

115. Zeghidour, N., Luebs, A., Omran, A., Skoglund, J., and Tagliasacchi, M.: SoundStream: An End-to-End Neural Audio Codec, July 2021. arXiv:2107.03312 [cs, eess].

116. Zhao, J. and Xia, G.: AccoMontage: Accompaniment Arrangement via Phrase Selection and Style Transfer, August 2021. arXiv:2108.11213 [cs, eess] version: 1.

117. Zhao, S., Song, J., and Ermon, S.: InfoVAE: Balancing Learning and Inference in Variational Autoencoders. Proceedings of the AAAI Conference on Artificial Intelligence , 33(01):5885–5892, July 2019.

CITED LITERATURE (continued)

118. Zhu, P., Pang, C., Wang, S., Chai, Y., Sun, Y., Tian, H., and Wu, H.: ERNIE-Music: Text-to-Waveform Music Generation with Diffusion Models, February 2023. arXiv:2302.04456 [cs, eess].

119. Ziegler, Z. M. and Rush, A. M.: Latent Normalizing Flows for Discrete Sequences, June 2019. arXiv:1901.10548 [cs, stat] version: 4.

VITA

NAME	Giuseppe Concialdi
EDUCATION	
	Alta Scuola Politecnica, Polytechnic of Milan, Italy, Sep 2023
	Master of Science in Computer Science, University of Illinois at Chicago, USA, Jul 2023
	Master of Science in Data Science and Engineering, Polytechnic of Turin, Italy, Jul 2023
	Bachelor of Science in Computer Engineering, Polytechnic of Turin, Italy, Jul 2021
LANGUAGE SKILLS	
Italian	Native speaker
English	Full working proficiency
	2021 - IELTS Academic 7.5
	A.Y. 2022/23 One year of study abroad in Chicago, Illinois
	A.Y. 2021/22. Lectures and exams attended exclusively in English
SCHOLARSHIPS	
2022-2023	Alta Scuola Politecnica honour program tuition waiver for the entire duration of the MSc at Polytechnic University of Turin
August 2022	TOP-UIC tuition waiver as best student of the Computer Science department
March 2022	Italian scholarship for TOP-UIC students