Towards Human-Quality Drum Accompaniment Using Deep Generative Models and Transformers

by

© **Arash Sadeghi Amjadi**

A thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of Master of Science.

Supervisor: Andrew Vardy

Co-Supervisor: Andrew Staniland

Department of Computer Science

Memorial University

May 2025

St. John's, Newfoundland and Labrador, Canada

Abstract

Automatic music generation has garnered significant interest among musicians and composers. In particular, the task of accompaniment in music generation presents unique challenges, as it involves generating an instrument track responsive to other played instruments. This project focuses on accompanying musicians with automatically generated tracks, specifically accompanying bass guitar players with AI-generated drum tracks. The proposed system was trained on multi-track songs to capture the connection between bass and drum tracks using the framework of Conditional Generative Adversarial Networks (CGANs). Unlike typical AI-generated drum tracks, which often lack nuanced dynamics, human-performed drums feature expressive elements such as velocity—the varying loudness of each strike. To capture this expressiveness, our transformer model is trained on human drum performances and focuses on assigning realistic velocities to the generated drum hits. An ablation study was conducted, and the results indicate that combining pitch and velocity generation into a single network significantly reduces music quality (measured by groove consistency), reinforcing our approach of separating velocity assignments to maintain coherent drum patterns while enhancing expressiveness. We also evaluate the generated music using objective metrics, demonstrating the models' performance and evolution during training. The drum generation system supports real-time interaction, enabling spontaneous live jamming sessions. Simplifications facilitate real-time operation, and we provide results from sample sessions.

To my family, for their unwavering support and endless love.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors, Professors Andrew Vardy and Andrew Staniland. Their invaluable guidance, continuous support, and belief in my work throughout this program have been instrumental in my success.

I am also deeply grateful for the financial support provided by SSHRC *Improvising Futures*, managed by IICSI (International Institution for Critical Studies in Improvisation) and Memorial University of Newfoundland. This funding was crucial in allowing me to pursue this degree.

I extend my immense thanks to my parents and my entire family for their unwavering encouragement and support throughout my academic journey.

Finally, I would like to thank Dr. Ali Emre Turgut for generously providing me with office space and resources during my visit to Middle East Technical University, Ankara, Turkey. This facilitated a productive writing environment during the critical thesis writing stage.

Citations to Published Work

Part of the work presented in this thesis was submitted and accepted in the following paper:

Sadeghi Amjadi, A., Staniland, A., & Vardy, A. (Accepted: January 2025). Towards Human-Quality Drum Accompaniment Using Deep Generative Models and Transformers. In International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar). Cham: Springer Nature Switzerland.

Table of contents

Title page i

Abstract ii

Acknowledgements iv

Citations to Published Work v

Table of contents vi

List of tables ix

List of figures x

List of abbreviations xi

1 Introduction 1

1.1 Motivation 2

1.2 Overview of the Drummer Companion System 3

1.3 Contributions 4

1.4 Thesis Outline 5

2 Related Works 7

2.1 Music Representation 7

2.2 Music Generation 10
2.3 Velocity Task 12
2.4 Music Accompaniment 13

3 Methodology **16**

3.1 Preliminary Work 16
3.2 Proposed Method 18
3.2.1 Drum Generator 19
3.2.2 Velocity Assigner 22

4 Implementation **24**

4.1 Dataset 24
4.1.1 Training Drum Generator 24
4.1.2 Training Velocity Assigner 25
4.2 Model Settings 26
4.2.1 Drum Generator 26
4.2.2 Velocity Assigner 27
4.3 Real-time Interaction 27

5 Experiments and Results **30**

5.1 Ablation Study on Joint Pitch-Velocity Generation 31
5.2 Model Errors 32
5.2.1 Drum Generator 32
5.2.2 Velocity Assigner 34
5.3 Music Performance Index 35
5.3.1 Empty Bar Ratio 36
5.3.2 Drum Pattern 36

5.4 Improvisation Samples 37

6 Conclusion **41**

6.1 Future Work 43

Bibliography **44**

List of tables

5.1 Groove Consistency (GC) deviation from GC of the training dataset for three models: 1) binary DG where DG was focused on only creating pitches without considering velocity. 2) DG with velocity generation along with pitch generation. 3) Untrained DG network, which basically just produces noise. 32

5.2 Performance index values for generated drum tracks and LPD dataset. 36

List of figures

2.1 Pianoroll representation of a music score sheet. 9

3.1 An example of Markov Chain trained on a simple music. 17

3.2 Proposed architecture. 19

4.1 Setup of a human interacting with the drum companion. 29

5.1 Discriminator loss 33

5.2 Evolution of piano roll representation 33

5.3 Training loss for VA on Groove dataset. 34

5.4 Validation loss for VA on Groove dataset. 35

5.5 Real-time improvisation sample of a human bassist with DG. 37

5.6 Offline improvisation sample of a human bassist with DG. 38

5.7 Velocity assigned to offline improvisation of DG with a human bassist. . 39

List of abbreviations

BERT	Bidirectional Encoder Representations from Transformers
CGAN	Conditional Generative Adversarial Network
C-RNN-GAN	Continuous Recurrent Neural Networks with Adversarial Training
D	Discriminator Network
DG	Drum Generator
DP	Drum Patern
E	Embedder Network
EB	Empty Bar Ration
G	Generator Network
GAN	Generative Adversarial Network
LSTM	Long Short-Term Memory
REMI	REvamped MIDI-derived events
RNN	Recurrent Neural Network
VA	Velocity Assigner

Chapter 1

Introduction

Automatic music generation has long captivated human imagination, with early attempts dating back to the 18th century when composers like Mozart devised musical dice games to randomly generate music sequences [25]. This enduring interest is driven by both the desire to deepen our understanding of music itself and the technical and artistic challenges it presents. With the advent of deep learning, this field has seen remarkable advancements, surpassing traditional statistical models and opening new horizons for musicians and composers. Today, automatic music generation not only provides novel tools for creativity but also poses unique challenges, particularly in generating responsive and cohesive accompaniments that interact dynamically with human performers [9, 16]. This project, motivated by the rich history and recent advancements, aims to address these challenges by developing an AI-driven system that accompanies bass guitar players with expressive, real-time generated drum tracks.

1.1 Motivation

Improvisation is widely regarded by educators and performers as a crucial skill in musical development, enhancing listening, timing, and creativity [17]. However, young musicians often lack access to a full band for consistent practice, limiting their ability to build ensemble skills in improvisational settings. By offering an intelligent, responsive AI drummer, our system addresses this gap—enabling solo musicians to rehearse and explore improvisation more effectively without needing a complete ensemble. This not only promotes technical growth but also nurtures musical intuition in a more accessible and engaging way.

The landscape of automatic music generation has undergone a profound transformation over the last sixty years, evolving from early grammar-based and probability model approaches to the revolutionary impact of deep learning methodologies. This transformative shift has spurred the development of numerous AI-driven systems, surpassing traditional statistical methods such as Markov models in the creation of high-quality musical compositions [6].

Generating realistic and aesthetically pleasing pieces stands out as one of the most intriguing challenges within the field of artificial intelligence. Recent advancements have witnessed significant progress in the generation of images, videos, and text, notably leveraging Generative Adversarial Networks (GANs) [13, 20]. Similar efforts have been made in the domain of symbolic music generation [9, 37, 23], although it remains a challenging task.

Our research aims to advance the field of AI-driven music generation by developing a system that can improvise alongside human musicians. Building upon existing efforts like Magenta's DrumBot [33], which generates real-time drum accompaniments,

we focus on enhancing the expressiveness and adaptability of AI-generated music. Specifically, our project seeks to create a real-time interactive system that generates dynamic drum tracks to accompany human bass guitar players, allowing for spontaneous and creative performances. By incorporating features such as expressive velocity variations and real-time genre switching, we aim to deepen the collaboration between musicians and AI, exploring new possibilities in human-AI improvisation.

1.2 Overview of the Drummer Companion System

Our research focuses on creating interactive popular multi-genre music accompaniment by generating expressive drum tracks that complement bass guitar performances by human musicians, both in real-time and offline. We also integrated genre adaptability, allowing musicians to select their desired genre for the generated drum accompaniment. We have named our system the Drummer Companion System (DAS).

The decision to accompany the bass with drums was based on the inherent compatibility of these two instruments in forming the rhythmic backbone of music. In many musical genres, the bass and drums constitute the rhythm section, working closely together to establish the groove and drive the tempo [5]. Additionally, we chose drums as the accompanying instrument to focus on enhancing its expressiveness by making it sound more human-like.

In addition to creating an accompanying drum track, we propose a novel approach to make the generated drum track expressive and sound human-like by training a separate network on drum tracks played by humans. Although velocity is often generated alongside pitch [2, 30, 37], we tested a single grayscale piano roll that encodes velocity as color intensity. Our ablation study revealed that combining both tasks in

one network lowered drum quality, as evidenced by a decline in 'groove consistency' [35]. Thus, our research proposes a drummer companion system that generates drum tracks for a given bass track in two phases:

1. Drum Generator (DG): Generates a drum track for a given bass track using a GAN (Generative Adversarial Networks).

2. Velocity Assigner (VA): Assigns velocity to the generated drum track using a transformer model trained solely on the human-played drum dataset.

1.3 Contributions

The contributions of this project can be listed as follows:

- **Separation of Drum Pitch Generation and Velocity Assignment:** We adopt a two-stage approach; our ablation study shows that combining pitch and velocity generation reduces drum quality (via 'groove consistency'). Splitting these tasks preserves coherent drum patterns and boosts expressiveness.

- **Genre-Conditioned Drum Generation with Real-Time Genre Switching:** We design our drum pitch generator using a Conditional Generative Adversarial Network (CGAN) that accepts the desired genre as input, enabling the generation of drum tracks in a specified genre. This feature allows musicians to change the genre during performances, providing flexibility and creative control. Incorporating real-time genre switching in drum accompaniment systems enhances adaptability and user interaction.

- **Development of a Real-Time Interactive System for Drum Accompaniment:** We develop a real-time interactive system capable of generating drum

tracks to accompany human bass guitar players, suitable for both live improvisation and offline use with MIDI files. Although real-time drum accompaniment has been explored in prior works, our system integrates the separation of pitch and velocity generation with real-time genre conditioning, aiming to enhance the expressiveness and adaptability of AI-generated drum tracks.

1.4 Thesis Outline

- **Chapter 2:** In this chapter, first we explore different music representations (Section 2.1). Then, we review similar works in the literature. We explore various methods of music generation and categorize research in this area while examining the different challenges that arise.

- **Chapter 3:** In this chapter, we explain the model we propose for drum track generation. We discuss the earlier version of our work and the conclusions that led to our current research. Additionally, se describe the deep learning framework used for generating drum tracks, detailing the different components of drum generation and the models used in each part.

- **Chapter 4:** In this chapter, we provide the implementation details for our drummer companion network. These details include the datasets used for training (Section 4.1), network configurations (Section 4.2), and implementation of real-time interaction (Section 4.3).

- **Chapter 5:** In this chapter, we provide details on the conducted experiments and their results. First, we show the results of the ablation test that justifies the separation of velocity generation and pitch generation in Section 5.1. Then we discuss training losses (Section 5.2), evaluate the generated music using objective

performance indices (Section 5.3), and provide a sample of human interaction (Section 5.4).

- **Chapter 6:** This chapter discusses the conclusions of our work and possible future research directions in music generation for accompanying humans.

Chapter 2

Related Works

This chapter reviews prior work related to automatic music generation, focusing on four key areas relevant to this thesis. Section 2.1 discusses various methods of representing music in a format suitable for machine learning models. Section 2.2 explores approaches used in generating symbolic music, particularly melody and multi-track compositions. Section 2.3 focuses on models that incorporate expressive elements such as velocity into music generation. Finally, Section 2.4 reviews existing methods for generating music accompaniment, with a particular emphasis on drum pattern generation.

2.1 Music Representation

The first problem to address in music generation is the representation of musical data. The music representation significantly affects the type of deep learning models that can be used and determines what kinds of musical information can be embedded in the representation.

Symbolic representations, such as MIDI (Musical Instrument Digital Interface), encode music through discrete events like pitch, duration, and velocity, using MIDI messages to convey performance data such as note on/off events. These can be represented in 1D sequences, facilitating communication between electronic instruments and software. The following list provides and example set of MIDI messages:

Listing 2.1: MIDI note samples.

```
Note(start=0.000000, end=0.123958, pitch=57, velocity=96)
Note(start=0.000000, end=0.373958, pitch=52, velocity=96)
Note(start=0.125000, end=0.373958, pitch=59, velocity=96)
Note(start=0.375000, end=0.498958, pitch=57, velocity=96)
Note(start=0.375000, end=0.748958, pitch=52, velocity=96)
Note(start=0.500000, end=0.748958, pitch=59, velocity=96)
Note(start=0.750000, end=0.873958, pitch=57, velocity=96)
Note(start=0.750000, end=0.998958, pitch=52, velocity=96)
Note(start=0.875000, end=0.998958, pitch=59, velocity=96)
Note(start=0.937500, end=1.123958, pitch=55, velocity=96)
```

For every note in this listing, *start* determines at which point in time (in seconds) the note should be played and *end* indicates when the note should end (in seconds). The pitch number is indicated by *pitch* which corresponds to a specific musical pitch on the MIDI note scale. For example, a pitch of 52 corresponds to E3, which is the note E in the third octave of the MIDI scale. The *velocity* indicates how forcefully a note is played, representing the intensity or volume of the note. The range for velocity values is from 0 to 127, where a higher velocity value (closer to 127) means the note is played with more intensity or louder and vice versa.

Another musical representation is the 2D pianoroll, where notes are depicted over time as a matrix. This method, although visually intuitive, faces challenges in distinguishing long and short notes and struggles with complicated rhythms, leading to proposals like the Conlon pianoroll [1] which explicitly represents note durations. The 2D matrix representation offers a visual advantage but requires careful consideration of time resolution and note duration representation to capture music nuances effectively.

Figure 2.1 illustrates a score sheet and pianoroll representation of a piece and also describes the musical terminology used for the rest of the paper.

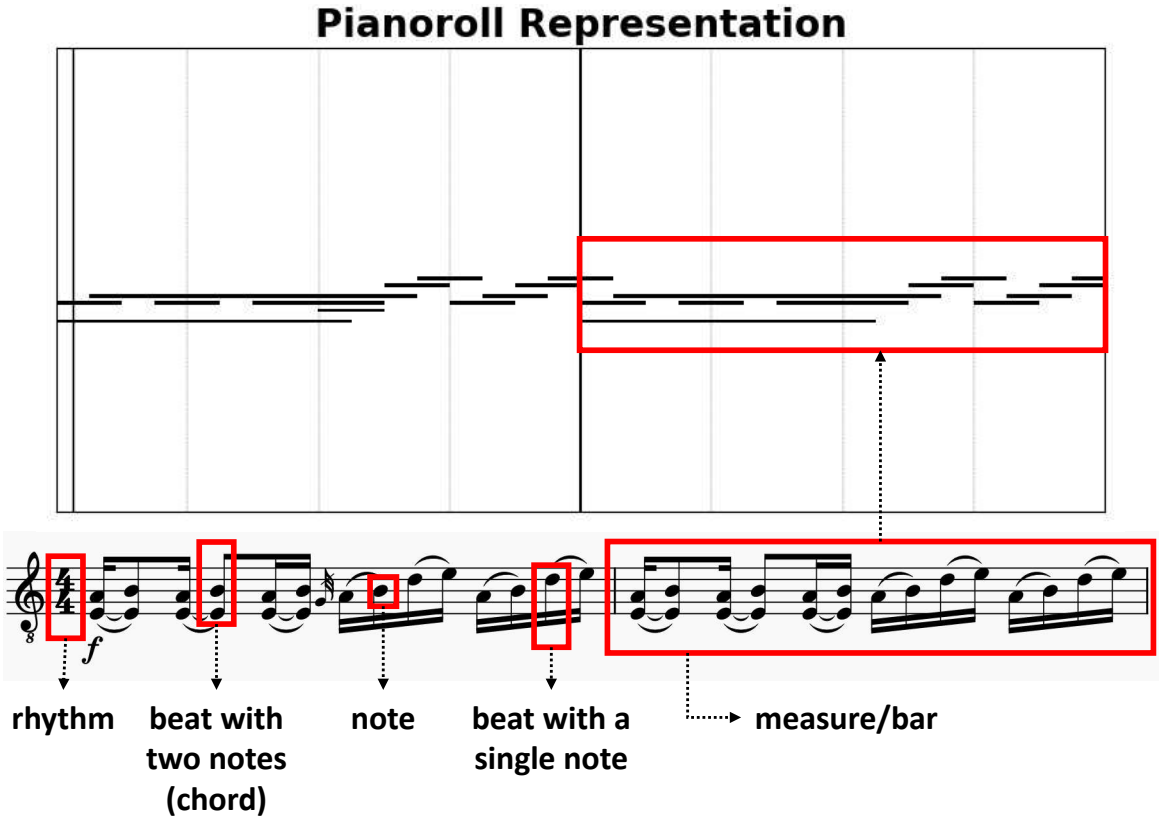

Figure 2.1: Pianoroll representation of a music score sheet. The number 4/4 defines the rhythm (time signature) of a measure (bar) and what will be the duration of a measure. Each measure consists of multiple beats. Each beat is a single note or a group of notes.

2.2 Music Generation

Advancements in GAN (Generative Adversarial Networks) architecture have yielded promising outcomes in music generation. This architecture not only demonstrates the ability to produce music but also offers the potential to exert control over the generated music by conditioning its generation based on given input. Researchers have approached the representation of sequential music data by adopting the piano roll format [9, 37], which can be treated as an image, thus facilitating processing by CNNs (Convolutional Neural Networks). As an example, in the MidiNet model [37], researchers employed a GAN structure with CNNs in both the generator and discriminator networks to generate multi-track music. Multi-track music refers to a composition that consists of multiple distinct instrumental or vocal parts (tracks), which are combined to create a richer and more layered sound. This model conditioned the generation of new bars on previously generated music bars, leading to melodies perceived as more appealing by test subjects compared to those generated by other state-of-the-art multi-track music generation models such as MelodyRNN [33].

The MuseGAN model [9] also addresses multi-track music generation, including drums, by using a GAN architecture. It allows for the simultaneous generation of multiple instruments, with drums being one of the key components. Although MuseGAN focuses on generating all tracks together, it highlights the importance of drums in the overall texture of generated music.

Furthermore, the MuseFlow model [8], which relies on an RNN (Recurrent Neural Network) structure, introduced flow-based music generation for creating music accompaniments, including drums, guitar, bass, and strings, based on the input piano melody. MuseFlow demonstrates superior performance in terms of accompaniment quality and harmony between tracks, closely mirroring the distributions of note pitch

and duration found in real music data.

In addition to treating music as images using the piano-roll format, numerous works have concentrated on generating music by treating it as sequential data. This approach allows for the utilization of architectures such as RNN [33], LSTM (Long Short-Term Memory) [31], and transformers [23, 38], which excel in capturing temporal dependencies within sequential data. Roberts et al. [31] integrated LSTM-based recurrent neural networks with Deep Q-learning, introducing a novel approach for real-time generation of musical sequences, particularly focusing on MIDI-encoded music scores.

Models like MelodyRNN, developed by the Magenta Project from the Google Brain team, have garnered attention for their ability to generate melodies from minimal input [33]. Minimal input refers to the small amount of initial musical information required for the model to begin the generation process. This could be a short sequence of notes, a single motif, or a starting melody, from which the model predicts and generates the subsequent notes using recurrent neural networks (RNNs). By learning musical patterns such as pitch and rhythm from large datasets, MelodyRNN can autonomously expand on this initial input, creating a complete and coherent melody.

MelodyRNN offers variants such as lookback RNN and attention RNN, enhancing its capability to understand longer-term structures and improve melody quality. Additionally, deep learning techniques have expanded symbolic music generation boundaries. DeepBach, by Sony CSL, utilizes an RNN-based architecture to compose polyphonic chorale music in J.S. Bach's style, allowing users to apply various constraints like rhythm and chords [14]. In contrast, C-RNN-GAN (Continuous Recurrent Neural Networks with Adversarial Training) pioneers the use of GANs in music generation, generating diverse melodies through random noise but lacks a conditional mechanism

for generating music based on priming melodies or chord sequences, showcasing the potential of GANs in innovative composition [22].

Addressing concerns regarding unpleasant melodies and insufficient harmony prevalent in existing multi-track music generation models, Zhao et al. [38] introduced a novel model that utilizes TransformerX (generator) - SpanBERT (discriminator) sequence conditions within a Generative Adversarial Network structure. In comparison to Transformer-GAN [23] and MuseGAN [9], their results demonstrate that the proposed model produces music with more harmonious tracks, enriched melody, rhythm, and coherence.

2.3 Velocity Task

In a velocity task, the system or model is tasked with predicting or assigning appropriate velocity values to each note, which can significantly affect the expressiveness and dynamics of the generated music. In the literature, velocity information of notes is often either ignored [9] or generated with the same network that generates the pitches. Several deep learning approaches have been explored for incorporating musical velocity into generated pieces. One approach utilizes a Deep Convolutional Generative Adversarial Network (DCGAN) to analyze MIDI data representations that include pitch, time, and velocity information [32]. This method enables the DCGAN to learn the inherent distribution of these elements from a given dataset and generate new music that incorporates these dynamics [32].

Expanding on the need to improve dynamic expressiveness in generated music, PopMAG introduced a novel MuMIDI representation [30]. MuMIDI allows for the simultaneous generation of multiple tracks within a single sequence, where each musical

note can include attributes like pitch, duration, and velocity [30]. This approach enhances the model's ability to capture the interplay between instruments but requires additional techniques to manage the increased sequence length due to incorporating multiple note attributes [30].

While some models focus on general expressive qualities, others target specific musical styles. The Pop Music Transformer leverages a Transformer architecture to generate expressive pop piano music [15]. However, it emphasizes the importance of data representation in achieving this expressiveness. By incorporating metrical structure into the input data, the model can more effectively capture the rhythmic and harmonic aspects of pop music, suggesting that specific data formatting can influence the model's ability to generate velocity variations [15].

Beyond stylistic considerations, research has also explored using deep learning for music style transfer. MIDI-VAE, a Variational Autoencoder based model, demonstrates the ability to manipulate musical dynamics by including note durations and velocities in its representation [2]. This allows MIDI-VAE to not only change pitches but also adjust the dynamics and instrumentation of a piece during style transfer tasks [2].

2.4 Music Accompaniment

Real-time music accompaniment by AI systems has seen significant development, focusing on both improvisation and following scores [18, 29, 27, 28]. One area of research explores AI accompaniment for jazz improvisation. Systems like the one presented in [18] analyze a soloist's input and musical score to generate accompanying chords in

real-time. This approach utilizes recurrent neural networks to predict soloist intentions and adapt the accompaniment accordingly.

Another approach focuses on real-time accompaniment for pre-composed pieces. Works like [29, 27, 28] analyze the soloist's performance (often through hidden Markov models) alongside the musical score. This information is then used to guide the generation or selection of accompaniment parts in real-time. These systems can learn from past rehearsals to personalize the accompaniment to the soloist's playing style [27, 28].

Recent advancements address challenges like latency and limited musical knowledge in real-time accompaniment. SongDriver [34] proposes a two-phase system separating chord arrangement and accompaniment generation. This avoids both logical latency and exposure bias, allowing for more natural-sounding accompaniment. Additionally, the model incorporates long-term musical information to compensate for shorter input sequences under real-time constraints.

Drums play a vital role in setting the rhythm and energy of a musical piece, and generating realistic drum patterns that complement other instruments is a complex task. One of the pioneering works in this domain is by Gillick et al. [12], who introduced a model for generating drum patterns conditioned on other instruments using sequence-to-sequence neural networks. Utilizing the Groove MIDI Dataset [12], which contains expressive human-performed drum recordings, they focused on capturing the nuances of human drumming, including timing and velocity variations.

In the realm of Generative Adversarial Networks (GANs), the DrumGAN model by Nistal et al. [24] leverages a GAN to synthesize drum sounds and patterns conditioned on latent embeddings. While primarily focused on sound synthesis, their approach contributes to the generation of drum accompaniments by enabling the creation of

realistic drum timbres that can be integrated into musical compositions.

Choi et al. [3] introduced a method for generating drum patterns using variational autoencoders (VAEs). Their model learns the latent representation of drum grooves and can generate new patterns by sampling from the latent space. This approach emphasizes the stylistic aspects of drumming and the ability to generate patterns that fit specific genres or moods.

Despite these advancements, existing models often struggle with capturing the expressive dynamics and intricate timing of human drumming, especially when conditioned on specific musical inputs like bass guitar tracks. Our work aims to fill this gap by introducing a GAN-based model that generates drum accompaniments conditioned on human-played bass guitar inputs. We emphasize the expressiveness of the generated drums by incorporating a separate network for velocity assignment, leveraging transformer architectures to capture the sequential nature of music dynamics.

Chapter 3

Methodology

3.1 Preliminary Work

In our initial attempts, we aimed to generate music in a simplistic and intuitive manner. We used Markov Chains on music scores to learn the underlying statistical characteristics of these scores. A Markov chain [4] is a mathematical model that describes a sequence of possible events where the probability of each event depends only on the state attained in the previous event. This property makes Markov chains suitable for modeling sequential data, including music, where the next note or chord is influenced by the current one.

In a musical context, each state in a Markov chain can represent a specific musical chord or note. For instance, consider a Markov chain where the states are chords, and the transitions between states represent the likelihood of moving from one chord to another based on trained data. By analyzing a piece of music, we can determine the probability of transitioning from one chord to another, thereby constructing a transition matrix that encodes these probabilities. This matrix can then be used to

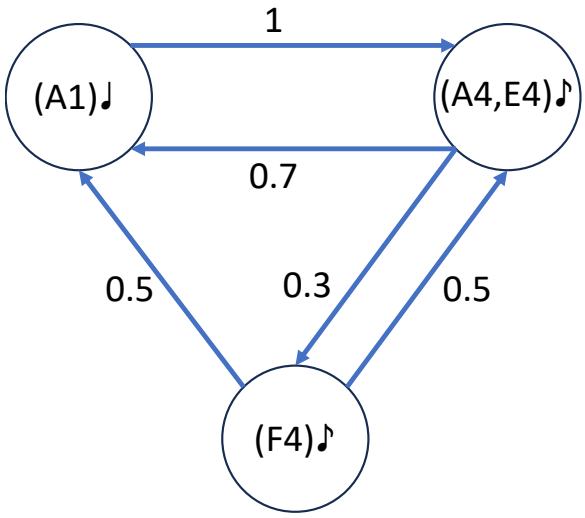

Figure 3.1: An example of Markov Chain trained on a simple music, showing transition probablities between chords and notes.

generate new music by following the chain's transitions, thus producing a sequence of chords or notes that are statistically similar to the original dataset. Figure 3.1 provides an example of Markov Chain trained on a minimal music sheet.

We trained a Markov Chain on five Metallica songs. Metallica's song notes are easily accessible in the format of GP3 files, making it easy to work with them and import them as symbolic music. We employed two methods: (1) taking musical measures as states, and (2) taking beats as states. In the first approach, there was little overlap between the musical measures of the five songs. Thus, during the Markov Chain random walk, whichever song the first measure belonged to, the Markov Chain tended to stay in that song. The only probable switches between different songs occurred at silent measures, which exist in all of the training songs.

In the second approach, when we took beats as states, the overlap between songs increased, as the training set had more overlapping beats. However, the rhythm of measures was sometimes violated, and the resulting music, despite not being pleasant to the ear, mimicked the training set.

One challenge we faced in the early stages was that the five songs had different tempos, and the measure rhythm and tempo changed during the songs. This problem manifested during the Markov Chain random walk when we generated a song with a single tempo, causing beats and measures from songs with different tempos to sound either slow or fast. Thus, ideally, a training set would have a single tempo and fixed rhythm. This finding helped us in choosing a dataset for training our system, which is discussed in detail in Section 4.1.

Additionally, the Markov Chain model was not suitable for our Drummer Companion purpose, as the Markov Chain only generates music but does not generate a track (drum) corresponding to another instrument (bass). Furthermore, the Markov Chain only captures simplistic statistical features of the training data. However, music generation for improvisation with humans requires a more complex generative model that could mimic the training dataset with greater complexity. For this purpose, we chose to use GANs [13] due to their success in generating new data instances.

3.2 Proposed Method

In this section, the proposed Drum Accompaniment System (DAS) is discussed in detail, with each part of the model explained. Overall, DAS consists of two independent parts trained separately and serving different purposes. The first part, the Drum Generator (DG), receives a human bass track and generates a drum track for it without velocity. The second part, the Velocity Assigner (VA), receives drum tracks generated by DG in the form of MIDI messages and assigns them velocity using the pre-trained BERT transformer [7] in cased English language. The entire pipeline is depicted in Figure 3.2. The following subsections explain both of these parts in detail.

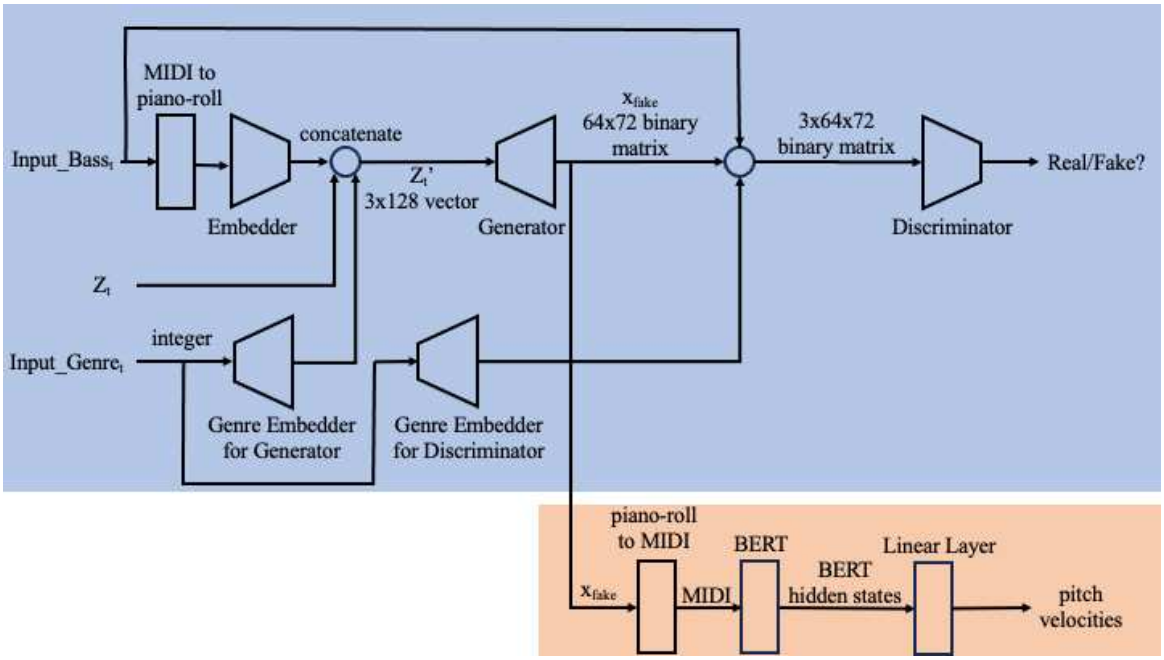

Figure 3.2: Proposed architecture for bass accompanying drum agent. The blue block represents DG, and the red block represents VA.

3.2.1 Drum Generator

Generating a responsive drum track requires solving three challenges: 1) generating drum tracks, 2) generating drum tracks responsive to human-played bass, and 3) generating drum tracks based on the required genre. Each of these challenges contributes to the final architecture of the model. The part of the model that handles these challenges is called Drum Generator (DG).

For drum generation, we used a binary pianoroll representation of music. The pianoroll representation is a binary 3D matrix, where the first dimension corresponds to pitch, the second dimension corresponds to measure (bar), and the third dimension corresponds to time (beat). By introducing measure as a separate dimension, we enforce our network to learn the structure of a bar. For the velocity assignment section (the red block in Figre 3.2), we used a one-dimensional MIDI representation

since, in that network, the main objective is to extract velocity information. We also tested a variant of DG using a grayscale piano-roll to simultaneously generate pitch and velocity. Velocity was encoded as the gray colour intensity in the piano-roll representation.

We chose our dataset based on the decided music representation and the information they carry. Further elaboration on dataset selection will be provided in Section 4.1. In the following subsections, each architectural challenge is described in detail.

Generating Drum Track with No Input

Considering that the music data is stored as 3D binary matrices, CNNs are a good choice to analyze these data. Since we are focused on generating innovative data based on the training dataset, a GAN framework is ideal for this purpose. The fundamental principle of GANs lies in adversarial learning, which involves the construction of two networks: the generator and the discriminator [13]. The generator is tasked with mapping random noise z sampled from a predefined distribution p_z to the data space p_{data}. Conversely, the discriminator is trained to differentiate between real data x_{real} and data generated by the generator x_{fake}, while the generator aims to deceive the discriminator. This training process is formally represented as a two-player minimax game between the generator G and the discriminator D. The optimization objective of a GAN can be formulated as Equation 3.1:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z))] \tag{3.1}$$

Using this framework, we can generate drum tracks that are similar to the training data but responsive to human inputs. For this purpose, a Conditional Generative Adversarial Network (CGAN) [21] is employed, a GAN model that incorporates conditioning. Conditioning in this context refers to the process of generating outputs based on additional input data, allowing the model to tailor the generated music to specific conditions, such as the bass line played by the musician.

Generating Drum Track Based on Human Input

In the CGAN framework, the generation process is guided by providing a condition y for both the generator and discriminator networks. Within the generator, the prior input noise $p_z(z)$ and condition y are combined into a joint hidden representation, offering substantial flexibility in the composition of this hidden representation. In the discriminator, the input is the concatenation of x and y. In our case, the condition y, which is the pianoroll representation of the bass track played by humans, can be concatenated with x to be fed to D but cannot be concatenated with z for the generator due to dimension differences. Thus, the embedder network, E, generates a learnable embedding of the condition, that can be concatenated with z, resulting in a vector which has compatible dimensions to be fed to the Generator network. The objective function of this two-player minimax game is expressed as Equation 3.2.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z|E(y))))] \quad (3.2)$$

Compared to Equation 3.1, we observe that $D(x)$ and $G(z)$ are replaced by $D(x|y)$ and $G(z|E(y))$, respectively. This indicates that the output of the discriminator, in

addition to the input data x, also depends on a condition—in this case, the human bass input y. Furthermore, $G(z|E(y))$ shows that the generation of the drum track depends not only on the latent vector z but also on the embedded representation of the human bass input, denoted as $E(y)$.

Human bass input is received as MIDI messages. With a buffer of 4 musical measures, these MIDI notes are then converted to a piano-roll representation, which can be considered as an image or binary matrix. The binary matrix, which in our CGAN context is y, passes through the embedder network to generate z'. All embedder, generator, and discriminator networks are implemented as CNNs.

Generating Drum Track Based on Desired Genre

The CGAN framework enables us to condition the generated drum track not only on the human bass input but also on the desired genre. Assuming the training dataset contains a finite number of genre labels, an embedder layer can map each genre to a corresponding matrix representation. This matrix representation can then be concatenated with other conditional inputs for both the discriminator and generator, allowing us to control the genre of the generated drum tracks.

3.2.2 Velocity Assigner

In this phase, to add velocity information to the drum track and make it expressive, we used the pre-trained Bidirectional Encoder Representations from Transformers (BERT) [7] on cased English, which retains the distinction between uppercase and lowercase letters, and fine-tuned it on human-played drum tracks which include velocity information. This part of the model which is tasked with assigning velocity to generated drum pitches is named Velocity Assigner (VA).

The objective here is to assign velocities, v, for drum notes that lack it. For this purpose, for drum tracks in the Groove dataset [12], we removed velocity information, and our goal was to train a deep learning model that would reacreate velocity information for each drum strike. The reason for choosing BERT model for this task is that velocity of a drum beat depends on the context of music and bar. This dependency includes beats before and after current beat, thus a bi-directional context of music increases the chances of better velocity assignment, compared to CGAN where context is local and limited.

A linear layer was attached to the last layer of the BERT model that transferred each input token's hidden state to its velocity. The velocity information is then incorporated into the original MIDI file, and the output is a MIDI file with a drum track with velocity. We used the Mean Squared Error (MSE) loss function to measure the error between the produced velocities $v_{\text{predicted}}$ and the expected velocities v_{real}:

$$\text{loss} = \frac{1}{N}\sum_{i=1}^{N}(v_{\text{real}} - v_{\text{predicted}})^2 \tag{3.3}$$

For tokenization, we used the Miditok library [11] (version 3.0.1) to tokenize MIDI messages. Then, we used the BERT tokenizer from the HuggingFace platform to tokenize these MIDI tokens into another set of tokens that are processable with BERT. The final output of the velocity assigner, after passing through the linear layer, consists of integers that correspond to $v_{\text{predicted}}$. To aid the training process, we scaled velocity information from the range of $[0, 127]$ to $[0, 1]$. Scaling the velocity values to a normalized range helps improve the numerical stability of the network during training by ensuring that the input features are on a similar scale, which facilitates faster convergence and more effective learning.

Chapter 4

Implementation

This chapter presents the implementation details of our proposed drum accompaniment system. We begin by introducing the datasets used for training both the Drum Generator (DG) and the Velocity Assigner (VA) in Section 4.1. Section 4.2 outlines the architectural and training configurations for each component of the system. Finally, Section 4.3 describes how the implemented system enables real-time interaction with human performers and discusses the practical considerations and simplifications applied to support live performance scenarios.

4.1 Dataset

4.1.1 Training Drum Generator

For training the Drum Generator (DG), we used the Lakh Piano-roll Dataset (LPD) [9], which includes piano rolls of multi-instrument songs. Specifically, we utilized the LPD-cleansed dataset [9], comprising 21,425 piano rolls of multitrack songs featuring

five instruments: bass, piano, strings, drums, and guitar. All measures in songs have a time signature of 4/4 and all songs start with the first beat at time zero.

The time signature constraint (being consistently in 4/4) was applied to the training songs, and it is recommended to test the model with songs that adhere to this constrain. If this constrain is not followed, the Drum Accompaniment System will still generate a drum track, but it may produce unexpected results. For the sake of this thesis and to simplify music generation, we adhered to this constrain.

Furthermore, each song belongs to one of 13 genres: Folk (45 songs), Country (512 songs), Rap (164 songs), Blues (22 songs), RnB (397 songs), New-Age (67 songs), Vocal (114 songs), Reggae (45 Songs), Pop Rock (4345 songs), Electronic (889 songs), International (206 songs), Jazz (157 songs), and Latin (360 songs). This genre information is used during training for conditioning drum generation based on the desired genre.

During each epoch of training, we used 4 measures of the bass track from each data instance as condition y, and the drum track from the data instance as real data x_{real} for the discriminator, along with the genre of music for conditioning the generator and discriminator.

4.1.2 Training Velocity Assigner

The Velocity Assigner (VA) operates on MIDI files, and its sole purpose is to assign velocity to generated drum pitches. The output of the generator network, x_{fake}, is converted from a piano roll to MIDI messages and fed to the BERT network for velocity assignment.

We chose the pretrained BERT model for cased English [7] and fine-tuned it using

the Groove MIDI Dataset (GMD) [12]. The GMD comprises 13.6 hours of synchronized MIDI and synthesized audio of human-performed, tempo-aligned expressive drumming, including 1,150 MIDI files and over 22,000 measures of drumming, with most performances in 4/4 time. Expressive drumming embodies the feel and groove that comes from a human's touch, making the music sound lively and engaging.

For each song in this dataset, we tokenized MIDI events using REMI (REvamped MIDI-derived events), implemented by the Miditok Python library [10]. Then, the tokens corresponding to velocity information were masked with the special BERT token,"[MASK]". The IDs corresponding to these tokens were tokenized using the BERT cased tokenizer [7]. Input lengths were limited to 510 tokens. The hidden states from the last layer of the BERT model, with dimensions of 768 (each hidden layer dimension) x 510 (number of output layers), were passed through a fully connected linear layer to reduce dimensions to 510 outputs. The outputs corresponding to masked input tokens were compared against initial velocity values, and the difference was backpropagated through the network.

4.2 Model Settings

4.2.1 Drum Generator

Networks G (Generator), E (Embedder), and D (Discriminator) are implemented as deep CNNs. G uses six 3D transposed convolution layers to increase the dimensions of the random noise vector z to the size of a piano roll with 4 measures, each measure with a 16-beat resolution. E uses six 3D convolution layers to reduce the size of human bass tracks to a vector of size 128, the same size as z. E has a similar layer size to G, only in reverse order. Both networks use 3D batch normalization. D uses

seven 3D convolution layers, with batch normalization in each layer. All G, D, and E are updated simultaneously. The training time for each model was less than 24 hours with an Nvidia RTX 3090 GPU.

4.2.2 Velocity Assigner

We used the BERT base model [7], which has 12 layers, 768 features in each layer, and 12 self-attention heads. We used the BERT pretrained on cased English and fine-tuned it for our purpose on the Groove MIDI dataset [12].

4.3 Real-time Interaction

The DG part was trained and tested on music files, so the music generation was not initially real-time. However, given a random vector z of size 128 and four bars of a bass track (along with the desired genre), G generates four bars of a drum track. This allows humans to interact with the model and jam with it since after every four measures of generated drums, a new random latent vector z conditioned with the desired genre and human bass input will trigger the generation of the next four bars of drums.

However, a few challenges need to be addressed. First, considering that DG works with a music buffer of four bars, and assuming a fixed tempo of 120 bpm and a rhythm of 4/4, each bar will take 2 seconds, and thus a buffer of four bars will take 8 seconds. During this whole 8-second period, DG listens to the human bassist, and once it receives the full four bars, it generates the associated drum track for the previously played bass bar. In other words, the drum track for the bass track played in the time window of t_0 to t_7 will be played in the time window of t_7 to t_{14}. Considering the

computation needed for DG, which was 2 seconds on an Apple MacBook Pro 2019, the drum track will be generated in the time window of t_9 to t_{16}. Thus, the drum track is played with a delay. If we try to assign velocities with VA, due to the larger size of the VA network, this time delay will increase.

This problem can be solved by different methods, one of which could be estimating the human bass track in the next time frame and generating the drum track for that estimation. However, for simplicity and to allow the human bassist to interact with the model in its current form, we relied on the repetitive nature of drum tracks [19] in songs and played the delayed drum track for the bassist. Since the human bassist plays four bars and DG generates four bars of drums, any computation time of DG will cause a silence between drum generations. In other words, when the human plays the bass track in the time window of t_0 to t_7, there will be a silence of 2 seconds for DG to generate the drum track. Thus, there will be silence in the time window of t_7 to t_9, and DG will start to play the drum track in the time window of t_9 to t_{16}. It should be noted that 2 seconds is for a MacBook Pro 2019. Different machines will have different computation times based on their computational power.

To avoid the time frame of silence caused by the computation time needed for DG, we proposed a simplistic solution of repeating the last bar of the generated drum track from the previous frame to fill the silence until DG generates the next four bars of drums. The reason we repeat only the last bar is due to our assumption that since the tempo is 120 bpm and the rhythm is 4/4, 2 seconds will correspond to one full bar. To avoid increasing the silence time frame and reduce computation time, we did not include VA for real-time interaction since, due to the large size of VA networks (BERT model), the computation time required for VA is relatively large (5 seconds on a MacBook Pro 2019). In future work, a smaller network for VA can be considered, or

if human bass track prediction is implemented, VA calculation time will not impose a problem.

By applying the aforementioned simplifications, DG is capable of interacting with human bassists in real time. Results for this interaction are provided in Section 5.4.

The interaction setup with the human player is depicted in Figure 4.1. We used a MIDI keyboard instead of a bass guitar for the ease of directly getting MIDI messages to the computer.

Figure 4.1: Setup of a human interacting with the drum companion. For the ease of generating MIDI messages, we used a MIDI keyboard instead of a bass guitar. The computer runs the DG model and plays the output, the generated drum track, to accompany the human player. To reduce generation latency, we did not use VA in real-time interaction.

Chapter 5

Experiments and Results

Assessing the quality of artistic creations, particularly those generated by algorithms, presents a unique challenge. Unlike tasks in machine learning focused on objective goals like classification or prediction, artistic domains such as music, literature, and cinema are inherently subjective [36]. While accuracy is paramount in tasks like image recognition, the evaluation of creative endeavors ultimately relies on human judgment as the arbiter of quality. This inherent subjectivity makes it difficult to establish a single, unified evaluation criterion for algorithmic art generation. However, objective evaluation metrics can still provide valuable insights.

The task of defining various objective musical measures is an active area of research [16], with each metric assessing different aspects of musical performance, such as melody and rhythm. Most deep learning research on music relies on subjective tests alongside a few objective metrics [9, 37, 16]. In our work, we sampled the output of our model for a human bass guitarist in both real-time and offline scenarios and report and compare the generated tracks in this chapter. Additionally, we report the training loss of the Drum Generator (DG) and Velocity Assigner (VA) (Section 5.2)

to provide an intuition about the models' learning processes.

After examining different musical performance indexes, we selected two performance indexes applicable to drum tracks: Empty Bar Ratio and Drum Pattern [9]. Further details on these performance metrics are provided in Section 5.3. The main goal with these performance indexes is to achieve values similar to the original dataset, as these values themselves do not directly indicate the quality of the generated music. However, having performance index values close to the original dataset can suggest that our trained model generates music similar to the original dataset.

Beyond Empty Bar Ratio and Drum Pattern, we also report a 'groove consistency' metric [35] to evaluate how stable the drum patterns remain across consecutive measures. The results of this study are reported in Section 5.1.

5.1 Ablation Study on Joint Pitch-Velocity Generation

Following [35], we use *groove consistency* to assess the quality of generated music. This metric quantifies differences in note placements between two consecutive measures without considering note velocity. Values close to those found in the training dataset typically correspond to a more pleasing rhythmic flow [26], whereas larger deviations often indicate abrupt or incoherent transitions between bars, making the track less musically appealing.

We conducted an ablation study by modifying the DG to generate pitch *and* velocity from a single grayscale piano roll representation. As shown in Table 5.1, this combined approach yielded a substantially higher deviation from the dataset's groove

consistency (from now on referred to as GC) baseline compared to the separate-task approach, indicating poorer rhythmic stability. This finding prompted us to maintain our two-stage design, where velocity assignment is modelled separately to preserve drum pattern quality and enhance expressiveness.

GC deviation for binary DG	**0.00427**
GC deviation for DG with velocity generation	0.0158
GC deviation for untrained DG	0.06811

Table 5.1: Groove Consistency (GC) deviation from GC of the training dataset for three models: 1) binary DG where DG was focused on only creating pitches without considering velocity. 2) DG with velocity generation along with pitch generation. 3) Untrained DG network, which basically just produces noise.

5.2 Model Errors

5.2.1 Drum Generator

We here report the loss values for the discriminator D. Positive values correspond to real samples, and negative values correspond to fake samples. As the CGAN operates in a Min-Max framework, we ideally expect the loss to converge to a steady value over training epochs [13]. The loss of the discriminator is reported in Figure 5.1. We have selected five critical points in the training process to observe the evolution of the model from different aspects.

The piano roll output of the generator model at these points is shown in Figure 5.2. It can be seen that the generated drum piano roll initially starts with a noisy pattern and then converges to common drum patterns.

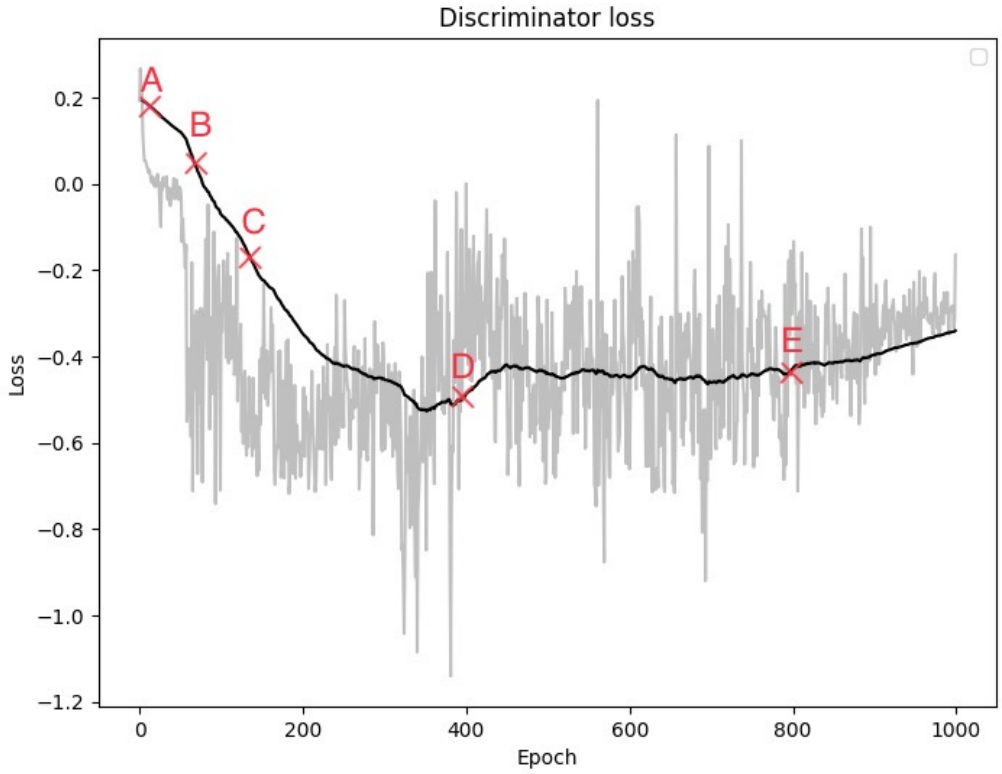

Figure 5.1: Discriminator loss over training epochs. The gray data represent the original losses and the black line is the smoothed data. Point A represents the initial untrained state, Points B and C represent mid-training states, Point D is the first instance of reaching a steady state value, and Point E is the converged state.

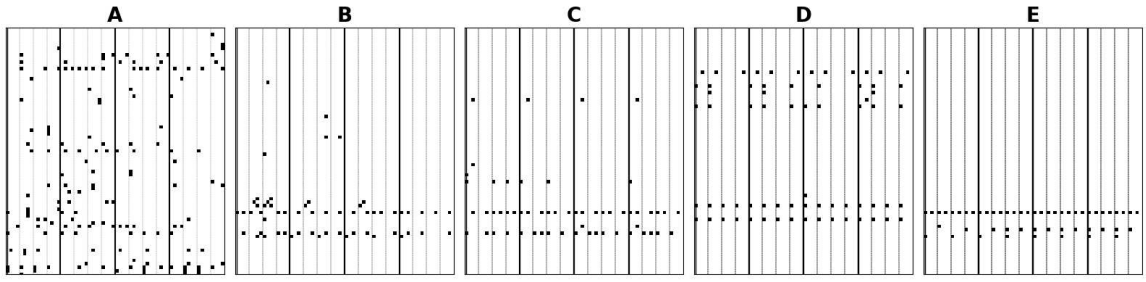

Figure 5.2: Evolution of piano roll representation corresponding to points in Figure 5.1.

5.2.2 Velocity Assigner

The training loss for VA is reported in Figure 5.3. As illustrated, the BERT model learns to associate drum pitch velocity with each drum pitch in the Groove dataset.

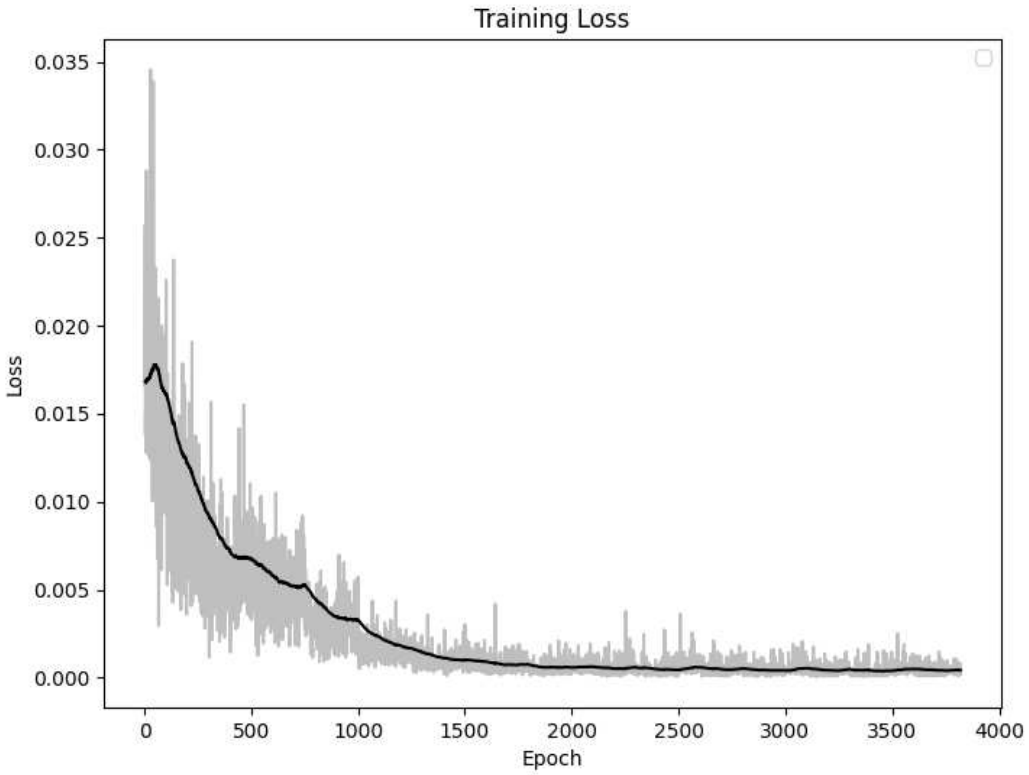

Figure 5.3: Training loss for VA on Groove dataset.

Figure 5.4 illustrates the validation loss for VA. Despite being lower than the training loss, the validation loss, (approximately) 0.18 for most epochs, oscillates. Considering that the training loss decreases as the network is trained over more-and-more epochs, the oscillation in the validation loss might indicate overfitting or a small validation dataset. Despite this phenomenon, the velocities generated for drum pitches in music pieces from the LPD dataset were satisfactory by subjective judgment (Refer to section 5.4 for more details.) In future work, it may be possible to

reduce validation loss by decreasing the learning rate over time and/or incorporating different regularization methods such as dropout.

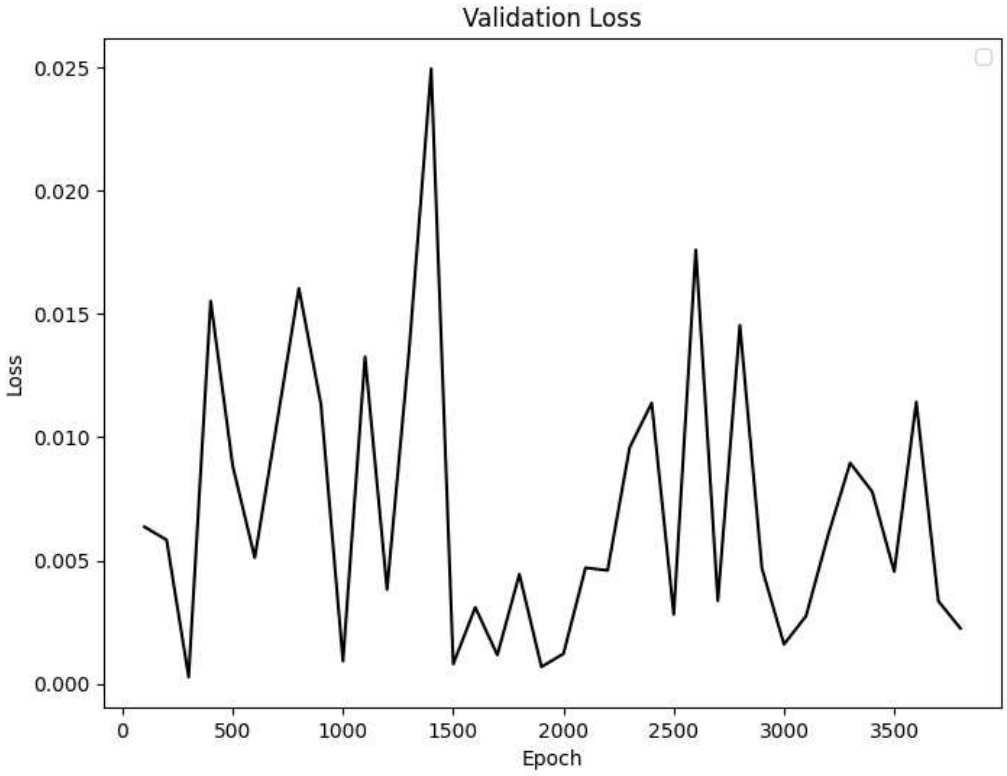

Figure 5.4: Validation loss for VA on Groove dataset.

5.3 Music Performance Index

The performance indexes reported in this section are applied to DG with its network weights at epoch 1000. This is the epoch where the discriminator shows stable convergence based on Figure 5.1.

5.3.1 Empty Bar Ratio

Empty Bar Ratio (EB) is defined as the ratio of empty bars to non-empty bars in percentage [9]. We calculated EB only for generated drum tracks over the entire bass track of the LPD dataset and compared it with the EB of the LPD dataset for drum tracks. EB is independent of drum pitch velocity and the pitch value.

The results are depicted in Table 5.2. As shown, both datasets have a low EB. However, there is a 6.09% difference between them. This occurs because DG sometimes creates a single pitch in an empty bar instead of leaving the whole bar empty. Due to the sensitive nature of EB, a single pitch in a bar disqualifies that bar from being empty. It was also observed that DG has less tendency to leave a whole bar empty and usually inserts a pitch or two in an otherwise empty bar.

5.3.2 Drum Pattern

Drum Pattern (DP) is defined as the "ratio of notes in 8- or 16-beat patterns, common ones for Rock songs in 4/4 time (in %)" [9]. DP is only calculated for songs with the Pop-Rock genre since this pattern is common in this genre. For the entire LPD dataset drum tracks, we counted the number of notes that follow this pattern and did the same for the drum tracks generated by DG based on LPD bass tracks. In Table 5.2, we reported these values in percentage. The small difference between the DP of the dataset and the generated drum tracks (0.31%) suggests that DG has captured the general drum pitch pattern for Pop-Rock songs in the LPD dataset.

	EB	DP
LPD dataset	0.42%	81.26%
Generated drum	6.51%	80.95%

Table 5.2: Performance index values for generated drum tracks and LPD dataset.

5.4 Improvisation Samples

Similar to the previous section, the generated drum tracks here are reported based on the DG's weights at epoch 1000. A human bass player improvises in real-time with DG, and the improvisation is shown in Figure 5.5. The selected genre for this live session was Pop-Rock. The video link for this improvisation along with audio sample and implementation of DAS is: `arash-sadeghi.github.io/MusicAiPage/`

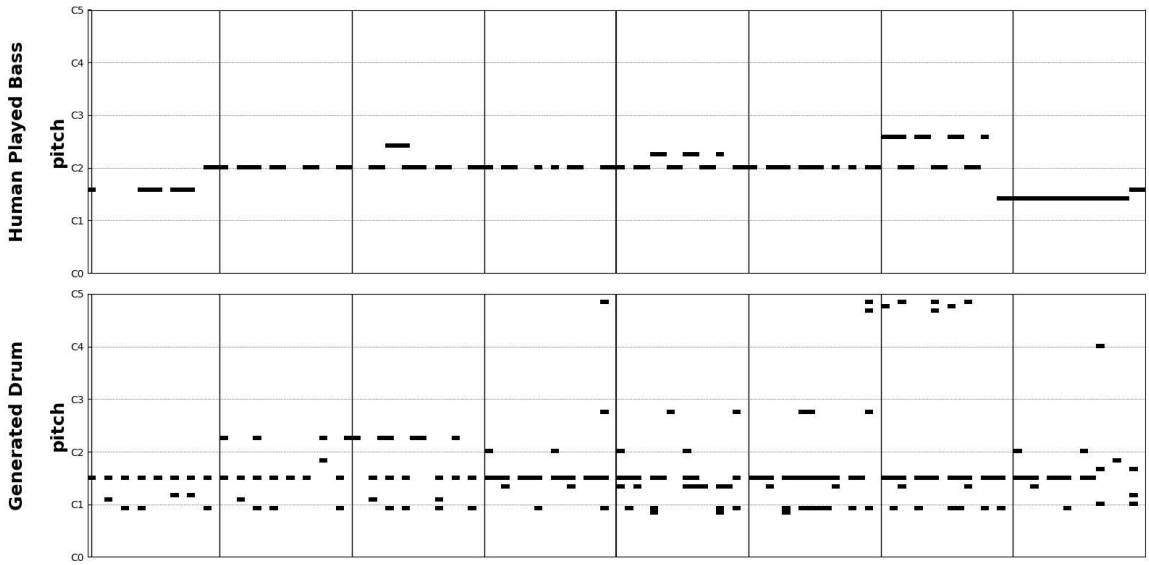

Figure 5.5: Real-time improvisation sample of a human bassist with DG.

In Figure 5.5, we observe the piano roll representation of the drum track generated by DG in real-time, responding to the live improvisation of the human bass player. Due to processing time constraints discussed in Section 4.3, there is a slight delay in the drum generation relative to the bass input. Despite this latency, the generated drum patterns seem to effectively complement the bass line, producing music that sounds coherent and musically acceptable. The drum hits are placed in a manner that aligns with the bass notes, resulting in a drum track that, in our subjective judgment, fits well with the bassist's performance even with real-time processing limitations. The

generated music does not resemble noise or disjointed notes; instead, it sounds like a proper drum accompaniment appropriate for the bass track and genre. We acknowledge that musical interpretation is subjective, and we encourage readers to listen to the provided audio samples available at `arash-sadeghi.github.io/MusicAiPage/` to form their own judgments.

From this live session, we extracted the bass track and also generated the drum track offline, meaning that we converted the played bass track to a piano roll and fed it to DG at once. The generated drum track is illustrated in Figure 5.6.

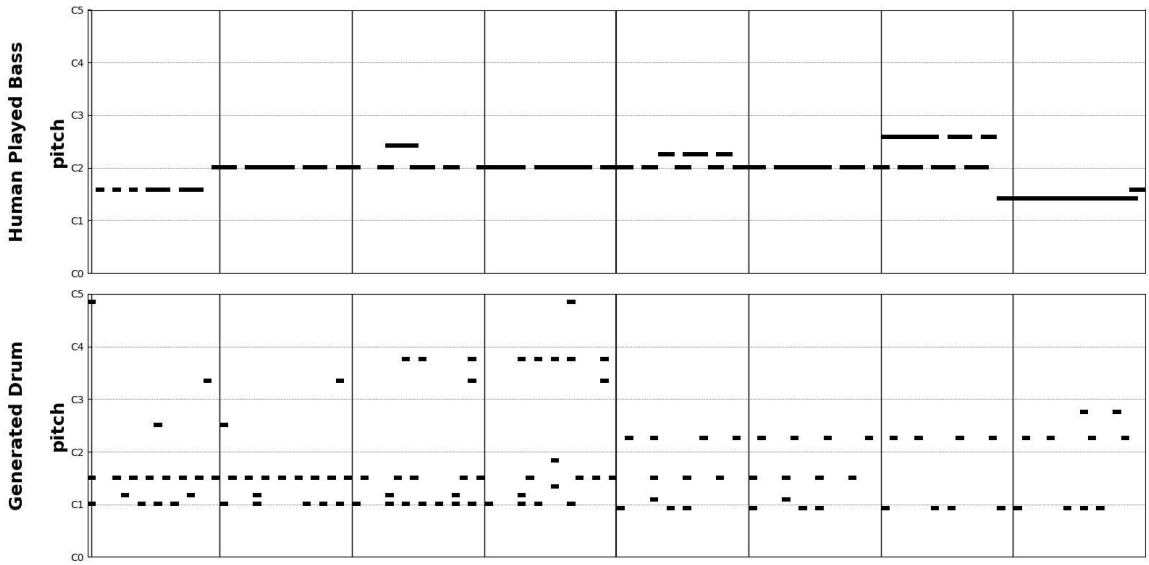

Figure 5.6: Offline improvisation sample of a human bassist with DG.

Figure 5.6 shows the piano roll of the drum track generated offline. In this scenario, DG has access to the full bass track without the constraints of real-time processing.

As a result, the drum hits in the offline generation are aligned with the bass notes without any delay. The generated drum hits are distributed across different percussion instruments, such as kick, snare, and hi-hat, which are essential components in Pop-Rock music. The music produced seems cohesive and fits well within the intended

genre, enhancing the overall musical piece. Again, we invite readers to listen to the audio samples on our website to make their own evaluations, understanding that perceptions of music are inherently subjective.

We then fed the generated drum track to VA, and the result is shown in Figure 5.7.

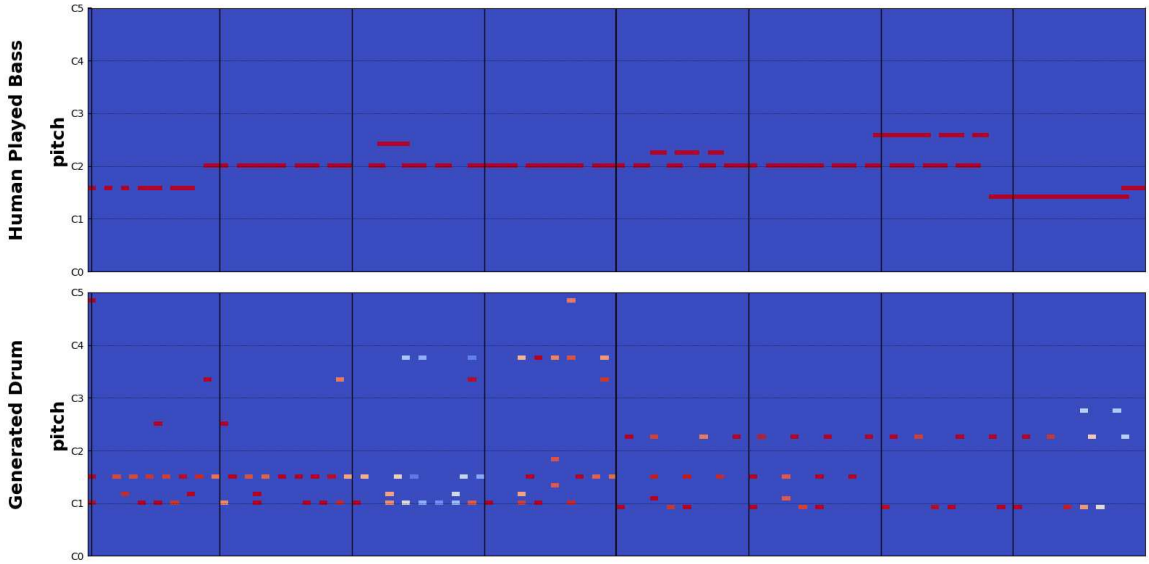

Figure 5.7: Velocity assigned to offline improvisation of DG with a human bassist. Red notes indicate higher velocity values and blue notes indicate velocity values close to zero.

In Figure 5.7, the assigned velocities add a dynamic layer to the drum track, reflecting variations in how forcefully each drum is struck. The velocity variations contribute to the expressiveness and realism of the drum accompaniment. For instance, stronger beats such as downbeats may have higher velocities (warmer shades), emphasizing their rhythmic importance, while lighter beats may have lower velocities (colder shades). This dynamic variation is characteristic of human drumming, where emphasis and articulation play crucial roles in musical expression.

These figures collectively demonstrate that our model is capable of generating

drum tracks that are rhythmically and dynamically aligned with a human-played bass track in both real-time and offline settings. The real-time generation shows DG's ability to interactively respond to live input, producing drum patterns that complement the bass improvisation. The offline generation allows DG to generate drum notes at precisely the right times without disruption of time delays. The velocity assignment by VA enhances the expressiveness of the drum tracks, introducing dynamic nuances that are essential for realistic and engaging music production.

Chapter 6

Conclusion

In this research, we proposed a novel drum accompaniment system designed to generate expressive drum tracks for human bass players, both in real-time and offline. The primary focus was on generating drum tracks based on played bass tracks, but the framework could be extended to accompany any human instrument player by training the CGAN model with different instruments. Our drum generation process incorporated human-played bass input and accepted a desired genre for generating the drum track, adding flexibility and adaptability to the system.

Our experiment of generating pitch and velocity in a single network showed a larger deviation from the training dataset's groove consistency, resulting in less coherent drum patterns and reinforcing our decision to separate pitch generation from velocity assignment.

Drum generation occurred in two phases: pitch generation, handled by the Drum Generator (DG), and velocity assignment, handled by the Velocity Assigner (VA). The purpose of VA was to enhance the expressiveness of the generated drum tracks by assigning appropriate velocity values, simulating the dynamic intensity of a human

drummer. This approach aimed to make the generated drum tracks sound more human-like and musically engaging.

We evaluated our models using objective metrics to provide insights into their learning processes and performance. However, we acknowledge the challenge of defining performance metrics for music generation, as music is inherently subjective and emotional. Due to time constraints, we were unable to perform a thorough subjective evaluation with organized test subjects. Instead, we provided improvisation samples, including piano roll visualizations and audio recordings, to give a sense of how the generated drums sound. These samples suggest that the generated drum tracks effectively accompany the bass tracks, producing music that sounds coherent and appropriate for the intended genre.

The improvisation samples demonstrated that our model can generate drum patterns that seem to correspond with the bass notes. In the offline setting, where processing delays are not a factor, the drum hits appear to align well with the bass input, resulting in cohesive and well-timed accompaniments. In the real-time setting, despite inherent processing delays, the generated drum patterns seem to complement the live bass improvisation, providing an interactive and engaging experience.

The velocity assignment by VA introduced dynamic nuances that contribute to the expressiveness and realism of the drum tracks. The variations in velocity add a layer of human-like dynamics, enhancing the overall musicality of the generated accompaniments.

In conclusion, our system demonstrates the potential for AI models to generate expressive and contextually appropriate drum accompaniments for human musicians. While our evaluations were limited by time constraints and the subjective nature of music assessment, the provided samples and analyses suggest that our approach is

effective in producing musically coherent, expressive and engaging drum tracks. Moreover, by offering an intelligent, responsive AI drummer, our work directly addresses the need for accessible improvisation practice highlighted in our motivation—enabling young musicians to develop ensemble skills without requiring a full band [17].

6.1 Future Work

Potential improvements and future directions for this work include:

- The songs used to train the DG model were based on the LPD-cleansed dataset, whose songs have constant tempo and no rhythm changes. Alternative CGAN networks, such as transformers, along with different music representations, could be used to reduce the model's dependency on these constraints, since time signatures and tempo changes are common in music.

- In real-time improvisation with DG, drum generation always lags behind the played bass track. This could be addressed by incorporating bass track estimation dynamics to predict the bass track that will be played in the next time frame and generate the drum track accordingly.

- The VA network's validation loss oscillated through different epochs. This could be addressed by adjusting network parameters such as the learning rate or incorporating different regularization methods such as dropout.

- For gaining subjective insight to generated music, a subjective evaluation could be proposed by designing listening tests with human participants to assess the musicality and expressiveness of the generated drum accompaniments.

Bibliography

[1] L. Angioloni, T. Borghuis, L. Brusci, P. Frasconi, et al. Conlon: A pseudo-song generator based on a new pianoroll, wasserstein autoencoders, and optimal interpolations. In *Proceedings of the 21st International Society for Music Information Retrieval Conference*, pages 876–883. International Society for Music Information Retrieval, 2020.

[2] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer. MIDI-VAE: Modeling dynamics and instrumentation of music with applications to style transfer. *arXiv preprint arXiv:1809.07600*, 2018.

[3] K. Choi, C. Hawthorne, I. Simon, M. Dinculescu, and J. Engel. Encoding musical style with transformer autoencoders. In *International Conference on Machine Learning*, pages 1899–1908. PMLR, 2020.

[4] K. L. Chung. Markov chains. *Springer-Verlag, New York*, 1967.

[5] J. R. Covach and A. Flory. *What's that sound?: An Introduction to Rock and its History*. WW Norton & Company New York., 2006.

[6] J. Cruz. Deep Learning vs Markov Model in Music Generation. Honors college thesis, Pace University, 2019.

[7] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.04805, 2018.

[8] F. Ding and Y. Cui. MuseFlow: Music accompaniment generation based on flow. *Applied Intelligence*, 53(20):23029–23038, 2023.

[9] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[10] N. Fradet, J.-P. Briot, F. Chhel, A. El Fallah Seghrouchni, and N. Gutowski. MidiTok: A Python package for MIDI file tokenization. In *Extended Abstracts*

for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference, 2021.

[11] N. Fradet, J.-P. Briot, F. Chhel, A. E. F. Seghrouchni, and N. Gutowski. MidiTok: A python package for MIDI file tokenization. *arXiv preprint arXiv:2310.17202*, 2023.

[12] J. Gillick, A. Roberts, J. Engel, D. Eck, and D. Bamman. Learning to groove with inverse sequence transformations. In *International Conference on Machine Learning (ICML)*, 2019.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014.

[14] G. Hadjeres, F. Pachet, and F. Nielsen. Deepbach: a steerable model for Bach chorales generation. In *International Conference on Machine Learning*, pages 1362–1371. PMLR, 2017.

[15] Y.-S. Huang and Y.-H. Yang. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1180–1188, 2020.

[16] S. Ji, J. Luo, and X. Yang. A comprehensive survey on deep music generation: Multi-level representations, algorithms, evaluations, and future directions. *arXiv preprint arXiv:2011.06801*, 2020.

[17] N. Kageyama. Why improvisation should be part of every young musician's training. `https://bulletproofmusician.com/why-improvisation-should-be-part-of-every-young-musicians-training/`, 2024. Accessed: 2025-03-31.

[18] K. Kritsis, T. Kylafi, M. Kaliakatsos-Papakostas, A. Pikrakis, and V. Katsouros. On the adaptability of recurrent neural networks for real-time jazz improvisation accompaniment. *Frontiers in Artificial Intelligence*, 3:508727, 2021.

[19] W. Liu. Literature survey of multi-track music generation model based on generative confrontation network in intelligent composition. *The Journal of Supercomputing*, 79(6):6560–6582, 2023.

[20] L. Metz, S. Chintala, and A. Radford. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations 2016*, 2016.

[21] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[22] O. Mogren. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

[23] A. Muhamed, L. Li, X. Shi, S. Yaddanapudi, W. Chi, R. Suresh, Z. Lipton, and A. Smola. Transformer-GAN: Symbolic music generation using a learned loss. In *NeurIPS 2020 Workshop on Machine Learning for Creativity and Design 4.0*, 2020.

[24] J. Nistal, S. Lattner, and G. Richard. Drumgan: Synthesis of drum sounds with timbral feature conditioning using generative adversarial networks. *arXiv preprint arXiv:2008.12073*, 2020.

[25] G. Papadopoulos and G. Wiggins. AI methods for algorithmic composition: A survey, a critical view and future prospects. In *AISB symposium on Musical Creativity*, volume 124, pages 110–117. Edinburgh, UK, 1999.

[26] T. Peer. *Comparing neural network architectures for drum pattern generation.* PhD thesis, Technische Universität Wien, 2023.

[27] C. Raphael. A Bayesian network for real-time musical accompaniment. *Advances in Neural Information Processing Systems*, 14, 2001.

[28] C. Raphael. Orchestra in a box: A system for real-time musical accompaniment. In *IJCAI Workshop Program APP-5*, pages 5–10. Citeseer, 2003.

[29] C. Raphael. Demonstration of Music Plus One-A Real-Time System for Automatic Orchestral Accompaniment. In *AAAI*, pages 1951–1952, 2006.

[30] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu. Popmag: Pop music accompaniment generation. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1198–1206, 2020.

[31] A. Roberts, J. Engel, C. Hawthorne, I. Simon, E. Waite, S. Oore, N. Jaques, C. Resnick, and D. Eck. Interactive musical improvisation with magenta. *Neural Information Processing Systems (NeurIPS)*, 2016.

[32] R. K. H. Toh and A. Sourin. Generation of music with dynamics using deep convolutional generative adversarial network. In *2021 International Conference on Cyberworlds (CW)*, pages 137–140. IEEE, 2021.

[33] E. Waite, D. Eck, A. Roberts, and D. Abolafia. Project Magenta: Generating long-term structure in songs and stories. *Online] https://magenta. tensorflow. org/2016/07/15/lookback-rnn-attention-rnn*, 2016.

[34] Z. Wang, K. Zhang, Y. Wang, C. Zhang, Q. Liang, P. Yu, Y. Feng, W. Liu, Y. Wang, Y. Bao, et al. Songdriver: Real-time music accompaniment generation without logical latency nor exposure bias. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 1057–1067, 2022.

[35] S.-L. Wu and Y.-H. Yang. The jazz transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures. *arXiv preprint arXiv:2008.01307*, 2020.

[36] I. P. Yamshchikov and A. Tikhonov. Music generation with variational recurrent autoencoder supported by history. *SN Applied Sciences*, 2(12):1937, 2020.

[37] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang. MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. *arXiv preprint arXiv:1703.10847*, 2017.

[38] H. Zhao, W. Su, and X. Zhang. Research on automatic music generation with multi-track based on melody constraints. In *International Conference on Computer, Artificial Intelligence, and Control Engineering (CAICE 2022)*, volume 12288, pages 352–361. SPIE, 2022.