UNIVERSITY
OF OSLO

Master's thesis

Interactive Music Generation Using Machine Learning and Multi-Agent Systems

Investigating the Effects of a Band-Like Multi-Agent Architecture on an Interactive Music Generation System That Uses Machine Learning

Trym Bø

Informatics: Robotics and Intelligent Systems
60 ECTS study points

Department of Informatics
Faculty of Mathematics and Natural Sciences

Spring 2024

Trym Bø

Interactive Music Generation Using Machine Learning and Multi-Agent Systems

Investigating the Effects of a Band-Like Multi-Agent Architecture on an Interactive Music Generation System That Uses Machine Learning

Supervisors:
Kyrre Glette, Çağrı Erdem

Abstract

The field of music generation has seen significant advancements through Machine Learning (ML), and more specifically Deep Neural Networks (DNNs). However, these technologies often introduce several challenges, from complex building and training processes to high computational demands. Consequently, researchers have suggested the potential for developing a band-like multi-agent architecture for music generation and human-computer interaction.

To address this, this thesis introduces ML-jam, a novel interactive music generation system that integrates ML with Multi Agent System (MAS) to create a dynamic and interactive music generation experience. ML-jam comprises five artificial agents: drum, bass, chord, melody, and harmony. In addition, a human agent directs these artificial agents through a digital User Interface (UI), facilitating creative human-computer interaction.

The core objective of ML-jam is to evaluate how a band-like multi-agent architecture can influence an interactive generative music system. This is achieved through a series of objective and subjective experiments that, via quantitative and qualitative analyses, focus on the modular multi-agent design, inter-agent communications, and performer-centered user experience.

The experiments demonstrate that this approach can achieve higher performance with simple models compared to larger traditional monolithic models, thanks to its inherent multi-agent modularity. They also show that collective intelligence emerges in the system through inter-agent communication. Furthermore, user evaluations indicate that this design promotes a more creative and interactive experience than standard end-to-end models. These insights contribute to the development of more sophisticated interactive music systems that emphasize creativity and human-computer co-creation.

Contents

List of Figures

List of Tables

List of Acronyms

AI Artificial Intelligence xiii, 1–5, 9, 22, 26, 27, 30, 32, 82, 84, 89, 90

ANOVA Analysis Of Variance 55, 56

API Application Programming Interface 27

BDI Belief-Desire-Intention 21

bpm beat per minute 34

DAW Digital Audio Workstation 13, 27, 30, 84

DNN Deep Neural Network i, 1, 2, 16

FC Fully-Connected 9, 48–50, 52–54, 122

FDR False Discovery Rate 56

GA Genetic Algorithm 22

GAN Generative Adversarial Networks 19

KL Kullback-Leibler 24

LLM Large Language Model 6, 20

LSTM Long Short Term Memory vii, 6–12, 17, 48, 50, 52, 53, 61, 122, 123, 129

MAS Multi Agent System i, 1–6, 16, 19–22, 39, 61, 64, 65, 84, 90–92, 95, 96

MIDI Musical Instrument Digital Interface 12–16, 26, 27, 30, 39–41, 125

ML Machine Learning i, 5, 6, 13, 15, 22, 26, 84, 89, 93, 94

MLLM Multimodal Large Language Model 6

MLP MultiLayer Perceptron 19

MOS Mean Option Score 72

NN Neural Network 2, 3, 5, 6, 11, 13, 14, 16, 18, 21, 28–31, 46–48, 69, 93, 121, 122

PDF Probability Distribution Function 24

PSO Particle Swarm Optimization 21

ReLU Rectified Linear Unit 54

REMI REvamped MIdi-derived events 15

RL Reinforcement Learning 22

RNN Recurrent Neural Network 6, 7, 10

UI User Interface i, 33, 34, 57

AI Declaration

In this scientific work, generative Artificial Intelligence (AI) has been used. All data and personal information has been handled in accordance with the regulations of the University of Oslo, and I, as the author of this document, take full responsibility for its content, claims, and references. An overview of the use of generative AI is provided below.

- ChatGPT-4 [1], developed by OpenAI has been used to help reformulate sentences, when I as the author struggled with achieving the desired articulation. The generated output was thoroughly reviewed and possibly altered before being used in the thesis.

  ChatGPT-4 [1] was also used to help write the code. The applications range from syntax help, debugging, and the use of unknown libraries or frameworks.

- Writefull [2] is a writing assistance tool using AI, similar to Grammarly [3]. It primarily helps with more advanced error highlighting, helping with grammar and sentence structure.

- Github Co-pilot [4] is an AI developer tool, which has been used during the development of the codebase.

Acknowledgements

There are many people to whom I owe gratitude for the completion of this master's thesis. Firstly, I am deeply grateful to my supervisor, Kyrre, who allowed me the freedom to pursue the thesis I wanted while providing invaluable guidance to steer me in the right direction. I am also grateful to you, Çağrı, for your expertise and creativity, both throughout this thesis and during your course. Both your continuous support and extensive knowledge have significantly shaped this work and it would not have been the same without you.

I would also like to thank everyone who took the time to participate in the survey and expert interviews. Your contributions provided the essential data needed for this thesis. Thank you!

Although this thesis has been a lot of hard work, I was fortunate enough to travel a lot, armed with my laptop in my backpack. This experience provided both motivation and much needed breaks. I would like to thank those who traveled with me, making this year more than just an academic journey.

To my friends from Navet, IF1, and BrewFI, thank you for all the good times. Finishing this thesis will be bittersweet because of you.

Lastly, to my family and my close ones who patiently listened to countless hours of generated music, from early stage noise to full-blown performances, I know you were tired, but thank you for always supporting me.

Chapter 1

Introduction

Music has consistently served as a universal medium, resonating in diverse cultures and epochs, profoundly influencing emotional states, interpersonal relationships, and cognitive processes [5, 6]. Historically, the production of music was an exclusively human endeavor. However, with the advent of AI, there is now the ability of machines to comprehend and create music, thus forging integrative links between the disciplines of musicology, computer science, and human-machine interaction.

In recent years, the field of AI music generation research has focused on end-to-end systems using ever-expanding DNN. Although the results are impressive [7–10], such large and complex network architectures require powerful machines, large amounts of data, and extensive knowledge to build [11, 12]. It is also challenging for humans to create genuine and creative music when they collaborate with such end-to-end systems in the music creation process [13].

However, some research has highlighted the potential of using different strategies. Dadman et al. [14] claim that there is undiscovered potential in using aMAS architecture. According to Dadman and colleagues, these systems offer a promising alternative by addressing specific challenges inherent to music generation tasks. Unlike traditional end-to-end models aimed primarily at commercial and entertainment applications, MAS architectures could also support artists through assisted composition systems. This is also supported by Huang et al. [13] who found that when humans interacted with generative musical systems to co-create music, they preferred to use smaller modular designs instead of larger end-to-end systems due to increased control and more creative freedom. Dadman et al. therefore argue that using MAS not only mitigates the limitations associated with large-scale models but also fosters a more collaborative and transparent environment for creative expression. To achieve this, they suggest the following:

> "By utilizing MAS architecture, we can simplify the representation of music by concentrating on one instrument at a time, where different agents can be assigned to a specific instrument. This is analogous to how musicians work in a band." [14]

Using MAS and modular design like this, by breaking up large tasks into multiple subtasks, is a common method to increase performance [15–19], and is even the foundation of the internet itself [20]. It is also suggested that the human brain is not a single agent, but rather an environment of several simple interactive agents doing smaller tasks to achieve a greater goal [21]. Naturally, MAS has also found its way into music generation, and band-like multi-agent architecture, as suggested by Dadman et al. is not a new concept [22–24], but since DNN has gained massive popularity in sequence processing, following great advances in the field of generative AI [1, 25, 26], it has been the main research topic. This has created a gap in the literature, where little new research has been done in the field of computer-generated music using MAS, mostly missing the integration of MAS with modern Neural Networks (NNs). Therefore, this thesis aims to take a step back from creating large and complex DNNs and instead look at the possibilities and advantages that lie in creating several agents that use simpler NNs and rule-based systems to create symbolic music, where the complexity will instead emerge from inter-agent communication.

In particular, this thesis will explore three main aspects of the MAS architecture: (1) naturally occurring modular design, (2) inter-agent communication, and (3) the effect this has on interactivity. By using a multi-agent architecture to create a modular design, Dadman et al. [14] suggest that an agent can concentrate on one instrument at a time and, by that, alleviate the shortcomings and challenges of the music generation tasks. Through communication, collective intelligence can emerge, forming a band-like system in which agents can create music together.

To do so, I have created ML-jam, a music generation system that includes five artificial agents – drum, bass, chord, melody, and harmony – along with a human agent. These artificial agents, employing NNs and rule-based approaches, work together using concepts from MAS, and through that try to generate aesthetically pleasing music. In addition, an interface has been developed so that the human agent can interact with the system effectively. This interface allows users to influence the music's characteristics, serving both as a creative tool for composing music and as a support tool for artists during the creative process and in practice situations. Instead of being an end-to-end generational system, it is built as an interactive system, where the user's live actions affect the musical output of the agents.

1.1 Research Questions

Inspired by the suggestion of Dadman et al. [14], this thesis investigates the unexplored potential of using MAS together with modern NNs to create an interactive musical system. Through the creation of the system and the evaluation of its potential benefits, this thesis aims to contribute to the field of interactive AI generated music by answering the research question:

> How can a band-like multi-agent architecture affect an interactive generative music system?

This question possesses a broad scope; thus, to refine the experimental focus, four sub-questions have been formulated:

- **RSQ1:** How does the modularity natural to a multi-agent architecture compare against a monolithic architecture in terms of performance and usability?
- **RSQ2:** How does communication between autonomous agents affect the musical output of individual agents?
- **RSQ3:** How does communication between autonomous agents affect the musical output of the system as a whole?
- **RSQ4:** How is the user experience of using an interactive musical system with a band-like multi-agent architecture?

Answering the questions requires a broad investigation, which is possible through multiple experiments using a wide range of evaluation methodologies. ML-jam is objectively and subjectively assessed to explore how the multi-agent architecture impacts system performance and the human interaction experience. Particular attention is paid to the influence of its modular design and communication. A comprehensive evaluation, using objective and subjective methodologies, is important for the evaluation of generational musical systems, as it is difficult to assess how well such a system performs purely based on a single evaluation methodology [27].

1.2 Contributions

This thesis has contributed to the field of interactive AI through the development of ML-jam, an open-source generative music system using a band-like multi-agent architecture. ML-jam consists of five artificial agents, some of which introduce new and unique network designs tailored for a multi-agent architecture. The contribution is further extended by conducting five experiments to address the research questions. This reveals how the modularity inherent to multi-agent architecture can be beneficial for system performance,

as well as possibly increasing control and enabling creativity during human interaction. The experiments also illustrate how effective inter-agent communication can lead to the emergence of collective intelligence. Furthermore, it explores strategies to mitigate the impact of noisy communication between agents.

In addition, the thesis contributes through its openness and accessibility. All the code used to create ML-jam is openly available [28], the music generated by the system for the listening test, which was chosen randomly, is accessible [29], and a video demonstrating the use of ML-jam is provided [30].

1.3 Outline

This thesis is comprised of five chapters:

- **Chapter 1: Introduction** - This chapter outlines the challenges encountered in AI-generated music. It sets the stage by describing the field's current state, identifying gaps in existing research, and highlighting the innovation potential.
- **Chapter 2: Background and Related Works** - This chapter provides a comprehensive overview of the related literature and establishes a foundational context to the thesis. It examines previous studies, methodologies, and results in the field, focusing on single and multi-instrument music generation and interactive systems.
- **Chapter 3: Methods** - In this chapter, the methods employed in the research are detailed. It covers the theoretical foundation, and gives insight into the implementation of ML-jam, with the the overarching MAS system design, as well as agent specific data representation and network designs.
- **Chapter 4: Experiments and Results** - In this chapter, features of ML-jam are tested. It focuses on the use of a multi-agent architecture with a modular design, communication between agents, and human interaction. The experiments will be targeted at both individual agents and the system as a whole, and the findings of each experiment will be presented and discussed individually.
- **Chapter 5: Discussion** - The final chapter will discuss the results of the previous chapter with a focus on the overarching findings. It will also address any limitations of the study, and suggest directions for future research, concluding the thesis with a reflection on the findings and its contributions to the understanding and development of interactive AI music systems.

Chapter 2

Background

The next chapter will present a review of the scientific literature related to the topics underlying this research. Firstly, it will provide a brief overview of the historical context of music generation methods that preceded this research. Afterward, an overview of the NNs relevant for this project is presented, before several sections about music generation. As this thesis aims to create multi-instrumental music based on several instrument-specific agents using different techniques, these sections will go into detail on multiple branches of the music generation field. This will include research on music representation, datasets, single- and multi-instrument generation, MAS, and interactive music systems. Lastly, this chapter will present research on the process of evaluating such a project.

2.1 History of Music Generation

Before immersing ourselves in the intricacies of AI-driven music creation and multi-agent orchestration, it is important to understand the landscape of generated music that precedes it. The quest for automated music generation dates far beyond the introduction of ML. Humans' natural interest in music and automation can perhaps be seen as the cause of the long-lasting interest in this topic. Despite the digital era introducing advanced techniques for automatic music creation, ways of generating music predate digital computers. In the 3rd century BCE, mechanical water-driven musical automata were made by famous mathematicians such as Archimedes and Apollonius of Perga [31]. In the late 18th century, musical dice games were published, making it possible for people without musical experience to compose pieces, using prefabricated music and chance in the form of dice [32].

When the computer was invented 200 years later, Lejaren Hiller and Leonard Isaacson [33] made the composition *Illiac Suite* in 1957, widely recognized as the first computer-generated music composition. It was composed using a high-speed digital computer, ILLIAC I at the University of Illinois. The piece was made using mathematical and probabilistic rules.

Later followed several decades with computer-generated music through such mathematical rule-based systems [34, 35]. After advancements in ML in the 1980s and the introduction of Recurrent Neural Network (RNN) by Rumelhart et al. [36] in 1986, a new method of sequence generation was possible. Several years later, Todd was one of the first to use a small RNN to create computer-generated music [37]. This is considered the first use of NNs to generate music. Although NNs are now used to create music [38], algorithmic composition was still popular and used by the likes of David Cope [39]. Following the introduction of the LSTM in 1997 [40], they became very convenient because of their memory component. They could handle temporal information, solving the shortcomings of vanilla RNNs. There are several influential models using RNNs and LSTMs [41–45].

Inspired by the works of MAS pioneers like Wooldridge and Kennedy [46, 47] from the 1990s, MAS found its way into music generation through popular works in the early 2000s, [48, 49], and often through musical swarms [50, 51].

The current state-of-the-art method for sequence generation is attention-based transformers [52]. Recently, these have captured significant public interest with the introduction of image generation diffusion models utilizing transformers, exemplified by DALL-E 2 and Midjourney [26, 53], and particularly through Large Language Models (LLMs) such as GPT-3 [54], and even Multimodal Large Language Models (MLLMs) like GPT-4 and Gemini [1, 25]. Given their proven success in text and image generation, a large field of study in recent years has been music generation using transformers. This has resulted in some promising results in both waveform-based music generation [7–9, 55] and symbolic music generation transformers [56–58].

2.2 Neural Networks

Many varieties of NNs have been used to make ML-generated music throughout the years, but they are often designed around the RNN, LSTM, and transformer architectures. These networks are designed to recognize patterns in sequences of data such as text, genomes, images, or numerical time-series data. They are therefore perfect for use in music generation. When processing sequences of data, they are designed to keep track of past information. This section will explain the fundamentals behind these three main network types, focusing on the LSTM and transformer, as they are the most relevant to this thesis.

2.2.1 RNN

Introduced by Rumelhart in 1986 [36], the RNN excelled in sequence processing by including an internal memory of state that allows it to better process sequences of inputs. The RNN cells are stacked in sequence, and the hidden state is passed on from cell to cell. The RNN cell is updated on the basis of the current input and the previous hidden state. This cell can be seen in Figure 2.1.

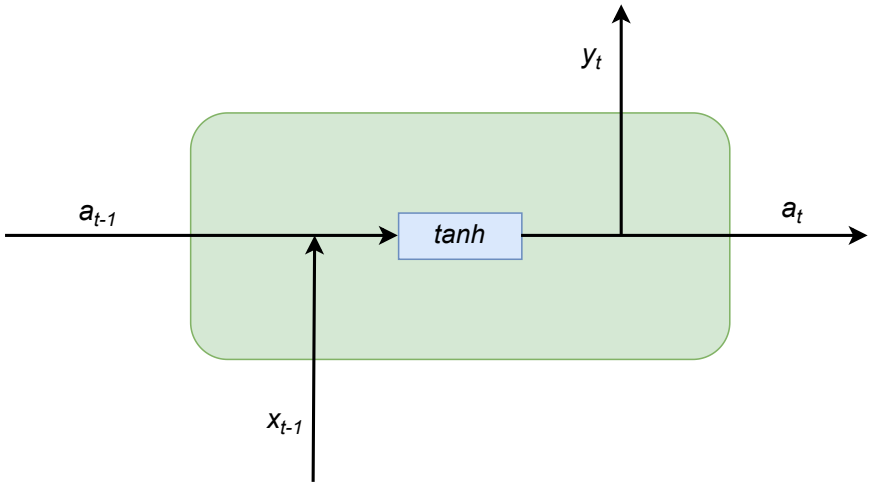


Figure 2.1: The architecture of a single RNN cell. a is the hidden state, x is the input, y is the output, and *tanh* is the hidden layer with tanh activation function.

2.2.2 LSTM

One limitation of RNNs is their struggle with long sequences where gradients tend to shrink during backpropagation, diminishing as they travel backward through the network's layers. This causes the earlier layers to remain relatively unchanged, and learning becomes difficult [59].

This was improved by the LSTM when Hochreiter and Schmidhuber introduced it in 1997 [40]. LSTMs allow for better preservation of long-term dependencies in sequential data. Rather than simply transmitting the hidden state as the RNN does, the LSTM additionally sends information through what is known as the cell state. At each time step, the cell state is altered by several gates, each responsible for regulating the flow of information. When the LSTM is fed new input, it is the job of the gate components to regulate how much of that information should affect the cell state. This regulation of information is what combats the vanishing gradient problem. The complete architecture of a single LSTM cell is shown in Figure 2.2.

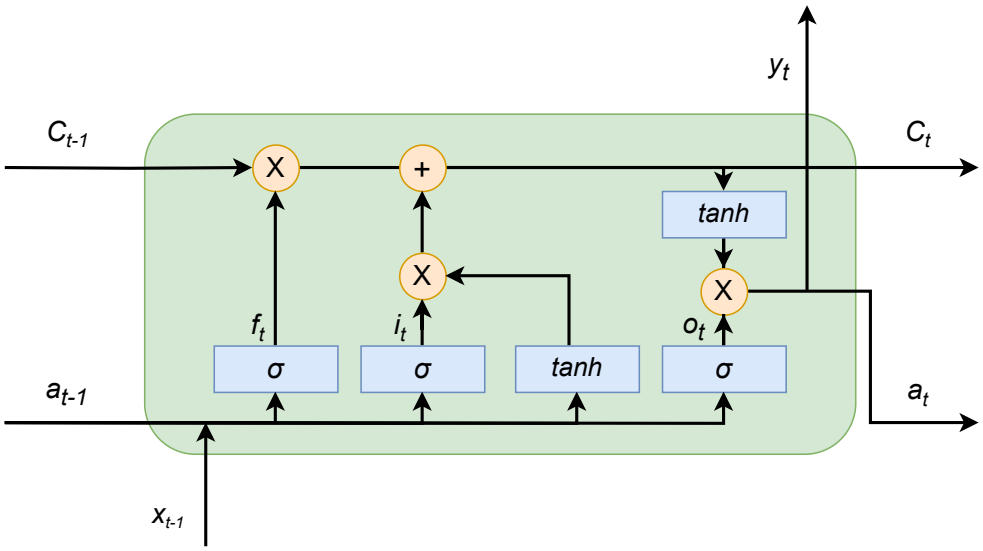


Figure 2.2: The architecture of a single LSTM cell. c is the cell state, a is the hidden state, x is the input, y is the output, σ is the sigmoid activation function, $tanh$ is the hidden layers with tanh activation function. f, i, and o are the forget, input, and output gates.

LSTMs includes these three gates:

- Input gate: Adjust how much information from the input should be passed to the cell state.
- Forget gate: Tells how much information from the cell state should be forgotten.
- Output gate: The final activation that provides the output of that cell.

The output of all the gates is passed through a sigmoid function. By doing this, the gates output a number between 0 and 1. If the gate outputs 0, all the information is blocked. If the gate outputs 1, all the information is passed through. The equations for the three gates are as follows:

$$
\begin{aligned}
inputgate_t &= \sigma(w_i[h_{t-1}, x_t] + b_i) \\
forgetgate_t &= \sigma(w_f[h_{t-1}, x_t] + b_f) \\
outputgate_t &= \sigma(w_o[h_{t-1}, x_t] + b_o)
\end{aligned}
$$

- $w \rightarrow$ weight for the respective gates neurons.
- $h_{t-1} \rightarrow$ output of the previous LSTM block (at timestamp $t-1$).
- $x_t \rightarrow$ input at current timestamp.
- $b \rightarrow$ biases for the respective gates.
- $\sigma \rightarrow$ the sigmoid activation function

2.2.3 Transformer

For a long time, LSTMs were considered state of the art for AI sequence generation. Still, it struggled with large sequences, and in music generation it is important to keep track of longer sequences to create choerence in the musical output over time [60]. Thus, following the introduction of the transformer by Vaswani et al. [52] in 2017, researchers focusing on music generation quickly embraced this technology, attracted by its enhanced memory capabilities enabled by self-attention. Transformers are applied in a broad spectrum of generative tasks, ranging from music [7], to text [1], and images [61].

As transformers are very complex, this section will only explain the fundamentals of the architecture enough to understand how the transformer excels in music generation. For further details, the full paper on Transformers [52] can be consulted.

The Transformer uses an encoder-decoder structure, where the encoder converts an input sequence into a latent representation through an input embedding. It subjects each of these latent vectors to self-attention, enabling interactions among all input sequence elements, regardless of their position. This means that elements can relate to others that come before and after them in the sequence. This interaction generates three vectors: query, key, and value, as illustrated in Equation 2.1. Subsequently, attention scores are standardized according to Equation 2.2, producing a specific attention score for the current input. This is passed through a Fully-Connected (FC) network.

$$\begin{gathered} Q_i = x_i W^Q \\ K_i = x_i W^K \\ V_i = x_i W^V \\ \text{Where } W^Q, W^K, W^V \text{ are trainable weights} \end{gathered} \tag{2.1}$$

$$\begin{gathered} \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \\ d_k \rightarrow \text{the dimension of the keys} \end{gathered} \tag{2.2}$$

The decoder receives this information produced by the encoder. In addition, it embeds and uses self-attention on the outputs generated so far. The output of the attention layers is passed through a FC network before a softmax layer creates the output probabilities. A simplified illustration of a transformer model is shown in Figure 2.3.

Using attention mechanisms, transformers can concentrate on specific segments of significance in the input sequence. Unlike LSTM networks that require sequential processing, the self-attention mechanism of transformers processes the entire input sequence simultaneously. This feature enables every component of the sequence to potentially influence every other component, regardless of their positions, including

future and past elements. This characteristic is particularly advantageous in music generation, where observing the entire input sequence at each step can give the network a greater understanding of the overall structure.

Furthermore, each part of the musical sequence receives equal consideration, overcoming the limitations of RNNs and LSTMs, which tend to lose information about earlier parts of the sequence gradually. Consequently, transformers offer improved opportunities to maintain long-term structure.

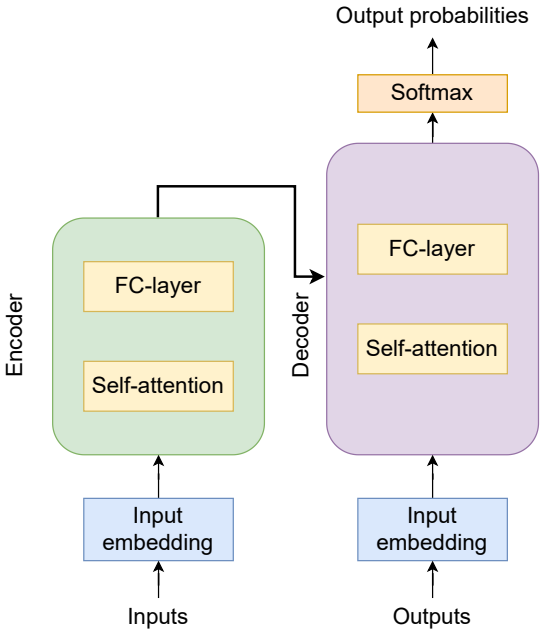


Figure 2.3: A simplified version of a transformer architecture.

2.2.4 Networks Modifications for Improved Structure

Both LSTMs and transformers can perform effectively with minimal adjustments, yet numerous innovative architectures have been developed to further enhance their capabilities. These enhancements often focus on refining the structure or expanding the memory capacity, thus creating better musical compositions. This project incorporates several such network modifications. The following is a summary of some notable variations.

Hierarchical Networks

Hierarchical networks are used in sequence generation to help the network focus on small segments and larger structures individually. This is important in sequence processing, especially in fields where humans can easily detect a lack of structure. Therefore, hierarchical networks are used in natural language processing [62], audio generation [63], as well as in symbolic music generation [64–67].

These networks are typically built by network cells that are structured in several layers, where certain layers are responsible for certain parts of the data processing. Each layer can then specialize in capturing one type of structure. These layers then share information, and the end product is a network that better understands the data. A simple hierarchical architecture can look like Figure 2.4.

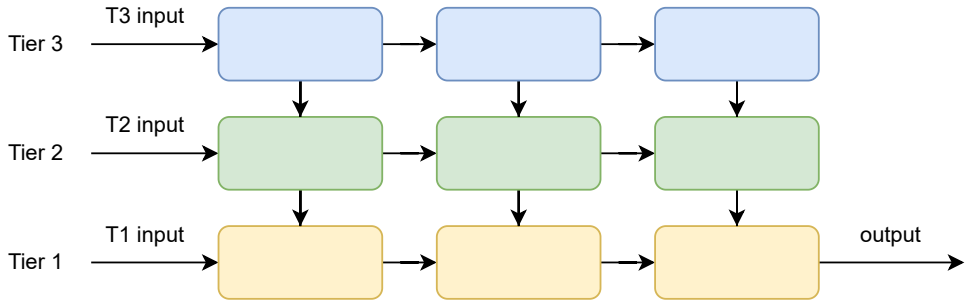


Figure 2.4: A simplified sketch of an arbitrary hierarchical architecture. Each cell passes information to the subsequent cell within the same tier and to the cell below. The final output is, therefore, a product of all cells, but since each tier has its unique input, it specializes in interpreting that specific type of information.

Similarly, Museformers [57] apply both fine- and coarse-grained attention mechanisms. Fine-grained attention targets specific, structurally critical parts, while coarse-grained attention covers the overall composition elements. Wei et al. [67] tackle this in a similar way using three networks to capture information on a note scale, bar scale, and track scale.

Bidirectional LSTM

Another network variation that has gained popularity in NN music generation is the bidirectional LSTM. This is a modification of an LSTM that to some degree allows LSTM to look at both previous and future inputs, similar to the transformer. As this is a trait that is beneficial in sequence processing, it has been used in multiple music generation systems [60, 66, 68, 69].

Transformer-XL

When it comes to transformers, many modified versions are available for benefit in image [70], text [71, 72] and music [57, 73] processing. Interestingly, one of the most popular transformer models for use in symbolic music generation has been the Transformer-XL [72], which originally was created for language processing. But because of the similarities in the way that music and language can be represented using tokens, it has also shown great results in music generation tasks [45, 58, 74]. A standard transformer has a fixed length learning dependency, based on the size of the length of the attention window (the maximum sequence length it can consider at one time). In addition, the Transformer-XL has hidden states that serve as memory. It passes the hidden state from sequence

to sequence a bit like an LSTM. This allows the transformer to model even longer dependencies than the native transformer.

Additionally, the Transformer-XL employs relative positional encodings instead of the standard absolute positional encodings. This change helps the model better understand the relative positioning between sequence elements. The use of relative positional encodings enables the model to be more aware of the distances between elements, which is particularly important for tasks that involve structured data like text and music, where the relationship between elements can be more important than their absolute positions in the sequence.

2.3 Music Representation

No matter how good the network choice is, good results require the data to be well represented, so the network can efficiently process the data. In the context of music generation, music can be represented in two principal forms: waveform-based or symbolic.

2.3.1 Waveform

Waveform-based representation captures music as digital audio waveforms, directly reflecting the raw auditory signals. This method aims to encapsulate the complete acoustic experience. Notable examples of music generation using waveform-based representation include WaveNet, MusicGen, and MusicLM [7, 8, 75]. Although this approach can yield high-quality audio output, it is often constrained by the need for extensive computational resources. This is attributed to the vast amounts of data that must be processed. Moreover, waveform-based methods are highly sensitive to small variations; even minor discrepancies in the generated waveforms can lead to significant auditory dissonance, an aspect to which human listeners are particularly attuned [76].

2.3.2 Symbolic

On the other hand, symbolic music representation encapsulates music at a more abstract level, focusing on the structural elements rather than the direct sound. Despite music's seemingly complex nature, symbolic representation simplifies music into elements like notes and duration, similar to how traditional sheet music operates. In the way sheet music acts as a guide for musicians to create sounds, in the world of computers, Musical Instrument Digital Interface (MIDI) does the same job. MIDI takes the place of sheet music by using digital symbols to represent musical attributes such as pitches and timings. Developed in the early 1980s, MIDI facilitates the communication of musical information, such as pitch, duration, and velocity, between electronic instruments and computers. This standard has become popular amongst music producers and performers,

with most modern music software, called Digital Audio Workstation (DAW), supporting the conversion of MIDI data into sound, much like musicians interpreting sheet music.

Symbolic music generation, exemplified by models such as DeepBach and Theme Transformer [43, 56], shifts focus from the complexities of sound wave manipulation to the construction of musical instructions. This approach enables a more straightforward manipulation of elements such as melody and rhythm, enhancing interpretability. However, the challenge with symbolic representation lies in its ability to convey the emotional depth and nuance found in human compositions. Instead of aspects such as timbre and feeling, symbolic generation emphasizes musical structure and intent.

For the purpose of developing agents capable of collaborative music creation, it not only allows for a more manageable computational load, facilitating simultaneous music creation by multiple agents, but it also simplifies the enforcing of musical elements such as key and tempo. This allows for quick and easy communication across different agents, in addition to the ability to easily apply post-generational changes to the music. Therefore, this method is chosen for ML-jam and serves as the topic of discussion in this chapter.

2.3.3 Representing MIDI for Machine Learning

Although MIDI is the standard in symbolic music representation, raw MIDI data is complex and noisy and does not perform well when used in ML. Therefore, feature extraction is commonly used to represent MIDI data in a way that facilitates pattern recognition by NNs. Feature extraction involves distilling raw MIDI data into attributes that effectively represent musical content, such as pitch, duration, and velocity of notes. This process is crucial in transforming the MIDI information into a structured form that NNs can interpret and learn more efficiently. This data pipeline from MIDI-data through an NN and converted to sound can be seen in Figure 2.5. This process is similar for most generations of symbolic music.

Finding the best way to represent the MIDI data for an NN is tricky. Quite a lot of research [45, 77–79] has been done on different ways of representing music that are complex enough to represent fine musical details but simple and structural enough for an ML algorithm to be able to understand and find patterns. The representation also varies according to the specific information that the network needs for the particular task it is designed to perform.

Regardless of the methods, a common way of representing symbolic music for ML is through the use of one-hot encoded vectors. This is a vector configured as a long list where each spot in the list represents a different musical feature (for instance, pitch) from a set list of features (for instance, pitches). In this setup, only one spot in the list is "hot" (assigned a value of 1) at any given time. This hot element indicates which specific musical feature is present at that moment. All other spots in the list are "cold" (assigned a value of 0), indicating that those features are not present. This binary representation

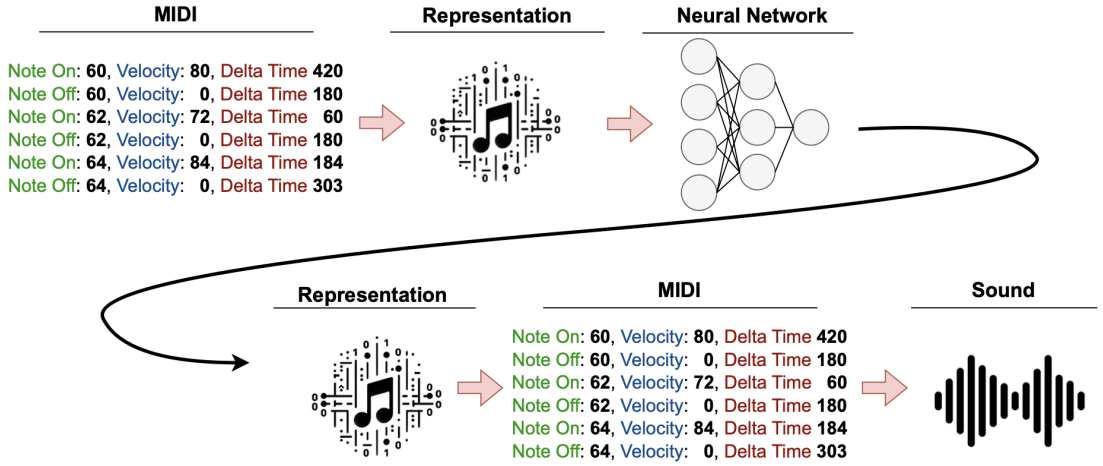


Figure 2.5: Illustrates the process from MIDI dataset to a format suitable for machine learning, via a neural network that outputs this machine learning-friendly format, which is then transformed back into MIDI and ultimately converted into sound.

makes it easier for computational models, such as NNs, to process and recognize distinct events or states without ambiguity. For example, if there are 10 possible pitches to play, a vector [0, 1, 0, 0, 0, 0, 0, 0, 0, 0] would indicate that the second pitch is occurring, while all others are not.

One-hot encoding in music representation has several variations, such as when representing pitches. Oore et al. [77] work with a vector of length 128, for a full range of 128 pitches, equivalent to the number of pitches MIDI can represent. Kumar et al. [80] focus on 88 pitches, the same as a full-size piano, while Zhao et al. [81] limit the representation to 48 pitches to conserve memory and computing power, covering only four octaves.

The standard one-hot encoding is binary. However, this binary approach cannot capture nuances such as the intensity of note (velocity). To overcome this, Mao et al. [82] use a floating number between 0 and 1 in the one-hot encoding. This number represents the velocity of the relevant note. Zhao et al. [81] expanded the note representation to include three aspects: play, articulate, and dynamic. Here, play and articulate are binary, but dynamic is a floating value from 0 to 1 representing velocity. It is also common to add an extra row in the vector representing silence or pause [66, 69].

2.3.4 Event-Based Representation

There are several ways these different one-hot encoded vectors are arranged, such as piano roll [83] or relative pitch [84], but perhaps the most popular is the event-based representation. It is a method closely related to MIDI, in which every musical event is represented by a set of characteristic, often one-hot encoded vectors. These events, varying by project, are actions such as starting or ending a note, pausing, or changing chords or pedal states.

Oore et al. [77] used a version very similar to MIDI, incorporating note-on, note-off, and time shift events. They also added velocity details for every note-on event. One issue with this format is that every note-on event must be matched with a corresponding note-off event, which is not always consistently managed by the network. They addressed this by introducing a sustain pedal that ends note-on events that were started during its activation period. Another challenge is that they use seconds as a time unit. It makes it difficult to explicitly express musical concepts such as quarter notes or bars.

A more recent investigation by Fradet et al. [85] examined different methods of representing time in music generation. They found that assigning a specific duration to each note-on event, rather than relying on separate note-off events, improved the network's performance. This approach helps prevent the common error of overlooking closing a note-on event with a corresponding note-off event. Moreover, the study suggests using musical time units like beats and bars instead of seconds for a clearer representation of rhythm and structure. This recommendation is supported by recent research in the field [45, 60, 66], which has moved towards using explicit duration for notes and musical time measures instead of seconds.

The REvamped MIdi-derived events (REMI) representation, developed by Huang et al. [45], is a type of event-based representation that has gained popularity and inspired various other formats [78, 79]. In REMI, each note-on event includes an attribute for pitch, ranging from 0 to 127 (corresponding to the full range of notes available in the MIDI format). Additionally, it introduces a note duration attribute that relies on a grid-based timing system, where each bar is segmented into 16 parts for precise timing. REMI expands its representation capabilities by including tempo and chord events, allowing for a more comprehensive and dynamic portrayal of musical pieces. The REMI-representation has also inspired more advanced representations that have a more compressed representation of tokens, allowing the processing of longer sequences [78, 79].

2.4 Dataset

Getting good results from an ML model will depend not only on the chosen representation, but also on the quality of the dataset [86]. Since MIDI is a common format for symbolic music, there are many available datasets, which vary significantly in size and quality. On the one end, there are smaller datasets like the JSB chorales [42], which contain 382 chorales, while on the other, larger datasets like the Lakh MIDI dataset [87], which boasts 176,581 MIDI files, are also accessible. Larger datasets tend to be less curated and labeled. The Lakh MIDI dataset even has a few thousand corrupt files. Its size and the fact that it is multi-instrumental still make it a popular choice [57, 74], but sometimes it is used with the need of considerable filtering [88]. The amount of data needed depends on the specific task. Smaller and better-structured datasets can be more beneficial for projects where structure is an important focus, while larger datasets

are useful when training complex DNNs [11].

Datasets also differ in style. Several popular musical generation systems have focused on classical music [43, 77]. Naturally, they use MIDI datasets of classical music, such as JSB chorales. Both the JSB chorales and the much-used MAESTRO piano e-competition dataset [89] are piano-specific datasets, and are therefore not be well suited for the task of multi-instrument generation.

A newer addition to the landscape of the MIDI dataset is the POP909 dataset [90]. It is a dataset consisting of 909 Chinese pop songs. This makes it genre-specific, which can be good for consistent learning. Every song is in three tracks, one for the monophonic lead melody, one for chords, and one for polyphonic melodic harmonization. Chord and timing information is also well documented in separate documentation files. The fact that it is multi-track makes it a good pick for multi-track generation. A drawback in that regard is that it does not include any percussive instruments.

There also exist drum-specific datasets, such as the Groove dataset [91], which is a MIDI drum dataset by Google's Magenta team. It provides 13.6 hours of drumming in different genres. It is well documented with metadata like genre. This is a popular dataset for symbolic drum generation [58, 91–93].

2.5 Music Generation: Neural Network Approach

The choice of network, dataset, and data representation is closely related to the type of music generation that is desired. This section will talk about different methods for generating music.

For the purpose of understanding the different methods relevant to this thesis, I have divided the music generation field into two, (1) the network approach and (2) the MAS approach. The network approach is usually done through a monolithic system focused on using a single NN. The network approach is further divided into single-instrument and multi-instrument generation. On the other hand, the MAS approach often uses several networks or rule-based entities. Nevertheless, the MAS-approach builds upon the network approach, making both parts relevant to this thesis. This first section will focus on the network approach.

2.5.1 Single Instrument Generation

Single-instrument generation is the stepping stone into multi-instrument generation through both the network approach and the MAS approach. It is important to understand how single-instrument generation works, especially if one is to generate autonomous agents, each in control of a single instrument. In general, the task of generating single instruments can be categorized as monophonic or polyphonic.

Monophonic Generation

In monophonic music only one note is played at a time, making its structure simpler and usually based on the creation of a single melody line [66, 69, 94, 95]. Some monophonic single-instrument melody generations are conditional. These are conditioned with information from another domain than the melody itself. In the work of Chen et al. and Zixun et al. [66, 69], monophonic melodies are generated using LSTM and bidirectional-LSTM networks, with the addition of chord information as a condition.

For this conditional task, Chen's [69] approach involves a one-hot encoded pitch vector and a one-hot encoded chord vector to represent the notes and the chord. Zixun et al. [66] adopt a more complex representation. Here, they use a large vector that is a concatenation of five one-hot encoded sub-vectors. For melody information, they have both a pitch and a duration vector. To add conditional information, they also include two one-hot encoded vectors representing the current and next chords, plus a vector marking the start or end of a musical bar. In addition, accumulated time information is fed into the network to enhance its awareness of timing. Their results demonstrate that incorporating these timing elements can significantly improve the network's timing capabilities.

Polyphonic Generation

Polyphonic music generation represents a more complex challenge compared to monophonic music, as it involves creating compositions in which multiple notes are played simultaneously. This requires the network to make decisions not just about whether to play a note or remain silent at a given time step, but also about the number of notes to play and their specific pitches. This complexity leads to richer, more harmonically dynamic, and intricate music, which is more reflective of most music that we listen to today.

To manage this complexity, systems that generate polyphonic music often make certain assumptions or simplifications about the nature of the music [77]. For example, they might divide the music into separate voices [80], restrict the number of simultaneous notes [96], or focus on the use of chords [66, 69, 97].

Chords in Polyphonic Generation

Chords are fundamental to Western music. A chord is a group of three or more different notes played simultaneously, creating a harmonious sound. They form the basis of harmony in Western music and influence the mood and direction of a composition. With only a few chords, one can lay the foundation for harmony in the piece. Therefore, using chords simplifies the generational process by reducing the complexity of the music while maintaining harmonic integrity.

Chu et al. [97] approach this by representing chords using the six main types of triad chords: major, minor, augmented, diminished, suspended 2nd, and suspended 4th.

Given that there are twelve notes in an octave and 6 different chord variations, this method provides $12 \times 6 = 72$ possible chords.

Zixun et al. [66] and Chen et al. [69] simplify the process even further by reducing the number of chord types used in their models. Zixun et al. include only major, minor, diminished and major 7th chords in their framework [66], while Chen et al. limit their model to only major and minor chords, but with the inclusion of an option for no chord played [69]. This further reduction in complexity reduces the dimensionality of the input data, making the generation process more manageable while still allowing for the creation of pleasing and harmonically correct music through chords.

2.5.2 Drum Generation

While the drum in modern Western music is typically a polyphonic instrument, it has its unique challenges due to the focus on rhythm and its repetitiveness. This demands precise timing and musical structure. To address these challenges, various effective methods have been developed, sometimes diverging from the conventional methods used for other instruments.

To achieve better timing, a normal approach is to quantify the input data [58, 98]. This sets all drum hits on beat, but removes microtimings, which some works try to preserve [92, 93]. The velocity with which the instrument is played is also an important aspect in percussive instruments such as drums. Therefore, many choose to include some kind of velocity information for each drum hit, be it as a floating number between 0 and 1 [91, 92], or by decreasing the complexity by grouping velocities into velocity buckets [58, 93].

Drum sets have a variable amount of drums, hi-hats, and cymbals. It is therefore common to do a process called pitch mapping, where several similar-sounding drums can be mapped into the same pitch. The exact number of drums they map down to differs, but can range from eight [58, 91], to seven [93] and only five [98]. This limits the dimensionality of the input data, which again reduces the complexity of the generational task. This reduction makes it easier for Makris et al. [98] to do the normally more challenging multi-hot encoding, as there are only five possible pitches. For example, "10010" could indicate playing the kick drum and hi-hat at the same time, followed by "01010" representing the simultaneous playing of the snare and hi-hat.

Using a different method, Chu et al. [97] identified the 100 most common one-bar patterns in their dataset. They found that these patterns covered 94.6% of all one-bar patterns in the dataset. They then used a one-hot encoded vector with 100 elements where each element represented one of these common patterns. The goal of their NN was to create music sequences by selecting from these 100 patterns. This guaranteed that within every one-bar window the structure and timing are consistently accurate.

In several multi-instrument systems, it is not uncommon to focus mainly on the melodic and harmonic aspects at the expense of the drums. Some ignore them completely

[67], while others choose to simplify the drum generation [97]. BandNet [99] even chose to generate only bass, chord, and melody, before assigning the composition with a pre-composed drum pattern.

2.5.3 Multi Instrument Generation

Just as generating polyphonic music represents a step up in complexity from monophonic music, creating music for multiple instruments is significantly more complex than for a single instrument. The challenge now is not only to produce distinct sounds for different instruments, but also to ensure that these sounds harmonize and create a cohesive piece. Similarly, while polyphonic music may be simplified into streams of monophonic sequences, or by using structured elements like chords, multi-instrument generation often employs similar strategies to manage its increased complexity.

One common strategy has been to combine the sounds of various instruments into a single sequence, which is similar to a single-instrument stream. This stream is then processed by a single neural network [57, 79, 99, 100]. It is common to add additional tokens to include instrument information.

Some have used networks with encoder-decoder structures such as Generative Adversarial Networks (GAN)-networks. Here, you can have a single encoder but with one decoder for each instrument [101, 102]. In this way, the encoder gets the context of the entire musical piece, while the decoders remain instrument-specific. Hierarchical networks are also used where each layer is responsible for one instrument, and communication between each layer ensures that the network makes decisions based on the other instruments [97].

Zhu et al. [103] develop a five-instrument framework, categorizing them into arrangement (violin, bass, guitar, drums) and melody (piano). The process starts with melody generation using predefined chords to create rhythm and melody, followed by arrangement generation to support the melody, emphasizing harmony among instruments. The two systems communicate through two information sharing cells; an Attention Cell to focus on relevant inputs and an MultiLayer Perceptron (MLP) Cell for managing the instruments' hidden states, ensuring coherence and harmony. The Jambot created by Brunner et al. [104] similarly creates an arrangement in the form of chords and a polyphonic melody on top. They use two LSTM networks, where the chord is first generated and the melody is created conditioned on the chord.

Although none of these systems mentions MAS or even the word agent, these systems are increasingly beginning to look like what could be defined as a MAS. This has probably not been the intention of the authors, but by using the definition of MAS, one could argue that several of these are, in fact, MAS. The line between what I have called the network approach and the MAS approach is not clear and will be discussed further in the next section, talking about music generation using MAS.

2.6 Music Generation: Multi-Agent System Approach

As mentioned, many music generation systems could be considered MAS, although they are never classified as such. The reason could be manyfold. Firstly, by the broad definition of MAS, many systems, both in music and in other fields, could be defined as such, making a lot of systems unintentionally fall into the category. This is perhaps even more prevalent in multi-track music, as it is a task that naturally leads itself to a multi-agent approach. However, MAS is a relatively small research field, and when these systems are used actively, they are often used in optimization and simulation tasks [15, 105]. They also traditionally include much simpler agents [106, 107], and often in swarms [22, 47, 108]. This is due to the fact that the birth of this field of research was in the creation of systems that simulated the cooperative behavior of social insects that acted in swarms [109, 110]. Now, MAS extend these boundaries and can even be seen as used in modern LLMs [111, 112]. But across these varied applications, the fundamental concept remains consistent, articulated by Wooldridge as:

> "Multiagent systems are systems composed of multiple interacting computing elements, known as agents." [113]

These agents described by Wooldridge can be very different in complexity and appearance, but they can all be defined as:

> "A computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its delegated objectives." [46]

These broad definitions ensure that there are many different works that can be categorized as MAS. One could argue that works such as that by Zhu et al., Brunner et al., and Chu et al. [97, 103, 104] mentioned in subsection 2.5.3, all fall under the definition of MAS, although they never refer to themselves as such or even mention the word agent.

2.6.1 Agent Types

MAS include agents with different degrees of intelligence. Agents are usually collaborative [24, 114], but sometimes have a competitive nature [115, 116]. However, they are designed in such a way that competitiveness helps them to work together for a common goal, as this is a key element of MAS [46].

The complexity of agents can vary from extremely simple, purely reflexive, to highly intelligent, capable of reasoning, communicating, and arguing. Agents that simply react to an environment, without reasoning about it or without knowledge of previous or

future states, are called reflexive agents. They are the very simplest form of agents and react to a single stimulus with a single action. Their behavior can be modeled as in Equation 2.3.

$$f : P \rightarrow A \tag{2.3}$$

Here, f is a function, P is a set of percepts, and A is a set of actions. Systems such as those of Blackwell and Bentley [50] inhabit reflexive agents that act on musical inputs using Particle Swarm Optimization (PSO), and emergence occurs. This is the collective intelligence that emerges from seemingly unintelligent actions.

Some systems include reactive agents. These agents are a bit more complex than reflexive agents by including some sort of memory. Virtualband and INMAMUSYS [23, 115] both create a band in which reactive agents have access to a database of music that they choose to harmonize with the music they listen to, but without NNs or complex algorithms.

By further increasing the complexity of the agents, we reach the intelligent agent. They are capable of acting using more complex methods such as advanced algorithms [117, 118] or NNs [114]. In addition, intelligent agents are capable of communication, which in turn gives them the ability to cooperate, negotiate, argue, and vote. In Solojam [116], agents use auctions to decide who should play solo. Miranda et al. [24] create a system in which agents are created by genetic code, and by listening to and rating other agents, their behavior evolves through learning and adoption. Similarly, the agents in MusicMas [117] rate the output of other agents. Certain systems exhibit more straightforward communication methods. In systems using NNs, it is common for agents to transmit sequences of music they have generated [97, 104] or specific features [114].

A common agent architecture for an agent has been the Belief-Desire-Intention (BDI) architecture, which in a way summarizes the design of an agent. Introduced by Bratman [119] as a philosophical theory of practical human reasoning, and later used in MAS, first by Rao and Georgeff [120]. When using this architecture, each agent should have the following:

- **Belief**: Represents the agent's knowledge about the world. It's the agent's current state of information. May be incorrect.
- **Desire**: Represents the goals or objectives the agent wants to achieve.
- **Intention**: Represents the agent's current plan or strategy to achieve its desires based on its beliefs.

This architecture has also been used in musical agents [117].

2.6.2 Interactive Music Systems

A benefit of using MAS is that as the goal is reached by distributing tasks to different interacting agents, it becomes natural and easy to include a human agent in the loop [22, 23, 114]. Systems not intentionally made to be MAS can also include human interaction [121, 122]. However, by doing so, one can argue that this in fact makes these systems multi-agent by introducing a human agent.

Several ways of achieving human interaction for music generation have been tested. Lewis' Voyager [22], MusicMas [117], and Virtualband [23] use systems where the human agent plays an instrument and the agents harmonize with it, creating a human-AI composition. Some have created programs that are fed human-made music, and the system continues to play [74], or human-made music with missing parts, where the system completes the incomplete composition [43, 123].

A different strategy has been to let the human agent rate the musical composition created by the system, steering a Reinforcement Learning (RL) agent [114] or a Genetic Algorithm (GA) [122, 124] towards creating music tailored to the human's taste.

A third approach has been to control the generation of music by changing the parameters. Louie et al. [121] include three *semantic sliders*. With these, humans can alter the parameters by moving a slider from major (happy) to minor (sad), conventional to surprising, and similar to different. Such sliders can alter the generation by changing the hyperparameters or the sampling distribution of an ML-model. Louie et al. [121] found that with the introduction of these Semantic Sliders, the user experienced greater creative ownership and better expressed the musical intent.

In newer music generation research, the focus has been on using complex transformers that receive text prompts and use them to generate music [7, 8]. While they are very impressive, research shows that when musicians attempt music co-creation with generative systems, they generally prefer to use several simpler models instead of large end-to-end models. They feel that this gives them better control and the possibility of being more creative [13].

2.7 Evaluation

Evaluating computer-generated music poses a unique set of challenges, distinct from many other domains of ML. At its core, music is a deeply subjective experience, and what resonates with one individual might not appeal to another. Even for a single individual, it is hard to evaluate one piece of music over another. Unlike tasks with clear metrics of success or failure, assessing the "quality" or "authenticity" of a musical piece is ill-defined. Although objective measures, such as the repetition of notes or the absence of dissonant intervals, can be quantified, they often fail to capture the emotive essence or the creative novelty of a composition.

Subjective evaluations, such as listener surveys or expert reviews, introduce their own biases and can vary widely based on cultural, personal, or contextual factors. Thus, determining the efficacy of a music generation algorithm requires a nuanced blend of objective metrics and subjective interpretations, making it a challenging task. This challenge has been the sole focus of several studies [27, 125]. A summary of some of the evaluation methods used will be provided in the following sections.

2.7.1 Subjective Evaluation

Subjective evaluation relies on human participants to evaluate the output of the generation. The perhaps most famous subjective evaluation for generative AI is the Turing test [126]. It is a test of whether human participants can distinguish a human-made (ground truth) example from a computer-made one. This was used by Donahue et al. [74]. Similar comparisons are often used in subjective evaluations. Often, participants are asked to rank two or more musical pieces against each other. This can be generated examples vs. real examples [81, 127], generated examples vs. other similar generative models [45, 57], or both generated vs. real examples vs. other similar generative models [128]. It is also normal to compare the system against itself using different criteria, such as system architecture or network configurations [73, 81, 97, 101].

Instead of ranking generated material in its entirety against other bits of audio, a different approach is to rank certain traits of the generated output. Some articles use certain general traits such as harmony, rhythm, musical structure, and coherence [81]. Some papers focus primarily on advancing in one aspect of the generation, such as emotion or style. In these cases, it is normal to let the participants classify the sound output as a certain emotion [81] or style [82]. During such comparative experiments, some collect information about the musical knowledge of the participants [45, 129]. This is done to see if there is any difference in preference based on the listener's musical background.

Others, like Oore et al. [77] did most of their evaluation objective, but also sent listening examples to professional musicians and composers for anecdotal subjective feedback that they added to show what a set of experts thought about the results. This can also be done through semi-structured interviews [130–132].

These subjective evaluations are often found when trying to evaluate a model for music generation. However, it is important to be aware of their shortcomings. Biases such as the mentioned personal and cultural ones are an important aspect. It has also been suggested that subjective evaluation has a bias towards models that overfit [27]. Overfitting will lead the model to create songs that are very similar to the training data and do not create music that is original, creative, or generalized. Thus, while subjective evaluation is a common practice, including objective evaluation is often useful.

2.7.2 Objective Evaluation

Objective evaluation allows for the evaluation of a large amount of generated material instead of only a small selection, and without personal bias. Although art is difficult to objectively conclusively evaluate, several methods are being used.

Yang et al. [125] claim that the realm of generated music moves much faster than the field of evaluation of generated music and suggests both an absolute and a relative metric to be used as objective evaluations. The absolute metric gives insight into the data on pitch and transitions through things like transition matrices, pitch intervals, and histograms. The relative metric uses Probability Distribution Function (PDF) of the model compared to PDF of the dataset. These PDFs can be analyzed using the overlapping area or the Kullback-Leibler (KL) divergence between them. This indicates how well the model has learned the traits of the dataset. These measures have been recognized by other researchers who have used KL divergence and overlapping area as part of their evaluation [133].

Another typical approach is to analyze the model through accuracy and log-likelihood. As opposed to using subjective evaluation which can be biased towards overfitting, log-likelihood favors models with large entropy [27]. Examples of this being used to evaluate generated examples can be found in a lot of research [73, 74, 77, 81]. Although it is an easy metric to measure and normally gives a way of quantifying how well the generated output aligns with the training data, it does not tell the whole picture. High likelihood does not guarantee good examples. Generated samples can take on an arbitrary form only a few bits from the optimum [125].

Chapter 3

Methods

This chapter describes the methods used to implement ML-jam. This mainly revolves around the creation and structuring of the five artificial agents (drum, bass, chord, melody, and harmony), in addition to the interactive aspects of the human agent. An overview of the agents and their relations is shown in Figure 3.1.

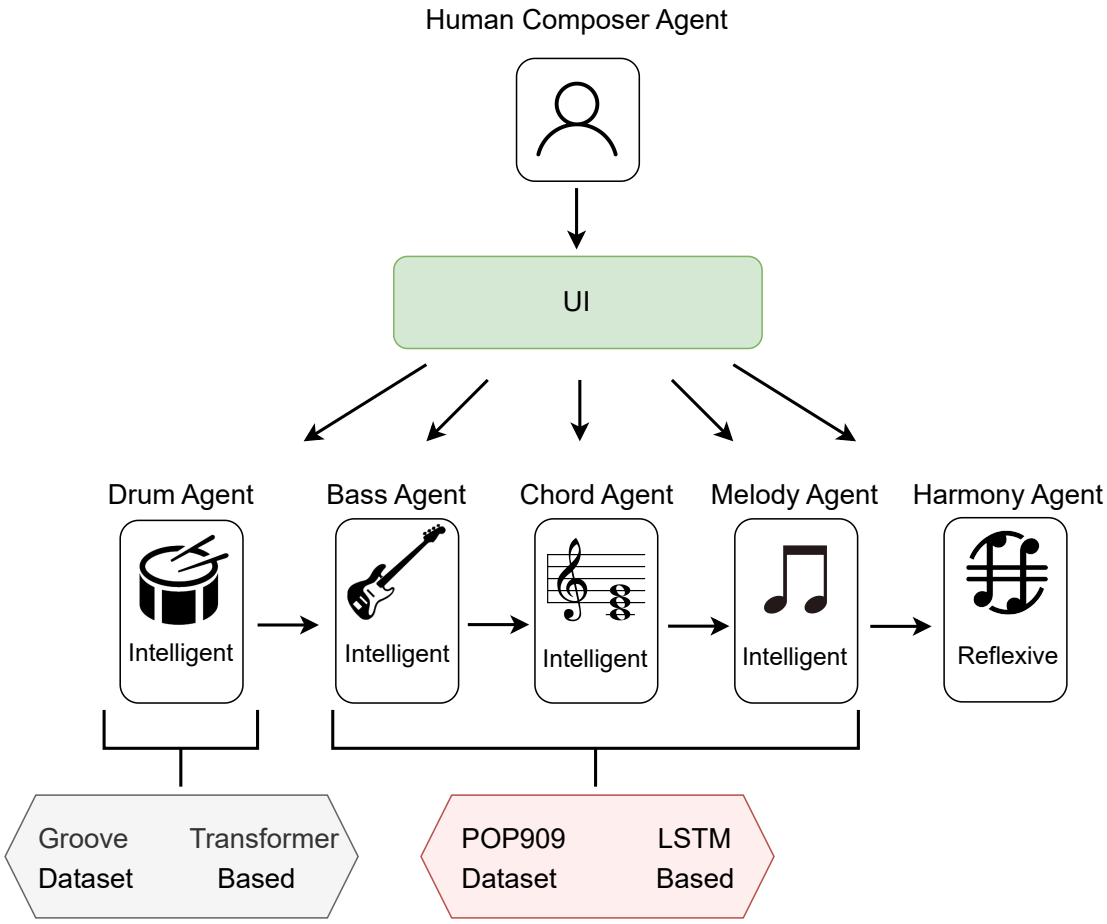


Figure 3.1: The system architecture with the different agents and how they interact. Arrows indicate information flow.

This chapter will address the individual data processing, creation, and tuning of each agent. However, before this, it is important to know how the system works on a larger level in order to understand the design choices made in data preparation and model design. Therefore, the chapter is structured as follows.

Firstly, an introduction to the software and tools used to create and use ML-jam is presented. Next, the assembly of musical agents to form the system is discussed, detailing the role of each artificial agent and how the human agent can control them. Following this, there is an introduction to the datasets and data processing methods for the individual agents. Finally, the chapter presents the network architecture, training, and tuning for each agent.

It is important to note that, like other studies [67, 99], this project focuses primarily on the melodic components of music generation, facilitated through the Bass, Chord, Melody and Harmony Agents. However, unlike the studies mentioned which exclude drum generation, this project incorporates a Drum Agent. Instead, data processing and network design for this agent were taken from the "Bumblebeat" project by Nuttall et al. [58]. This approach allows for the inclusion of a drum generation into the multi-agent loop while primarily focusing on other instruments and on the aspects of co-play and human interaction.

3.1 Software and Tools

In the development of this AI-generated music project, several programming languages, frameworks, and applications were utilized to create a comprehensive and interactive system. Figure 3.2 shows an overview of the architecture used.

The project is primarily developed in *Python 3.9.5*, serving as the backbone for the ML components and the server-side logic. Python is the leading programming language used in ML tasks due to its simplicity, and its huge arsenal of integrated solutions and external libraries. The most important ones used in this project are mentioned below, but for the full list of Python libraries used, refer to the project codebase [28].

For the ML agents responsible for the generation of music, PyTorch was the primary framework used due to its flexibility and efficiency in handling deep learning models. More specifically, this project uses *Torch 2.2.0*. This library is used for all agents except for the Drum Agent, which diverges in its technical foundation. The Drum Agent is heavily inspired by the "Bumblebeat" project by Nuttall et al. [58] and consequently utilizes TensorFlow to maintain consistency with the original architecture and methodologies of the project. ML-jam was implemented using an Apple Silicon M1 MacBook, and Therefore uses *Tensorflow-macos 2.14.0* and *Tensorflow-metal 1.1.0*. The drum generation in ML-jam is also possible to run using the more conventional *Tensorflow 2.11.0*. For the creation, processing, and manipulation of MIDI, I used the python library *Pretty-midi 0.2.10*, while *Rtmidi 2.5.0* is used to broadcast the MIDI

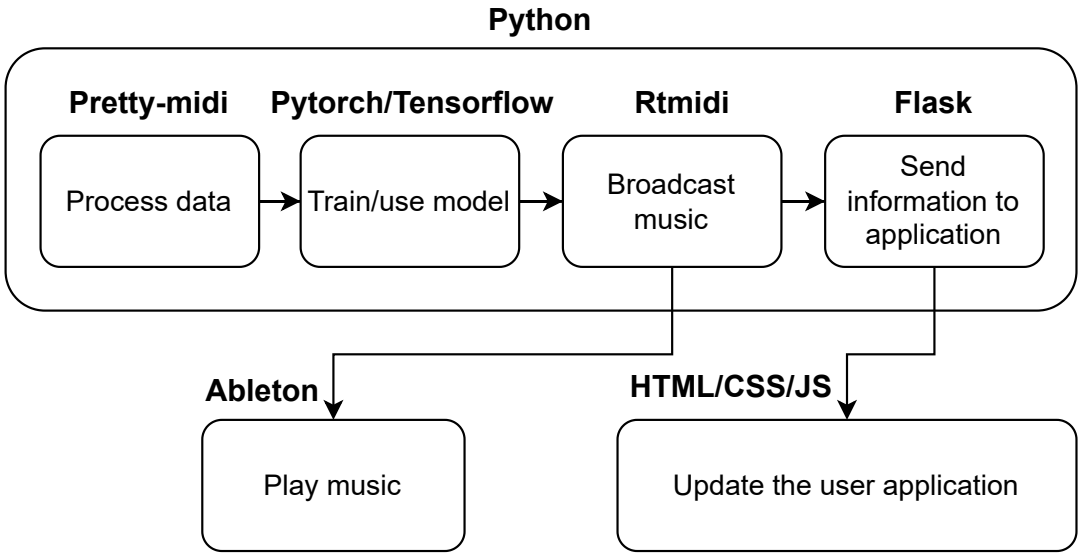


Figure 3.2: Shows the system architecture with the most important frameworks and libraries. Each box is a process with the corresponding library or framework in bold. The arrows indicate which process follows next.

signals.

The front-end application was crafted using standard *HTML*, *CSS*, and *JavaScript*, ensuring a user-friendly interface and seamless user experience. The interaction between the Python backend and JavaScript frontend is facilitated through a *Flask Application Programming Interface (API)* using *Flask 3.0.0*, which acts as a bridge for data exchange and real-time updates between the server and the client.

Lastly, the project integrates *Ableton Live 11 Suite*, a popular DAW, to play the musical output. Ableton listens to the virtual MIDI ports on the computer and plays back the music that is broadcasted to them. This integration enables the use of different instrument sounds and effects available in Ableton.

This combination of technologies and frameworks ensures a robust system capable of generating, playing, and manipulating AI-generated music in an interactive setting.

3.2 Assembling the Musical Agents

The project aims to connect several agents together to form a musical band. This segment will discuss how these agents act together to form the band through architecture and communication in addition to individual agent design on a broad functional level. A more thorough description of the data representation and network architecture of the intelligent artificial agents is available in section 3.5 and section 3.6, respectively.

Table 3.1: Overview of the different agents that create the band.

Agent Name	Intelligent	Reflexive	Artificial	Human
Drum Agent	✓		✓	
Bass Agent	✓		✓	
Chord Agent	✓		✓	
Melody Agent	✓		✓	
Harmony Agent		✓	✓	
Composer Agent	✓			✓

The band consists of six agents, see Table 3.1; five artificial agents and one Human Composer Agent. The goal is to use these five artificial agents to replicate the structure and communication found in a band with human musicians during a jamming session. The human agent's role is to control the generation of music by the artificial agents.

To do this, the process of generating musical outputs for the artificial agents has been divided into two phases, where in both phases, the Human Composer Agent controls the other agents. Some agents go through both phases, while others only partake in one phase:

- **Phase 1:** The agent generate or acquires the musical sequence. This can be through the use of an NN or by listening to other agents.

- **Phase 2:** Rule-based methods are used to enhance the output of Phase 1. This is done to improve the interaction between agents, give the human composer more control of the composition, and create more variation to make the music sound more alive. Some modifications are constant, while most are optional and controlled by the Human Composer Agent.

These rule-based modifications are relatively simple to implement and modify. Thus, the program can be easily adapted to new play styles through the introduction of new Phase 2 modifications.

3.2.1 Band Pipeline

A common challenge for interactive music generation systems is the need for a real-time response. For simpler systems that use fast algorithms, real-time can be naturally integrated [22, 134]. Systems using a more advanced generation method like NNs have a harder time maintaining the feeling of a system that responds in real-time to human actions due to inference time. A disadvantage of the lack of real-time reaction by the system is that it might not feel as responsive, particularly in interactive setups where humans collaborate with the system by playing together. Here, latency can remove all feelings of responsiveness. Therefore, studies focus on ways to combat this [135, 136].

This project not only uses time-consuming NNs to generate music, but also has communication between agents that create dependencies. For example, certain agents must complete Phase 1 before others can proceed, which requires a structured generation pipeline as depicted in Figure 3.3. On a standard computer, the cumulative inference time for the entire pipeline ranges from 2 to 4 seconds, which poses a barrier to achieving seamless real-time performance.

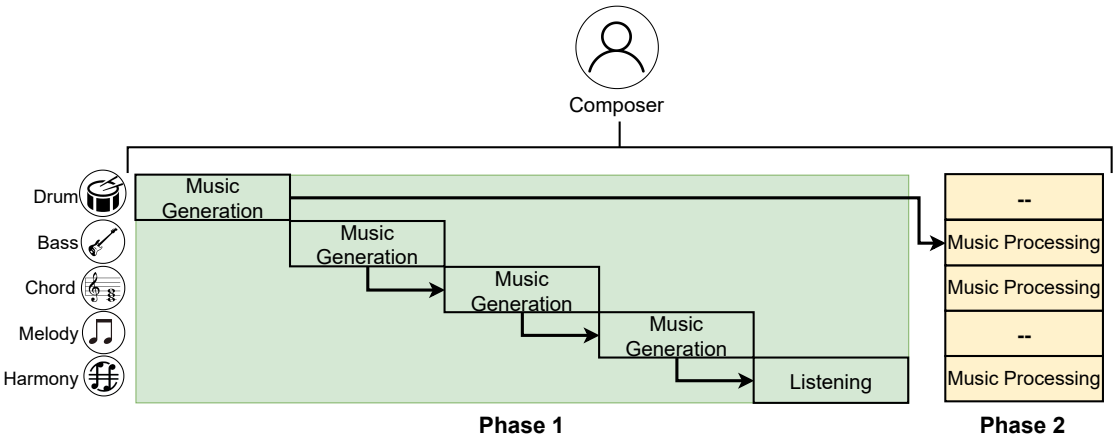


Figure 3.3: The pipeline of the agents. The arrows show the flow of information between different agents. In Phase 1, the Drum, Bass, Chord, and Melody Agents generate music through their NNs, while the Harmony Agent listens to the Melody Agent. The Bass, Chord and Harmony Agents undergo Phase 2, altering the musical input they acquired in Phase 1. The Composer Agent has control over the operations in both phases. Everything in Phase 2 does not actually happen simultaneously, but as there are no dependencies, they are visualized as such.

To accommodate the system's large combined inference times while still enabling it to be used in tandem with a human performer (which requires minimal latency), a stable semi-real-time approach is adopted in this project through the use of musical loops. Once the agents have generated a music segment of the desired length, this segment is looped continuously until a new segment is ready. The transition to the new loop occurs seamlessly after the preceding loop, ensuring that the human user experiences a predictable delay between their actions and the system response, thus maintaining control over the musical output.

By using loops like this, it is also possible to only recreate music from selected agents,

giving the user the flexibility to preserve sequences from one agent while updating others. Furthermore, the use of loops naturally introduces repetition into the music. Research indicates that long-term musical structure is the key to creating repetition [56, 60, 95]. These systems focus on long-term structures and are large and complex. So, by instead adding repetition to the system's design, the aim is to achieve a larger sense of musical structure than the agent's network actually has.

Post Generation

After the different agents have completed Phase 1 and Phase 2, the music generation is completed, and the output of their different representations is translated into MIDI format. These MIDI signals are then sent to virtual MIDI ports within the computer. A DAW capable of receiving input from multiple MIDI ports simultaneously interprets these signals to produce sound. An advantage of using a DAW to output the sound lies in its ability to apply various sounds and effects, enabling the shaping of the soundscape even after the music generation process has been completed. This flexibility allows for further customization and refinement of the musical piece.

3.2.2 Artificial Agents

The system consists primarily of artificial agents. This section will describe how the different agents communicate and play on a superior level. The Human Composer Agent that controls all the artificial agents will be described in section 3.3.

Drum Agent

In most popular music, the drum is the foundation of rhythm and tempo and acts as the backbone of the composition [137]. That is also the case in this system. The Drum Agent is the first one to generate music and is the only artificial agent not affected by any other artificial agent.

- In Phase 1, the Drum Agent uses a NN to create a 1-bar drum beat that is looped for the duration of the generation.
- The Drum Agent has no Phase 2, the AI-generated output of Phase 1 is used raw without any modifications.

Bass Agent

In a band setting, the bass primarily establishes the harmonic foundation through the chord progression and fills the low end of the frequency spectrum. It often partners with the drums to enhance the rhythm of the piece and create a groove. Therefore, the Bass Agent is responsible for creating the baseline of the chord progression through the generation of root notes, and simultaneously the groove along with the drum.

- In Phase 1, the Bass Agent uses its NN to generate root notes and durations. This process lays the harmonic groundwork for the chord progression.
- In Phase 2, the Bass Agent cooperates with the Drum Agent. The Drum Agent passes on information about when the bass drum is played, enabling the Bass Agent to synchronize its notes with the bass drum hits. The aim of the synchronization is to produce a dynamic soundscape that sounds more alive and less flat. The interaction is depicted in Figure 3.4. On instruction from the Composer Agent, the agent can also use several rule-based methods to play groovy transitions from one note to the next.

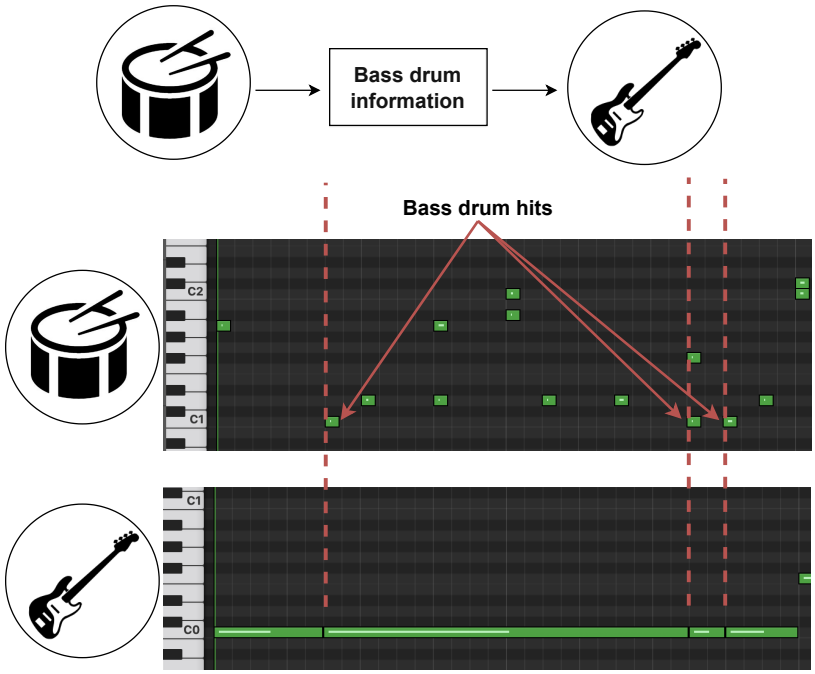


Figure 3.4: The communication between drum and Bass Agent. The Bass Agent plays a C0 note on the bass drum hits from the Drum Agent.

Chord Agent

Chords consist of several notes that are derived from the root note and the chord variation. Getting both the root notes and the chord variation fulfills the chord progression. The notes of a chord are harmonically aligned and serve as the structural basis of the melody [138].

Following the methodology of Chu et al. [97], the agent knows six types of triad chords: major, minor, diminished, augmented, suspended 2nd, and suspended 4th. Chords in Western pop music have mathematical foundations, and triad chords, in

particular, are constructed of three different notes where the interval between the notes defines the chord variation. The six chords used in this project can be seen in Figure 3.5.

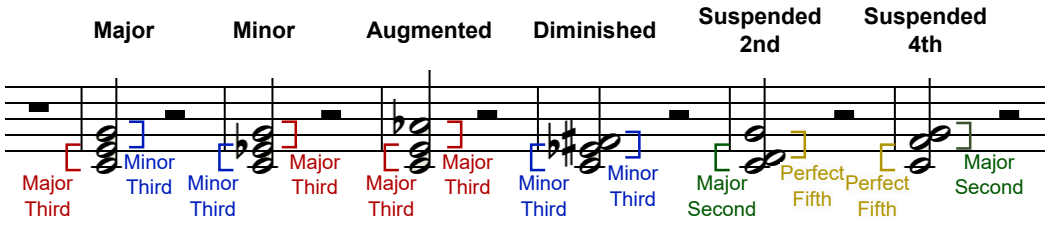


Figure 3.5: The six triad chords known to the Chord Agent, and the interval between the notes.

In this system, the Chord Agent receives the root notes from the Bass Agent and creates chord variations on top of these.

- In Phase 1, the Chord Agent communicates with the Bass Agent, which passes along root note information. Conditioned with this data and an awareness of previously generated chords, the Chord Agent generates chord variations aligning with the root note.
- In Phase 2, the Chord Agent offers a range of playing styles. These styles are interchangeable and can be activated or adjusted according to the Composer Agent needs. Throughout the different playing styles, the chords are always played in the root position, meaning no chord inversions.

Melody Agent

In addition to the foundational groove established by the drums and bass, and the harmonic structure provided by the chord progressions, the melody emerges as the central and most distinctive element of the composition.

- In Phase 1, the Melody Agent communicates with the Chord Agent, which passes along three information metrics used by the Melody Agent: (1) the current chord, (2) the upcoming chord, and (3) the duration until the next chord begins. Conditioned with this information, the Melody Agent generates melodies that integrates smoothly with the underlying harmonic context.
- The Melody Agent has no Phase 2, the AI-generated melody from Phase 1 is not processed.

Harmony Agent

To give the music more depth, a melody might not act on its own but is sometimes harmonized by a secondary melody. This can create a more complex soundscape.

- In Phase 1, the Harmony Agent listens to the melody generated by the Melody Agent.

- In Phase 2, the Harmony Agent alters the melody line using rule-based systems. The Composer Agent controls these modifications.

3.2.3 Ties To The Composer Agent

Ultimately, the tuning of the networks used in Phase 1, or the choices of post-processing in Phase 2 is controlled by the Human Composer Agent. The specific choices available, in addition to the corresponding underlying modifications done to the artificial agents, will be discussed in section 3.3.

3.3 Human Composer Agent

The previously described artificial agents work together to create a musical output, but they are all guided by a Human Composer Agent. The composer controls the musical production from what is played to how it is played. This is done through a digital UI, which gives the user control over sliders, dropdown menus, and checkboxes that affect both the Phase 1 generational process and the subsequent execution in Phase 2[1]. Sliders like these have been shown to give the user greater control and ownership of the composition [121].

3.3.1 User Interface

Depicted in Figure 3.6 is the UI utilized by the Human Composer Agent. This interface provides each agent with a set of parameters that can be adjusted, in addition to overarching composition parameters that affect all agents, and controls for generating new loops and adjusting the volume of the agents. The next section will go into detail on every available parameter.

3.3.2 Tunable Parameters and Play Styles

This section will explore the various parameters and play styles that the Composer Agent manages, as well as the impact these choices have on the musical output of the agents.

It is important to remember that, as detailed in subsection 3.2.1, the system functions based on loops due to inference time constraints and the required pipeline. Consequently, any modifications made by the composer to a parameter will take effect only once a new loop is generated.

[1] A video showcasing the use of ML-jam is available here: [30].

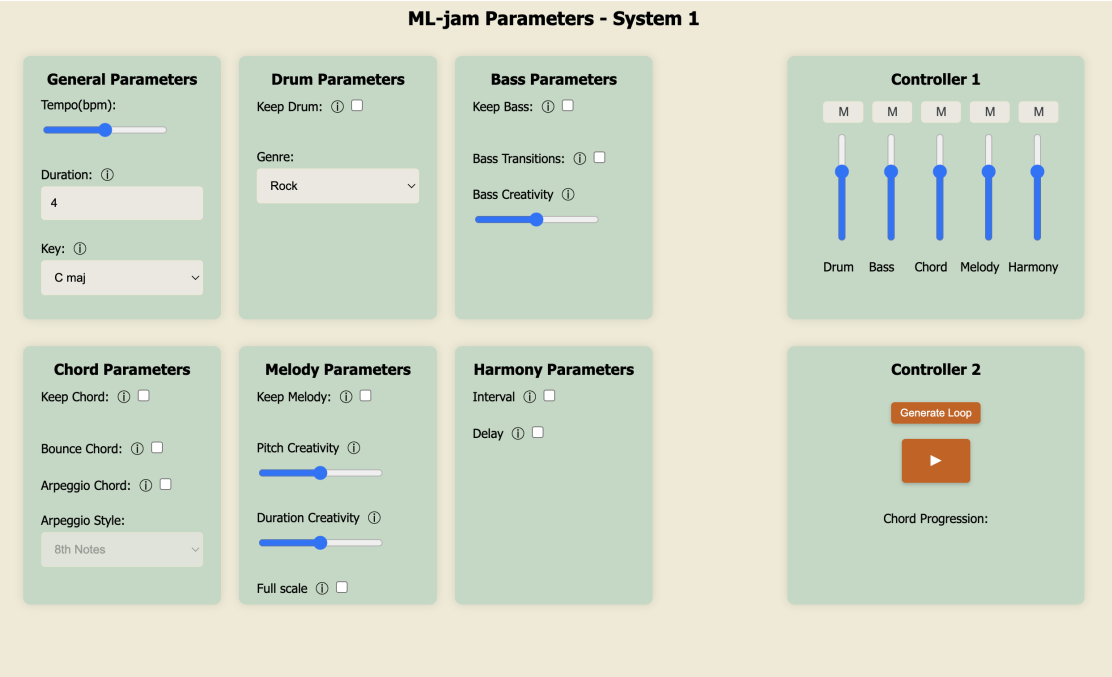


Figure 3.6: The digital UI the Human Composer Agent interacts with. The general parameters control the composition as a whole, specifically the *Tempo* (beat per minute (bpm)), the *Key* and the *Duration* of one loop. The drum parameters include *Keep* option (keeps track in future generations) and *Genre* control. The bass parameters include the same *Keep* control, a *Bass Transition* option, and a bass *Creativity Slider*. The slider works by restricting the durations the Bass Agent can generate. Chord parameters include the *Keep* option and the playing styles *Bounce Chord* and *Arpeggio Chord*, where the latter has multiple versions. The melody parameters include *Keep*, *Pitch Creativity*, *Duration Creativity*, and *Full Scale*. Excluding *Keep*, these parameters affect the temperature and restrictions on the duration and pitch generation of the agent. The harmony parameters are *Interval* and *Delay*. These apply pitch adjusting and delay effects on the melody. Controller 1 is a sound mixer with volume sliders and mute buttons, while Controller 2 has a *Generate Loop* and a *Play/Pause* button, in addition to showcasing the chord progression.

General Parameters

Most parameters are agent-specific, but some affect factors that influence all agents. They are categorized as general parameters:

- *Tempo:* It ranges from 60 bpm to 180 bpm. This slider sets the tempo of the music piece and defaults to 120 bpm.
- *Duration:* Altering it allows the Human Composer Agent to control the length of the loops that are produced.
- *Key:* It allows the composer to change the key of the entire composition. This adjustment is achieved by shifting every note of every agent upwards by the required semitones.

Drum Parameters

The drum has two parameters. Firstly, it has a *Keep* option, a feature common to all intelligent artificial agents (drum, bass, chord, and melody) within ML-jam. By selecting this option, the agent is instructed not to generate a new loop but instead to continue using the current one. Keeping loops is beneficial for helping the composer tailor the music, as well as reduces the inference time, as agents that keep the loop do not need to generate a new one.

The other option is a dropdown menu for genre selection. The composer can choose between different genres to get different styles of drum play. This is done through a genre-specific primer sequence and will be discussed in further detail in subsection 3.6.1.

Bass Parameters

The first parameter, similar to the other agents, is the *Keep* option. Additionally, there is a *Bass Transition* option. Activating this feature enables several rule-based transitions between notes during phase 2 of the bass generation process. These transitions are designed to enhance the groove, creating smoother and more engaging movement from one note to the next.

Lastly, the Bass Agent is equipped with a creativity slider. This slider controls the diversity in note durations that the bass can produce. At a low creativity setting, the bass is restricted to producing longer, more stable root notes, typically 4 beats in length. As the creativity setting is increased, the agent gains the freedom to use a wider range of durations, thereby injecting more variability and complexity into the bass line. This change in duration choices is achieved by adjusting the probability distribution that guides the agent's selection process, ensuring that the choices conform to the desired level of restriction, as detailed in Figure 3.7. The durations are at all times restricted to the duration of the current loop, regardless of creativity levels.

Chord Parameters

As with the drum and bass, the Chord Agent also includes the standard *Keep* option. In addition to the *Keep* option, the Chord Agent offers two distinctive play styles that alter the agent's way of playing the generated chord:

- *Bounce Chord:* Selecting this option causes the chords to be played with the duration of 8th notes in a staccato style. This play style adds a rhythmic, percussive quality to the chords, enhancing the dynamic texture of the piece and contributing to a more pronounced rhythmic drive.

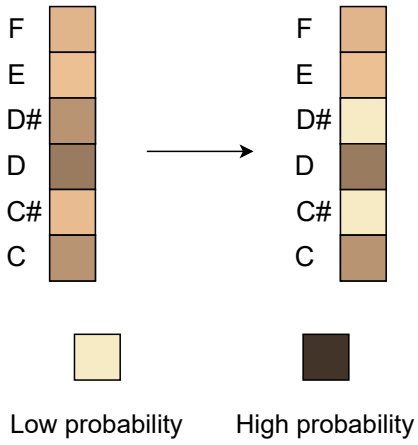


Figure 3.7: The effect of a reduced creativity, where the distribution is being altered restrict notes that are outside of the C major scale (C# and D#). This restriction is applied to the Bass Agent's duration, and the Melody Agents durations and pitch.

- *Arpeggio Chord:* When this option is activated, the chords are played as arpeggios. This means that instead of the chord notes being played simultaneously, they are played in sequence, one after another, creating a melodious and flowing effect. The composer can choose from four different arpeggio styles, which vary in range and note duration, allowing for adjustments that can match the desired stylistic preferences of the composition.

Melody Parameters

In addition to the *Keep* option, the Melody Agent is equipped with two creativity sliders. One for pitch, the other for duration. The pitch slider alters the temperature in the generation and activates scale restrictions. If set low, the temperature is reduced, which sharpens the probability distribution so that the agent has a higher chance of choosing what it considers the most probable option. This results in more predictable and conventional melodies.

As the slider increases, the temperature increases, which smooths out the probability distribution. This change encourages the agent to select less probable options, leading to more unconventional and varied melodies. The effect of temperature is shown in Figure 3.8.

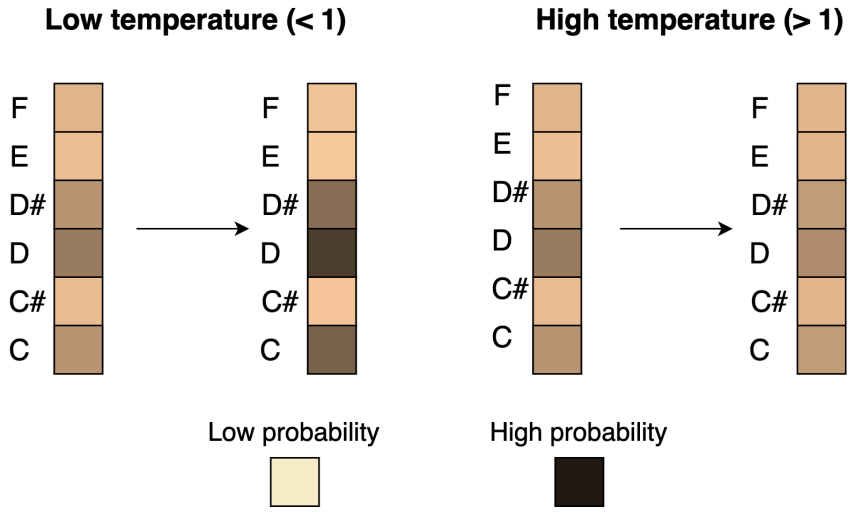


Figure 3.8: The two figures show how the probability distribution of pitch selection changes when temperature is applied. Low temperature spikes the distribution, causing pitches deemed safer to be chosen. High temperature smooths the distribution, causing a higher chance for pitches deemed less likely correct to be chosen.

In addition to affecting temperature, the creativity slider also restricts pitches and durations, similarly to how it is shown in Figure 3.7. At a low setting, the agent's probability distribution is altered such that the pitch choices are confined to the pentatonic scale, which includes only five of the standard 12 notes in an octave. At higher temperatures, the selection of notes is less restrictive and includes the full major scale, consisting of 8 out of 12 notes in an octave. If it increases further, the scale restriction is removed, allowing the agent to play every note, in or out of scale.

The duration creative slider works in a similar way. If it is set low, the duration temperature is low, and the duration distribution is altered to restrict notes like 16th note or dotted notes, steering the melody towards simpler, more straightforward rhythms. A higher level of creativity results in a higher temperature and fewer restrictions up to the point of no restriction.

These restrictions and temperature settings are designed to provide composers with the ability to influence the mood and style of melody generation. For more experimental and unconventional melodies, one can increase the creativity settings. To achieve a more traditional melody, it is best to keep the creativity at a moderate level. If your goal is to create a slower, more tranquil melody, then lowering the sliders is the way to go. This flexibility allows composers to tailor the music to match their artistic intention.

Lastly, there is an additional option on the melody controller, *Full Scale*, that will remove any scale restrictions, regardless of the creativity setting. Activating this feature ensures that the pitch distribution is free from any scale constraints. This allows no scale restrictions, but without needing to increase the temperature.

Harmony Parameters

The last controllable agent is the Harmony Agent. Unlike the other agents, this one is purely reactive, meaning that there are no parameters that change the Phase 1 behavior of the agent. In Phase 1, the agent receives the melody by listening to the Melody Agent. However, for Phase 2, the Harmony Agent offers two options that alter the playback of the melody: *Delay* and *Interval*.

- *Delay:* When activated, the *Delay* option modifies the melody by adding a delay effect, similar to the functionality of a delay plugin used with synthesizers or guitars. This effect repeats the melody notes at intervals; each repetition is quieter and slightly delayed compared to the previous one.
- *Interval:* The *Interval* modification plays the melody note simultaneously with the original but shifts it higher by 5 semitones. This creates a harmonic layer that complements the main melody.

Both the *Delay* and *Interval* options can be used one by one or simultaneously. Achieving delayed notes, played with a 5-note interval from the melody.

Main Controls

- *Controller 1:* It includes mute buttons and allows for control over the volume of each artificial agent.
- *Controller 2:* This interface element houses two buttons. One is for generating new loops, and one is for play/pause. It also displays the chord progression together with the duration of each chord. This feature aims to make it easier for musicians to play along.

3.4 Datasets

This project uses two different datasets. The first is the Groove dataset by Google's Magenta team [91]. This dataset is used by the Drum Agent. Although the dataset is smaller than others, such as the Lakh MIDI dataset [87], it is frequently used in drum generation [58, 92, 93] due to its reliability, good structure, and documentation. The Groove dataset consists of 13.6 hours of MIDI drum music, divided into 1150 MIDI files, played by 10 different artists. Most of the performances are played in a 4/4 time signature, and all of them with a metronome. The dataset is provided in a training split of *train*, *test*, and *validation*. The test and validation sets are each 10% of the dataset, making the training data 10.8 hours long. The drum beats are divided into genres that allow for genre-specific generation.

For the other instruments, I have used the POP909 dataset from Wang et al. [90]. It is a dataset consisting of 909 pop songs played by professional musicians, totaling about

60 hours. The songs are divided into three instrument tracks, none of which includes any percussive instruments:

- **Melody Track**: the lead melody transcription. Purely monophonic.
- **Bridge Track**: the arrangement of secondary melodies or lead instruments. Both monophonic and polyphonic.
- **Piano Track**: the arrangement of the main body of the accompaniment, including broken chords, arpeggios, and many other textures. This track is mostly polyphonic.

Of these tracks, the only one that is used directly is the melody track. In addition to the MIDI tracks, each song has information on the beat, key, and chord, divided into 6 different additional files. This documentation is crucial for this project; some agents are conditioned on information from the domain of other agents. This requires them to have precise and accurate knowledge about the surrounding composition. Other datasets would require me to extract this information from the MIDI data, which is a time-consuming and inaccurate process. This is one of the strengths of the POP909 dataset.

Another benefit of this dataset is that every song is a pop song. This is beneficial as all of the music is in the same genre, but also since pop music is quite uniform in structure. This allows for easier training and faster convergence.

3.5 Data Processing

While two suitable datasets have been chosen, the MIDI files and the included text files cannot be used raw, but must undergo processing in form of feature extraction.

Traditional music generation systems for multi-track generation use complex representations, often trying to capture the music by the different instruments in a single stream of tokens [57, 79, 100]. A benefit of using a MAS architecture is that, by using a modular design, agents only need to know how to represent music in their domain, reducing the complexity of music representation for each agent.

Because of this, feature extraction is implemented individually, so that each agent can get the information needed to suit the specific traits and challenges of their generational process. This is similar to how the data representation can be tailored for single-instrument generational tasks [58, 66]. This section will go into detail about the features that are used, how they are extracted, and what type of representation is used for each agent.

3.5.1 Groove Dataset

This project uses the drum generation created by Nuttall et al. in the «Bumblebeat» project [58]. Therefore, the data processing is performed in the same way as in the paper by Nuttall et al. [58], which again draws a lot of inspiration from the original paper where the Groove Dataset is introduced by Gillick et al. [91]. The data processing will be described here, but for more details, refer to the work of Nuttall et al. [58].

The MIDI data for the drums are transformed into a continuous one-dimensional stream of tokens. Unlike melodic instruments, where the duration or end time of a note is important, drum data primarily require tracking the pitch, velocity, and start time of each hit. These attributes are extracted from each drum event and undergo transformations to be encoded properly into the token stream.

Pitch Mapping

The Groove dataset features 22 unique pitches, each representing a different drum sound. However, many of these pitches appear infrequently within the dataset, contributing to an unnecessary high dimensionality in the input data that can adversely affect the training process. To address this, the pitches are consolidated into nine distinct categories according to their sound characteristics. This consolidation mapping, which reduces the complexity of the input data, is available in Appendix B as a table showing the mapping between the Roland TD-11 drum kit used to record the dataset and the nine pitches used.

Velocity Bucket

In MIDI, the velocity is represented as a number between 0 and 127. As with the large amount of drum sounds, this create an unnecessary high dimensionality and leave velocities that are sparse in the dataset. Therefore, the velocities are mapped to four equally spaced buckets. Equation 3.1 shows how the different buckets are created.

$$b_n = \begin{cases} 1 & \text{if } v_n \in [0, 32] \\ 2 & \text{if } v_n \in [33, 64] \\ 3 & \text{if } v_n \in [65, 96] \\ 4 & \text{if } v_n \in [97, 127] \end{cases} \tag{3.1}$$

b_n = bucket velocity of note n

v_n = original velocity of note n

Pitch-Velocity Token

Nuttall et al. [58] found out through qualitative evaluation that the model performed better when they did not use pitch and velocity as separate tokens but instead combined them into a pitch-velocity token. As there are 9 different pitches and 4 velocity buckets, the total number of unique pitch-velocity tokens is 36.

Time Tokenization

Firstly, to keep the rhythm consistent, every element in these sequences is quantized to the nearest 16th note. To further reduce the dimensionality of the input sequence, time is tokenized into only five different tokens. Here, Nuttall et al. are inspired by Donahue et al. [74]. The mapping uses ticks as a time unit. It is native to MIDI and represents the lowest resolution possible in a certain MIDI file. In the Groove dataset, the tick resolution is 480 ticks per second. Table 3.2 shows the mapping from ticks to time tokens.

Table 3.2: Mapping between ticks and time tokens.

Time Token	Number of ticks
1	1
10	2
100	3
1000	4
10000	5

These 5 tokens can be combined to represent the duration of every pause between two notes. If a pause is 345 ticks long, it is represented by [3, 3, 3, 2, 2, 2, 2, 1, 1, 1, 1, 1]. If a pause is 5003 ticks long, it is represented by [4, 4, 4, 4, 4, 1, 1, 1]. If two notes are played simultaneously, no time token is used at all.

The Final Representation

By combining the mentioned representation, in the end every sequence can be formulated as in Equation 3.2.

$$MIDIsequence = [(pv_1, t_1), (pv_2, t_2), \ldots, (pv_n, t_n)] \tag{3.2}$$

n = number of notes in a sequence n

pv_n = pitch-velocity token of note n

t_n = time tokens of note n

3.5.2 POP909 Dataset

The rest of the intelligent artificial agents (Bass Agent, Chord Agent, and Melody Agent) are trained on the POP909 dataset. All of them have their own data representation tailored to the complexity of the agent's task and the individual challenges faced by the generation process. However, there are some common modifications made to the dataset for all agents.

Firstly, every song that is not in the 4/4 or 1/4 time signature is discarded. In addition, every track is transposed into the C major or A minor scale. These two scales share the same 7 notes (C, D, E, F, G, A, B) and are therefore compatible. Transposing songs to the same key reduces noise and is inspired by the positive results observed in related studies [80, 104, 129, 139]. Standardization of the key of all training data is expected to facilitate improved model learning. Musical timings as beats are used instead of seconds across all the different agents, as this improves the network performance. This is also the reason why none of the representations use note-off events but simply rely on the duration of a note [85]. It should be noted that the tempo in the POP909 dataset varies between songs, but this implementation does not discriminate based on the tempo, since every temporal element is measured in terms of beats.

Bass

The data used by the Bass Agent are extracted from one of the text files of the POP909 dataset. This file includes information about what chords are being played and for what number of ticks. From the chords, the root note can be easily extracted and the number of beats is derived from the number of ticks and the tempo of the song. As the generation task of the Bass Agent is not very complicated, it can use a simple data representation. The pitch vocabulary comprises 12 tokens, one for each of the 12 pitches in an octave, and the duration vocabulary has 8 tokens. This allows for the representation of $1 - 8$ beats long notes in the span of one octave.

Chord

The POP909 dataset includes many different chord variations. In order to reduce the dimensionality and simplify the generation process, the Chord Agent can only generate triad chord varieties, similar to the work of others [66, 69, 97]. The Chord Agent can play these six triad chords: major chord, minor chord, augmented chord, diminished chord, suspended 2nd chord, and suspended 4th chord.

The chord generation is a conditional task, where the chord variation is conditioned on the root note from the Bass Agent. Meaning, the Chord Agent must know both the previous chord variation and the previous root notes. Therefore, a pair of root note – chord variation is extracted from the dataset documentation files. Both the root note and chord variation information are represented by their own individual token. To generate

a chord variation, a special token masks the unknown chord in the last slot of the chord vector, signaling the agent to generate the chord based on the known root note.

This means that the vocabulary size for the chords is 7, representing the six possible chord variations and the special token. Since there are 12 root notes, the vocabulary size for the root note is 12.

Melody

The Melody Agent is more complex than the two previous agents. Not only shall it keep track of pitch and duration, but the agent is conditioned on both the current and the next chord. Also, to enable better timing, it receives information about the time until the next chord starts. The input is therefore a 117-element long vector consisting of five one-hot encoded sub-vectors: pitch, duration, current chord, next chord, and time left on chord.

Melody - Pitch

The melody information (pitch and duration) is extracted from the monophonic melody track of the POP909 dataset using the *PrettyMIDI* Python library. Since the melody is monophonic, a one-hot encoded vector is chosen to represent pitch. From an analysis of the dataset, it is clear that 98.8% of the pitches are within three octaves, between C4 and C7 (MIDI nr. 60 and 96). This allows for a reduction in the size of the pitch vector to include only 36 pitches and still be able to represent the melody sufficiently. This reduction in vector size is a common practice [80, 81]. By doing so, complexity is reduced, resulting in faster and better training. Notes outside of the three octaves are brought within it by the modulus 12 operator. To be able to represent pauses, an extra element is included for silence. This makes the pitch sub-vector 37 elements long, as illustrated in Figure 3.9.

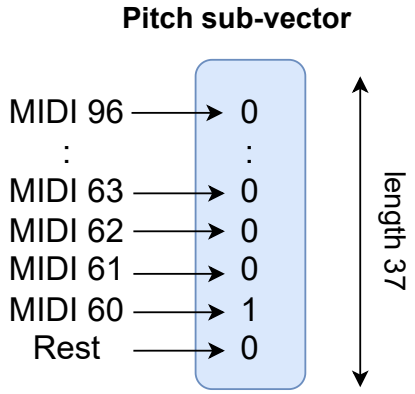


Figure 3.9: The sub-vector that represents the pitch for the Melody Agent.

Melody - Duration

Each pitch is converted into a note with a duration, represented by a one-hot encoded vector of 16 elements, each corresponding to a 16th note. The vector is computed by determining the number of 16th notes covering the note's duration. For example, a quarter note lasts for four 16th notes, activating the 4th element (at index 3) in the vector, as indicated by Equation 3.3. This allows the Melody Agent to play durations from 16th notes to whole notes. The duration sub-vector is shown in Figure 3.10.

$$index = num\_16th\_notes - 1 \tag{3.3}$$

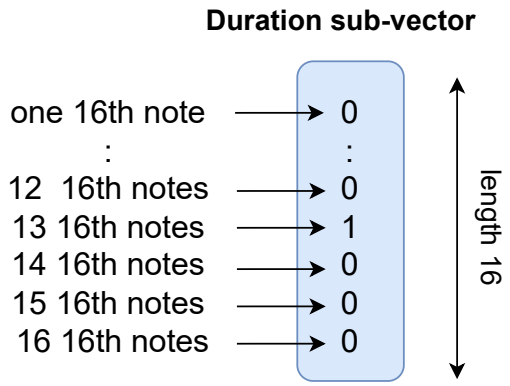


Figure 3.10: The sub-vector that represents the duration for the Melody Agent.

Melody - Current Chord and Next Chord

Since the Melody Agent is conditioned on the chord of the Chord Agent, it must know the chord that is currently playing. Inspired by previous works, [66, 95] the agent also gets information about the next chord the Chord Agent is going to play. The aim is to let the agent prepare for the coming chord and allow it to produce a melody thereby. This is possible as the inference of the different agents does not occur simultaneously but after each other in a hierarchical structure, as discussed in subsection 3.2.1.

The Chord Agent is capable of generating six different types of triad chords: major, minor, augmented, diminished, suspended 2nd and suspended 4th. However, for the melody representation, the complexity has been further simplified. Following the approach suggested by Chen et al. [69], the representation has been restricted to only the major and minor chords. The augmented, diminished, suspended 2nd, and suspended 4th chords are transformed into major chords, as detailed in Equation 3.4. This reduces

the size of the full melody vector by 45%, which likely increases the ability of the agent to learn.

$$ct_n = \begin{cases} \text{Major} & \text{if } oc_n \in \{\text{Major, Aug, Dim, Sus2, Sus4}\} \\ \text{Minor} & \text{if } oc_n \in \{\text{Minor}\} \end{cases} \tag{3.4}$$

ct_n = Chord transformed at place n

oc_n = Original chord at place n

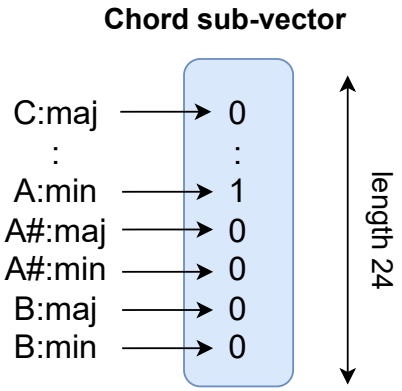


Figure 3.11: The figure shows the sub-vector that represents both the current chord and the next chord for the Melody Agent.

This leaves a one-hot encoded vector of length 24 as shown in Figure 3.11. The same representation is used for both the current and next chord.

Melody - Time left on chord

Lastly, the agent needs some way of keeping track of the time other than just the duration. By using only duration, the agent will know nothing about when a chord is going to end, as chord durations vary. It was first experimented with a vector that indicates the start or end of a bar such as in [66]. Through anecdotal tests, better results were achieved using a more explicit measure of timing, based on the duration vector in subsubsection 3.5.2. Instead of measuring the duration of a note, it measures the duration until the next chord starts, represented in the same way as the duration vector.

By adding this, the Melody Agent seemed to play much more fluently together with the Chord Agent, creating better harmony in the band. The *time left on chord* sub-vector is therefore identical in structure to the melody network's duration vector and can be seen in Figure 3.12.

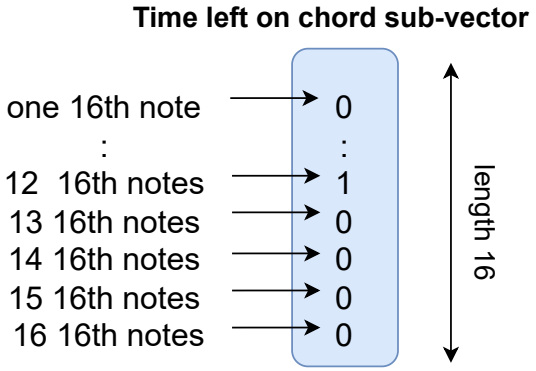


Figure 3.12: The figure shows the sub-vector that represents the time left on chord for the Melody Agent.

Melody - Final Representation

Concatenating all these sub-vectors into one input vector results in a vector of length 117, organized as shown in Equation 3.5. This vector is used as input for the NN of the Melody Agent.

$$i_n = [p_n, d_n, cc_n, nc_n, tloc_n] \tag{3.5}$$

i_n = input vector at place n

p_n = pitches sub-vector at place n

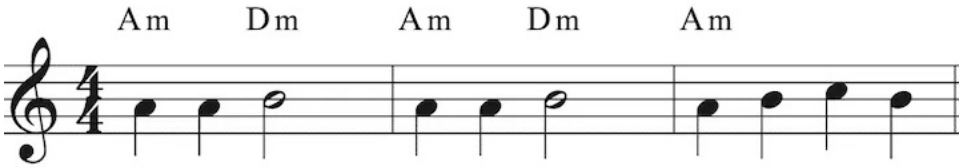
d_n = duration sub-vector at place n

cc_n = current chord sub-vector at place n

nc_n = next chord sub-vector at place n

$tloc_n$ = time left on chord sub-vector at place n

To demonstrate the transformation from sheet music to the melody representation, Figure 3.13 shows a note sequence that has been transformed into an input vector consisting of the five sub-vectors.



P*	A (9)	A (9)	B (11)	A (9)	A (9)	B (11)	A (9)	B (11)	C (12)	B (11)
D*	4 16th (3)	4 16th (3)	8 16th (7)	4 16th (3)	4 16th (3)	8 16th (7)	4 16th (3)	4 16th (3)	4 16th (3)	4 16th (3)
CC*	Am (19)	Am (19)	Dm (5)	Am (19)	Am (19)	Dm (5)	Am (19)	Am (19)	Am (19)	Am (19)
NC*	Dm (5)	Dm (5)	Am (19)	Dm (5)	Dm (5)	Am (19)	Dm (5)	Dm (5)	Dm (5)	Dm (5)
TLOC*	8 16th (7)	4 16th (3)	8 16th (7)	8 16th (7)	4 16th (3)	8 16th (7)	16 16th (15)	12 16th (11)	8 16th (7)	4 16th (3)

* *P* = Pitch, *D* = Duration, *CC* = Current Chord, *NC* = Next Chord, *TLOC* = Time Left On Chord

Figure 3.13: The figure shows how a sequence of notes would be represented. *Pitch* shows the index of the pitch starting from C4, *duration* shows the duration in number of 16th notes, *current chord* and *next chord* shows the chord type, and *time left on chord* shows the number of 16th notes until the next chord starts. The numbers in parenthesis are the indices of the respective sub-vector.

3.6 Networks and Training

The final step in understanding the agents is to look at the components responsible for data processing and music generation: their NNs. This section will look at the architecture of each agent's network, detailing both the training methodologies and the generation processes employed. For a complete insight into the structure of each model, refer to the project code [28]. Detailed information on hyperparameters and other network details is provided in Appendix A.

3.6.1 Drum

Similar to the data processing approach for the drum dataset (Groove), the network architecture and training draw significant inspiration from the "Bumblebeat" project by Nuttall et al. [58]. Therefore, for a full explanation of drum generation, please refer to the work of Nuttall et al. [58].

Network

The Drum Agent uses a Transformer-XL [72] network. It is an improved version of the vanilla transformer, which has been shown to struggle when sequences become too long [74]. This is addressed in the Transformer-XL architecture by introducing hidden states and relative positional encoding.

Compared to the network by Nuttall et al. [58], the size and complexity of the network in this project are reduced. These parameters have been halved: The number of transformer layers, the memory length (cache), the positional dimension size, the input / output embedding dimension, the number of attention heads, the dimension of the attention head, and the hidden size dimension for the feedforward NN.

To mitigate the effects of reduced complexity, this project limits the generation to a single bar of drum music, which is then looped rather than generating extended sequences as seen in the original drum generation project. This approach also significantly reduces the inference time, thus accelerating the generation process.

In the work by Nuttall et al. [58], they experiment with two methods for generation: (1) the generation of a new sequence and (2) the continuation of an existing sequence. They achieve the best performance by continuing an existing sequence. This method works by feeding the network a primer sequence of events in a specific genre and letting the network generate the next events. This method of continuation has been adopted in this project and is how music is generated in a specific genre, chosen by the Human Composer Agent.

Training

The Drum Agent uses several different training techniques. First, it uses the Adam optimizer with a learning rate of 0.00001. A cosine annealing learning rate scheduler gradually reduces the learning rate throughout the training to allow for finer adjustments to the weights as the model converges. The training also applies gradient clipping, preventing the exploding gradient problem by capping the gradients at 0.25. This ensures that the training remains stable.

3.6.2 Bass

Network

The bass network is built around an LSTM. The inputs are embedded using two separate embeddings, one for the pitch input and one for the duration input. These embeddings are then concatenated before being passed onto a 4-layered bidirectional LSTM network. The output of the LSTM is passed through two parallel FC layers, one that predicts the next pitch and one that predicts the next duration. A FC network, also known as a dense network, refers to a network architecture where every node in one layer is connected to every node in the subsequent layer. The model architecture can be seen in Figure 3.14.

Training

The bass network is trained using an Adam optimizer with a constant learning rate of 0.0001. Due to the simplicity of the model and the large amounts of data available in the dataset, the model is not trained on all the data in the training set at each of

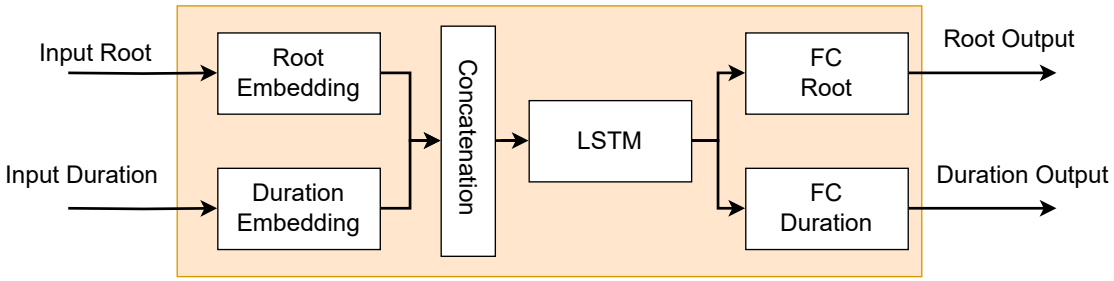


Figure 3.14: The architecture of the bass network.

its 100 epochs. Instead, each epoch stops after 1000 batches. This allows the agent to potentially be exposed to the entire training set, without having to train for an extensive amount of time.

The network outputs two distinct tokens: one representing the root note and another for the duration. These elements vary in both their difficulty to predict and their significance for the overall composition. To address this, two alpha (α) parameters are introduced, influencing the loss function and thereby adjusting the focus of the network. This allows for a differentiated emphasis, penalizing inaccuracies in one output more than the other. The total loss of the network is calculated as specified in Equation 3.6.

$$\text{Loss} = \alpha_1 \times \text{loss}_{\text{root}} + \alpha_2 \times \text{loss}_{\text{duration}} \tag{3.6}$$

Through trial and error by subjective listening and analyzing the accuracy and log-likelihood of the model, α_1 (root) is set to 0.7, while α_2 (duration) is set to 0.3.

3.6.3 Chord

Network

The chord network is built with an architecture similar to that of the bass network but with some minor adjustments. Unlike the Bass Agent, the Chord Agent's task is to predict only one element, the chord variation. Therefore it only requires one FC layer in the network's output layer. See Figure 3.15 for the full architecture.

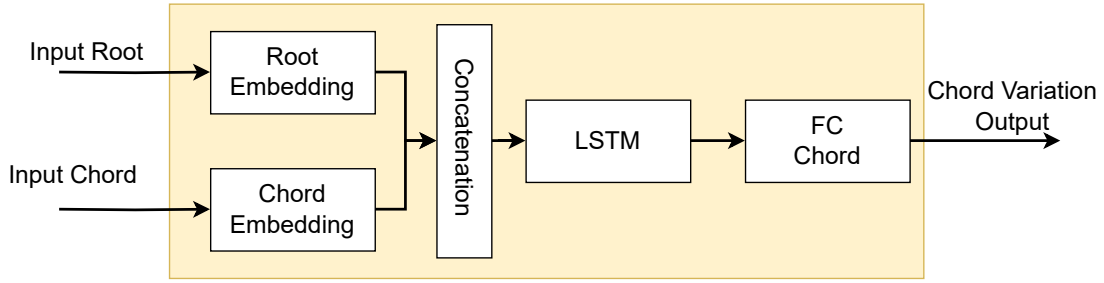


Figure 3.15: The architecture of the chord network.

Training

The chord network training uses the same Adam optimizer with a learning rate of 0.0001. However, since generating only one chord variation is a simpler task than generating a root and duration, the agent trains for only 50 epochs, and each epoch sees only 200 batches.

3.6.4 Melody

Network

The melody network is far more complex than the bass and chord networks. In melody generation, capturing structure is very important, and the network is therefore constructed hierarchically, as this has been proven to be good at capturing structure in sequences [62–64]. The Melody Network has three tiers, with higher tiers focusing more on the larger structures. Each tier consists of several network blocks. Tier 2 and 3 consist of LSTM blocks and use them to process the sequential data, while Tier 1 consists of FC blocks.

The network receives inputs of 16 events at a time, each represented as a vector detailed in subsubsection 3.5.2. These events represent individual notes, including silences, in addition to chord and timing information. In ML-jam, the default length of a loop in this system is set to 4 bars. During these bars, there are usually somewhere between 8–16 melody events. The length of an input sequence is therefore set to 16 events. For a more complex network and a greater understanding of large structures in the music sequence, one could have a larger input sequence. This was done by Zixuan et al. [66], which has a similar hierarchical architecture, but a 44-event long input sequence. This is not deemed necessary in this project due to the mentioned default loop length and the need for a low inference time, as the systems should feel quick and responsive, and an increased complexity will lead to an increased inference time.

The 16 divided events are spread out and given as inputs to the individual blocks. The upper tiers receive a concatenation of several events as input per block, thus focusing on larger structures, while the lowest tier only receives one event per block. The latent output of the upper tiers is passed down to the tiers below. By doing this, tiers that initially focus only on structures on a note-by-note basis are influenced by the larger structures of the sequence. In addition to passing latent information to the tiers below, each network block passes information onto the next block on its tier. By doing this, the last block in each tier will have indirect information about every event in the sequence. The last network block in Tier 1 is a prediction block consisting of two parallel FC layers that predict the next pitch and duration. The complete network architecture can be seen in figure 3.16.

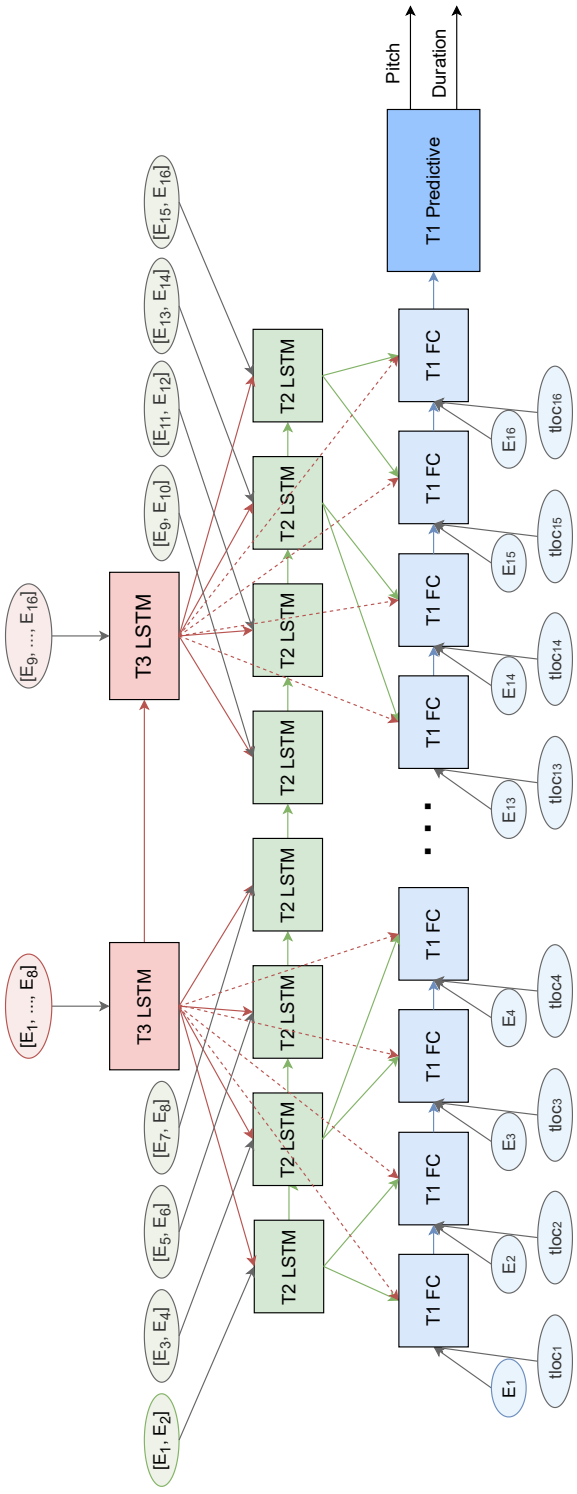


Figure 3.16: The full 3-tier hierarchical melody network. E_n is Events, and $tloc_n$ is the *Time Left On Chord* vector. The upper tier has two T3 blocks that process eight new melody events each. The middle tier has eight T2 blocks that process two new events each, while the lower tier has 16 T1 blocks, each processing just a single new event. Each block passes on information to the next block. T2 and T3 blocks also pass on information to blocks in layers below.

Tier 3 LSTM block

The upper tier primarily handles the processing of large-scale structures. Each block within this tier, as depicted in Figure 3.17, processes eight consecutive events. Notably, the second block also incorporates the output from the first block, concatenating it before processing through a bidirectional LSTM. The LSTM configuration includes two layers, each with a hidden size of 256, and employs a dropout rate of 0.5 to prevent overfitting. The output of the LSTM is then downsampled through an FC layer. This is for the latent information to match the Tier 1 and Tier 2 input in dimension.

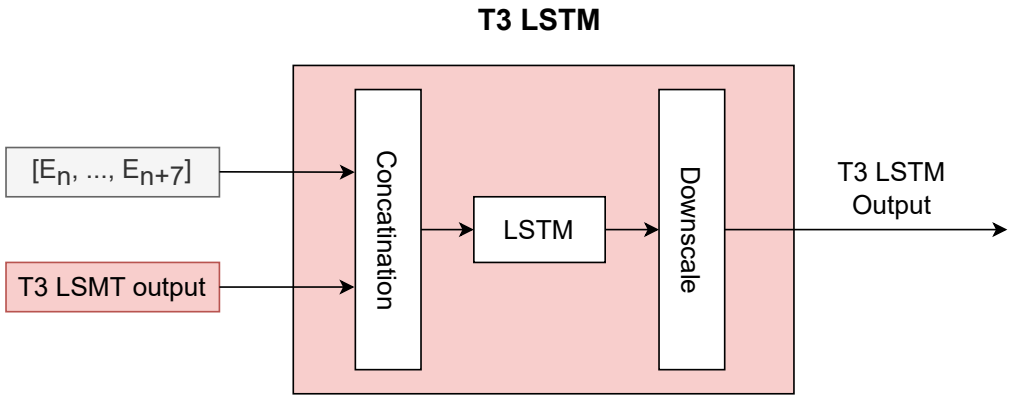


Figure 3.17: The internal structure of a Tier 3 LSTM block. The first Tier 3 block does not get a T3 LSTM output, as it is not yet generated one.

Tier 2 LSTM block

Depicted in Figure 3.18 is the block that makes up the second tier of the melody network. It is the same as Tier 3 in terms of internal structure with a bidirectional LSTM equipped with the same two layers, a hidden size of 256, and a dropout of 0.5. The main difference is that the blocks in this tier only input two subsequent events each. This creates the need for 8 blocks in series to process the 16 event input. Unlike in Tier 3, each block (except for the first) will get latent information not only from the previous block but also from the blocks in Tier 3.

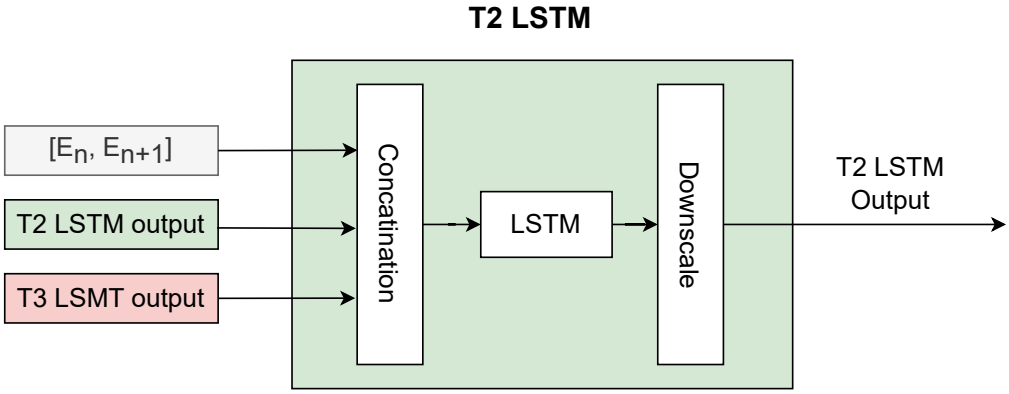


Figure 3.18: The internal structure of a Tier 2 LSTM block. The first block in the sequence will not get a T2 LSTM output, as it is not created one yet.

Tier 1 FC block

The first Tier is different from the two previous ones. It only receives one event at a time and includes a small Convolutional Block that processes the event input. After the Convolutional Block is done, the output is concatenated with the latent output of the corresponding blocks in Tier 2 and Tier 3, as well as the previous T1 FC Block (if there is one). This concatenated vector is then passed through a single FC layer. The block can be seen in Figure 3.19.

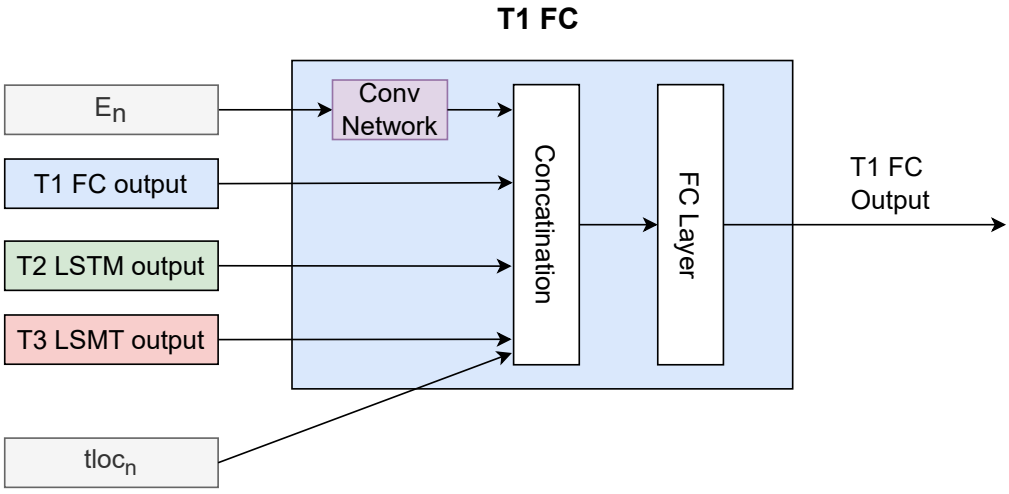


Figure 3.19: The internal structure of a Tier 1 FC block. If this is the first block in the sequence, there is no T1 FC output yet.

Convolutional Block

The small convolutional block is added because it has been shown to have a good effect in similar studies [69, 140, 141]. The convolutional block can be seen in Figure 3.20, and has the goal of helping the Tier 1 FC blocks by extracting features from events. The convolution is a 1d convolution over the event with a kernel size of 4 and a stride of 1. After the convolution, the output is passed through a Rectified Linear Unit (ReLU) activation function, before being average pooled and downsampled.

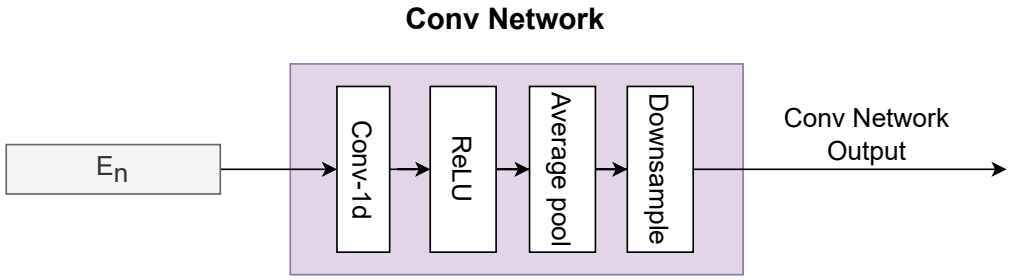


Figure 3.20: The internal structure of a convolutional block.

T1 Prediction block

After the last T1 FC block has processed what is indirectly every element three times, a final latent output is produced, This output is fed into the last block, a prediction block as seen in Figure 3.21. It consists only of two FC layers. The two layers try to predict the next pitch and duration.

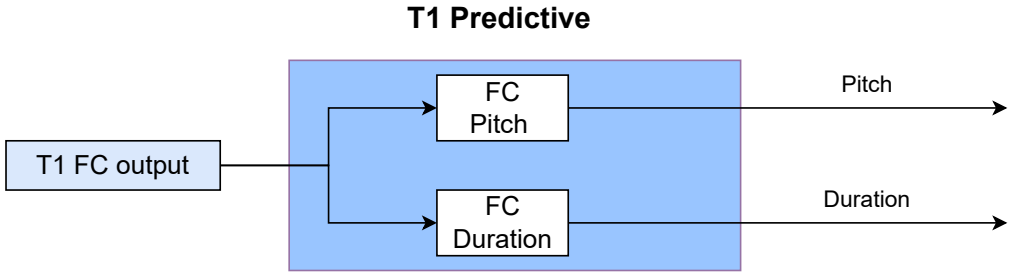


Figure 3.21: The internal structure of a Tier 1 Predictive block.

Training

Similarly to the other agents, the melody agent trains using the Adam optimizer, but with a learning rate of 0.005. To combat overfitting, weight decay was added. This punishes large weights that often lead to overfitting. Unlike the bass and chord networks, the melody network is trained on the entire POP909 dataset for all of its 100 epochs

for a total of 29766 events. Similarly to the bass network, it produces two outputs, pitch and duration. Therefore, two alpha values are added, which work as described in Equation 3.6 for the bass network. α_1 (pitch) has been set to 0.6, and α_2 (duration) has been set to 0.4.

3.6.5 Generation

Primer Sequence

The generation process is similar for every agent. A primer sequence is chosen. When it is the system's first generation, the primer sequence is chosen randomly. However, the bass, chord, and melody primers are chosen from the same sequence in the dataset. This alignment is crucial as it ensures that all agents operate on a consistent foundational basis and facilitates inter-agent communication.

For subsequent generations, if a new loop is created, a primer is assembled from the events of the previous loop, supplemented by the prior primer sequence when necessary to meet the length requirements of the new primer. The Drum Agent selects a new primer sequence for every generation, choosing it from the genre to which it is selected to play.

Sliding Window Approach

As each agent predicts a new event, the primer sequence is dynamically updated using a sliding window approach: the last event is removed, and the newly predicted event is appended at the end. This method maintains a continuous flow in the generation process.

3.7 Statistical Methods

A variety of scientific experiments will be carried out in chapter 4. Then, statistics will be used to show whether the results are significant and therefore unlikely to appear by chance. There are several different tests, useful in different scenarios based on the type of data. This section will go over some of the tests that are relevant in the coming experiments.

3.7.1 Number of Comparable Groups

When performing statistical tests, the objective is often to determine whether there is a statistically significant difference between multiple distributions. For scenarios involving only two groups, tests such as the t-test [142] or the Wilcoxon signed rank test [143] are typically used. However, when the analysis involves more than two groups, alternative methods must be employed. These include the Analysis Of Variance (ANOVA) test

[144], which is designed for comparisons between multiple groups, the Mann-Whitney U test [145], and the Friedman test [146][2].

3.7.2 Parametric vs. Nonparametric

The variety of statistical tests mentioned caters to different analytical needs depending on the nature of the data distributions involved. Parametric tests, such as the T test and ANOVA, require the assumption that the data are normally distributed. Various tests are available to verify normality, where the Shapiro-Wilk test [148] is often preferred due to its reliability [149, 150].

When data do not adhere to a normal distribution, non-parametric tests become necessary. These tests, including the Wilcoxon signed rank test, the Mann-Whitney U test, and the Friedman test, do not assume normality and are thus suitable for analyzing distributions that deviate from this condition. By choosing the appropriate statistical method based on the characteristics of the distribution, it can be ensured that the conclusions are valid and reliable.

3.7.3 Comparing Proportions

When categorical data such as accuracy or error rates are evaluated, which categorize outcomes into binary results such as "success" or "failure," analysis often involves comparing proportions rather than continuous distributions. In such cases, the two-proportion z-test is a simple and effective statistical tool to determine whether there is a significant difference between two proportions [151].

3.7.4 Correction

In research where multiple hypotheses are tested simultaneously, the likelihood of encountering a false positive or a Type I error increases. This phenomenon, known as the multiple comparison problem, can lead to misleading conclusions if not properly managed. To mitigate this risk, statistical corrections are applied to adjust the significance thresholds. Common methods include the Bonferroni correction [152] and False Discovery Rate (FDR) [153].

[2]Statistical tests for multiple groups are often extensions or adaptations of methods originally developed for two-group comparisons [147].

3.8 Summary

This chapter has showcased the implementation of ML-jam, covering both broad and fine aspects. Key points to remember before proceeding to the experiments include the modular construction of ML-jam, which comprises five independent artificial agents and a human agent. These artificial agents communicate with each other and can be steered by the human agent through a digital UI. To summarize, Figure 3.22 from the introduction of the methods chapter is revisited, highlighting the system's modular structure and the interactions between the agents.

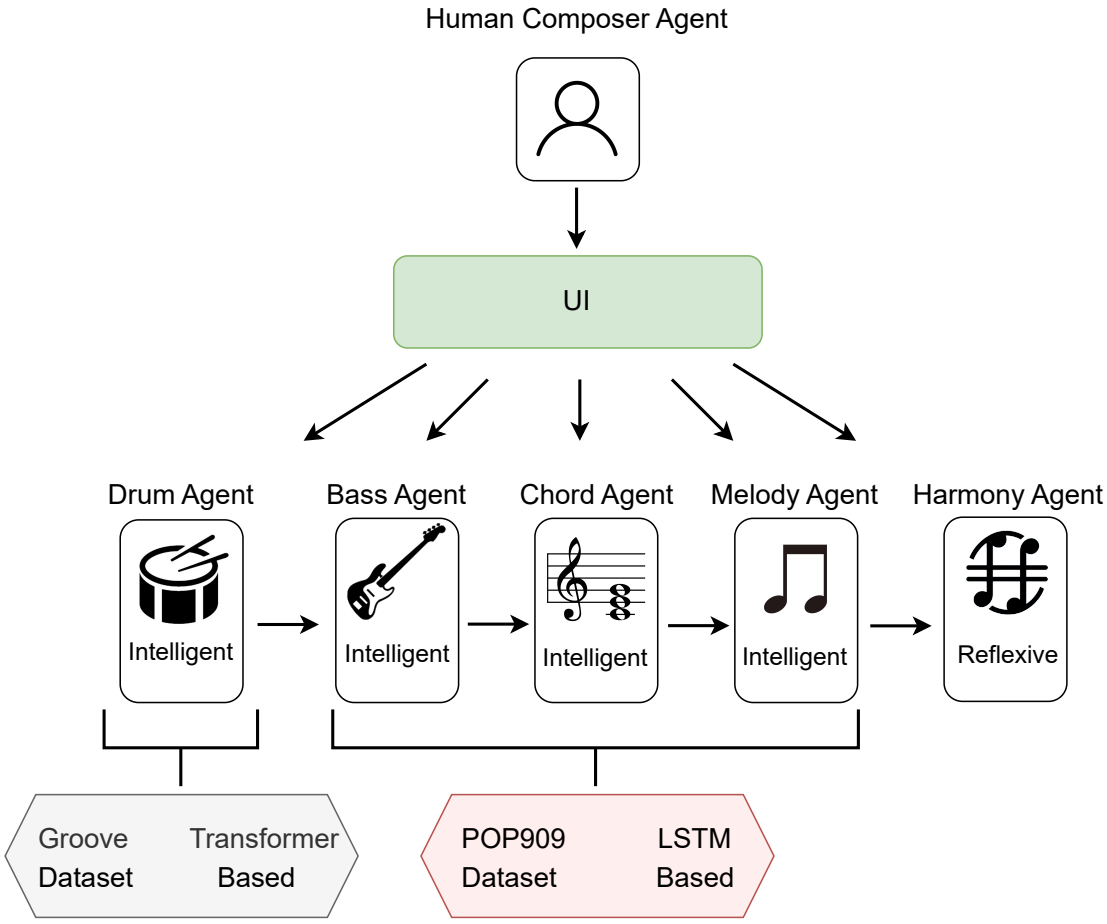


Figure 3.22: The system architecture with the different agents and how they interact. Arrows indicate information flow.

Chapter 4

Experiments and Results

ML-jam has been developed to explore the hypothesis proposed by Dadman et al. [14], which suggests that using a band-like multi-agent architecture for music generation could offer advantages over traditional methods. Aiding in this exploration, the thesis poses one main research question and four sub-questions, introduced in section 1.1, with the main focuses being the effects of the modular multi-agent architecture, inter-agent communication, and human interaction. Five different experiments have been designed to answer these research questions and in turn assess the validity of the claim by Dadman et al. An overview of the five experiments, their main methodology and the research sub-questions they are primarily aimed at, is provided in Table 4.1.

Table 4.1: An overview of the five experiments conducted in this section, together with the main methodology and research sub-question the experiments are aimed towards.

Experiment	Methodology	Research Sub-Question
Multi-Agent vs Monolithic	Quantitative	RSQ1
Investigating the Impact of Communication on the Melody Agent	Quantitative	RSQ2
Scrambled Communication - System Review	Quantitative	RSQ2, RSQ3
Evaluation of Communication Effects using Listening Test	Quantitative	RSQ3
Interactive User Study - Informal Feedback from Experienced Musicians	Qualitative	RSQ4

Several studies [27, 154, 155] point to the challenges associated with the evaluation of musical systems and therefore the need for multiple evaluation techniques. They caution that relying solely on one type of evaluation can be misleading, as different tests may reveal different facets of the system. Thus, achieving favorable results in one specific experiment does not necessarily predict similar outcomes in other dimensions of the

system. An example pointed out by Theis et al. [27] is that listening tests are prone to favor models that overfit. These models should therefore also be evaluated by methods such as log-likelihood, which favors models with higher entropy.

Because of this, the experiments conducted in this thesis incorporate a variety of subjective and objective evaluations. Accuracy and log-likelihood were chosen as evaluation metrics in part because they are simple, quantifiable metrics commonly used in the field of generative music evaluation [73, 74, 77, 81]. This makes them more relatable and comparable. Both accuracy and log-likelihood try to capture how well the model manages to reflect the underlying patterns in the data but through slightly different approaches. Accuracy measures the proportion of correct predictions made by a model compared to the ground truth. Log-likelihood, on the other hand, evaluates how well the probability distributions output by the model align with the observed data (ground truth), quantifying the plausibility of the data given the model parameters. Log-likelihood is mentioned as a good metric to use to supplement subjective evaluations [27], which are also included in this thesis.

Since the drums network and data representation in this project are created by Nuttall et al. [58], and thoroughly examined by them, no experiment will focus solely on the evaluation of the drums; instead, they will be evaluated as part of the system.

Each experiment will be conducted individually, and the results will be presented after each experiment. A brief discussion of the results will follow after each experiment, while a more comprehensive discussion that contextualizes the findings within the broader context of the system will be presented in section 5.7.

4.1 Experiment 1: Multi-Agent vs. Monolithic

The theory of Dadman et al. [14], on which this thesis is based, states that by using a MAS architecture, individual agents can concentrate on generating instrument-specific music. This is through the naturally occurring modular design achieved by creating a band-like multi-agent architecture.

Modular design and task decomposition are used to increase performance in a range of applications [16–19]. Nordaunet et al. [156] demonstrated that an agent using a network selection architecture, which alternates between two networks randomly, outperforms an agent using a single network. This suggests significant potential for modular designs with specialized networks in enhancing performance. There are also several music generation systems that, although not designed to be multi-agent, use modular designs, be it through several decoders [102], hierarchical network designs [97], or by using multiple networks [98].

This separation of tasks is one of the main attributes of MAS, and this first experiment focuses on finding out if it can help improve the performance of the musical system. Ultimately, it tries to help answer the research question through the first sub-question:

> How does the modularity natural to a multi-agent architecture compare against a monolithic architecture in terms of performance and usability?

More specifically, the experiment tries to answer if the modular design naturally occurring in the band-like multi-agent architecture improves performance compared to a more traditional monolithic design, using only a single network.

4.1.1 Experiment Outline

To investigate this, a quantitative experiment has been designed around a new network. This network is a combination of the Bass Agent and the Chord Agent. It has the same inner architecture as the bass and chord network, but with adjustments made in the input and output layers. It single handedly aims to do the same job as the networks of the Bass and Chord Agent do together; meaning it will receive the previous root note, duration, and chord variation before trying to predict the next root note, duration, and chord variation.

For this experiment, the bass and chord networks are trained using the same hyperparameters, where the LSTM depth is set to 4 layers deep. The monolithic bass-chord network is tuned identically, but with an 8-layer deep LSTM. This matches the combined length of the two LSTMs from the bass and chord networks. The full list of hyperparameters for this experiment is available in Appendix C.

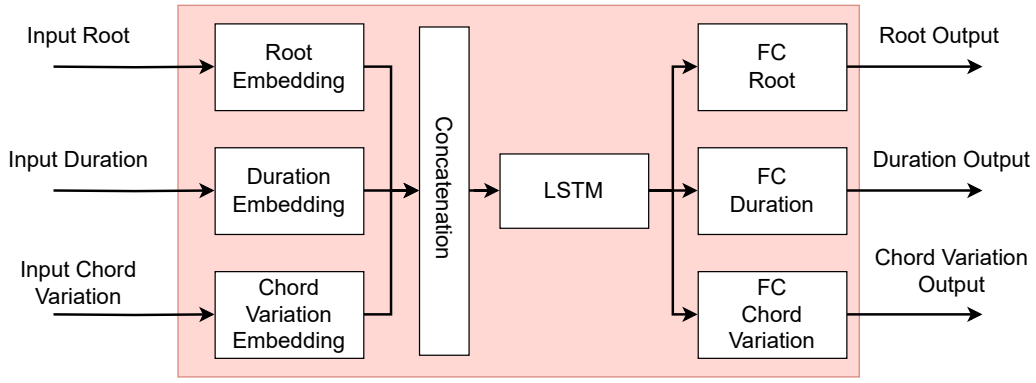


Figure 4.1: The architecture of the monolithic bass-chord network.

After identical training, the networks are evaluated on 10,000 unseen random sequences from the test dataset. Performance is evaluated based on accuracy and log-likelihood across four aspects: root note generation, duration generation, chord variation generation, and a combined assessment. The combined assessment examines the network's capability to simultaneously generate all three of the previously mentioned features.

4.1.2 Results and Discussion

The distribution of log-likelihood for the different metrics is shown in Figure 4.2.

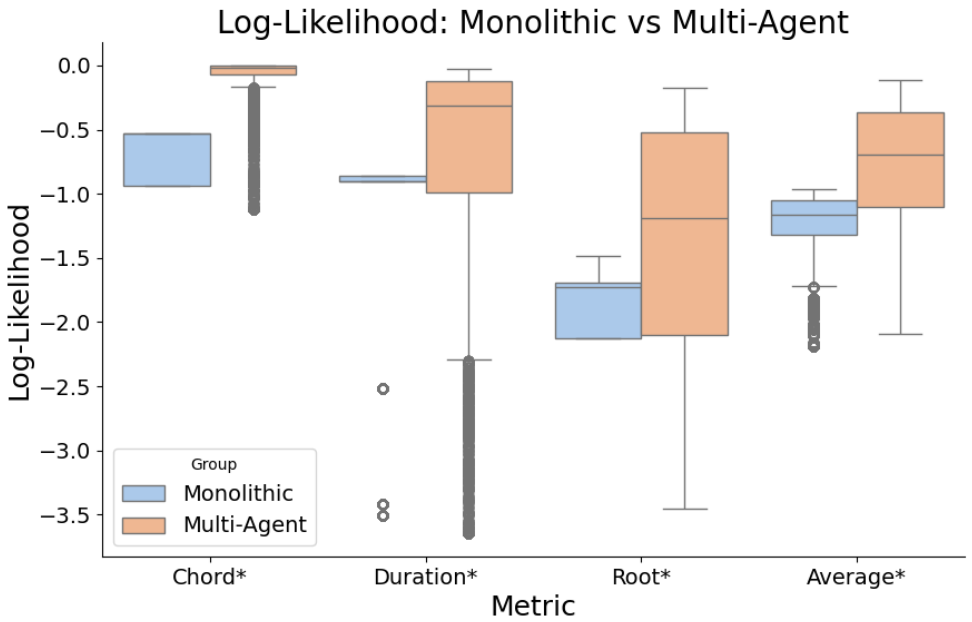


Figure 4.2: Log-likelihood distribution for all metrics. A star (*) indicates statistically statistical difference between performance of multi-agent vs. monolithic. The siginficance level is Bonferroni corrected ($m = 4$) to $p = 0.013$. For an improved visual representation, the data have undergone a Winsorization transformation for the upper and lower 5th percentiles.

Since a Shapiro-Wilks test [148] shows that the data are not normally distributed, the statistical significance between the two groups (monolithic and multi-agent) is calculated using a Wilcoxon signed rank test [143], before being Bonferroni corrected for the four hypotheses tested ($\alpha = 0.013$). In the figure, statistical significance between the groups is marked with a star (*). For a better visual representation, the data have undergone a Winsorization transformation for the upper and lower 5th percentiles, limiting the extreme values.

As the figure shows, there is a statistical significance across every metric, showing that in ML-jam, bass and chord generation through multi-agent architecture is superior to the presented monolithic network.

Although there are a total of 10.000 data points for each metric in each group (multi-agent and monolithic), an interesting observation is the large number of outliers for the multi-agent group. This high variance in the predicted data from the multi-agent model could be the product of overfitting [157], where the model is specialized on the training data, but performs poorly on new unseen data.

To check this, a similar evaluation was performed, but instead of comparing monolithic vs. multi-agent, both network systems now had the same multi-agent architecture. However, one network system was evaluated on the seen training dataset, and the other on the unseen test dataset. The results are shown in Figure 4.3 (these data are not Winsorization transformed).

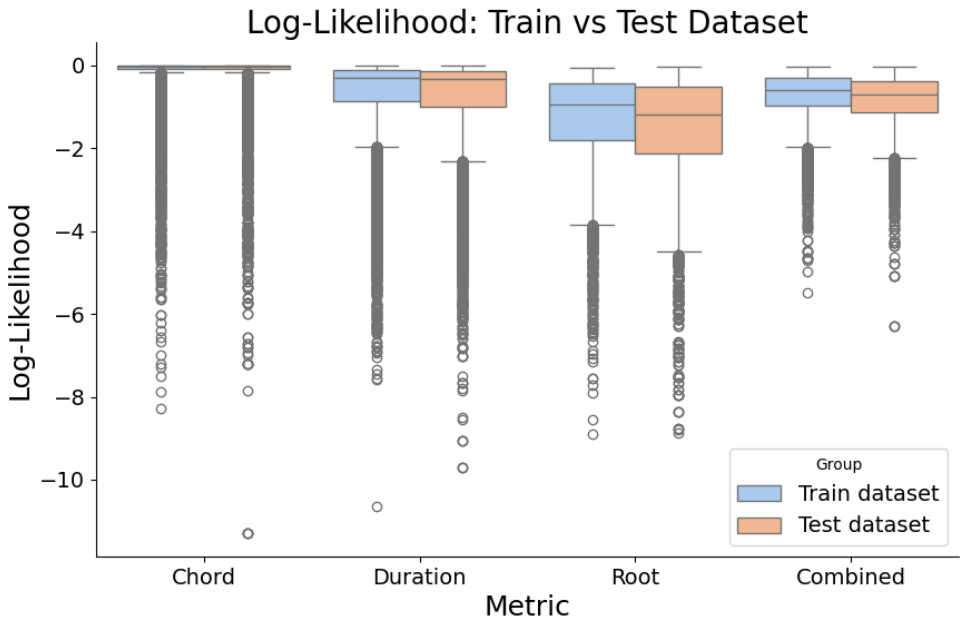


Figure 4.3: The log-likelihood distributions of two multi-agent networks, one evaluated on training data the other on test data. This test is conducted to see if there are signs of overfitting.

The graph shows that the previously seen training dataset performs slightly better than the test dataset. This is as expected. It also shows a similar amount of outliers in both groups, indicating that the large amount of outliers for the multi-agent approach in Figure 4.2 is not caused by overfitting. The reason is probably rather that the monolithic network is unable to learn much from the data, resulting in underfitting and a consistently mediocre performance.

Returning to the main experiment, Table 4.2 shows the accuracy of the two models. The multi-agent approach beats the monolithic in all four accounts on both accuracy and log-likelihood and with statistical significance, calculated by a z-test [151] and Bonferroni corrected ($m = 4$) ($p = 0.013$).

Table 4.2: Accuracy (higher is better) of the multi-agent and the monolithic architecture. The systems are tested on the generation of root, duration, and chord variation, as well as the ability to generate the combination of all three. Bold indicates best performance.

Generation	**Multi-Agent**	**Monolithic**	***p*-value**
Root Note	**36.7%**	21.3%	< 0.013
Duration	**61.5%**	57.6%	< 0.013
Chord Variation	**90.5%**	57.6%	< 0.013
Combined	**21.8%**	8.3%	< 0.013

Concluding Discussion

The results demonstrate the effectiveness of modular design and, to some extent MAS in scenarios where tasks naturally lend themselves to distribution among multiple agents. In this experiment, the MAS network design yields superior results compared to monolithic. A comparable result using a monolithic approach would require either a deeper, more computationally heavy network or a more sophisticated network design. This is both costly and time consuming and requires more knowledge of complex network design to achieve [158]. In contrast, the MAS architecture can achieve better results with several simple networks, making it an attractive alternative in suitable contexts.

This improved performance can also answer why several music generation projects already use modular-like designs [60, 66, 97, 101–104], as they ultimately are a way of creating specialized networks or network sections in a similar way to the multi-agent approach used in this project, possibly benefiting from the same improved performance.

This finding reinforces the assertion of Dadman et al. that "modular and hybrid characteristics can help to alleviate the shortcomings and challenges of the music generation tasks" [14]. The results demonstrate that the modularity inherent in a multi-agent design can indeed enhance performance relative to monolithic architectures in generative music systems, while allowing the use of simple network designs. This effectively answers the first research sub-question, showcasing the practical benefits of employing a modular approach in the context of music generation.

4.2 Experiment 2: Investigating the Impact of Communication on the Melody Agent

Although evidence from the previous experiment suggests that dividing tasks among multiple agents can enhance the performance of a model compared to a singular monolithic approach, this alone does not confirm the utility of features inherent to MAS. The next couple of experiments therefore aim to investigate one of the core elements of ML-jam, which is also fundamental in many other systems using intelligent agents: communication. Specifically, the main goal of this experiment is to assess whether communication between agents leads to improved musical output for the Melody Agent, and subsequently help answering the second research sub-question:

> How does communication between autonomous agents affect the musical output of individual agents?

4.2.1 Experiment Outline

This second experiment is a quantitative comparative analysis between two versions of the melody network. The first network is the exact network used in this system by the Melody Agent, presented in subsection 3.6.4. It will also have the same input vector as presented in subsubsection 3.5.2, which means that it communicates with the Chord Agent, conditioning it with information from the chord domain. In this experiment, this will be called a coop network. The other network is a modified version of the coop network that is not conditioned on information in the chord domain. This means that the input vector now only contains the two sub-vectors: pitch and duration. This network will be referred to as the non-coop network. With the exception of the mentioned difference in input data, the network architecture and data representation are the same. The exact hyperparameters used can be seen in Appendix D.

In similar MAS where agents communicate information among themselves, networks are often conditioned in some way on the information communicated [104, 114]. In cases like this, the dimensionality of the input data can increase. In this experiment, the input sequence at each event for the coop network has a length of 117 elements, while the non-coop network's input sequence is only 53 elements long. This is a 121% increase in the input dimensionality of the coop network.

Although this is a previously studied matter [159–161], the results will vary as musical representation and the complexity of the model is different. Conditioning may be beneficial with some systems and not for others. Genchel et al. [159] experiment with conditioning a melody network with different types of chord information, and performance vary. Sometimes, conditioning on more information results in lower performance. As [162] states, the conditional generation of musical components is faced with the challenge that the data structure is not robust enough. The experiment will

therefore to some extent also capture how well the music representation and network complexity is adjusted to chord conditioning.

This experiment aims to see whether communication leading to agents conditioned on data from other domains increases the performance of the agent or if the negative effect of a larger input dimensionality outweighs the possible benefits. This will be done by training the two different versions of the melody network using the same parameters and then evaluating them on a large amount of unseen data.

After training the networks, they are evaluated on the entire melody test dataset. This is done to get results as accurate as possible. This dataset consists of 29,677 melody sequences previously unseen to the networks. Like in the previous experiment, the models are then evaluated in terms of accuracy and log-likelihood.

4.2.2 Results

Since both the coop and non-coop networks have the same output, the loss values during training are comparable. Figure 4.4 shows the loss curve of both agents during training.

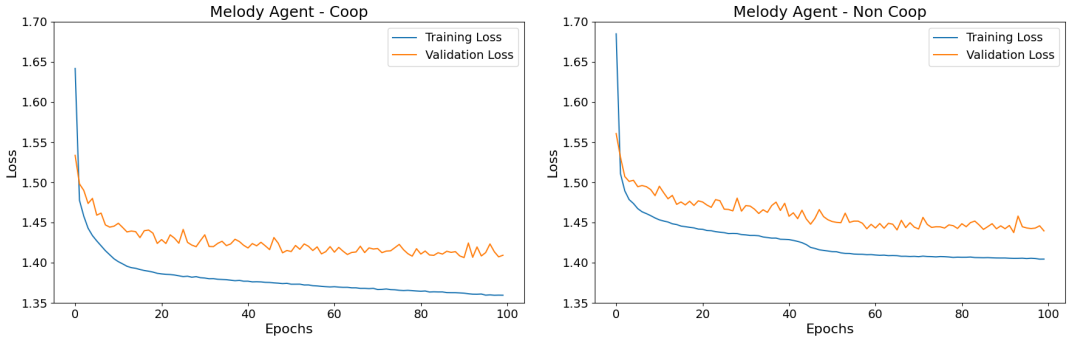


Figure 4.4: The training and validation loss of the coop and non-coop networks.

The curve shows signs of similarity in terms of training, but with a bit more stable curves for the coop network. In addition, the coop network ends with a lower training and validation loss.

The distribution of the log-likelihood shown in Figure 4.5 illustrates two pairs of distributions for each of the two metrics, pitch and duration. The data are Winsorization transformed for an improved visual representation. Again, a Shapiro-Wilk test [148] shows a nonnormal distribution, so a Bonferroni-corrected [152] ($m = 2$) Wilcoxon signed rank test [143] is performed, revealing a statistical significant difference ($\alpha = 0.025$) between the two groups for pitch and duration. Although the two distributions appear visually to be quite similar, the large number of samples ($n = 29766$) ensures that even smaller differences can be statistically significant.

Through Table 4.3, the accuracy of the two models is presented. A Bonferroni corrected ($m = 2$) z-test [151, 152] is used to calculate the p-value, again showing statistical significance ($\alpha = 0.025$). This further strengthens the idea that the

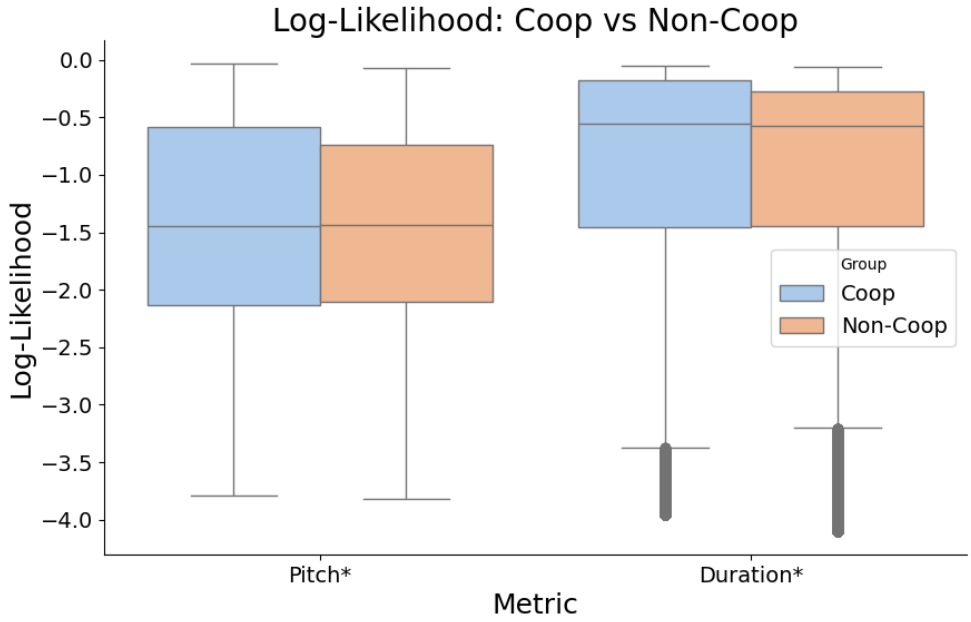


Figure 4.5: Log-likelihood distribution for pitch and duration. A star (*) indicates statistical statistical difference between performance of coop and non-coop. The significance level is Bonferroni corrected ($m = 2$) to $\alpha = 0.025$.

performance of the melody network is improved by being conditioned on chord information.

Table 4.3: Comparison of network accuracy for coop and non-coop networks. Numbers in bold indicate best performance.

Measurement	Coop	Non-Coop	p-value
Pitch	**35.1%**	33.9%	< 0.025
Duration	**52.8%**	50.9%	< 0.025

Discussion with Respect to Comparable Work

In addition to log-likelihood and accuracy, the evaluation negative log-likelihood loss was calculated for the two different networks. This was done to compare the results with the work of Genchel et al. [159], who use this as a metric to compare different conditionings of a generative melody network. Similar to the melody network used in this experiment, the network by Genchel et al. is trained to generate pitch and duration. They compared several different conditioning methods, but I will focus on their comparison of *No conditioning* and *Chord + Next Chord*.

In their research, Genchel et al. achieved 22.5% and 59.0% difference in negative log-likelihood loss between *No conditioning* and conditioned with *Chord + Next Chord* for pitch and duration respectively [159]. Comparably, my networks only achieved a difference of 1.3% and 3.8%. Although direct comparison is inaccurate and depends on

several variable factors, such as the dataset and training, this highlights a clear limitation of my melody network.

The reason for this is unclear, but some main differences between my network and that of Genchel et al. [159] is that firstly, they use two networks, one for pitch generation and the other for duration generation. This may exploit the positive effects of modularity hypothesized in section 4.1. Secondly, they process the pitch, duration, currect chord, and next chord information separately before merging the latent information, instead of processing everything as one large input vector. The results can also be affected by differences in the data representation, although it is difficult to point out which excact differences this might be.

4.2.3 Overall Discussion and Limitations

The results indicate that although conditioning the Melody Agent with chord information increases the input dimensionality by 121%, it is statistically significantly beneficial in terms of better representing the underlying dataset. The experiment therefore reflects positively on the impact communication can have on autonomous agents in terms of their musical output.

On a different note, the experiment revealed some limitations of the melody network, particularly the minimal performance difference between the conditioned and nonconditioned setups compared to previous work. This finding suggests that the Melody Agent could possibly benefit from a redesign in network architecture and data representation. Compared with the network structures used by Genchel et al. [159], it appears that greater modularity could improve the effectiveness of the Melody Agent. One potential approach could be to split the Melody Agent into several specialized agents, each focusing on different aspects of melody generation, as Experiment 1 showed the potential of modular network design. It is also probable that a deeper network with more data and training could increase the difference, since more complex networks are better suited to handle data of higher dimensions [163].

4.3 Experiment 3: Scrambled Communication - System Review

The previous experiment tried to find out how the removal of communication affected a single agent. This experiment continues to explore this, while also focusing on the effect communication has on the system as a whole, and in turn helps answering the second and third research sub-question:

> How does communication between autonomous agents affect the musical output of individual agents?
>
> How does communication between autonomous agents affect the musical output of the system as a whole?

Instead of doing this by removing the communication between agents like in experiment 2, which as mentioned is already a heavily researched topic, this experiment will instead create a new system where the communications between the different agents are scrambled.

4.3.1 Experiment Outline

In this new system, every communication line between agents is to some extent replaced by noise. The amount of noise used can be adjusted by a parameter γ, which indicates the probability that a single number in the communication vector will be replaced by a random number within the range of the recipient's vocabulary. By increasing γ, the agents will have an increasingly wrongful perception of their musical surroundings. This scrambling affects two communication lines:

- Bass Agent $\rightarrow$ Chord Agent. The information being transmitted between the two agents are root note and duration. These are used to generate the chord variation.
- Chord Agent $\rightarrow$ Melody Agent. The information being transmitted between the two agents are current chord, next chord, and time until next chord. These are used to generate the melody pitch and duration.

Since the communication between the Drum Agent and the Bass Agent does not affect the generation of the bass network (Phase 1), only the post-processing of the musical output (Phase 2), this communication line is not included in this experiment. The same goes for the Harmony Agent that does not use an NN to generate music. Hence, the agent is not part of the experiment.

The experiment is carried out by evaluating the chord and melody networks on their test dataset in terms of log-likelihood. Starting with $\gamma = 0$, it gradually increases by increments of 0.1, until it reaches $\gamma = 1$, where all communication is purely random.

Each network is evaluated with 1,000 samples for each γ-interval. At each γ, the average log-likelihood of the 1,000 samples is calculated for each metric.

4.3.2 Results and Discussion

In Figure 4.6, the average log-likelihood for each metric is shown with γ ranging from 0 to 1. In addition to the *chord variation*, *pitch*, and *duration*, a fourth metric called *system* is added. This is the average of the other three metrics and acts as an indicator of how the system as a whole will react to disruption in the communication between agents.

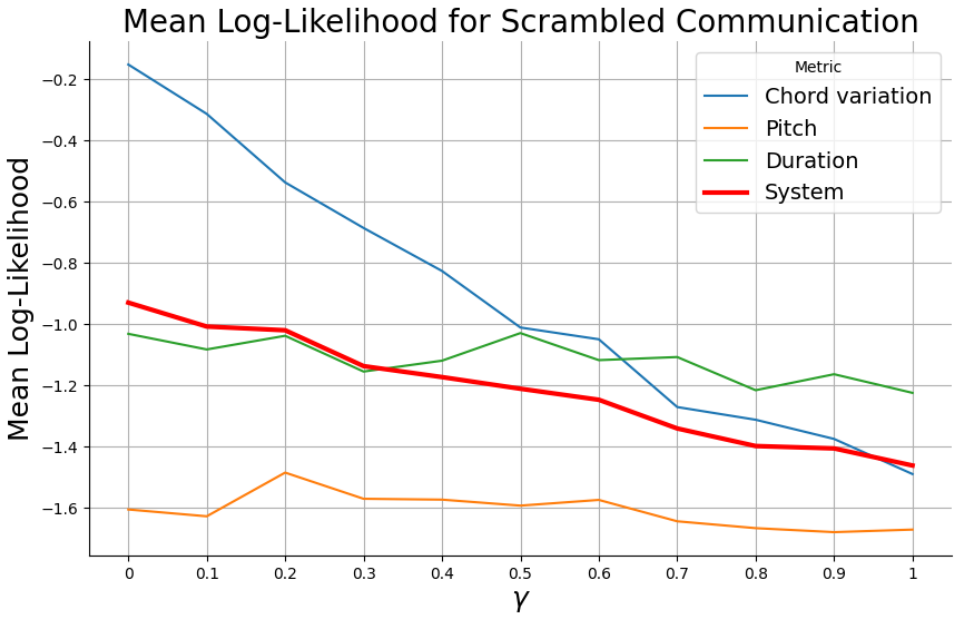


Figure 4.6: The mean log-likelihood of three generated metrics plus system (average of the three metrics) as γ increases.

Examining the generation of the Melody Agent first (the pitch and duration graphs), we see that, although not greatly, an increase in randomness in the communications seems to lead to a decrease in performance. Duration seems to be more affected than pitch. This is synonymous with the insight derived from Experiment 2. Also interesting is that for both pitch and duration, it seems like there is no clear worsening of performance before γ reaches 0.5.

The chord variation generated by the Chord Agent, on the other hand, has a consistently large decrease corresponding to each increase in γ. Using Spearman's rank correlation coefficient [164], we can calculate how likely it is that there is a correlation between the parameter γ and the performance of each metric. The test is non-parametric, as the log-likelihood data have a non-normal distribution. The test is Bonferroni corrected for four hypotheses ($m = 4$) making the threshold for statistical

significance $\alpha = 0.013$. Table 4.4 shows the calculated correlation coefficients and their corresponding p-values. They show that significance is found for all metrics except pitch.

Table 4.4: Spearman's correlation between γ and log-likelihood for metrics and system. p-values are Bonferroni corrected (m=4).

Metric	**Correlation coefficient**	***p*-value**
Chord variation	-1.0	< 0.013
Pitch	-0.66	0.025
Duration	-0.72	< 0.013
System	-1.0	< 0.013

This research further strengthens the conclusion made in Experiment 2, that inter-agent communication can have a positive impact on the musical output of the individual agents. Although a non-significant correlation between communication noise and log-likelihood was observed in one of the three metrics, the collective evidence from both experiments supports the positive role of agent interaction.

Interestingly, this also enforces the claim from Experiment 2 that the melody network could benefit from a design more capable of leveraging the positive effects of communication, as the experiment found no significant correlation between the amount of noise in communication and pitch generation performance.

4.4 Experiment 4: Evaluation of Communication Effects using Listening Test

The previous experiment concludes the objective evaluations of the system. The following experiment tries to answer the research questions by involving subjective human evaluation. This first subjective experiment is related to Experiment 3 as it uses the same system with scrambled communications as described in subsection 4.3.1. Instead of evaluating individual agents and system performance through objective measures, the musical output of the entire system is now directly evaluated through a user survey. Thus, helping to answer the third research sub-question:

> How does communication between autonomous agents affect the musical output of the system as a whole?

4.4.1 Methods

There are multiple different methods for doing a listening test, depending on the questions being answered and the structure of the audio clips. Some use the Turing test [74] or try to classify a certain style or genre [81, 82], while others give their participants two or more sound clips and let them select the preferred one [45, 57, 128]. Ribeiro et al. [165] promote Mean Option Score (MOS) through what is a 5-point Likert scale. They claim that this is the most popular choice for subjective listening tests, in addition to being better than paired comparison in terms of data requirement to achieve statistical significance.

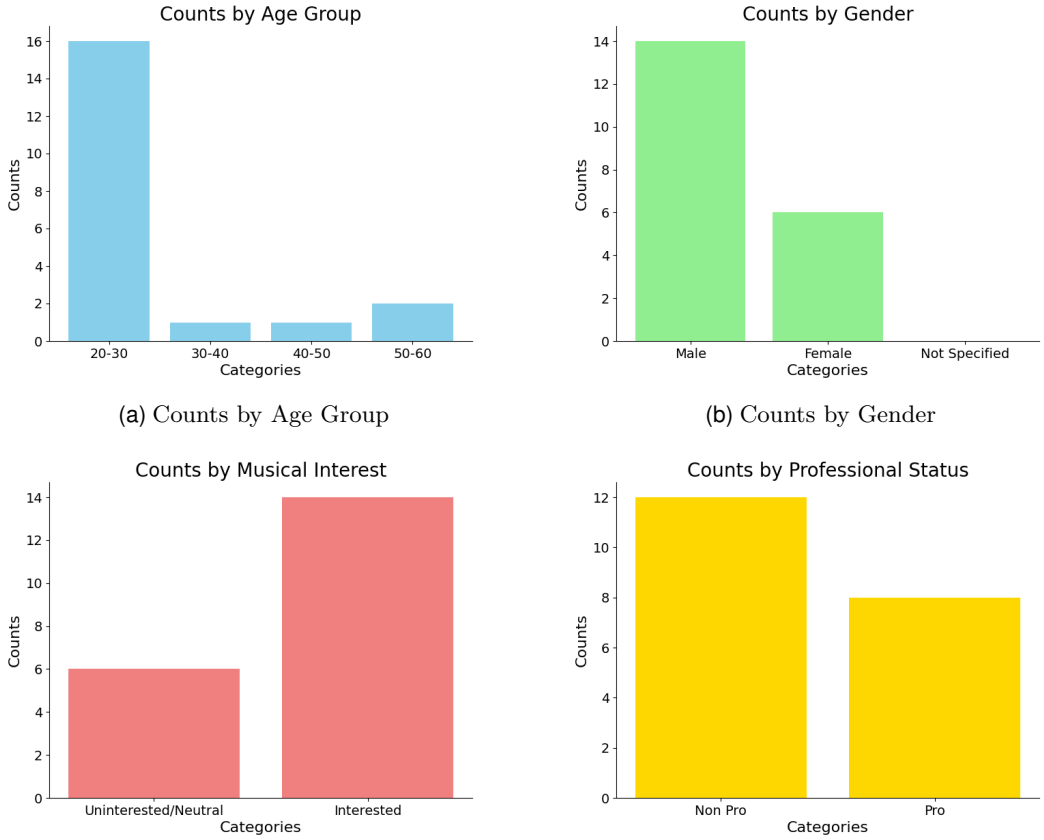
In a 5-point Likert scale test, the participants are asked to rate each question a scale of 1-5. This gives them the option to be positively or negatively modest or extreme in addition to neutral. Although criticized for not being able to reflect subtle differences in the respondent's attributes [166], the simplicity makes it one of the most used psychometric tools [166]. For simplicity, comparability, and user friendliness, the 5-point Likert scale was therefore chosen as survey method for this listening test.

4.4.2 Recruitment and Participants

The survey was distributed on the Internet, mainly among my social circle, but also internally in university research groups. The test was conducted online through a form. In total, the study had 20 participants.

This type of internal sampling is not ideal and could make the study vulnerable to limited representation and reduced reliability [167]. It is beneficial for these studies that the participants have some diversity in terms of demographic traits [165]. Therefore, a small preliminary questioning was conducted for every participant to extract some basic information about the demography of the user.

Participants were asked about age and gender in addition to their interest in music, as well as their experience playing or creating music. This last question was in order to categorize participants as pro or non-pro, similar to other studies [45, 60, 102, 129].



(a) Counts by Age Group

(b) Counts by Gender

(c) Counts by Musical Interest

(d) Counts by Pro Status

Figure 4.7: Overview of participant demographics.

As can be seen from Figure 4.7 the age of the participants is heavily weighted towards the group 20-30 representing 80% of the participants. The gender is also skewed towards men, with 70% of the participants being male. In general, a more uniform distribution primarily in age would be preferred. This uneven distribution can be attributed to the natural outreach of the survey by distribution mainly among friends. To balance this, one could use crowd-sourcing services as suggested by [165], where participants are paid to participate in such user studies. This approach is not used, as it is both costly and criticized for ethical and data quality reasons [168, 169]. Although they can give a higher participation rate, they have been shown to favor the participation of low-income individuals [168], again skewing the demographics.

The musical interest of the participants was given based on the answer to the question *How interested are you in music?*, with the option to be *Very Uninterested*, *Uninterested*, *Neutral*, *Interested*, and *Very Interested*. They were then grouped so that participants answering *Interested* or *Very Interested* were considered to have a musical interest,

while the rest were deemed neutral/not interested. A similar assessment was made to distinguish pro users from non-pro users, but this time they were asked *How much experience do you have in playing music?*.

4.4.3 Audio Examples

In total, there are 24 audio clips in the study[1]. They are divided into 12 counts of scrambled communication and 12 counts of normal communication. The audio clips with scrambled communication have a $\gamma = 1$, which means that the inter-agent communication is completely random.

In addition, each of these communication methods is divided into 3 categories of creativity related to the bass creativity described in subsubsection 3.3.2 and the melody creativity described in subsubsection 3.3.2. In short, this agent creativity affects the probability distribution and restriction in the model selection process. This causes the agent to generate notes that are deemed more probable or less probable, affecting the presumed novelty of the musical output.

The audio examples are categorized as either *Low Creativity*, *Medium Creativity*, or *High Creativity*, where low and high creativity is created by maximizing or minimizing the creativity slider of the Bass Agent, and both the pitch and duration creativity sliders of the Melody Agent. This ensures a broad distribution of audio samples and allows user preference of creativity levels to be analyzed.

The Harmony Agent does not generate its own music, but rather purely listens and reacts to the Melody Agent. Although it can be difficult to determine what is listening and what is communication, it is made as a reflexive agent, which means that it technically does not have the ability to communicate. It was therefore excluded from the generated audio clips.

4.4.4 Experiment Outline

The study was conducted online, where the participant listened to the 24 audio clips as much as they wanted before rating the clips on 4 different metrics on a 5-point Lickert scale. The metrics are inspired by [102] and include harmony, rhythm, musical structure, and subjective rating. To give every participant the same foundation, each metric was clearly defined to the users:

- **Harmony:** How harmonic is the interplay between the different instruments?
- **Rhythm:** How consistent is the rhythm that the different instruments create together?

[1]The clips used in the user study are available here: [29].

- **Musical Intention:** How well does the music connect from one point in time to the next?

- **Subjective Evaluation:** How well did you personally like the music?

Users are also recommended to use headphones, as it further causes the stimuli of the participants to be similar. In addition, the use of headphones increases the participant's discriminatory capacity [165].

Every audio clip requires an answer, and it is not possible to leave blank responses. This is done to obtain as much data as possible and so that the statistical analyzes can be free of adjustments due to non-responses [170].

4.4.5 Statistical Power Analysis

Statistical power is a measure of how probable it is for a test to find statistically significant results of a certain effect size in the data, depending on the significance criterion (alpha), the sample size, and the effect size. This makes it well suited to test whether the sample size of a study is sufficient to discover a result of a certain effect size [171].

In my study, there are 20 participants, 24 audio clips and 4 metrics, which means that 1920 data points are gathered in total. These data points are divided into groups depending on the specific statistical test conducted and the number of data points in each group of the test. Therefore, statistical power analysis is performed for the different statistical methods used in this experiment and specifically for the tests performed using the lowest sample size in each group. The power levels are shown in Table 4.5. Power levels are performed with conventional standards for the significance level (alpha = 0.05) and effect sizes. For the two group tests, Cohen's d of 0.2, 0.5 and 0.8 is used for small, medium, and large effect sizes respectably, while the three group test use 0.1, 0.25 and 0.4 for the same effect sizes.

Table 4.5: Statistical Power Analysis of the two-sided and three-sided statistical test performed in the experiment.

Effect Size	Two-sided Test	Three-sided Test
Large Effect	1.0	0.99
Medium Effect	1.0	0.98
Small Effect	0.71	0.34

Power levels above 0.8 are considered good, which means that power analysis suggests that there is a very high chance that the study will discover statistical differences of medium effect. Still, more samples could be beneficial to discover even smaller differences in the data.

4.4.6 Results

Scrambled vs. Unscrambled Communications

There are multiple ways to examine the data from the user survey, but first a possible difference in user feedback between the two main groups, unscrambled and scrambled communications, will be investigated. Figure 4.8 displays the results in a violin plot for the four different metrics in addition to the average score. Since a high score on each of the four metrics positively reflects how users perceived musical output, averaging the metrics can be a good way to measure overall rating. From the plot, it can be seen that human listeners favor the original system with unscrambled communications.

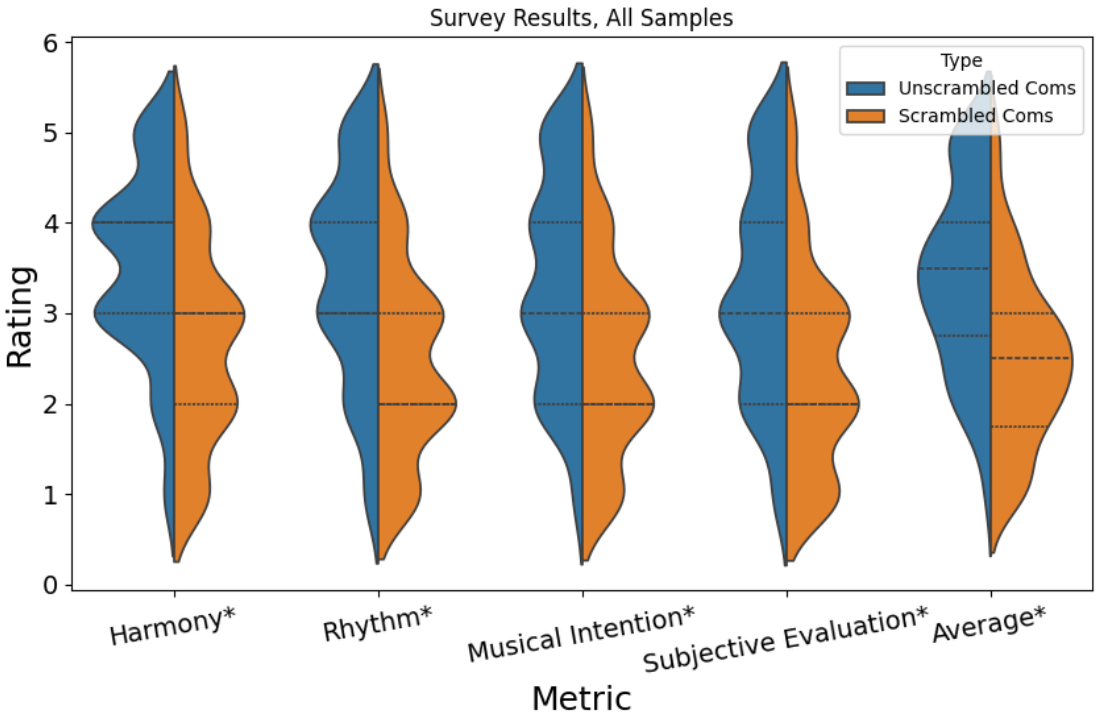


Figure 4.8: Violin plot of the user study data for the four metrics: Harmony, Rhythm, Musical Intention, and Subjective Evaluation, in addition to the average of all metrics. Statistical significance between the two classes, unscrambled and scrambled, is marked with a star (*). Bonferroni correction (m=5) is used.

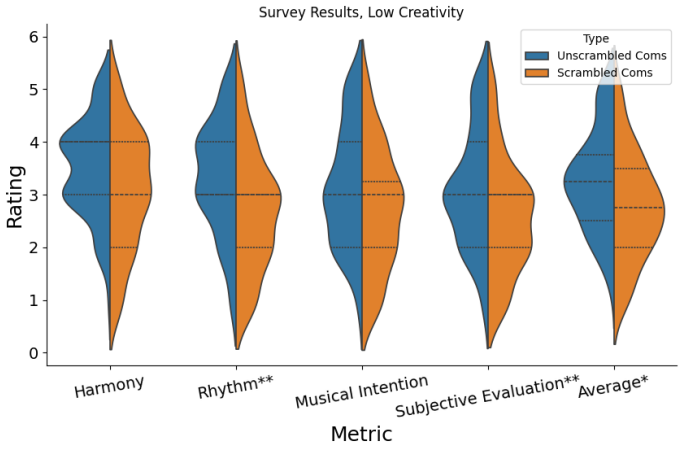
There is a debate on whether data from Likert scales approximate intervals well enough for the data to be considered normally distributed [172]. In terms of my user data, a Shapiro-Wilk test [148] concluded that the data are not normally distributed. Therefore, the non-parametric Wilcoxon signed rank test [143] is performed on every metric to detect whether the data from the two distributions (unscrambled and scrambled) are significantly different. The p-values are corrected for the five different metrics using the Bonferroni correction ($\alpha = 0.010$). Significance is marked in Figure 4.8 with a star (*). Although all metrics are statistically significant, the largest differences between the two communication methods are seen in *Harmony* and *Rhythm*. Interestingly, no

significant differences between ratings of the pro users compared to the non-pro users were discovered on any metric.

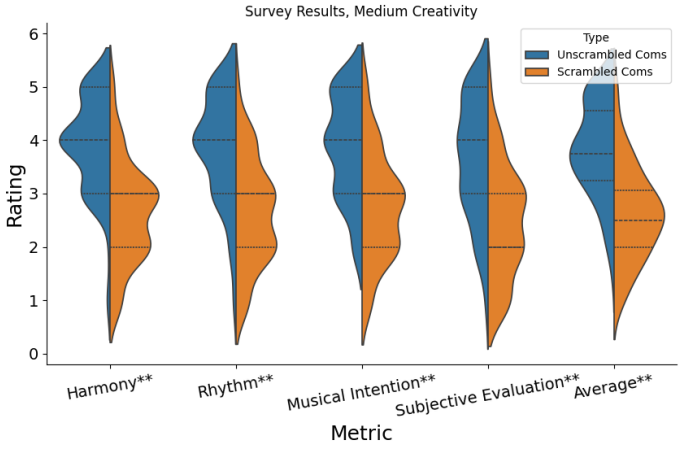
Agent Creativity

Since the data are divided not only into scrambled and unscrambled, but also into agent creativity levels, it can be examined how the different levels of creativity affect user ratings in different metrics and communication styles. In Figure 4.9 similar violin plots as in Figure 4.8 can be seen, but now each subplot represents a level of creativity. The significance levels are again calculated using the Wilcoxon signed rank test [143], comparing the communication methods for each metric. Again, the p-values are Benferroni corrected, but this time two levels of significance are used. The first level uses a Bonferroni correction with $m = 5$ ($\alpha = 0.010$). If the p-value is below this level, it is considered statistically significant. This is marked with one star (*). The second level corrects for all 15 tests in different levels of creativity ($\alpha = 0.0033$). This is considered strictly statistically significant and is marked with two stars (**).

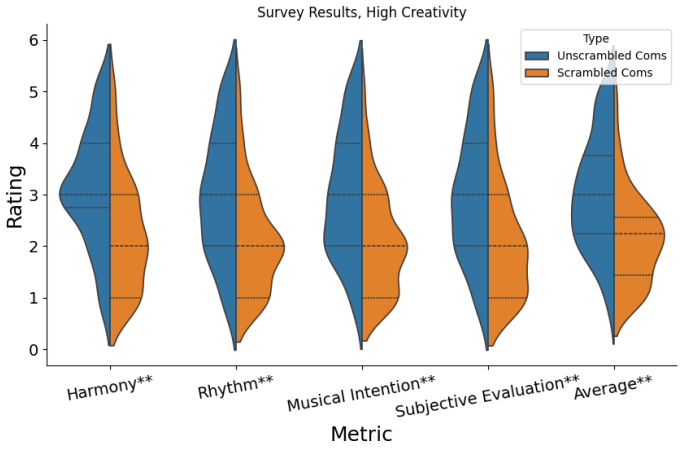
When dividing the clips into levels of creativity like this, it is clear that agent creativity changes the effect communication has on the user ratings. Although 13 of 15 metrics still show significant benefits of using unscrambled communications, the two metrics that do not yield these results are achieved using low agent creativity. The violin plots also show that among the significant metrics for low creativity, the difference in distribution is smaller than the other levels of creativity. In particular, medium creativity has the largest difference in performance between the two distributions.



(a) Low Creativity



(b) Medium Creativity



(c) High Creativity

Figure 4.9: Comparison of agent creativity levels — high, medium, and low — using four metrics, with average. Violins depict unscrambled versus scrambled communications. One star (*) indicates statistical significance (corrected for 5) ($\alpha = 0.010$), two stars (**) indicate strict statistical significance (corrected for 15) ($\alpha = 0.0033$).

Table 4.6 shows the average user score based on creativity rating for each of the communication methods. It shows how agent creativity levels affect the user ratings across all metric. To check if there is statistical significance between the three levels of agent creativity for each of the communication methods, a Wilcoxon signed rank test can no longer be used, as this only works when comparing two classes. Instead, Friedman's test is used to rank the three groups, revealing that the data in the three groups are significantly different from each other. Lastly, Wilcoxon's post hoc text is performed with the highest ranking creativity level as reference (Medium for unscrambled and Low for scrambled). The adjusted p-values are shown in Table 4.7 and Table 4.8.

Table 4.6: Comparison of scores for unscrambled and scrambled communication across every metric and agent creativity level. Bold indicate highest scoring creativity option for the current metric.

	Unscrambled					Scrambled				
Creativity	**H**	**R**	**MI**	**SE**	**A**	**H**	**R**	**MI**	**SE**	**A**
Average	3.51	3.39	3.34	3.23	3.37	2.66	2.49	2.53	2.38	2.52
High	3.13	2.99	2.99	2.96	3.02	2.26	2.14	2.1	2.09	2.15
Medium	**3.91**	**3.85**	**3.86**	**3.64**	**3.82**	2.65	2.65	**2.76**	2.46	2.63
Low	3.5	3.34	3.16	3.1	3.3	**3.01**	**2.68**	2.72	**2.58**	**2.76**

H = Harmony, R = Rhythm, MI = Musical Intention
SE = Subjective Evaluation, A = Average

Table 4.7: Corrected p-value using Friedman and Wilcoxon's post-hoc for Unscrambled.

	High	Medium	Low
High		< 0.05	< 0.05
Medium	< 0.05		< 0.05
Low	< 0.05	< 0.05	

Table 4.8: Corrected p-value using Friedman and Wilcoxon's post-hoc for Scrambled.

	High	Medium	Low
High		< 0.05	< 0.05
Medium	< 0.05		0.12
Low	< 0.05	0.12	

Looking at the user rating of the audio clips in Table 4.6, we can see how different levels of creativity affect the user ratings. We can see that for the normal unscrambled communication mode, medium agent creativity is by far the highest rated creativity setting, with low creativity second, and high creativity being the worst performing option.

When we instead look at the scrambled communication mode, medium agent creativity only beats out low creativity on the metric musical intention, while low agent creativity receives better scores on all other metrics, including average score. Again, high creativity is the worst performer.

4.4.7 Discussion

There are several interesting take-aways from the user survey. Most importantly, it helps answering the third research sub-question:

> How does communication between autonomous agents affect the musical output of the system as a whole?

It is clear that the users preferred musical clips generated with unscrambled communications over clips generated with scrambled communications. This is true for all four metrics with statistical significance.

In particular, the two metrics with the highest discrepancy between the two communication modes were harmony and rhythm. It is easy to see how harmony and rhythm are lost when agents receive misleading information about the surrounding musical landscape. The difference is smallest for musical intention. This can be due to the fact that, although there are miscommunications amongst the agents, the knowledge about their previously played notes is still intact, making the individual agent's musical intention remain.

It is also interesting to see that for normal unscrambled communication, across the different metrics, the highest rated sound clips are from a system with agent creativity set to medium. This is not surprising, as the creativity scores are tuned subjectively in the making of the system, and the medium score was set as a good standard. Another observation that is interesting, yet not surprising, is that the lowest discrepancy between the two communication modes was achieved when creativity was set to low. Here, there are no statistical difference between the user rating for musical intention and harmony. By having a lower creativity, the agents are more restricted and choose safer options. Although this level of creativity is not preferred for the normal system with unscrambled communications, the conservative nature of the agents ensures positive results when communications are scrambled relative to the normal unscrambled communication mode. However, while low creativity was the highest rated level of creativity in scrambled communication, it performed far worse than the worst performing level of creativity in unscrambled communication (high creativity), showing how creativity adjustments can not fully compensate for scrambled communications.

Limitations

The meta-analysis of the survey shows that it could benefit from a more uniform distribution amongst the participants, especially in the age category. Research on

whether age is a significant factor in music preferences is contradictory [173, 174], but as the difference between the main groups (scrambled versus unscrambled) is so significant ($p = 5.63 \times 10^{-25}$ for the average metric), I would still argue that the findings are relevant.

In addition, the statistical power analysis shows that the statistical tests with the smallest sample size that are performed are expected to have a high chance of detecting large and medium, but not small, effects in the data. The study therefore could benefit from more data, but I would argue that the experiments did not try to find any small effects, but rather large overarching differences in the data, making the necessity for a larger sample size less relevant.

4.5 Experiment 5: Interactive User Study - Informal Feedback from Experienced Musicians

This final experiment aims to evaluate user experiences with ML-jam, focusing on the design choices influenced by the multi-agent architecture. The research by Huang et al. [13] showed that when musicians were to incorporate generative musical AI into their creative musical process, they preferred to use simpler modular architectures instead of large and complex end-to-end models. By gathering user feedback, the goal is to derive insights into how ML-jam can be used in creative processes by musicians, and whether the multi-agent architecture can be beneficial. More precisely, the experiment will address the fourth and final research sub-question:

> How is the user experience of using an interactive musical system with a band-like multi-agent architecture?

4.5.1 Methods

In an interactive generative musical system, there are several stakeholders (performers, audience, composer, etc.) [155]. Interactive user studies involving hands-on experience are an often used method to gain insight into how the system performs for the stakeholder who will ultimately use it, the performer [130–132, 134]. These experiments often take place as semi-structured interviews. Quite a lot of research is conducted on how these interviews should be performed and how data should be presented [154, 175]. Although this structured approach is valuable, another method used in end-to-end music generation systems has been to gather informal feedback to supplement a thorough scientific analysis [73, 77]. These are not restricted to strict methodologies, but act as additional insight into the system that can be hard to obtain elsewhere.

This experiment is a combination of the two methods. The format is a structured hands-on user experience followed by interviews, all based on scientific theories. However, because of the scope of the thesis in addition to the project time frame, it does not extend into a comprehensive thematic analysis. Instead, like others [73, 77], it provides an informal insight through user reflections, which is important to evaluate an interactive generative music system.

Execution

The experiment was carried out individually for each participant. Before starting, a brief introduction to the functionality of the system was given through an oral guide of the functionality of the system in addition to each adjustable parameter. After that, the experiment consisted of four phases:

- **Free play:** The participant was given as much time as they wanted to explore the system. No restrictions or guides were given.
- **Co-play:** After the participants were satisfied with the free play session, they were asked to plug in their instrument of choice and play along with the system. The emphasis was on using it as a jamming tool, but also as a way of co-creating new music.
- **Task:** Finally, when the participants were familiar with the system through the previous phases, they were asked to record a composition with the system.
- **Interview:** When the participant felt done with the co-play session, a interview was conducted to gather insight into the musician's experience using the system.

By assigning predefined tasks to users, such as creating a composition, they were provided with clear goals to achieve with the system. This guidance is meant to steer the participants into experimenting with use cases of the system. The use of tasks is also suggested as a way of quantitatively evaluating systems in terms of metrics such as task completion rate [175]. This kind of quantitative approach to tasks was not used in this experiment; the tasks acted merely as a push to encourage the participants to be creative with the system. In addition, it put the participants in a setting where they had to integrate ML-jam into their music-writing processes. This is in good agreement with the focus of the experiment, which is a performer-centered evaluation, favored by Stowell et al. [154], who criticized task evaluation for not being able to capture creative aspects. Performer-centered evaluation focuses on the participant's experience while interacting with the system, including analyzing how the participants integrate the system into their performance. Therefore, these phases aim to give participants a good balance between the freedom to be creative and still being structured enough to guide them to experience ML-jam as a tool for music creation and as a musical companion.

In the interviews, the participants were encouraged to be honest but comprehensive in their responses. The interview was conducted as a free-flowing conversation, but revolved around some key points, as suggested by Wanderlay and Orio [175]. These were control, co-play, learnability, responsiveness, enjoyment, frustration, and usability. A consensual audio recording of the interview was made and translated from Norwegian to English by the author.

Participants

Since the goal is to use the system as a tool for co-creating music as well as being a jamming partner, the participants had to master one or more instruments and be experienced in music creation. In addition to this, participants need to be motivated. This is an important aspect in a user study that evaluates a system by integrating it into the creative process of a musician, as motivation in a task has been shown to increase creativity [176].

Two participants in their mid to late 20s were chosen, experienced in various musical instruments, band settings, and music production using a DAW. Although it is a small sample size, it still gives some personal insight into the use of ML-jam. One of the participants had experience in programming and both had a modest interest in ML, but had no experience with similar musical AI systems. The study was carried out at the participant's preferred location. In addition, during free play, co-play, and the composition task, the participants were left to themselves without being interrupted while interacting with the system, but with the opportunity to ask questions. This was done to encourage creativity through a feeling of security and familiarity, as the opposite has been shown to decrease creativity [177]. In addition, interrupting the playing through questions, or encouraging the participants to *think-aloud* during interaction with the system can disrupt the creative process [155, 178].

The participants are held anonymous, the participation was voluntary, and the use of their responses in the thesis is consensual. All recorded data were collected with consent and, when no longer needed, discarded.

4.5.2 Results

The results will be presented as quotes from the interview and are representative of the feedback received. The quotes will be categorized under the key points that they were asked about. Not every quote ties into the research question about interactive MAS, but they are included to give an insight into the general user experience.

Control

> "I felt like I had good control to generate a melody and a chord progression that could fit what I wanted to achieve."

> "You can spend some time if you initially don't find something that fits perfectly for the purpose you are using the program for."

> "There are good opportunities to change both the melody and the chord progression and how it is played. I felt like I had pretty good control over most of what I might need."

Co-play

> "The system has a predictable tempo and chord progression, which makes it easy to play along with."

> "If you got something you don't want to play together with, you could easily change it, and it was great that you could keep the things you liked, and instead generate the parts you wanted to change."

Learnability

"It was very intuitive and straightforward, and it took a very short time to learn. Of course, mastering it will take a bit longer, but I felt like I very quickly understood what would change in the system if I generated a new loop after clicking on some of the boxes or dragging some of the sliders."

Responsiveness

"I liked the loop system, and I thought it was nice that it waited out the progression before changing. It gave a realistic feel and made for a smooth transition."

Enjoyment

"I liked that you had a lot of freedom to do different things. You have a lot of choices if you want to change play style."

Frustration

"It could sometimes generate things that did not fit."

"Sometimes it felt like the drums did not play along with the rest of the agents."

Usability

"It suited jamming at home alone very well. You can generate whatever you want to practice alone. If you lower the creativity on the melody, you can use it to practice harmonizing with melodies. You can use it both to learn new things yourself and to avoid searching for backing tracks on YouTube. When I was learning guitar at the music school, the teacher often searched for *backing tracks G-major*, but it was usually just four chords played in the background. This is much more dynamic."

"I can see how it can be used to practice, but personally I would like to practice to a real song."

"It can also be used for songwriting; you can get a lot of inspiration from the melodies and chord progressions that are generated."

"It could be used in a performance as a fun thing to play along with."

> "In worst case you can generate things that do not match. If you could preview or save generated melodies, you could prepare an entire show on things you know are going to sound good, but as it is now if you generate something that does not fit, I assume people can notice that things do not fit well together."

4.5.3 Discussion

The participants expressed great control over the systems and the individual parameters.

> "I very quickly understood what would change in the system if I generated a new loop after clicking on some of the boxes or dragging some of the sliders."

A point was also made about the ease of adjusting a single agent parameter to create simple melodies that are easily harmonizable, which is a testament to the ability to precisely adjust single agent parameters to create music aligned with the preferences of the user. This is also true for the post-processing of the generated music, which was an aspect mentioned when asked about enjoyment.

In addition, a particularly appreciated aspect was the ability to keep certain parts while regenerating instruments that did not fit their liking. It is reasonable to believe that both this and the precise control are qualities that can partly be attributed to using a band-like multi-agent architecture with a modular design as in ML-jam. This dynamic control was also pointed out as a reason why this system would be a good jamming tool. This aligns with the findings of Huang et al. [13], who found that musicians had better control when creating music using a modular approach, using instrument-specific models. This also ensured that they could regenerate specific instruments until they were satisfied, very similar to how the participants used ML-jam.

A notable comment was made on the loop system. It was originally made as a solution to the inherent latency in the agent generation process and a suboptimal alternative to true real-time, albeit with some perceived benefits such as predictability and repetitiveness, as mentioned in subsection 3.2.1. Nevertheless, the participants expressed a fondness for the loop system, and even when specifically asked if it was frustrating that the changes did not happen in real time; both disagreed and instead expressed a positive attitude toward it.

In general, there were two things that were mentioned as negative. The first thing was the integration of the drums. Both participants felt that it did not integrate as well into the mix as the other agents. This fits very well with how the system is constructed. As mentioned in subsubsection 3.2.2, the Drum Agent is the only agent that does not receive information from any other agent. It is also the agent that has received the least attention in the making, as the data representation and network design were not created for this thesis.

The second negative mention was that sometimes the program could make unpleasing music. This aligns with the results from Experiment 4, where it is clear that for some parameter settings, the chances of getting unpleasing results are significantly higher. For *High Creativity*, it received ratings below the Likert scale average of 3. However, users appreciated the ability to easily regenerate an agent's instrument that was not preferred, highlighting this as a valuable feature.

4.5.4 Addressing the Research Question

We can then answer the fourth research sub-question:

> How is the user experience of using an interactive musical system with a band-like multi-agent architecture?

Experienced musicians who had hands-on experience with the system mentioned several benefits of the system that are the natural results of a band-like multi-agent architecture. This includes control over individual agents and their generation and the ability to change parts of the composition, without having to change all the instruments. This made the system work well as a jamming partner.

Chapter 5

Discussion

This final chapter will initiate the discussion towards the overarching findings of the five experiments carried out, interpreting them within the context of the research questions and related theoretical frameworks proposed by Dadman et al. [14]. Here, I critically examine the implications of these results, discuss the inherent limitations of the thesis, and suggest future research directions before concluding. However, first I will address the ethical implications and concerns related to the work carried out in this thesis before discussing the overarching results.

5.1 Ethical statement

As the landscape of music technology evolves, ethical considerations become important, especially in the field of AI-creativity, and the integration of ML and art. Therefore, it is important to address the ethical implications and justify how this work tries to minimize the potential downsides. It is also important to address the uses in which the author is aware of the potential negative broader impact of the work [179].

5.1.1 Intellectual Property Rights

A common legal and ethical question is the use of intellectual property rights [180, 181]. Although generative models might fall under fair use [182], it is still an open legal question [183]. Generative models are generally trained on the work of humans. It is important that the data are ethically sourced, meaning that the owner of the data consents to it being used. This project uses two datasets. The first, known as the Groove dataset, consists of drum beats improvised by 10 professional drummers with the purpose of creating the dataset. It is licensed appropriately for use in projects such as this [91]. However, ethical sourcing of music data, as discussed by Thickstun et al. [184], becomes a concern with datasets such as LakhNES [74], which compiles music by scraping MIDI files from the Web. This raises concerns regarding the preservation of the original artists' intellectual property.

The second dataset used in this study, the POP909 [45], similar to LakhNES, poses potential ethical concerns. Although the POP909 dataset features performances by professional musicians specifically recorded for the dataset and is licensed for use, it is not explicitly clear whether the musical pieces played were ethically sourced in terms of copyright and intellectual property rights related to the original author of the music. This uncertainty highlights the need for transparency in the origin and licensing of music data used in research to ensure the respect for and protection of artistic rights.

5.1.2 AI and Art - Impact on Labor and Creativity

Music is art, and a common ethical concern raised by several researchers revolves around the fact that by giving computers the ability to generate art, it could disrupt the industry, partially eliminating the need for humans to generate music [184, 185]. It is also suggested that AI could potentially stifle human creativity and innovation [186].

This is an important concern; however, ML-jam distinguishes itself from many AI applications that are designed as autonomous, end-to-end solutions for commercial music production [14]. ML-jam is developed as an interactive tool that works in collaboration with musicians. The intent behind ML-jam is to enhance the creative capabilities of human artists, not to replace them. This approach respects the intrinsic value of human creativity and expertise in the music-making process, ensuring that technology serves as an aid rather than a substitute. This is something Clancy highlights the importance of [185].

5.1.3 Conclusion

Throughout the development of this thesis, ethical considerations have been addressed. However, it is impossible to protect against all the broader negative implications related to the scientific research conducted. The focus has remained on creating a tool that complements human skill and creativity, with the aim of enriching rather than diminishing the music creation landscape.

5.2 Discussing Multi-Agent Modularity

To start the discussion surrounding the experiment, the first sub-question is revisited:

> How does the modularity natural to a multi-agent architecture compare against a monolithic architecture in terms of performance and usability?

The first experiment provides an argument for the effectiveness of a MAS architecture in music generation systems. This modular approach outperforms traditional monolithic structures and potentially offers a more manageable alternative for system development. Unlike the difficult process required to build and optimize complex deep monolithic

networks, MAS allows developers to deploy several simpler, task-specific networks. This capability facilitates precise network design and fine-tuning of parameters tailored to each agent's specific functions. This is beneficial in multi-track music generation, where the task naturally lends itself to a multi-agent design.

In contrast to systems such as those discussed in [121], where parameter adjustments impact the system globally, the ML-jam benefits from the flexibility of agent-specific parameter settings. For example, ML-jam creativity adjustments can be applied for individual agents. Likewise, there is the ability to control individual agents' post-processing of musical data. In Experiment 5, expert interviews showed that these features were appreciated in ML-jam and come naturally to a band-like modular multi-agent design. Previous research also shows that musicians preferred this way of co-creating music with generative music systems [13].

5.3 Evaluating Communication Impact on Musical Agents

The system proposed by Dadman et al. [14] is mainly constructed of what the literature calls intelligent agents, as described in subsection 2.6.1. These agents have the ability to communicate. It is therefore important to evaluate how communication between such autonomous agents affects the musical output of individual agents and the system as a whole. The second and third research sub-questions focus on this, and they are:

> How does communication between autonomous agents affect the musical output of individual agents?
>
> How does communication between autonomous agents affect the musical output of the system as a whole?

The findings of Experiments 2 and 3 underline the importance of effective communication between agents. The inter-agent communication facilitated better individual agent performance and improved the coherence and quality of the overall musical output. User feedback in Experiment 4 further corroborated this, where disruptions in communication (scrambled signals) led to a noticeable degradation in music quality, confirming the value of precise and reliable inter-agent communication. In addition, Experiment 5 also showed that the participants agreed that the drums sometimes did not feel like a part of the composition. Since this is the only agent who does not receive information from other agents, it is natural to assume that this is due in part to lack of communication.

Interestingly, Experiments 3 and 4 found a correlation between the increase in communication noise and a decline in performance for the system and for individual agents – except for the pitch generation by the Melody Agent. This insight is beneficial because it emphasizes that good data representations that are precise – so that they

convey the necessary information without introducing noise – can directly cause an increase in performance. It is also important to know when designing systems where communication is not necessarily reliable. This could be through lossy compression, real-time systems where time issues cause incomplete communication, or other scenarios where data loss occurs. The user study conducted as part of Experiment 4 revealed that some negative effects of poor communication could be mitigated by restricting agents to more conventional choices and reducing the variability of their actions (reduced temperature).

5.4 User Experience

Insight into the user experience of using ML-jam as an interactive creative tool for musicians was gained in Experiment 5, helping to address the final research sub-question, which is:

> How is the user experience of using an interactive musical system with a band-like multi-agent architecture?

Unlike previous evaluation methods, which focused on objectively and quantitatively measuring different aspects of the system, this part of the study targeted the subjective experiences of two musical artists. Firstly, the participants felt like they had good control over the agents by tuning the parameters. It was also mentioned that it was nice that instrument-specific parameters could be adjusted to get the desired instrument-specific output, as well as the options for changing play styles post-generation. The participants also viewed the interchangeability of the selected agents as a positive feature, highlighting a natural advantage of the ML-jam architecture. This dynamic control was cited as a reason for ML-jam's effectiveness as a practice and jamming tool. Similarly, Huang et al. [13] noted that musicians benefit from using multiple simpler models and a modular approach, allowing them to pick and choose various musical outputs and combine them in their compositions. This introduced more control and creativity. ML-jam enhances this approach by allowing users to individually tune separate instruments while maintaining inter-instrument coherence. This setup not only provides musicians with the control afforded by simpler models but also integrates these models through inter-agent communication, ensuring a cohesive musical output.

Agent-specific parameter tuning and generation are features that naturally arise from the band-like multi-agent architecture. In contrast, monolithic multi-track architectures generate all instruments simultaneously, lacking the same level of precise agent-specific control. So, although all the positive attributes mentioned in Experiment 5 may not be directly caused by the MAS architecture, it is reasonable to assume that using a band-like multi-agent architecture enhances control over agent-specific parameter tuning and generation.

5.5 Addressing the Main Research Question

Finally, I will use the points discussed previously to address the primary research question:

> How can a band-like multi-agent architecture affect an interactive generative music system?

The answer can be summarized in three key points, addressing the modular design, the user experience, and the communication.

- By using the modularity naturally occurring through multi-agent design, system performance can increase while maintaining simplicity in individual network design.
- This modularity can also give better control over the individual agent's generated music, both during generation and post-processing. This can facilitate creative and controllable human–computer co-creation.
- The communication between agents can create inter-agent cohesion, and the emergence of a higher collective intelligence. This allows agents to co-play and improve their own performance by collaborating with other agents.

5.6 Limitations and Future Work

Throughout the experiments, limitations for the individual experiments have been revealed. This section addresses broader limitations that extend beyond individual experiments to include the whole system and the thesis' scope.

More Data and Network Research

Although Experiment 2 concluded with the same result as previous work [159], namely that receiving information from the chord domain could lead to improved performance for an NN generating a melody, perhaps a more interesting finding was that the difference between improved performance was very low compared to previous work [159]. This makes me believe that there is room for improvement when it comes to network architecture but also when it comes to the representation of the data that is communicated. It would be interesting to try to improve the musical output by using networks and data representations that are more optimized for the multi-agent approach.

A key point that Dadman et al. [14] point out is that multi-agent music generation systems can benefit from simpler task-specific data representation. ML-jam uses such a modular data representation design with instrument-specific representations, but the effects of this have not been researched. As is the case for all ML, the results you get

are only as good as the data you use [187]. Li et al. [188] found out that cleaning data could significantly improve the performance of an ML-model. Perhaps such a modular data representation could function as data cleaning?

This is especially interesting, as Experiment 2 indicated that some parts of the system were not optimized for learning from different domains. This could be due in part to the representation of the data. Since there is a lot of work involved in creating good data representations for music generation [45, 77–79], it would be interesting to see future research on data representations for multi-agent musical systems.

Increased Complexity

This paper suggests multi-agent architecture as a possible method of creating generative musical systems that do not need overly complex networks. However, scalability remains an unexplored challenge, as the study does not examine how this approach works when complexity increases. Han et al. claim that this requires careful planning and an in-depth understanding of each agent's capabilities and goals [112], which could make this approach difficult and potentially limit its efficiency or the applicability to larger or more diverse musical compositions.

Better Communication

To further develop ML-jam, introducing a communication loop where agents can interact back and forth with all other agents would be intriguing. This could be facilitated by implementing a consensus memory [112]. The agents would then have a real-time update on the state of all other agents all the time. Such a system would possibly increase the coherence and responsiveness of the musical output, making the interactions between agents more fluid and synchronized. This would mitigate one of the problems with the waterfall communication pipeline in ML-jam, where information only flows one way, and the Drum Agent ends up without receiving any information. This was possibly the reason for the perceived lack of cohesion the drum had with the rest of the composition.

Long-Term Coherence

Generative music systems often face challenges in maintaining long-term musical coherence, a topic that has been extensively explored in research [56, 60, 78]. ML-jam, employing models with short sequence lengths (information about previously played notes), tends to quickly lose track of previously played content. Although the cohesiveness of long sequences generated within this system has not been evaluated, smaller models are typically less capable of generating coherent music over extended periods, complicating their use in co-creative music settings [13]. This limitation suggests that ML-jam might struggle to produce long sequences that maintain long-term coherence.

Currently, the system employs loops to introduce a semblance of long-term structure, creating a repetitive pattern that gives the illusion of coherence. Looking ahead, it would be valuable to explore how a MAS with relatively simple agents could effectively manage long-term coherence. One potential approach could involve introducing a new agent specifically designed to oversee and maintain larger structural elements of the music. The primary role of this agent would be to track and integrate these larger musical structures, ensuring continuity and coherence throughout the compositions.

5.7 Conclusion

This thesis has introduced ML-jam. Through objective and subjective evaluation, a MAS interactive music generation system has been used to show how the MAS architecture can affect an interactive music generation system. The research focuses on how a multi-agent architecture can be used to create musical systems that can take advantage of features that often come naturally to the MAS architecture, like modularity and inter-agent communication.

The objective evaluation shows how MAS designs can provide increased performance compared to a similar monolithic network. It also shows how communication between agents can ensure that agents are aware of the surrounding musical environment, creating music that is coherent and influenced by the other agents.

A listening test shows how proper agent communication creates music that human listeners prefer compared to agents where the communications are scrambled. If communication is noisy, reducing the agent's creativity can increase user preference. In ML-jam, this is done by restricting the generation of untypical notes and duration, and by sharpening the selection distribution by reducing the temperature.

An expert user study shows how users interacting with the system appreciate features that are naturally occurring due to the modular multi-agent design, such as precise single-agent parameter tuning and single-agent interchangeability. This could foster a more creative and controllable human-computer interaction.

Bibliography

[1] OpenAI et al. *GPT-4 Technical Report*. arXiv:2303.08774 [cs]. Dec. 2023. DOI: 10.48550/arXiv.2303.08774. URL: http://arxiv.org/abs/2303.08774 (visited on 02/27/2024).

[2] *Writefull*. URL: https://www.writefull.com/ (visited on 05/08/2024).

[3] *Grammarly: Free AI Writing Assistance*. en-US. URL: https://www.grammarly.com/ (visited on 05/08/2024).

[4] *GitHub Copilot · Your AI pair programmer*. en. 2024. URL: https://github.com/features/copilot (visited on 05/08/2024).

[5] Stefan Koelsch. "Brain correlates of music-evoked emotions". en. In: *Nature Reviews Neuroscience* 15.3 (Mar. 2014). Number: 3 Publisher: Nature Publishing Group, pp. 170–180. ISSN: 1471-0048. DOI: 10.1038/nrn3666. URL: https://www.nature.com/articles/nrn3666 (visited on 08/30/2023).

[6] Gottfried Schlaug et al. "Effects of Music Training on the Child's Brain and Cognitive Development". en. In: *Annals of the New York Academy of Sciences* 1060.1 (2005). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1196/annals.1360.015, pp. 219–230. ISSN: 1749-6632. DOI: 10.1196/annals.1360.015. URL: https://onlinelibrary.wiley.com/doi/abs/10.1196/annals.1360.015 (visited on 08/30/2023).

[7] Jade Copet et al. "Simple and Controllable Music Generation". en. In: *Advances in Neural Information Processing Systems* 36 (Dec. 2023), pp. 47704–47720. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/94b472a1842cd7c56dcb125fb2765fbd-Abstract-Conference.html (visited on 03/14/2024).

[8] Andrea Agostinelli et al. *MusicLM: Generating Music From Text*. arXiv:2301.11325 [cs, eess]. Jan. 2023. URL: http://arxiv.org/abs/2301.11325 (visited on 09/01/2023).

[9] Shansong Liu et al. “Music Understanding LLaMA: Advancing Text-to-Music Generation with Question Answering and Captioning”.
In: *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X. Apr. 2024, pp. 286–290. DOI: 10.1109/ICASSP48485.2024.10447027. URL: https://ieeexplore.ieee.org/abstract/document/10447027?casa_token=946v3a0Lq-0AAAAA:RUha4Yn5LdITNpWJALPa4QdKS-Ps4Ttlm6Ah-UUZdimKGhFZ5_HrpoiC9zYPQg38KQwV0mUPPsA (visited on 05/01/2024).

[10] Max W. Y. Lam et al. “Efficient Neural Music Generation”. en. In: *Advances in Neural Information Processing Systems* 36 (Dec. 2023), pp. 17450–17463. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/38b23e2328096520e9c889ae03e372c9-Abstract-Conference.html (visited on 05/01/2024).

[11] Laith Alzubaidi et al. “Review of deep learning: concepts, CNN architectures, challenges, applications, future directions”.
In: *Journal of Big Data* 8.1 (Mar. 2021), p. 53. ISSN: 2196-1115. DOI: 10.1186/s40537-021-00444-8.
URL: https://doi.org/10.1186/s40537-021-00444-8 (visited on 05/12/2024).

[12] Hamed Taherdoost.
“Deep Learning and Neural Networks: Decision-Making Implications”. en.
In: *Symmetry* 15.9 (Sept. 2023). Number: 9 Publisher: Multidisciplinary Digital Publishing Institute, p. 1723. ISSN: 2073-8994. DOI: 10.3390/sym15091723.
URL: https://www.mdpi.com/2073-8994/15/9/1723 (visited on 05/12/2024).

[13] Cheng-Zhi Anna Huang et al.
AI Song Contest: Human-AI Co-Creation in Songwriting.
arXiv:2010.05388 [cs, eess]. Oct. 2020. DOI: 10.48550/arXiv.2010.05388.
URL: http://arxiv.org/abs/2010.05388 (visited on 05/09/2024).

[14] Shayan Dadman et al.
“Toward Interactive Music Generation: A Position Paper”. eng.
In: (Nov. 2022). Accepted: 2023-01-04T12:34:54Z Publisher: Institute of Electrical and Electronics Engineers. ISSN: 2169-3536.
DOI: 10.1109/ACCESS.2022.3225689.
URL: https://munin.uit.no/handle/10037/28026 (visited on 09/25/2023).

[15] Roberto Dominguez and Salvatore Cannella. “Insights on Multi-Agent Systems Applications for Supply Chain Management”. en.
In: *Sustainability* 12.5 (Jan. 2020). Number: 5 Publisher: Multidisciplinary Digital Publishing Institute, p. 1935. ISSN: 2071-1050. DOI: 10.3390/su12051935.
URL: https://www.mdpi.com/2071-1050/12/5/1935 (visited on 09/27/2023).

[16] Gasser Auda and Mohamed Kamel. “Modular neural networks: a survey”. In: *International Journal of Neural Systems* 09.02 (Apr. 1999). Publisher: World Scientific Publishing Co., pp. 129–151. ISSN: 0129-0657. DOI: 10.1142/S0129065799000125. URL: https://www.worldscientific.com/doi/abs/10.1142/S0129065799000125 (visited on 04/21/2024).

[17] C.P. Kruskal and A. Weiss. “Allocating Independent Subtasks on Parallel Processors”. In: *IEEE Transactions on Software Engineering* SE-11.10 (Oct. 1985). Conference Name: IEEE Transactions on Software Engineering, pp. 1001–1016. ISSN: 1939-3520. DOI: 10.1109/TSE.1985.231547. URL: https://ieeexplore.ieee.org/abstract/document/1701915 (visited on 03/15/2024).

[18] Patrick R. Amestoy et al. “A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling”. In: *SIAM Journal on Matrix Analysis and Applications* 23.1 (Jan. 2001). Publisher: Society for Industrial and Applied Mathematics, pp. 15–41. ISSN: 0895-4798. DOI: 10.1137/S0895479899358194. URL: https://epubs.siam.org/doi/abs/10.1137/S0895479899358194 (visited on 03/15/2024).

[19] Justin Reppert et al. *Iterated Decomposition: Improving Science Q&A by Supervising Reasoning Processes*. arXiv:2301.01751 [cs]. Jan. 2023. DOI: 10.48550/arXiv.2301.01751. URL: http://arxiv.org/abs/2301.01751 (visited on 03/15/2024).

[20] Edmund Durfee, D. L. Kiskis, and W.P. Birmingham. *IET Digital Library: The agent architecture of the University of Michigan Digital Library*. 1997. URL: https://digital-library.theiet.org/content/journals/10.1049/ip-sen_19971024 (visited on 04/21/2024).

[21] Marvin Minsky. *Society Of Mind*. en. Google-Books-ID: bLDLllfRpdkC. Simon and Schuster, Mar. 1988. ISBN: 978-0-671-65713-0.

[22] George E. Lewis. “Too many notes: Computers, complexity, and culture in voyager”. In: *New Media*. Num Pages: 14. Routledge, 2003. ISBN: 978-0-203-95385-3.

[23] Julian Moreira, Pierre Roy, and Francois Pachet. “VIRTUALBAND: INTERACTING WITH STYLISTICALLY CONSISTENT AGENTS”. en. In: (2013).

[24] Eduardo R. Miranda, Alexis Kirke, and Qijun Zhang. “Artificial Evolution of Expressive Performance of Music: An Imitative Multi-Agent Systems Approach”. en. In: *Guide to Computing for Expressive Music Performance*.

Ed. by Alexis Kirke and Eduardo R. Miranda. London: Springer, 2013, pp. 99–121. ISBN: 978-1-4471-4123-5. DOI: 10.1007/978-1-4471-4123-5_4. URL: https://doi.org/10.1007/978-1-4471-4123-5_4 (visited on 09/07/2023).

[25] Gemini Team et al. *Gemini: A Family of Highly Capable Multimodal Models*. arXiv:2312.11805 [cs]. Dec. 2023. DOI: 10.48550/arXiv.2312.11805. URL: http://arxiv.org/abs/2312.11805 (visited on 02/27/2024).

[26] Aditya Ramesh et al. "Zero-Shot Text-to-Image Generation". en. In: *Proceedings of the 38th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, July 2021, pp. 8821–8831. URL: https://proceedings.mlr.press/v139/ramesh21a.html (visited on 03/14/2024).

[27] Lucas Theis, Aäron van den Oord, and Matthias Bethge. *A note on the evaluation of generative models*. arXiv:1511.01844 [cs, stat]. Apr. 2016. URL: http://arxiv.org/abs/1511.01844 (visited on 09/26/2023).

[28] Trym Bø. *trymboe/mas_music_generation: Symbolic music generation using multi agent systems*. URL: https://github.com/trymboe/mas_music_generation (visited on 03/10/2024).

[29] Trym Bø. *Audio examples – Google Disk*. URL: https://drive.google.com/drive/u/0/folders/1OaxiKUGSwl5xi2mJShtslJL19ofKYd4t (visited on 05/01/2024).

[30] Trym Bø. *ML-jam in use*. URL: https://drive.google.com/file/d/1UGECq2rLIj9Vfn4tyNQhX82IZItzyA-w/view?usp=sharing (visited on 05/04/2024).

[31] Fumihide Tanaka and Shizuko Matsuzoe. "Children Teach a Care-Receiving Robot to Promote Their Learning: Field Experiments in a Classroom for Vocabulary Learning". en. In: *Journal of Human-Robot Interaction* (Aug. 2012), pp. 78–95. ISSN: 21630364. DOI: 10.5898/JHRI.1.1.Tanaka. URL: http://dl.acm.org/citation.cfm?id=3109685 (visited on 03/27/2024).

[32] Stephen A. Hedges. "Dice Music in the Eighteenth Century". In: *Music & Letters* 59.2 (1978). Publisher: Oxford University Press, pp. 180–187. ISSN: 0027-4224. URL: https://www.jstor.org/stable/734136 (visited on 09/06/2023).

[33] Jr Hiller and L. M. Isaacson. "Musical Composition with a High Speed Digital Computer". English. In: Audio Engineering Society, Oct. 1957. URL: https://www.aes.org/e-lib/browse.cfm?elib=189 (visited on 09/01/2023).

[34] J. L. Divilbiss. “The Real-Time Generation of Music with a Digital Computer”. In: *Journal of Music Theory* 8.1 (1964). Publisher: [Duke University Press, Yale University Department of Music], pp. 99–111. ISSN: 0022-2909. DOI: 10.2307/842975. URL: https://www.jstor.org/stable/842975 (visited on 05/01/2024).

[35] James Anderson Moorer. “Music and computer composition”. In: *Communications of the ACM* 15.2 (Feb. 1972), pp. 104–113. ISSN: 0001-0782. DOI: 10.1145/361254.361265. URL: https://dl.acm.org/doi/10.1145/361254.361265 (visited on 05/01/2024).

[36] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors”. en. In: *Nature* 323.6088 (Oct. 1986). Number: 6088 Publisher: Nature Publishing Group, pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0. URL: https://www.nature.com/articles/323533a0 (visited on 09/01/2023).

[37] Peter M. Todd. “A Connectionist Approach to Algorithmic Composition”. In: *Computer Music Journal* 13.4 (1989). Publisher: The MIT Press, pp. 27–43. ISSN: 0148-9267. DOI: 10.2307/3679551. URL: https://www.jstor.org/stable/3679551 (visited on 09/01/2023).

[38] MICHAEL C. MOZER. “Neural Network Music Composition by Prediction: Exploring the Benefits of Psychoacoustic Constraints and Multi-scale Processing”. In: *Connection Science* 6.2-3 (Jan. 1994). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/09540099408915726, pp. 247–280. ISSN: 0954-0091. DOI: 10.1080/09540099408915726. URL: https://doi.org/10.1080/09540099408915726 (visited on 03/27/2024).

[39] David Cope. “Experiments in musical intelligence (EMI): Non-linear linguistic-based composition”. In: *Interface* 18.1-2 (Jan. 1989). Publisher: Routledge _eprint: https://doi.org/10.1080/09298218908570541, pp. 117–139. ISSN: 0303-3902. DOI: 10.1080/09298218908570541. URL: https://doi.org/10.1080/09298218908570541 (visited on 04/08/2024).

[40] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (Nov. 1997). Conference Name: Neural Computation, pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[41] D. Eck and J. Schmidhuber. “Finding temporal structure in music: blues improvisation with LSTM recurrent networks”. In: *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*. Sept. 2002, pp. 747–756. DOI: 10.1109/NNSP.2002.1030094. URL: https://ieeexplore.ieee.org/abstract/document/1030094?casa_token=wDzOp72Qd_oAAAAA:12h0kLXp5vo2CQurSl_CPIVUwkyf-

3ahWsZAsSpxLsMCwHkxyR1aEvDYwySkr0LEO_5NNtp7F_Y (visited on 03/27/2024).

[42] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. *Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription*. arXiv:1206.6392 [cs, stat]. June 2012. DOI: 10.48550/arXiv.1206.6392.
URL: http://arxiv.org/abs/1206.6392 (visited on 09/25/2023).

[43] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. "DeepBach: a Steerable Model for Bach Chorales Generation". en. In: *Proceedings of the 34th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, July 2017, pp. 1362–1371.
URL: https://proceedings.mlr.press/v70/hadjeres17a.html (visited on 09/04/2023).

[44] Bob L. Sturm et al. *Music transcription modelling and composition using deep learning*. arXiv:1604.08723 [cs]. Apr. 2016. DOI: 10.48550/arXiv.1604.08723.
URL: http://arxiv.org/abs/1604.08723 (visited on 09/25/2023).

[45] Yu-Siang Huang and Yi-Hsuan Yang. "Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions". en. In: *Proceedings of the 28th ACM International Conference on Multimedia*. Seattle WA USA: ACM, Oct. 2020, pp. 1180–1188. ISBN: 978-1-4503-7988-5. DOI: 10.1145/3394171.3413671.
URL: https://dl.acm.org/doi/10.1145/3394171.3413671 (visited on 09/08/2023).

[46] Michael Wooldridge and Nicholas R. Jennings. "Intelligent agents: theory and practice". en. In: *The Knowledge Engineering Review* 10.2 (June 1995). Publisher: Cambridge University Press, pp. 115–152. ISSN: 1469-8005, 0269-8889. DOI: 10.1017/S0269888900008122.
URL: https://www.cambridge.org/core/journals/knowledge-engineering-review/article/intelligent-agents-theory-and-practice/CF2A6AAEEA1DBD486EF019F6217F1597 (visited on 09/27/2023).

[47] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Vol. 4. Nov. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.
URL: https://ieeexplore.ieee.org/abstract/document/488968?casa_token=n1H8WVmHG_wAAAAA:oefGkcY93hkuEbCku0Q4Dr22tEN239fb5dxhtQlTSZ93l_TZVvn0hCrx1B6F1ALeDXZWl6VxqAI (visited on 03/19/2024).

[48] Eduardo Reck Miranda and Al Biles, eds. *Evolutionary computer music*. en. OCLC: ocm80332658. London: Springer, 2007. ISBN: 978-1-84628-599-8.

[49] Palle Dahlstedt. “A MutaSynth in parameter space: interactive composition through evolution”. en. In: *Organised Sound* 6.2 (Aug. 2001), pp. 121–124. ISSN: 1469-8153, 1355-7718. DOI: 10.1017/S1355771801002084. URL: https://www.cambridge.org/core/journals/organised-sound/article/abs/mutasynth-in-parameter-space-interactive-composition-through-evolution/EC5207907A5B017E390EC5B8A6326E59 (visited on 03/28/2024).

[50] T.M. Blackwell and P. Bentley. “Improvised music with swarms”. In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*. Vol. 2. May 2002, 1462–1467 vol.2. DOI: 10.1109/CEC.2002.1004458. URL: https://ieeexplore.ieee.org/abstract/document/1004458?casa_token=TG11oWHCGOkAAAAA:Rp9pTQLT-aYO9587t4dIASAzAAJsr4azsva4tLMBSNTXF-HEpzV5LU26ttGUL1djZPLYEr4kdA (visited on 04/01/2024).

[51] Eduardo Reck Miranda. “At the Crossroads of Evolutionary Computation and Music: Self-Programming Synthesizers, Swarm Orchestras and the Origins of Melody”. In: *Evolutionary Computation* 12.2 (June 2004). Conference Name: Evolutionary Computation, pp. 137–158. ISSN: 1063-6560. DOI: 10.1162/106365604773955120. URL: https://ieeexplore.ieee.org/abstract/document/6790498 (visited on 05/10/2024).

[52] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (visited on 03/14/2024).

[53] *Midjourney*. URL: https://www.midjourney.com/home (visited on 02/27/2024).

[54] Tom Brown et al. “Language Models are Few-Shot Learners”. en. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 1877–1901. (Visited on 09/01/2023).

[55] Chris Donahue et al. *SingSong: Generating musical accompaniments from singing*. arXiv:2301.12662 [cs, eess]. Jan. 2023. URL: http://arxiv.org/abs/2301.12662 (visited on 09/01/2023).

[56] Yi-Jen Shih et al. “Theme Transformer: Symbolic Music Generation with Theme-Conditioned Transformer”. In: *IEEE Transactions on Multimedia* (2022). Conference Name: IEEE Transactions on Multimedia, pp. 1–1. ISSN: 1941-0077. DOI: 10.1109/TMM.2022.3161851.

[57] Botao Yu et al. “Museformer: Transformer with Fine- and Coarse-Grained Attention for Music Generation”. en. In: *Advances in Neural Information Processing Systems* 35 (Dec. 2022), pp. 1376–1388.
URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/092c2d45005ea2db40fc24c470663416-Abstract-Conference.html (visited on 08/31/2023).

[58] Thomas Nuttall, Behzad Haki, and Sergi Jorda.
“Transformer Neural Networks for Automated Rhythm Generation”. en. In: *International Conference on New Interfaces for Musical Expression* (Apr. 2021).
DOI: 10.21428/92fbeb44.fe9a0d82.
URL: https://nime.pubpub.org/pub/8947fhly/release/1 (visited on 09/30/2023).

[59] Seol-Hyun Noh.
“Analysis of Gradient Vanishing of RNNs and Performance Comparison”. en. In: *Information* 12.11 (Nov. 2021). Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, p. 442. ISSN: 2078-2489. DOI: 10.3390/info12110442.
URL: https://www.mdpi.com/2078-2489/12/11/442 (visited on 05/13/2024).

[60] Yang Qin et al. “Bar transformer: a hierarchical model for learning long-term structure and generating impressive pop music”. en.
In: *Applied Intelligence* 53.9 (May 2023), pp. 10130–10148. ISSN: 1573-7497.
DOI: 10.1007/s10489-022-04049-3.
URL: https://doi.org/10.1007/s10489-022-04049-3 (visited on 09/29/2023).

[61] Aditya Ramesh et al.
Hierarchical Text-Conditional Image Generation with CLIP Latents. en. arXiv:2204.06125 [cs]. Apr. 2022.
URL: http://arxiv.org/abs/2204.06125 (visited on 09/01/2023).

[62] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio.
Hierarchical Multiscale Recurrent Neural Networks. arXiv:1609.01704 [cs]. Mar. 2017. DOI: 10.48550/arXiv.1609.01704.
URL: http://arxiv.org/abs/1609.01704 (visited on 01/18/2024).

[63] Soroush Mehri et al.
SampleRNN: An Unconditional End-to-End Neural Audio Generation Model. arXiv:1612.07837 [cs]. Feb. 2017. DOI: 10.48550/arXiv.1612.07837.
URL: http://arxiv.org/abs/1612.07837 (visited on 01/18/2024).

[64] Jian Wu et al. *A Hierarchical Recurrent Neural Network for Symbolic Melody Generation | IEEE Journals & Magazine | IEEE Xplore*. 2019.
URL: https://ieeexplore.ieee.org/abstract/document/8918424?casa_token=wsmq6w4jjHkAAAAA:cpWejHyGUqnF8OG9k8qHMRsso5ToNzycjd1584euOxw-f930kjDtfnR8C1nN3TJy8iHcWH5Z (visited on 01/18/2024).

[65] Adam Roberts et al. “A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music”. en.
In: *Proceedings of the 35th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, July 2018, pp. 4364–4373.
URL: https://proceedings.mlr.press/v80/roberts18a.html (visited on 01/18/2024).

[66] Guo Zixun, Dimos Makris, and Dorien Herremans.
“Hierarchical Recurrent Neural Networks for Conditional Melody Generation with Long-term Structure”. en.
In: *2021 International Joint Conference on Neural Networks (IJCNN)*. Shenzhen, China: IEEE, July 2021, pp. 1–8. ISBN: 978-1-66543-900-8.
DOI: 10.1109/IJCNN52387.2021.9533493.
URL: https://ieeexplore.ieee.org/document/9533493/ (visited on 10/26/2023).

[67] Xipin Wei et al. *A Multi-Scale Attentive Transformer for Multi-Instrument Symbolic Music Generation*. arXiv:2305.16592 [cs, eess] version: 1. May 2023.
URL: http://arxiv.org/abs/2305.16592 (visited on 01/09/2024).

[68] Satish Chikkamath and Nirmala S R.
“Melody generation using LSTM and BI-LSTM Network”. en.
In: *2021 International Conference on Computational Intelligence and Computing Applications (ICCICA)*. Nagpur, India: IEEE, Nov. 2021, pp. 1–6.
ISBN: 978-1-66542-040-2. DOI: 10.1109/ICCICA52458.2021.9697286.
URL: https://ieeexplore.ieee.org/document/9697286/ (visited on 11/17/2023).

[69] Ke Chen et al. “The effect of explicit structure encoding of deep neural networks for symbolic music generation: 2019 International Workshop on Multilayer Music Representation and Processing, MMRP 2019”.
In: *Proceedings - 2019 International Workshop on Multilayer Music Representation and Processing, MMRP 2019*. Proceedings - 2019 International Workshop on Multilayer Music Representation and Processing, MMRP 2019 (Mar. 2019). Ed. by Adriano Barate et al. Publisher: Institute of Electrical and Electronics Engineers, pp. 77–84. DOI: 10.1109/MMRP.2019.8665362. URL: http://www.scopus.com/inward/record.url?scp=85064108523&partnerID=8YFLogxK (visited on 09/29/2023).

[70] Hugo Touvron et al. “Going Deeper With Image Transformers”. en. In: 2021, pp. 32–42. URL: https://openaccess.thecvf.com/content/ICCV2021/html/Touvron_Going_Deeper_With_Image_Transformers_ICCV_2021_paper.html (visited on 03/21/2024).

[71] Iz Beltagy, Matthew E. Peters, and Arman Cohan.
Longformer: The Long-Document Transformer. arXiv:2004.05150 [cs].
Dec. 2020. DOI: 10.48550/arXiv.2004.05150.
URL: http://arxiv.org/abs/2004.05150 (visited on 03/21/2024).

[72] Zihang Dai et al.
Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context.
arXiv:1901.02860 [cs, stat]. June 2019. DOI: 10.48550/arXiv.1901.02860.
URL: http://arxiv.org/abs/1901.02860 (visited on 03/21/2024).

[73] Aashiq Muhamed et al.
"Symbolic Music Generation with Transformer-GANs". en.
In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.1 (May 2021). Number: 1, pp. 408–417. ISSN: 2374-3468. DOI: 10.1609/aaai.v35i1.16117.
URL: https://ojs.aaai.org/index.php/AAAI/article/view/16117 (visited on 08/31/2023).

[74] Chris Donahue et al. *LakhNES: Improving multi-instrumental music generation with cross-domain pre-training.* arXiv:1907.04868 [cs, eess, stat]. July 2019.
URL: http://arxiv.org/abs/1907.04868 (visited on 08/31/2023).

[75] Aaron van den Oord et al. *WaveNet: A Generative Model for Raw Audio.*
arXiv:1609.03499 [cs]. Sept. 2016.
URL: http://arxiv.org/abs/1609.03499 (visited on 09/04/2023).

[76] Sam V. Norman-Haignere et al.
"Divergence in the functional organization of human and macaque auditory cortex revealed by fMRI responses to harmonic tones". en.
In: *Nature Neuroscience* 22.7 (July 2019). Number: 7 Publisher: Nature Publishing Group, pp. 1057–1060. ISSN: 1546-1726.
DOI: 10.1038/s41593-019-0410-7.
URL: https://www.nature.com/articles/s41593-019-0410-7 (visited on 09/04/2023).

[77] Sageev Oore et al.
"This time with feeling: learning expressive musical performance". en.
In: *Neural Computing and Applications* 32.4 (Feb. 2018). Company: Springer Distributor: Springer Institution: Springer Label: Springer Number: 4 Publisher: Springer London, pp. 955–967. ISSN: 1433-3058.
DOI: 10.1007/s00521-018-3758-9.
URL: https://link.springer.com/article/10.1007/s00521-018-3758-9 (visited on 08/30/2023).

[78] Mingliang Zeng et al.
MusicBERT: Symbolic Music Understanding with Large-Scale Pre-Training.
arXiv:2106.05630 [cs, eess]. June 2021. DOI: 10.48550/arXiv.2106.05630.
URL: http://arxiv.org/abs/2106.05630 (visited on 03/18/2024).

[79] Yi Ren et al. "PopMAG: Pop Music Accompaniment Generation".
In: *Proceedings of the 28th ACM International Conference on Multimedia.*
MM '20.
New York, NY, USA: Association for Computing Machinery, Oct. 2020,

pp. 1198–1206. ISBN: 978-1-4503-7988-5. DOI: 10.1145/3394171.3413721.
URL: https://dl.acm.org/doi/10.1145/3394171.3413721 (visited on 03/18/2024).

[80] Harish Kumar and Balaraman Ravindran. *Polyphonic Music Composition with LSTM Neural Networks and Reinforcement Learning.*
arXiv:1902.01973 [cs, eess, stat]. Mar. 2019.
URL: http://arxiv.org/abs/1902.01973 (visited on 09/06/2023).

[81] Kun Zhao et al.
"An Emotional Symbolic Music Generation System based on LSTM Networks".
In: *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. Mar. 2019, pp. 2039–2043.
DOI: 10.1109/ITNEC.2019.8729266.

[82] Huanru Henry Mao, Taylor Shin, and Garrison Cottrell.
"DeepJ: Style-Specific Music Generation".
In: *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*.
Jan. 2018, pp. 377–382. DOI: 10.1109/ICSC.2018.00077. URL:
https://ieeexplore.ieee.org/abstract/document/8334500 (visited on 03/21/2024).

[83] Christian Walder. "Modelling Symbolic Music: Beyond the Piano Roll". en.
In: *Proceedings of The 8th Asian Conference on Machine Learning.*
ISSN: 1938-7228. PMLR, Nov. 2016, pp. 174–189.
URL: https://proceedings.mlr.press/v63/walder88.html (visited on 05/01/2024).

[84] Hooman Rafraf. *Differential Music: Automated Music Generation Using LSTM Networks with Representation Based on Melodic and Harmonic Intervals.*
Publication Title: arXiv e-prints ADS Bibcode: 2021arXiv210810449R.
Aug. 2021. DOI: 10.48550/arXiv.2108.10449. URL:
https://ui.adsabs.harvard.edu/abs/2021arXiv210810449R (visited on 05/01/2024).

[85] Nathan Fradet et al. *Impact of time and note duration tokenizations on deep learning symbolic music modeling.* arXiv:2310.08497 [cs, eess]. Oct. 2023.
DOI: 10.48550/arXiv.2310.08497.
URL: http://arxiv.org/abs/2310.08497 (visited on 02/22/2024).

[86] Abhinav Jain et al.
"Overview and Importance of Data Quality for Machine Learning Tasks".
In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* KDD '20.
New York, NY, USA: Association for Computing Machinery, Aug. 2020,
pp. 3561–3562. ISBN: 978-1-4503-7998-4. DOI: 10.1145/3394486.3406477.
URL: https://dl.acm.org/doi/10.1145/3394486.3406477 (visited on 03/18/2024).

[87] C. Raffel. "[a]Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching, º Ph. D".

PhD Thesis. dissertation, Columbia University, 2016.
URL: https://colinraffel.com/projects/lmd/#license.

[88] Serkan Sulun, Matthew E. P. Davies, and Paula Viana.
"Symbolic Music Generation Conditioned on Continuous-Valued Emotions". en.
In: *IEEE Access* 10 (2022), pp. 44617–44626. ISSN: 2169-3536.
DOI: 10.1109/ACCESS.2022.3169744.
URL: https://ieeexplore.ieee.org/document/9762257/ (visited on 03/18/2024).

[89] Curtis Hawthorne et al. *Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset*. arXiv:1810.12247 [cs, eess, stat].
Jan. 2019. DOI: 10.48550/arXiv.1810.12247.
URL: http://arxiv.org/abs/1810.12247 (visited on 09/25/2023).

[90] Ziyu Wang et al.
POP909: A Pop-song Dataset for Music Arrangement Generation.
arXiv:2008.07142 [cs, eess]. Aug. 2020. DOI: 10.48550/arXiv.2008.07142.
URL: http://arxiv.org/abs/2008.07142 (visited on 09/25/2023).

[91] Jon Gillick et al.
"Learning to Groove with Inverse Sequence Transformations". en.
In: *Proceedings of the 36th International Conference on Machine Learning*.
ISSN: 2640-3498. PMLR, May 2019, pp. 2269–2279.
URL: https://proceedings.mlr.press/v97/gillick19a.html (visited on 09/25/2023).

[92] Jon Gillick and David Bamman. "What to Play and How to Play it: Guiding Generative Music Models with Multiple Demonstrations". en.
In: *International Conference on New Interfaces for Musical Expression*.
June 2021. DOI: 10.21428/92fbeb44.06e2d5f4.
URL: https://nime.pubpub.org/pub/s3x60926/release/1 (visited on 01/14/2024).

[93] Kyungyun Lee, Wonil Kim, and Juhan Nam.
PocketVAE: A Two-step Model for Groove Generation and Control.
arXiv:2107.05009 [cs, eess]. July 2021. DOI: 10.48550/arXiv.2107.05009.
URL: http://arxiv.org/abs/2107.05009 (visited on 01/14/2024).

[94] Ke Chen et al. *Music SketchNet: Controllable Music Generation via Factorized Representations of Pitch and Rhythm*. arXiv:2008.01291 [cs, eess, stat].
Aug. 2020. URL: http://arxiv.org/abs/2008.01291 (visited on 09/26/2023).

[95] Shuqi Dai et al. *Controllable deep melody generation via hierarchical music structure representation*. arXiv:2109.00663 [cs, eess]. Sept. 2021.
URL: http://arxiv.org/abs/2109.00663 (visited on 03/31/2024).

[96] Allen Huang and Raymond Wu. *Deep Learning for Music*.
arXiv:1606.04930 [cs]. June 2016. DOI: 10.48550/arXiv.1606.04930.
URL: http://arxiv.org/abs/1606.04930 (visited on 03/19/2024).

[97] Hang Chu, Raquel Urtasun, and Sanja Fidler.
Song From PI: A Musically Plausible Network for Pop Music Generation.
arXiv:1611.03477 [cs]. Nov. 2016.
URL: http://arxiv.org/abs/1611.03477 (visited on 09/11/2023).

[98] Dimos Makris et al. "Combining LSTM and Feed Forward Neural Networks for Conditional Rhythm Composition". en.
In: *Engineering Applications of Neural Networks.*
Ed. by Giacomo Boracchi et al.
Communications in Computer and Information Science.
Cham: Springer International Publishing, 2017, pp. 570–582.
ISBN: 978-3-319-65172-9. DOI: 10.1007/978-3-319-65172-9_48.

[99] Yichao Zhou et al. *BandNet: A Neural Network-based, Multi-Instrument Beatles-Style MIDI Music Composition Machine.* en. arXiv:1812.07126 [cs, eess]. Dec. 2018. URL: http://arxiv.org/abs/1812.07126 (visited on 01/09/2024).

[100] Jeff Ens and Philippe Pasquier. *MMM : Exploring Conditional Multi-Track Music Generation with the Transformer.* en. arXiv:2008.06048 [cs]. Aug. 2020.
URL: http://arxiv.org/abs/2008.06048 (visited on 01/09/2024).

[101] Hao-Wen Dong and Yi-Hsuan Yang. *Convolutional Generative Adversarial Networks with Binary Neurons for Polyphonic Music Generation.*
arXiv:1804.09399 [cs, eess, stat]. Oct. 2018. DOI: 10.48550/arXiv.1804.09399.
URL: http://arxiv.org/abs/1804.09399 (visited on 03/05/2024).

[102] Hao-Wen Dong et al. "MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment". en.
In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (Apr. 2018). Number: 1. ISSN: 2374-3468. DOI: 10.1609/aaai.v32i1.11312. URL:
https://ojs.aaai.org/index.php/AAAI/article/view/11312 (visited on 01/09/2024).

[103] Hongyuan Zhu et al. "XiaoIce Band: A Melody and Arrangement Generation Framework for Pop Music". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* KDD '18.
New York, NY, USA: Association for Computing Machinery, July 2018,
pp. 2837–2846. ISBN: 978-1-4503-5552-0. DOI: 10.1145/3219819.3220105.
URL: https://dl.acm.org/doi/10.1145/3219819.3220105 (visited on 03/05/2024).

[104] Gino Brunner et al. "JamBot: Music Theory Aware Chord Based Generation of Polyphonic Music with LSTMs". In: *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI).* ISSN: 2375-0197. Nov. 2017,
pp. 519–526. DOI: 10.1109/ICTAI.2017.00085. URL:
https://ieeexplore.ieee.org/abstract/document/8371988 (visited on 03/21/2024).

[105] Alfonso González-Briones et al. “Multi-Agent Systems Applications in Energy Optimization Problems: A State-of-the-Art Review”. en.
In: *Energies* 11.8 (Aug. 2018). Number: 8 Publisher: Multidisciplinary Digital Publishing Institute, p. 1928. ISSN: 1996-1073. DOI: 10.3390/en11081928.
URL: https://www.mdpi.com/1996-1073/11/8/1928 (visited on 09/27/2023).

[106] Thomas Schmickl and Heiko Hamann.
BEECLUST: A swarm algorithm derived from honeybees. Vol. 4.
Bio-inspired computing and communication networks. CRC Press, 2011.

[107] Thomas Stützle and Holger H. Hoos. “MAX-MIN Ant System”.
In: *Future Generation Computer Systems* 16.8 (June 2000), pp. 889–914.
ISSN: 0167-739X. DOI: 10.1016/S0167-739X(00)00043-1.
URL: https://www.sciencedirect.com/science/article/pii/S0167739X00000431
(visited on 03/19/2024).

[108] J. -L. Deneubourg et al.
“The self-organizing exploratory pattern of the argentine ant”. en.
In: *Journal of Insect Behavior* 3.2 (Mar. 1990), pp. 159–168. ISSN: 1572-8889.
DOI: 10.1007/BF01417909.
URL: https://doi.org/10.1007/BF01417909 (visited on 03/19/2024).

[109] Bert Hölldobler and Edward O. Wilson. *The Ants*. en.
Harvard University Press, 1990. ISBN: 978-0-674-04075-5.

[110] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz.
Swarm Intelligence: From Natural to Artificial Systems. en.
Google-Books-ID: fcTcHvSsRMYC. Oxford University Press, Sept. 1999.
ISBN: 978-0-19-803015-7.

[111] Shi Jinxin et al.
CGMI: Configurable General Multi-Agent Interaction Framework. en.
arXiv:2308.12503 [cs]. Aug. 2023.
URL: http://arxiv.org/abs/2308.12503 (visited on 04/30/2024).

[112] Shanshan Han et al. *LLM Multi-Agent Systems: Challenges and Open Problems*.
arXiv:2402.03578 [cs]. Feb. 2024. DOI: 10.48550/arXiv.2402.03578.
URL: http://arxiv.org/abs/2402.03578 (visited on 04/30/2024).

[113] Michael Wooldridge. *An Introduction to MultiAgent Systems*. en.
Google-Books-ID: X3ZQ7yeDn2IC. John Wiley & Sons, June 2009.
ISBN: 978-0-470-51946-2.

[114] Shayan Dadman and Bernt Arild Bremdal. “Multi-agent Reinforcement Learning for Structured Symbolic Music Generation”. en.
In: *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection*. Ed. by Philippe Mathieu et al.

Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2023, pp. 52–63. ISBN: 978-3-031-37616-0. DOI: 10.1007/978-3-031-37616-0_5.

[115] Miguel Delgado, Waldo Fajardo, and Miguel Molina-Solana. "Inmamusys: Intelligent multiagent music system". In: *Expert Systems with Applications* 36.3, Part 1 (Apr. 2009), pp. 4574–4580. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2008.05.028. URL: https://www.sciencedirect.com/science/article/pii/S0957417408002194 (visited on 03/21/2024).

[116] Pierre Potel, Frank Veenstra, and Kyrre Glette. "SoloJam Island : a platform for rhythmic experimentation in spatially segmented multi-agent systems". EN. In: (2021). Accepted: 2022-02-17T18:40:14Z Publisher: Aalborg University Copenhagen. DOI: 10.5281/zenodo.5717860. URL: https://www.duo.uio.no/handle/10852/91069 (visited on 09/28/2023).

[117] Maria Navarro, Juan Manuel Corchado, and Yves Demazeau. "MUSIC-MAS: Modeling a harmonic composition system with virtual organizations to assist novice composers". In: *Expert Systems with Applications* 57 (Sept. 2016), pp. 345–355. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2016.01.058. URL: https://www.sciencedirect.com/science/article/pii/S0957417416300227 (visited on 10/30/2023).

[118] Kıvanç Tatar and Philippe Pasquier. *MASOM: A Musical Agent Architecture based on Self-Organizing Maps, Affective Computing, and Variable Markov Models*. June 2017.

[119] Michael Bratman. *Intention, plans, and practical reason*. en. 1987. URL: https://philpapers.org/rec/braipa (visited on 03/23/2024).

[120] Anand Rao and Michael Georgeff. *Modeling rational agents within a BDI-architecture.* en. Google-Books-ID: TNl1Eb4LgmsC. Morgan Kaufmann, 1998. ISBN: 978-1-55860-495-7.

[121] Ryan Louie et al. "Novice-AI Music Co-Creation via AI-Steering Tools for Deep Generative Models". In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI '20. New York, NY, USA: Association for Computing Machinery, Apr. 2020, pp. 1–13. ISBN: 978-1-4503-6708-0. DOI: 10.1145/3313831.3376739. URL: https://dl.acm.org/doi/10.1145/3313831.3376739 (visited on 03/07/2024).

[122] Björn Þór Jónsson, Amy K. Hoover, and Sebastian Risi. "Interactively Evolving Compositional Sound Synthesis Networks". en. In: *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. Madrid Spain: ACM, July 2015, pp. 321–328.

ISBN: 978-1-4503-3472-3. DOI: 10.1145/2739480.2754796.
URL: https://dl.acm.org/doi/10.1145/2739480.2754796 (visited on 03/23/2024).

[123] Cheng-Zhi Anna Huang et al. *The Bach Doodle: Approachable music composition with machine learning at scale*. arXiv:1907.06637 [cs, eess, stat]. July 2019. URL: http://arxiv.org/abs/1907.06637 (visited on 03/23/2024).

[124] Maximos A. Kaliakatsos-Papakostas, Andreas Floros, and Michael N. Vrahatis. "Interactive music composition driven by feature evolution". en. In: *SpringerPlus* 5.1 (June 2016), p. 826. ISSN: 2193-1801. DOI: 10.1186/s40064-016-2398-8.
URL: https://doi.org/10.1186/s40064-016-2398-8 (visited on 03/05/2024).

[125] Li-Chia Yang and Alexander Lerch. "On the evaluation of generative models in music". en. In: *Neural Computing and Applications* 32.9 (May 2020), pp. 4773–4784. ISSN: 1433-3058. DOI: 10.1007/s00521-018-3849-7.
URL: https://doi.org/10.1007/s00521-018-3849-7 (visited on 09/26/2023).

[126] Alan M. Turing. "Computing Machinery and Intelligence". en. In: *Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer*. Ed. by Robert Epstein, Gary Roberts, and Grace Beber. Dordrecht: Springer Netherlands, 2009, pp. 23–65. ISBN: 978-1-4020-6710-5. DOI: 10.1007/978-1-4020-6710-5_3.
URL: https://doi.org/10.1007/978-1-4020-6710-5_3 (visited on 09/25/2023).

[127] Ning Zhang. "Learning Adversarial Transformer for Symbolic Music Generation". In: *IEEE Transactions on Neural Networks and Learning Systems* 34.4 (Apr. 2023). Conference Name: IEEE Transactions on Neural Networks and Learning Systems, pp. 1754–1763. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2020.2990746.

[128] Cheng-Zhi Anna Huang et al. *Music Transformer*. arXiv:1809.04281 [cs, eess, stat]. Dec. 2018.
URL: http://arxiv.org/abs/1809.04281 (visited on 08/30/2023).

[129] Feynman Liang and Mark Gotham. "AUTOMATIC STYLISTIC COMPOSITION OF BACH CHORALES WITH DEEP LSTM". en. In: (2017).

[130] Benedikte Wallace. "Predictive songwriting with concatenative accompaniment". eng. Accepted: 2018-08-08T22:01:00Z. MA thesis. 2018.
URL: https://www.duo.uio.no/handle/10852/62811 (visited on 04/19/2024).

[131] Hans Lindetorp et al. “Collaborative music-making: special educational needs school assistants as facilitators in performances with accessible digital musical instruments”. English.
In: *Frontiers in Computer Science* 5 (July 2023). Publisher: Frontiers.
ISSN: 2624-9898. DOI: 10.3389/fcomp.2023.1165442.
URL: https://www.frontiersin.org/articles/10.3389/fcomp.2023.1165442 (visited on 04/19/2024).

[132] Nanette Lindrupsen. “Exploring Textile Controllers for Computer Music”. eng. Accepted: 2021-09-24T22:04:41Z. MA thesis. 2021.
URL: https://www.duo.uio.no/handle/10852/88496 (visited on 04/19/2024).

[133] Abudukelimu Wuerkaixi et al. “COLLAGENET: FUSING ARBITRARY MELODY AND ACCOMPANIMENT INTO A COHERENT SONG”. en.
In: (2021).

[134] Halvor Sogn Haug. “Exploring interaction with a musical interface based on the BEECLUST algorithm”. eng. Accepted: 2022-08-23T22:00:17Z.
MA thesis. 2022.
URL: https://www.duo.uio.no/handle/10852/95572 (visited on 04/18/2024).

[135] Marco Pasini and Jan Schlüter.
Musika! Fast Infinite Waveform Music Generation. arXiv:2208.08706 [cs, eess]. Aug. 2022. DOI: 10.48550/arXiv.2208.08706.
URL: http://arxiv.org/abs/2208.08706 (visited on 04/18/2024).

[136] Nan Jiang et al. “RL-Duet: Online Music Accompaniment Generation Using Deep Reinforcement Learning”. en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.01 (Apr. 2020). Number: 01, pp. 710–718.
ISSN: 2374-3468. DOI: 10.1609/aaai.v34i01.5413.
URL: https://ojs.aaai.org/index.php/AAAI/article/view/5413 (visited on 04/18/2024).

[137] Matt Dean. *The Drum: A History*. en. Google-Books-ID: 9RmN7w8kVpAC. Scarecrow Press, Dec. 2011. ISBN: 978-0-8108-8171-6.

[138] Dmitri Tymoczko. “The Geometry of Musical Chords”.
In: *Science* 313.5783 (July 2006). Publisher: American Association for the Advancement of Science, pp. 72–74. DOI: 10.1126/science.1126287. URL: https://www.science.org/doi/full/10.1126/science.1126287 (visited on 04/18/2024).

[139] Keunwoo Choi, George Fazekas, and Mark Sandler.
Text-based LSTM networks for Automatic Music Composition.
arXiv:1604.05358 [cs]. Apr. 2016. DOI: 10.48550/arXiv.1604.05358.
URL: http://arxiv.org/abs/1604.05358 (visited on 01/14/2024).

[140] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. *MidiNet: A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation.* arXiv:1703.10847 [cs]. July 2017.
URL: http://arxiv.org/abs/1703.10847 (visited on 09/29/2023).

[141] Yongjie Huang, Xiaofeng Huang, and Qiakai Cai.
"Music Generation Based on Convolution-LSTM". en.
In: *Computer and Information Science* 11.3 (June 2018), p. 50.
ISSN: 1913-8997, 1913-8989. DOI: 10.5539/cis.v11n3p50.
URL: http://www.ccsenet.org/journal/index.php/cis/article/view/75550 (visited on 11/17/2023).

[142] Gordon B. Drummond and Brian D. M. Tom. "Statistics, probability, significance, likelihood: words mean what we define them to mean".
In: *Advances in Physiology Education* 35.4 (Dec. 2011). Publisher: American Physiological Society, pp. 361–364. ISSN: 1043-4046.
DOI: 10.1152/advan.00060.2011.
URL: https://journals.physiology.org/doi/full/10.1152/advan.00060.2011 (visited on 04/26/2024).

[143] R. F. Woolson. "Wilcoxon Signed-Rank Test". en.
In: *Wiley Encyclopedia of Clinical Trials*. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780471462422.eoct979. John Wiley & Sons, Ltd, 2008, pp. 1–3. ISBN: 978-0-471-46242-2.
DOI: 10.1002/9780471462422.eoct979.
URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780471462422.eoct979 (visited on 04/25/2024).

[144] R. A. Fisher. "Statistical Methods for Research Workers". en.
In: *Breakthroughs in Statistics: Methodology and Distribution.* Ed. by Samuel Kotz and Norman L. Johnson. New York, NY: Springer, 1925, pp. 66–70. ISBN: 978-1-4612-4380-9. DOI: 10.1007/978-1-4612-4380-9_6.
URL: https://doi.org/10.1007/978-1-4612-4380-9_6 (visited on 04/25/2024).

[145] H. B. Mann and D. R. Whitney. "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other".
In: *The Annals of Mathematical Statistics* 18.1 (1947). Publisher: Institute of Mathematical Statistics, pp. 50–60. ISSN: 0003-4851.
URL: https://www.jstor.org/stable/2236101 (visited on 04/26/2024).

[146] Milton Friedman. "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance". EN. In: *Journal of the American Statistical Association* (Dec. 1937). Publisher: Taylor & Francis Group.
URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1937.10503522 (visited on 04/25/2024).

[147] Donald W. Zimmerman and Bruno D. Zumbo. “Relative Power of the Wilcoxon Test, the Friedman Test, and Repeated-Measures ANOVA on Ranks”. In: *The Journal of Experimental Education* 62.1 (1993). Publisher: Taylor & Francis, Ltd., pp. 75–86. ISSN: 0022-0973. URL: https://www.jstor.org/stable/20152399 (visited on 04/25/2024).

[148] S. S. SHAPIRO and M. B. WILK. “An analysis of variance test for normality (complete samples)”. In: *Biometrika* 52.3-4 (Dec. 1965), pp. 591–611. ISSN: 0006-3444. DOI: 10.1093/biomet/52.3-4.591. URL: https://doi.org/10.1093/biomet/52.3-4.591 (visited on 04/25/2024).

[149] B. W. Yap and C. H. Sim. “Comparisons of various types of normality tests”. In: *Journal of Statistical Computation and Simulation* 81.12 (Dec. 2011). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00949655.2010.520163, pp. 2141–2155. ISSN: 0094-9655. DOI: 10.1080/00949655.2010.520163. URL: https://doi.org/10.1080/00949655.2010.520163 (visited on 04/25/2024).

[150] Nornadiah Mohd Razali and Yap Bee Wah. “Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests”. en. In: ().

[151] Kelly H. Zou et al. “Hypothesis Testing I: Proportions”. In: *Radiology* 226.3 (Mar. 2003). Publisher: Radiological Society of North America, pp. 609–613. ISSN: 0033-8419. DOI: 10.1148/radiol.2263011500. URL: https://pubs.rsna.org/doi/full/10.1148/radiol.2263011500 (visited on 04/27/2024).

[152] Olive Jean Dunn. “Multiple Comparisons among Means”. In: *Journal of the American Statistical Association* 56.293 (Mar. 1961). Publisher: Taylor & Francis _eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1961.10482090, pp. 52–64. ISSN: 0162-1459. DOI: 10.1080/01621459.1961.10482090. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1961.10482090 (visited on 04/27/2024).

[153] Sigrid Rouam. “False Discovery Rate (FDR)”. en. In: *Encyclopedia of Systems Biology*. Ed. by Werner Dubitzky et al. New York, NY: Springer, 2013, pp. 731–732. ISBN: 978-1-4419-9863-7. DOI: 10.1007/978-1-4419-9863-7_223. URL: https://doi.org/10.1007/978-1-4419-9863-7_223 (visited on 05/01/2024).

[154] D. Stowell et al. “Evaluation of live human–computer music-making: Quantitative and qualitative approaches”. In: *International Journal of Human-Computer Studies*. Special issue on Sonic Interaction Design 67.11 (Nov. 2009), pp. 960–975. ISSN: 1071-5819.

DOI: 10.1016/j.ijhcs.2009.05.007.
URL: https://www.sciencedirect.com/science/article/pii/S107158190900069X (visited on 04/19/2024).

[155] Sile O'Modhrain.
"A Framework for the Evaluation of Digital Musical Instruments".
In: *Computer Music Journal* 35.1 (2011). Publisher: The MIT Press, pp. 28–42.
ISSN: 0148-9267.
URL: https://www.jstor.org/stable/41241705 (visited on 04/19/2024).

[156] Herman Nordaunet et al.
Competitive Reinforcement Learning Agents with Adaptive Networks. EN.
Nov. 2023.
URL: https://drive.google.com/file/d/1prHOFA2kE6WAJ5qbkb4FGMywqWyAfLFs/view?usp=embed_facebook (visited on 01/21/2024).

[157] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. en. 2006.
URL: https://link.springer.com/book/10.1007/978-0-387-45528-0 (visited on 04/26/2024).

[158] Herman Pareli Nordaunet. "Multi-Speed Reinforcement Network: Enhancing Reinforcement Learning through Early Exit Neural Networks". eng.
Accepted: 2024-03-07T00:30:19Z. MA thesis. 2023.
URL: https://www.duo.uio.no/handle/10852/109209 (visited on 05/02/2024).

[159] Benjamin Genchel, Ashis Pati, and Alexander Lerch. *Explicitly Conditioned Melody Generation: A Case Study with Interdependent RNNs*. en.
arXiv:1907.05208 [cs, eess]. July 2019.
URL: http://arxiv.org/abs/1907.05208 (visited on 04/27/2024).

[160] Ke Chen, Gus Xia, and Shlomo Dubnov. "Continuous Melody Generation via Disentangled Short-Term Representations and Structural Conditions".
In: *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*.
ISSN: 2325-6516. Feb. 2020, pp. 128–135. DOI: 10.1109/ICSC.2020.00025. URL: https://ieeexplore.ieee.org/abstract/document/9031528 (visited on 04/27/2024).

[161] Kyoyun Choi et al.
"Chord Conditioned Melody Generation With Transformer Based Decoders".
In: *IEEE Access* 9 (2021). Conference Name: IEEE Access, pp. 42071–42080.
ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3065831. URL: https://ieeexplore.ieee.org/abstract/document/9376975 (visited on 04/27/2024).

[162] Yu Seung-yeon. *Improving Conditional Generation of Musical Components: Focusing on Chord and Expression*. 2023.
URL: https://s-space.snu.ac.kr/handle/10371/194079 (visited on 04/27/2024).

[163] Neil C. Thompson et al. *The Computational Limits of Deep Learning*. arXiv:2007.05558 [cs, stat]. July 2022. DOI: 10.48550/arXiv.2007.05558. URL: http://arxiv.org/abs/2007.05558 (visited on 04/27/2024).

[164] C. Spearman. *The Proof and Measurement of Association Between Two Things*. Studies in individual differences: The search for intelligence. Pages: 58. East Norwalk, CT, US: Appleton-Century-Crofts, 1961. DOI: 10.1037/11491-005.

[165] Flavio Ribeiro et al. "CROWDMOS: An approach for crowdsourcing mean opinion score studies". en. In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Prague, Czech Republic: IEEE, May 2011, pp. 2416–2419. ISBN: 978-1-4577-0538-0. DOI: 10.1109/ICASSP.2011.5946971. URL: http://ieeexplore.ieee.org/document/5946971/ (visited on 04/22/2024).

[166] Ankur Joshi et al. "Likert Scale: Explored and Explained". en. In: *British Journal of Applied Science & Technology* 7.4 (Jan. 2015), pp. 396–403. ISSN: 22310843. DOI: 10.9734/BJAST/2015/14975. URL: https://journalcjast.com/index.php/CJAST/article/view/381 (visited on 04/22/2024).

[167] Richard A. Berk. "An Introduction to Sample Selection Bias in Sociological Data". In: *American Sociological Review* 48.3 (1983). Publisher: [American Sociological Association, Sage Publications, Inc.], pp. 386–398. ISSN: 0003-1224. DOI: 10.2307/2095230. URL: https://www.jstor.org/stable/2095230 (visited on 05/01/2024).

[168] Robert A Moffitt, John L Adams, and Thomas Corbett. *Studies of Welfare Populations: Data Collection and Research Issues*. en. May 2002. URL: https://aspe.hhs.gov/reports/studies-welfare-populations-data-collection-research-issues-0 (visited on 05/01/2024).

[169] Emily A. Largent et al. "MONEY, COERCION, AND UNDUE INDUCEMENT: A SURVEY OF ATTITUDES ABOUT PAYMENTS TO RESEARCH PARTICIPANTS". In: *IRB* 34.1 (2012), pp. 1–8. ISSN: 0193-7758. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4214066/ (visited on 04/22/2024).

[170] James R Chromy and Savitri Abeyasekera. "Chapter 19 Statistical analysis of survey data". en. In: (2005).

[171] Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences*. 2nd ed. New York: Routledge, July 1988. ISBN: 978-0-203-77158-7. DOI: 10.4324/9780203771587.

[172] Gail M. Sullivan and Anthony R. Artino.
"Analyzing and Interpreting Data From Likert-Type Scales".
In: *Journal of Graduate Medical Education* 5.4 (Dec. 2013), pp. 541–542.
ISSN: 1949-8349. DOI: 10.4300/JGME-5-4-18. URL:
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3886444/ (visited on 04/25/2024).

[173] Albert LeBlanc et al. "Music Style Preferences of Different Age Listeners".
In: *Journal of Research in Music Education* 44.1 (1996). Publisher: [MENC: The National Association for Music Education, Sage Publications, Inc.], pp. 49–59.
ISSN: 0022-4294. DOI: 10.2307/3345413.
URL: https://www.jstor.org/stable/3345413 (visited on 04/26/2024).

[174] Arielle Bonneville-Roussy et al. "Age trends in musical preferences in adulthood: 1. Conceptualization and empirical investigation". en. In: *Musicae Scientiae* 21.4 (Dec. 2017). Publisher: SAGE Publications Ltd, pp. 369–389.
ISSN: 1029-8649. DOI: 10.1177/1029864917691571.
URL: https://doi.org/10.1177/1029864917691571 (visited on 04/26/2024).

[175] Marcelo Mortensen Wanderley and Nicola Orio. "Evaluation of Input Devices for Musical Expression: Borrowing Tools from HCI".
In: *Computer Music Journal* 26.3 (2002). Publisher: The MIT Press, pp. 62–76.
ISSN: 0148-9267.
URL: https://www.jstor.org/stable/3681979 (visited on 04/19/2024).

[176] Theresa M. Amabile.
A Model of Creativity and Innovation in Organizations | CiNii Research. 1998.
URL: https://cir.nii.ac.jp/crid/1570572700177521536 (visited on 04/20/2024).

[177] T. M. Amabile, C. M. Hadley, and S. J. Kramer.
"Creativity under the gun". en. In: (2002). Accepted: 2015-09-24T12:26:36Z
Publisher: Harvard Business Review. URL:
https://evidence.thinkportal.org/handle/123456789/31555 (visited on 04/20/2024).

[178] Maaike van den Haak, Menno De Jong, and Peter Jan Schellens.
"Retrospective vs. concurrent think-aloud protocols: Testing the usability of an online library catalogue".
In: *Behaviour & Information Technology* 22.5 (Sept. 2003). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/0044929031000, pp. 339–351.
ISSN: 0144-929X. DOI: 10.1080/0044929031000.
URL: https://doi.org/10.1080/0044929031000 (visited on 04/20/2024).

[179] Julia Barnett. "The Ethical Implications of Generative Audio Models: A Systematic Literature Review".
In: *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*.
arXiv:2307.05527 [cs, eess]. Aug. 2023, pp. 146–161.

DOI: 10.1145/3600211.3604686.
URL: http://arxiv.org/abs/2307.05527 (visited on 05/01/2024).

[180] Jan Smits and Tijn Borghuis.
"Generative AI and Intellectual Property Rights". en. In: *Law and Artificial Intelligence: Regulating AI and Applying AI in Legal Practice*.
Ed. by Bart Custers and Eduard Fosch-Villaronga.
The Hague: T.M.C. Asser Press, 2022, pp. 323–344. ISBN: 978-94-6265-523-2.
DOI: 10.1007/978-94-6265-523-2_17.
URL: https://doi.org/10.1007/978-94-6265-523-2_17 (visited on 05/01/2024).

[181] Varun Shah and Shubham Shukla. "Creative Computing and Harnessing the Power of Generative Artificial Intelligence". en. In: *Journal Environmental Sciences And Technology* 2.1 (Mar. 2023). Number: 1, pp. 556–579.
ISSN: 3006-046X.
URL: https://jest.com.pk/index.php/jest/article/view/123 (visited on 05/01/2024).

[182] Victoria Campbell. "Authors Guild v. Google, Inc. Case Summaries". eng.
In: *DePaul Journal of Art, Technology and Intellectual Property Law* 27.1 (2016), pp. 59–72. URL: https://heinonline.org/HOL/P?h=hein.journals/dael27&i=69 (visited on 05/01/2024).

[183] Peter Henderson et al. *Foundation Models and Fair Use*. arXiv:2303.15715 [cs]. Mar. 2023. DOI: 10.48550/arXiv.2303.15715.
URL: http://arxiv.org/abs/2303.15715 (visited on 05/01/2024).

[184] John Thickstun et al. *Anticipatory Music Transformer*. en.
arXiv:2306.08620 [cs, eess, stat]. June 2023.
URL: http://arxiv.org/abs/2306.08620 (visited on 05/01/2024).

[185] Martin Clancy. "Reflections on the financial and ethical implications of music generated by artificial intelligence". PhD Thesis. University of Dublin, 2021.
URL: http://www.tara.tcd.ie/bitstream/handle/2262/94880/REFLECTIONS%20ON%20THE%20FINANCIAL%20AND%20ETHICAL%20IMPLICATIONS%20OF%20MUSIC%20GENERATED%20BY%20ARTIFICIAL%20INTELLIGENCE.pdf (visited on 05/01/2024).

[186] K Egon, Kaledio Potter, and Mark Lord. "AI in Art and Creativity: Exploring the Boundaries of Human-Machine Collaboration".
In: *International Journal of Art and Art History* (Oct. 2023).

[187] Lukas Budach et al.
The Effects of Data Quality on Machine Learning Performance.
arXiv:2207.14529 [cs]. Nov. 2022. DOI: 10.48550/arXiv.2207.14529.
URL: http://arxiv.org/abs/2207.14529 (visited on 05/02/2024).

[188] Peng Li et al. “CleanML: A Study for Evaluating the Impact of Data Cleaning on ML Classification Tasks”.
In: *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. ISSN: 2375-026X. Apr. 2021, pp. 13–24. DOI: 10.1109/ICDE51399.2021.00009. URL: https://ieeexplore.ieee.org/abstract/document/9458702 (visited on 04/30/2024).

Appendix A

System Hyperparameters

A.1 Drum

Table A.1: Hyperparameters for Drum Agent.

Hyperparameter	Value
Optimizer	Adam
Learning Rate	0.00001
Number of Epochs	10000
Memory Length (cache)	256
Number of Layers	6
Batch Size	6
Dimension of Positional Embedding	256
Dimension of Input/Output Embedding	256
Number of Attention Heads	4
Dimension of Attention Heads	32
Hidden Size of Feed-Forward NN	1024
Dropout	0.2
Attention Dropout	0.1
Gradient Clip	0.25

A.2 Bass

Table A.2: Hyperparameters for Bass Agent.

Hyperparameter	Value
Optimizer	Adam
Number of Epochs	100
Max Batches Per Epoch	1000
Learning Rate	0.0001
Batch Size	8
Dimension of Embedding Pitch	128
Dimension of Embedding Duration	128
Number of Layers	4
Hidden Size of LSTM	128
Alpha 1 (Pitch)	0.7
Alpha 2 (Duration)	0.3

A.3 Chord

Table A.3: Hyperparameters for Chord Agent.

Hyperparameter	Value
Optimizer	Adam
Number of Epochs	50
Max Batches Per Epoch	200
Learning Rate	0.0001
Dimension of Embedding Root	128
Dimension of Embedding Chord	128
Number of Layers	4
Batch Size	8
Hidden Size of FC NN Pitch	64
Hidden Size of FC NN Duration	64

A.4 Melody

Table A.4: Hyperparameters for Melody Agent.

Hyperparameter	Value
Optimizer	Adam
Number of Epochs	100
Learning Rate	0.0005
Batch Size	64
Weight Decay	0.001
Dropout	0.5
Hidden Size LSTM	256
Number of Layers of Each LSTM	2
T1 Activation Function	ReLU
Kernel Size Conv Layer	4
T3 Cell Length	2
T2 Cell Length	8
T1 Cell Length	16

Appendix B

Drum Mappings

Table B.1: Mapping between the Roland TD-11 drum kit used to record the dataset, and the pitches used in this project. The numbers in parentheses are the corresponding MIDI pitches.

Roland TD-11	Mapped To
Kick	Bass (35)
Snare (Head)	Snare (38)
Snare (Rim)	Snare (38)
Snare X-Stick	Snare (38)
Tom 1	High Tom (50)
Tom 1 (Rim)	High Tom (50)
Tom 2	Low-Mid Tom (48)
Tom 2 (Rim)	Low-Mid Tom (48)
Tom 3 (Head)	High Floor Tom (45)
Tom 3 (Rim)	High Floor Tom (45)
HH Open (Bow)	Open Hi-Hat (46)
HH Open (Edge)	Open Hi-Hat (46)
HH Closed (Bow)	Closed Hi-Hat (42)
HH Closed (Edge)	Closed Hi-Hat (42)
HH Pedal	Closed Hi-Hat (42)
Crash 1 (Bow)	Crash Cymbal (49)
Crash 1 (Edge)	Crash Cymbal (49)
Crash 2 (Bow)	Crash Cymbal (49)
Crash 2 (Edge)	Crash Cymbal (49)
Ride (Bow)	Ride Cymbal (51)
Ride (Edge)	Ride Cymbal (51)
Ride (Bell)	Ride Cymbal (51)

Appendix C

Experiment 1 Hyperparameters

C.1 Multi-Agent Approach

Table C.1: Hyperparameters for the bass and chord networks from experiment 1 (they are both the same).

Hyperparameter	Value
Optimizer	Adam
Learning Rate	0.0001
Number of Epochs	50
Max Batches Per Epoch	500
Batch Size	8
Number of Layers	4
Hidden Size	128
Sequence Length	8
Dimension of Embedding (All)	128

C.2 Monolithic Approach

Table C.2: Hyperparameters for the monolithic network from experiment 1.

Hyperparameter	Value
Optimizer	Adam
Learning Rate	0.0001
Number of Epochs	50
Max Batches Per Epoch	500
Batch Size	8
Number of Layers	8
Hidden Size	128
Sequence Length	8
Dimension of Embedding (All)	128

Appendix D

Experiment 2 Hyperparameters

D.1 Coop and Non-Coop Network

Table D.1: Hyperparameters for both the Coop and Non-Coop networks.

Hyperparameter	Value
Optimizer	Adam
Number of Epochs	100
Learning Rate	0.0005
Batch Size	64
Weight Decay	0.001
Dropout	0.5
Hidden Size LSTM	256
Number of Layers of Each LSTM	2
T1 Activation Function	ReLU
Kernel Size Conv Layer	4
T3 Cell Length	2
T2 Cell Length	8
T1 Cell Length	16