electronics

MDPI

Article

Crafting Creative Melodies: A User-Centric Approach for Symbolic Music Generation

Shayan Dadman * and Bernt Arild Bremdal

Department of Computer Science, UiT, The Arctic University of Tromsø, Lodve Langesgate 2, 8514 Narvik, Norway; bernt.a.bremdal@uit.no
* Correspondence: shayan.dadman@uit.no

Abstract: Composing coherent and structured music is one of the main challenges in symbolic music generation. Our research aims to propose a user-centric framework design that promotes a collaborative environment between users and knowledge agents. The primary objective is to improve the music creation process by actively involving users who provide qualitative feedback and emotional assessments. The proposed framework design constructs an abstract format in which a musical piece is represented as a sequence of musical samples. It consists of multiple agents that embody the dynamics of musical creation, emphasizing user-driven creativity and control. This user-centric approach can benefit individuals with different musical backgrounds, encouraging creative exploration and autonomy in personalized, adaptive environments. To guide the design of this framework, we investigate several key research questions, including the optimal balance between system autonomy and user involvement, the extraction of rhythmic and melodic features through musical sampling, and the effectiveness of topological and hierarchical data representations. Our discussion will highlight the different aspects of the framework in relation to the research questions, expected outcomes, and its potential effectiveness in achieving objectives. Through establishing a theoretical foundation and addressing the research questions, this work has laid the groundwork for future empirical studies to validate the framework and its potential in symbolic music generation.

Keywords: artificial intelligence; deep reinforcement learning; human computer interaction; machine learning; multi-agent systems; symbolic music generation

Citation: Dadman, S.; Bremdal, B.A. Crafting Creative Melodies: A User-Centric Approach for Symbolic Music Generation. *Electronics* **2024**, *13*, 1116. https://doi.org/10.3390/electronics13061116

Academic Editors: Francisco A. Gómez Vela, Miguel García-Torres and Aurelio López-Fernández

Received: 30 December 2023
Revised: 1 March 2024
Accepted: 13 March 2024
Published: 18 March 2024

1. Introduction

Symbolic music generation refers to creating music in a format that represents musical notes and symbols rather than audio recordings. Indeed, the output of such models is a structured sequence of musical symbols that can be rendered into sound using software or played by musicians. The structure in symbolic music generation refers to the organization and arrangement of musical elements that conform to specific patterns and rules. It creates a coherent and logical progression within a composition. The symbolic music generation models are often trained to recognize and replicate these structural elements by exposing them to a large dataset of music examples [1]. However, simply mimicking existing structures is not sufficient for generating music that is perceived as original and engaging [2]. The models should be able to manipulate such structures creatively to produce original compositions while maintaining coherence and musical logic.

To enhance the structural aspect of generated music, we can incorporate domain-specific knowledge, such as music theory or compositional techniques, into the model's architecture or training process [3]. We may encode these rules directly into the model to ensure that the music conforms to specific music-related rules, such as harmonic progression. Similarly, we can use specialized loss functions that penalize structural incoherence. In addition, we may utilize hierarchical modeling techniques, where different model layers focus on different levels of structure, ranging from the micro level, such as note-to-note transitions, to the macro level, such as the overall form of the piece [4].

The process of creating symbolic music involves a delicate balance between structure and creativity [5]. The tension between these two elements is inherent in the composition process. We can consider creativity in music as generating novel ideas, motifs, and progressions that are aesthetically pleasing and evoke emotions. It requires the ability to break free from established patterns and to introduce elements of surprise and innovation. In this context, the structure provides a framework that helps to understand and appreciate creativity. It includes repeating themes, considering rhythmic and harmonic conventions, and organizing musical ideas into coherent forms such as sonatas or rondos.

Indeed, repetitions in music create a sense of form and familiarity for the listeners [6]. It enables them to identify and anticipate particular components within a composition. Variations, on the other hand, introduce diversity and complexity. It prevents monotony and sustains the listener's interest. The interplay between repetition and variation is a distinguishing feature of musical creativity and structure. It leads to forms like the theme, where a fundamental musical idea is repeated but altered in subsequent iterations.

The structure and creativity challenge is further complicated by the subjective nature of music. What one listener considers a creatively successful piece of music may differ from another listener's opinion, and this can vary widely across cultural contexts. Similarly, the degree of structure considered optimal may differ based on the genre of music and the audience's expectations. However, a user-centered approach can help music generation systems navigate these subjective complexities.

A user-centered approach to music generation involves actively involving users in the system's design, development, and refinement [7]. This approach prioritizes users' needs, preferences, and feedback to create a more personalized and satisfying musical experience. The goal is to generate music tailored to individual tastes, cultural contexts, and specific use cases. By incorporating user feedback, the system can adapt and evolve to create music that resonates more with its audience. This approach enhances the relevance and appeal of the generated music. It also ensures that the system remains flexible and responsive to the diverse and changing human musical preferences.

1.1. Design Considerations

The challenge for music generation models in this context lies in their tendency to either over-generalize, which leads to excessive repetition and lack of variation, or to produce random sequences, resulting in a lack of coherence and structure. These models are required to internalize the complex and often implicit rules that govern musical composition. They learn from examples in existing music datasets to create and balance the recurrence of specific patterns. However, these datasets often lack the diversity of musical tastes and the contextual nature of music preferences [2].

As [8] suggests in their work on MuseMorphose, there is a need for models that can exceed simple pattern replication to capture the essence of musical creativity. Potentially, we can achieve this through a framework using reinforcement learning algorithms. However, the definition of a reward function that covers objective and subjective characteristics of music remains challenging [2]. Furthermore, RL algorithms are limited in scalability, modularity, and homogeneous architecture, which can restrict their ability to handle the diverse and dynamic aspects of music generation. Therefore, multi-agent systems can be effective solutions with their distributed nature and collaborative approach. Such systems can effectively address issues of scalability and modularity by allowing multiple specialized agents to collaborate.

However, we need to ensure that learning and coordinating the actions of multiple agents is simple enough. Indeed, high complexity can lead to high computational demands and difficulties in maintaining the whole system's coherence while ensuring individual agents' autonomy. The integration of RL within a multi-agent systems (MAS) framework can be particularly effective [9]. By employing RL, agents within a MAS can independently refine their behaviors through a self-learning process considering exploration and exploita-

tion. This combination allows the agents to adapt not only to the musical context but also to the actions and behaviors of other agents.

In this direction, this paper introduces a new framework for generating symbolic music that puts the user at the center of the process. Our approach involves multiple knowledge agents and users working collaboratively to generate music. Our scientific contribution lies in the development of a multi-agent system (MAS) that integrates reinforcement learning (RL) with hierarchical data representations to foster a dynamic and adaptive music creation process. Our framework aims to balance user control and system autonomy, allowing users to guide the music generation process while enabling the system to learn and suggest creative ideas. This approach enhances the user's creative agency while improving the system's understanding of music composition through continuous user interaction and feedback.

1.2. Research Objectives

The objective of our research has been to analyze the shortcomings of existing approaches and attempt to create a synthesis of methods that overcome these shortcomings. Hence, our objective has been to design a framework that can lay the foundation for the symbolic music generation.

One of the crucial aspects we have considered is the user's role. Symbolic music generation needs to engage the user in the music creation process. The user is required to be in the loop, provide qualitative feedback, add emotional assessments, and lay the premises for the music generation. Hence, the framework needs to be user-centric in the sense that it assigns an active role for the user as an agent that can take an equal part in a collaborative effort with other knowledge agents. This means the user should be encouraged to interact and provide feedback to make his musical preferences salient. As such, we are pursuing an effective symbiosis of the system and the user.

In this direction, the framework is designed to adapt to new challenges and recover from potential failures in a practical and efficient manner. Agents within this framework can share and develop ideas, fostering a collaborative and rich process. The generative agent in such a framework can evolve by learning from past compositions, efficient agent communication, and active user interaction to enhance future results. Therefore, such a collaborative and adaptive framework needs to include an effective method for the underlying agent's interaction. This method will help the agents build on each other's strengths and compensate for weaknesses. Such collaborative interaction is essential when facing unforeseen challenges or recovering from errors and failures. Indeed, a shared understanding of the problem space allows agents to quickly realign their strategies and coordinate their efforts to address issues effectively.

Furthermore, the type of framework targeted needs to strike a balance between structured repetition and creative variations. This framework should be able to capture both short-term and long-term musical structures. Instead of traditional musical notes, such a framework is designed to use pre-recorded samples representing melodic patterns. Therefore, the agents within the framework aim to arrange these samples into compositions that replicate human-made music. By incorporating user feedback, they can improve their performance over time. This will result in customized musical outputs that cater to each user's taste and preference. In addition to user feedback, incorporating musical metrics as part of the reward function is crucial for guiding the agents toward generating high-quality music. It ensures that the generated music meets a certain level of technical proficiency while catering to individual user tastes.

Moreover, it is essential to consider the topological accuracy of the framework, as it ensures that the intrinsic relationships within music data are preserved. This allows for meaningful clustering and continuity, reflecting the perceptual and structural similarities in the musical samples. Indeed, topological integrity is crucial for maintaining the proximity of similar musical elements, which is essential for such a framework to acquire required musical knowledge and analyze musical influence across different generations. In addition

to topological accuracy, the framework needs to consider the hierarchical aspect of music data due to the layered nature of musical composition and production. The hierarchical structure allows for the representation of music at different levels of abstraction. Thus, the combined topological and hierarchical properties provide an exhaustive framework for analyzing and understanding music samples.

To achieve these objectives, some key research questions need to be answered:

- RQ1: What level of system autonomy should be traded in exchange for user involvement?
- RQ2: Can musical patterns and underlying relationships be revealed by extracting an exhaustive set of rhythmic and melodic features through musical sampling?
- RQ3: How effective can the topological and hierarchical representation of data be in organizing and adapting musical features at different levels of abstraction?
- RQ4: Can we establish metrics to ensure that a generative element of a system based on this framework maintains consistent musical styles or structures?
- RQ5: How effective can user involvement be in such a framework in navigating the agents toward understanding the user's musical preferences?
- RQ6: How can we incorporate user feedback into the agent's learning process while adapting to various stages of musical development within a framework?
- RQ7: Can we establish an effective communication method for agents to share and develop musical experiences and ideas?

The remainder of this paper is organized as follows: Section 2 reviews the existing relevant music generation models, highlighting their strengths and limitations and situating our work within this context. Sections 3 and 4 detail the computational techniques and algorithms employed by our framework, including data preprocessing, feature extraction, and the mechanics of the multi-agent system. Section 5 describes the distinct functions of the perceiving, generative, and human agents and how they collaborate within the music generation process. Sections 6 and 7 outline the iterative procedure for creating and refining music within our framework, and interface design considerations, respectively. Section 8 reflects on the implications of our user-centric approach and the potential for system expansion. Finally, Section 9 summarizes our findings and proposes directions for future research.

2. Related Works

Deep learning models can address several challenges in symbolic music generation in various ways, as highlighted by [1,2]. Among these models, MusicVAE [4] employs a hierarchical variational autoencoder framework to generate music with a structured approach. The Music Transformer [10] leverages the self-attention mechanism to capture long-range dependencies within musical pieces, thereby enhancing the coherence of the generated output. MuseGAN [11] utilizes generative adversarial networks to produce polyphonic music by learning and mimicking the input data distribution. Refer to [1,12] for a comprehensive overview of the deep learning models in symbolic music generation.

As stated earlier in Section 1, deep learning models for music generation mainly use a note-based approach. This can limit the models' ability to create music naturally incorporating repetitions and variations, particularly in longer compositions. These elements are crucial to the aesthetic and structural aspects of musical composition. Various models, including Museformer [13], GetMusic [14], MeloForm [15], and Music Transformer [10], address the challenges of generating music with structural integrity. However, these models rely on large datasets to learn and generate music and do not incorporate direct user feedback. Additionally, these models are complex and require significant computational resources for training and inference.

Table 1 summarizes the related studies in reinforcement learning (RL), user preference, and multi-agent systems (MAS). They offer insights into addressing the shortcomings of deep learning (DL) models in symbolic music generation. These studies highlight the potential of user interaction and adaptive learning mechanisms to enhance the music generation process.

Table 1. Relation of existing studies to the proposed framework.

Study Reference	Methodology	Results	Relation to Proposed Framework
RL-Tuner [3]	DQN and RNN models	Generated melodies using user-defined constraints	Suggests reinforcement learning integration in proposed framework
RE-RLTuner [16]	Extension of RL-Tuner with LDA	Improved handling of complex user constraints	Supports user involvement aspect of proposed framework
LSTM RNN composition [17]	LSTM RNN	Composed melody and chords	Provides a basis for sequential pattern learning in proposed framework
RL-Duet [18]	Actor-critic with GAE	Generated melodic and harmonic parts in real time	Informs real-time interaction and feedback in proposed framework
RL-Chord [19]	RL with conditional LSTM	Generated chord progression	Informs real-time interaction and feedback in proposed framework
Guzheng music [20]	RL and LSTM with DQN	Composed Guzheng music	Shows adaptability to specific musical styles for proposed framework
MusicRL [21]	RL-based method	Adapted to human preferences and Fine-tuned music generation	Emphasizes user feedback for personalization in proposed framework
NLP transformer [22]	NLP with transformer architecture	Created music based on user preferences	Supports user preference integration in proposed framework
Statistical ML [23]	Four modules for different music aspects	Generated music imitating styles from a seed song	Informs style imitation and personalization in proposed framework
Interactive GA [24]	DNN and transformer with genetic algorithm	Personalized music generation	Informs user interaction for refinement in proposed framework
Hierarchical CRNN-SA [25]	CRNN with self-attention	Categorized and suggested music based on user history	Informs user history-based adaptation in proposed framework
NN music [26]	MLP neural networks with PQF architecture	Trained networks by similarity algorithm	Suggests modular architecture for proposed framework
Musical agent with ART and RL [27]	ART and RL	Generated monophonic melody	Informs continuous learning and agent interaction in proposed framework
Improvagent [28]	Sarsa RL algorithm	Generated music with feedback from human musician	Supports improvisational and feedback aspects of proposed framework
CTRNNs and GA musical agent [29]	CTRNNs and genetic algorithms	Evolved multiple CTRNNs for sound events	Informs evolutionary aspects and sound event generation in proposed framework
Autonomous agents for melody and harmony [30]	RNN for melody and statistical model for harmony	Generated melodies and harmonies with feedback loop	Demonstrates agent specialization and feedback in proposed framework

2.1. Reinforcement Learning

Reinforcement learning (RL) algorithms emphasize the importance of interaction with the environment in the learning process. RL models engage in a process of trial and

error to adapt to new tasks and environmental changes. The adaptability of RL models originates from their ability to learn optimal policies through evaluating actions based on the rewards received. In symbolic music generation, an RL agent can be trained to recognize which sequences of notes, rhythms, or harmonies are more likely to lead to a structured and aesthetically pleasing composition. The agent's interaction with the environment, which in this case includes the musical framework and possibly feedback from listeners, allows it to refine its understanding of what constitutes successful melodic patterns. RL is especially useful in music generation, where creative exploration and the ability to respond to changing constraints are essential. In this context, the objective function is designed to capture various aspects of musicality. These aspects may include harmony, melody, rhythm, structure, novelty, tension-resolution patterns, and adherence to a particular musical style or genre. The design of this function is quite challenging as it needs to balance the core aspects of music theory with the subjective nature of music.

RL-Tuner [3] utilizes two DQN and two RNN models to generate melodies using user-defined constraints. Later, ref. [16] proposed an extension to the RL-Tuner that uses the latent Dirichlet allocation (LDA) called RE-RLTuner. This extension aims to expand the possibilities of music generation by improving the model's ability to handle complex user constraints. Ref. [17] used LSTM RNN to compose melody and chords, where the agent's objective is to find a suitable combination of sequences. RL-Duet [18] can generate melodic and harmonic parts in an online accompaniment framework using an actor-critic with a generalized advantage estimator (GAE). This model can generate music in response to input, like a human musician improvising in real time. RL-Chord [19] is a melody harmonization system using RL and conditional LSTM (CLSTM) to generate chord progression. Ref. [20] proposed a method using RL and LSTM to compose Guzheng music. They first trained the LSTM model on MIDI examples and optimized it by introducing the Guzheng playing techniques using the DQN algorithm.

2.2. User Preference

User preference studies aim to create systems that can adapt and respond to individual preferences by placing users at the center of the process. Within music generation tasks, this approach recognizes the subjective nature of music and seeks to personalize the music generation process by tailoring it to personal tastes, cultural contexts, and specific use cases. Researchers have explored various methods for capturing and integrating user feedback to achieve this. User preference studies and reinforcement learning models in music generation serve different but complementary purposes. User preference studies are primarily concerned with personalizing music to individual tastes, while reinforcement learning models focus on learning strategies to generate music that meets specific compositional goals. The following related user-centric music generation systems showcase the integration of machine learning techniques with user feedback mechanisms to refine and personalize the music creation process.

MusicRL [21] is an RL-based method for generating music that adapts to human preferences. It utilizes MusicLM to learn from user feedback and fine-tune music generation. MusicRL aims for ecologically valid music tested in real-world settings and is evaluated by human raters. Researchers in [22] combine natural language processing (NLP) with transformer architecture to create music based on user preferences. They trained the model on the TheoryTab database. User ratings guide the system to refine music generation, with higher scores leading to similar future segments. Listening tests assess the system's genre and mode accuracy. A statistical machine learning system in [23] generates music by imitating styles from a seed song. It uses four modules for different music aspects and a dataset of MIDI songs. The system is evaluated with ten pop songs to capture musical structure and style nuances.

The author in [24] suggests an interactive genetic algorithm system that uses a deep neural network (DNN) and transformer architecture to personalize music generation. The system is trained on the MAESTRO dataset, and user feedback is used to refine the system.

This ensures that the generated music aligns with individual tastes measured by user satisfaction. The system in [25] aims to enhance music-driven services by categorizing and suggesting music based on user history and preferences. It uses a hierarchical convolutional recurrent neural network with self-attention (CRNN-SA) to extract user music preferences from audio features. The system is trained on the Echo Nest dataset. It employs contrastive learning to capture user preferences. The model's performance is evaluated through objective measures, such as validation losses and positioning, and subjective experiments involving user satisfaction ratings.

2.3. Multi-Agent Systems

A multi-agent system (MAS) is a type of distributed system consisting of multiple intelligent agents interacting with each other [31]. In a MAS, each agent is an independent entity with specific abilities, knowledge, and objectives. It can perceive the environment, process information, make decisions, and act independently without relying on other agents. The agents in a MAS can either be homogeneous, meaning they have similar abilities, or heterogeneous, meaning they differ in their capabilities, knowledge, and goals. The interactions between these agents can lead to complex behaviors, enabling the system to solve problems beyond individual agents' abilities. The agents communicate and work together through these interactions to achieve a common goal. Effective communication is crucial as it allows the agents to exchange knowledge and intentions, while cooperation requires them to harmonize their efforts. Indeed, the cooperation mechanism aligns agents' contributions and actions, preventing conflicts.

In symbolic music generation, MAS translates to a system where different agents can be responsible for various aspects of music composition. Each agent optimizes its musical task by utilizing its specific set of rules or learned behaviors. NN music [26] is a musical agent that uses multi-layer perceptron (MLP) neural networks with PQF modular architecture [32]. The PQF architecture includes three modules: listening and analysis (P), performing/synthesis (Q), and generation (F). The model uses a similarity algorithm to train the networks by checking the similarity of the current states with the ones used in training. Smith and Garnett [27] propose a musical agent with adaptive resonance theory (ART) and reinforcement learning (RL) to generate monophonic melody. ART [33] is a framework that helps multiple autonomous agents learn to interact with their environment and each other in a stable and adaptive manner. The RL agents employ the ART to update their internal representations of the musical corpus continuously. They communicate through a shared representation of the musical state. This representation includes current chord progression, melody, and improvisational context information.

Improvagent [28] is a musical agent that utilizes the Sarsa reinforcement learning algorithm. Improvagent represents each note by pitch, duration, and velocity. The agent interacts with its environment by generating music and receiving feedback from a human musician or another agent. Given the inputs, the agent computes a set of features like onset, pitch, and rhythm. It considers the features as the states of the environment and clusters them using the k-nearest neighbors algorithm with Euclidean distance. Bown [29] introduces a musical agent that combines continuous-time recurrent neural networks (CTRNNs) and genetic algorithms (GA). The musical agent utilizes the CTRNN output to map synthesis parameters and trigger sound events. GA evaluates the CTRNN's success and evolves multiple CTRNNs in parallel.

The system proposed by [30] utilizes two autonomous agents, one focused on generating melodies and the other on harmonies. The melody agent uses a recurrent neural network to create melodic patterns, while the harmony agent employs a statistical model based on musical grammar to create harmonic progressions. The two agents respond to each other's outputs and communicate explicitly through this feedback loop. The system also includes a feature to generate different musical styles for the harmony agent by implementing various statistical models.

2.4. Research Gaps and Contributions

Studies such as RL-Tuner and RL-Duet have made notable progress utilizing reinforcement learning (RL) for generating music. They focus on optimizing musical compositions based on predefined reward functions that capture musicality and structure. However, these models often lack mechanisms for adapting to real-time user feedback, which is crucial for personalizing the music generation process. On the other hand, studies considering user preferences in music generation have shown the importance of tailoring the musical output to individual tastes. These studies usually rely on static datasets of user interactions or feedback to inform the generative process. While they achieve some level of personalization, they offer a different level of adaptability and responsiveness compared to an RL approach.

On the other hand, the MAS-based models mimic collaborative human interactions and often provide an improvisational system. These multi-agent systems demonstrate the potential for complex, emergent behavior arising from the interactions of multiple agents. However, agents within these systems must be exposed to a wide range of musical examples or scenarios to interact effectively and produce results that correspond to human musicians. In fact, the degree to which users can directly or creatively influence the system's interactions and outcomes is limited.

Additionally, there is a concern about the applicability of these methods across different musical genres and styles. Models tend to perform best within the specific context they were trained in but often struggle to replicate the nuances of other musical styles accurately. This difficulty is due to each genre's unique structural and rhythmic characteristics, which may not be fully captured within the model's learned feature space. Furthermore, these models utilize datasets that require time and expertise to construct and use effectively. Assembling and curating such datasets requires resources, which can be challenging for users, research teams, or creative projects with limited means. Consequently, adapting these models to diverse musical preferences is not an easy task.

Here, we present a new framework that combines reinforcement learning (RL) models with user preference studies to create adaptive music. The framework incorporates a multi-agent system (MAS) that uses hierarchical data representation to respond to user feedback and generate music. The framework's user-centered approach allows the system to learn from both the deep learning model's structural patterns and the evolving preferences of users through continuous interaction. Figure 1 illustrates the components and procedures of the proposed multi-agent framework for symbolic music generation.

Our study extends RL applications by introducing a framework that actively involves the user in the RL loop. Indeed, our framework benefits from the adaptability and long-term planning capabilities of RL models while also ensuring that the music generated is personalized, as emphasized in user preference studies. Integrating these methods allows for a more holistic approach to music generation that respects the composition's artistic integrity and the listener's subjective experience.

Moreover, our study proposes a new approach to maintaining coherence in the music generated by multi-agent systems (MAS) while balancing the autonomy of individual agents. This is achieved through an organizational structure where each agent specializes in music generation and user interaction. It allows for a refined approach to music generation that can adapt to the user's evolving tastes. Our framework learns from user interactions and adapts the generative strategies based on feedback. It ensures that the music produced is aligned with the user's current preferences.

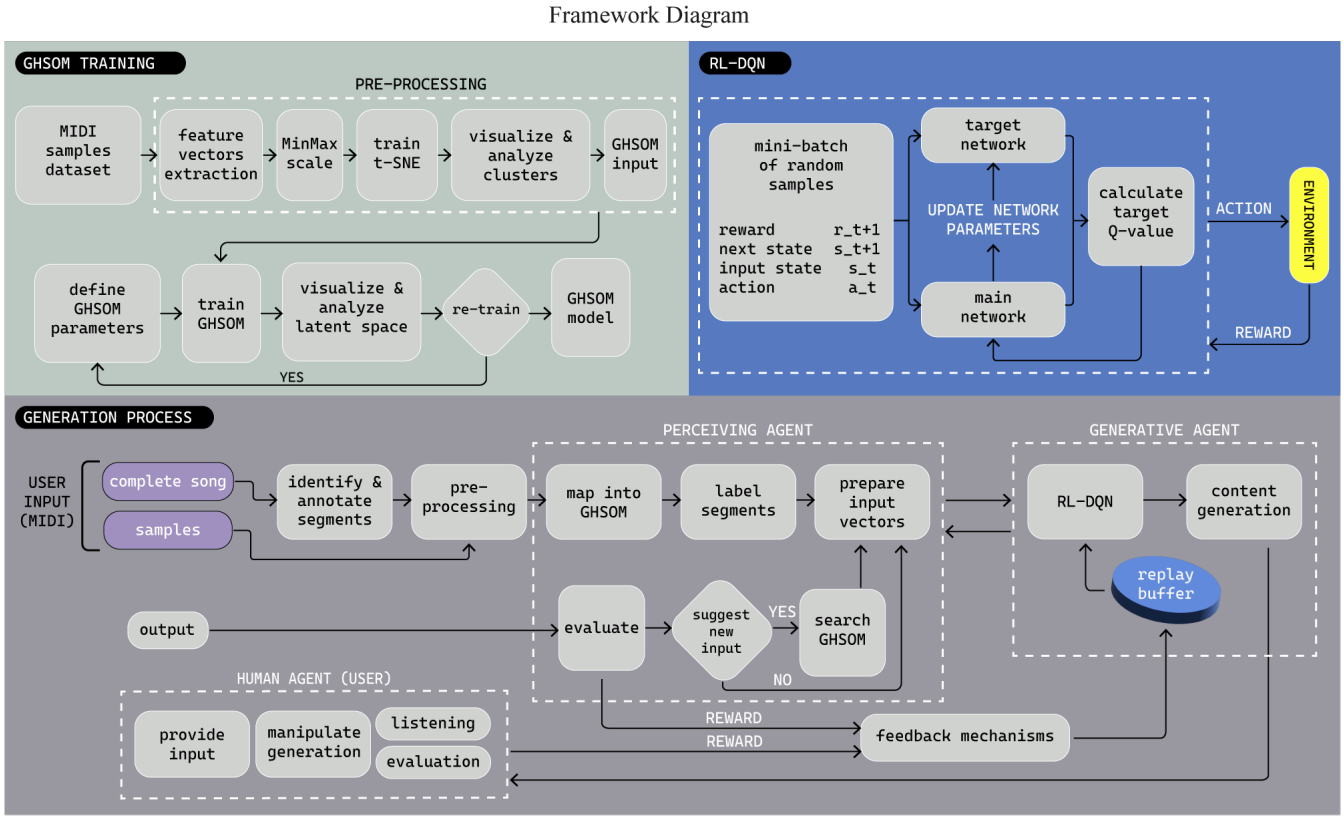


Figure 1. This diagram demonstrates the components and procedures of the proposed multi-agent framework for symbolic music generation. The "GHSOM Training" process explains how to create a hierarchical representation of music samples. To begin with, MIDI samples are pre-processed to obtain feature vectors, which are then prepared to train the growing hierarchical self-organizing maps (GHSOM). The obtained clusters are then visualized and analyzed for optimization. The reinforcement learning deep Q-network, "RL DQN", which interacts with the user and the environment, represents the framework's learning component. The "Generation Process" demonstrates the different steps and agent interactions involved in the music generation process. For more details, refer to Sections 5 and 6.

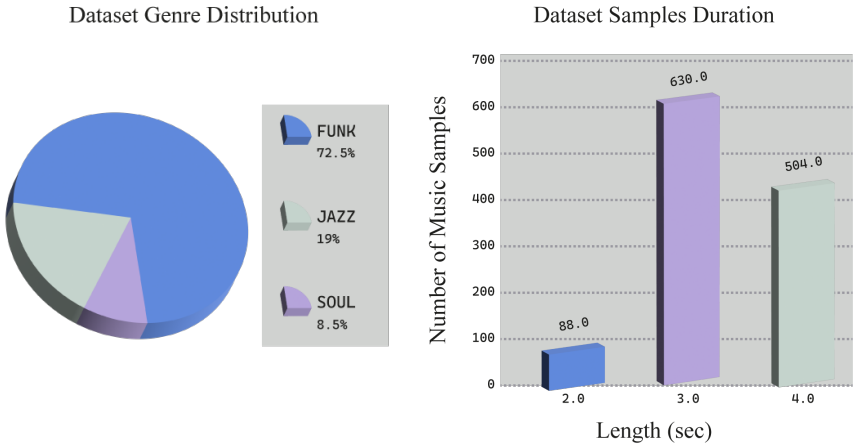
3. Methods of Use

The designed framework consists of methods that work together to achieve a subtle understanding and create musical sequences. The framework's musical knowledge relies on a set of MIDI samples providing structured musical information for learning and generation. The growing hierarchical self-organizing map (GHSOM) organizes musical knowledge in a hierarchical structure. The recurrent neural network with long short-term memory (RNN LSTM) captures temporal relationships within musical sequences. The framework's output is optimized through reinforcement learning (RL), which balances objective musicality with subjective aesthetic judgment. Additionally, t-distributed stochastic neighbor embedding (t-SNE) helps with dimensionality reduction and dataset structuring. Moreover, it facilitates guiding the GHSOM's configuration to represent musical samples better. These methods form the main framework's components and contribute to the overall functionality and performance.

3.1. Data

The framework uses a dataset of samples to acquire musical knowledge. This dataset includes a collection of melodic patterns. Samples can be recordings of real instruments, like guitar riffs, piano chords, or a sequence of musical notes. They are repeating sections of a sound recording or musical piece that provide repetition and structure. They can be

played back in sequence to create a continuous and rhythmic pattern. They come in the form of audio or MIDI representation. Percussive samples, like drum patterns or grooves, are commonly used to maintain a consistent beat and drive the music forward. Additionally, harmonic and melodic samples consist of repeating melodic phrases or chord progressions. These samples can serve as the foundation for the composition and can be layered to create musical arrangements. In this study, we consider a dataset of melodic samples in MIDI representation to explain the underlying processes within the framework. The dataset comprises 1222 MIDI samples in three main genres: Jazz, Funk, and Soul. Figure 2 presents the distribution of the genre and the sample length.



(**a**) (**b**)

Figure 2. (**a**) The genre distribution of dataset samples. (**b**) The distribution of samples duration in seconds. The annotations on top of the bars indicate the corresponding number of samples for the specific value (bar).

Musical timing is out of the scope of this study. Therefore, to ensure consistent timing throughout the dataset, we impose a fixed time signature of 4/4, a velocity of 80, and a tempo of 120 BPM for all samples. This standardization allows for a more uniform comparison of musical features across samples. Furthermore, all the samples are quantized to 16th s, meaning that the timing of the musical events is aligned to a predefined grid or rhythmic structure. The quantization ensures that all samples have a consistent rhythmic alignment. This allows for a fair comparison of their melodic and harmonic features and facilitates extracting meaningful musical features.

The framework uses this dataset to train the GHSOM model, which serves as a knowledge base for the system. During the generation process, the framework can also receive complete MIDI songs that serve as structural guidance for the RL agent. We will elaborate on this later in Section 6.

3.2. Growing Hierarchical Self-Organizing Maps (GHSOM)

Therefore, in the designed framework, the growing hierarchical self-organizing maps (GHSOM) represents a potential method that can be used as musical knowledge. The GHSOM can better represent music data due to its topological and hierarchical properties. Topologically, GHSOM preserves the spatial arrangement of the input music data in a lower-dimensional grid, which helps maintain the relationships between different musical features. This is important in clustering samples with similar characteristics such as tempo, rhythm, and harmonic progressions. Hierarchically, GHSOM reflects the layered structure of music by adaptively growing its architecture to various levels of granularity based on the complexity of the data.

To elaborate more, GHSOM is an unsupervised machine learning algorithm that learns to represent high-dimensional input data in a lower-dimensional space [34]. GHSOM extends the capabilities of the traditional self-organizing map (SOM) [35] to better handle

complex and hierarchical data structures. It has the ability to dynamically grow and adapt its architecture to represent the input data more accurately. This feature allows it to provide a more nuanced and detailed representation of high-dimensional datasets.

The GHSOM algorithm uses competitive learning to operate. It achieves this by adjusting prototype vectors, also known as codebook vectors, through iterative processes to represent input data best. These vectors serve as neurons or map nodes. The algorithm organizes them in a way that preserves the topological properties of the input space. This ensures that similar data points are mapped close to each other in the lower-dimensional space.

The GHSOM is characterized by its hierarchical structure, which is not predetermined but obtained from the data. This means the algorithm adapts to the data's structure without prior knowledge or assumptions. This approach enables the GHSOM to represent the hierarchical relationships present in the data. One can choose a broader, flatter representation with detailed refinements at each layer or a deeper hierarchy that emphasizes the separation of subclusters into distinct maps. This feature allows customizing the level of detail in data representation based on specific needs.

Furthermore, the architecture of GHSOM is defined by parameters that determine the balance between shallow and deep hierarchies. Its goal is to distribute data samples evenly across the map space to facilitate exploratory data analysis. This uniform distribution is critical in densely populated regions of the data space where subtle differences between clusters can be captured. The algorithm uses the quantization error of individual units to better represent these nuances instead of the mean quantization error.

The GHSOM uses parameter τ to determine the global stopping criterion. This criterion affects the size of the map and its level of detail. One noteworthy characteristic of the GHSOM is that its training process often results in an imbalanced hierarchy with branches of varying depths. However, this is actually an advantage over other hierarchical models because it allows the GHSOM to adapt its structure to the complexity and diversity of the input data, especially musical data, which can have many aspects or features.

3.3. Recurrent Neural Networks (RNNs)

Recurrent neural networks (RNNs) are a class of neural networks that process sequential data by allowing information to persist over time. The recursive nature of RNNs enables them to retain a form of "memory". This mechanism makes them proficient at tasks where context from previous inputs is crucial for understanding current and future data points. However, despite their usefulness, RNNs can suffer from the vanishing and exploding gradient problem. This limits their ability to capture long-term dependencies in sequential data.

During the backpropagation phase, the gradients of the loss function are sent backward in time to update the network's weights. However, when these gradients are sent over multiple time steps, they can either shrink exponentially, leading to vanishing gradients, or increase exponentially, resulting in exploding gradients. The vanishing gradient problem makes it challenging for the network to learn and maintain information over long sequences, thus limiting its ability to capture long-term dependencies. On the other hand, exploding gradients can cause learning to diverge, leading to unstable network behavior.

Long short-term memory (LSTM) is a type of RNN that effectively addresses this problem with a gating mechanism that regulates information flow. This mechanism consists of three sigmoidal gates and one hyperbolic tangent gate that selectively update, forget, and output information. The input gate plays a role in regulating the inflow of new information into the cell state. On the other hand, the forget gate determines which part of the existing information needs to be disregarded. Finally, the output gate determines the information that should be outputted at the current time step.

Therefore, LSTMs can maintain a stable gradient over time by selectively updating and forgetting information. This allows them to learn from and remember information over extended sequences. This characteristic makes LSTMs highly effective for many applications that require modeling long-term dependencies, such as music structure. Therefore,

the designed framework utilizes the LSTM model to capture short-term and long-term musical structures.

3.4. Reinforcement Learning (RL)

Reinforcement learning (RL) is a machine learning subfield that trains agents to make sequential decisions. RL is characterized by an interactive process where an agent learns to perform tasks by operating within an environment and receiving feedback through rewards or punishments. Through the interaction, the agent learns from feedback and adapts behavior to achieve a specific goal. The agent aims to maximize its cumulative reward over time by learning a policy that maps states to actions.

RL algorithms can be value based or policy based. Value-based methods aim to learn the optimal value function. The optimal value function represents the expected long-term return for each state, or state-action pair, under an optimal policy. The value function quantifies the quality of states and actions. It guides the agent towards the most rewarding decisions. One of the most well-known value-based algorithms is Q-learning, where the agent learns an action-value function that gives the expected utility of taking a given action in a given state and following the optimal policy after that.

On the other hand, policy-based methods aim to learn the best policy directly without requiring value estimation as an intermediate step. These approaches adjust the parameters of the policy in a manner that maximizes the expected return. Policy-based methods can be more effective in high-dimensional or continuous action spaces where value-based methods may have limitations.

Additionally, there are hybrid approaches, such as deep Q-networks (DQNs) [36], which combine Q-learning with deep neural networks to approximate the Q-value function and handle large and continuous state spaces. DQN tackles the challenge of Q-learning becoming impractical in environments with large or continuous state spaces. This is due to the Q-function becoming too complex to represent with a simple tabular approach. In the DQN algorithm, the neural network receives the current state of the environment as input and produces a Q-value for each available action. The main advantage of DQN is its capability to generalize from the observed states to the unobserved ones, which empowers the agent to make informed decisions even in unfamiliar regions of the state space.

The designed framework uses the DQN algorithm to understand and optimize the learning of musical composition and structure. By framing the task of music structure learning as a sequential decision-making problem, DQNs can identify patterns and transitions that are specific to different musical styles. Here, the deep neural network component of DQNs consists of the LSTM model, which enables the RL agent to predict and generate musically coherent sequences. Additionally, the agent's ability to continuously enhance its policy through interaction with a musical environment allows for dynamic adaptation to various genres or preferences.

3.5. t-Distributed Stochastic Neighbor Embedding (t-SNE)

Dimensionality reduction is a technique to decrease the number of features or variables in a dataset without losing important information. There are two types of dimensionality reduction techniques: linear and nonlinear. The t-distributed stochastic neighbor embedding (t-SNE) [37] algorithm is a nonlinear technique that can be used to visualize high-dimensional data. The main objective of t-SNE is to reduce the high-dimensional data space and display it in a lower-dimensional space while maintaining its local structure as much as possible. t-SNE is highly effective for analyzing complex datasets where the relationships between data points are not easily visible in the original high-dimensional space.

t-SNE has an advantage in capturing data's local and global structure by revealing clusters at various scales. This is accomplished by the perplexity parameter, which functions as a knob that determines the number of effective nearest neighbors. It balances attention between the local and global aspects of the data and can be adjusted to emphasize different structures within the data. For music, Refs. [38,39] compared various dimensionality

reduction methods using extracted meaningful features from music and textural sound data. They observed that t-SNE performs much better in preserving the local structure of the original data and keeping the distinct sub-groups separated in visualization.

The designed framework uses the t-SNE algorithm as a preprocessing step to prepare training examples for the GHSOM model. The t-SNE algorithm helps visualize high-dimensional data in two or three dimensions, which can reveal the data's intrinsic clustering and structure. This visualization assists in making decisions about the GHSOM's parameters to control its hierarchical depth and architecture. Furthermore, the t-SNE algorithm can effectively separate clusters even when they are nonlinearly related. This results in more distinct training examples for the GHSOM model, which can improve its ability to identify fine-grained patterns and hierarchical relationships. The algorithm's focus on preserving local neighborhoods can also act as a feature denoising mechanism, reducing noise and emphasizing relevant features, resulting in more robust training outcomes.

Moreover, the reduced dimensionality can make it more efficient to train the GHSOM model, particularly with large datasets. The nonlinear relationships that t-SNE can uncover help to reveal complex data interactions suitable for GHSOM learning. This can potentially lead to faster convergence and improved model performance. However, it is essential to remember that t-SNE can be computationally intensive, and it is necessary to avoid overfitting the visual patterns it discovers.

4. Data Preprocessing

Data preprocessing is a crucial step before training machine learning models. It involves several steps to clean and organize the raw data, making it suitable for model training. It ensures that the machine learning model can effectively learn from the information provided. The designed framework, after feature extraction, performs data transformation and reduction.

Data transformation is a powerful tool that converts data into a more suitable format for model training. It includes encoding categorical variables into numerical representations, a process allowing the model to understand and learn from them. It also involves scaling features to a standard range, ensuring that all features have an equal opportunity to influence the model's learning. Lastly, it includes normalizing data, ensuring all features contribute equally to the model's learning process.

Data reduction is the final step in data preprocessing. It aims to reduce the dimensionality of the data while preserving its essential information and the underlying structure. This is particularly useful when dealing with large and complex datasets. Indeed, reducing the complexity of the dataset while maintaining its quality helps to decrease the computational cost associated with clustering tasks. Additionally, it can filter out irrelevant or redundant features to minimize the impact of noise on the model's performance.

4.1. Feature Extraction

To facilitate the analysis of the samples within the dataset, the designed framework processes the MIDI files to extract a set of relevant musical features. The feature extraction is performed using the `pretty_midi` [40] and `MusPy` [41] libraries. These features are represented as a vector for each music sample, which provides a structured and quantifiable profile of the musical content. The resulting feature vector includes various aspects of a musical piece, from pitch characteristics to rhythmic patterns. Table 2 summarizes the musical features included in the feature vector.

These features are expected to assist the agents in recognizing musical patterns and analyzing the dataset for learning and generating coherent music.

To elaborate, structural features such as the number of notes, the duration of samples, and note density maintain the music's form, pacing, and complexity characteristics. Tonal features, including the pitch range and the average pitch, provide insights into the tonal characteristics of the sample. They help to maintain a consistent tonal center and utilize a musically relevant pitch range during music analysis and generation.

Table 2. Summary of musical features.

Parameter	Description
`number_of_notes`	Number of notes of the sample.
`length_sec`	Duration of the piece in seconds.
`pitch_range_min_note`	Lowest note value in the sample's MIDI note range.
`pitch_range_max_note`	Maximum note value in the sample's MIDI note range.
`number_pitches_used`	The total number of unique pitches in the sample.
`number_pitch_classes_used`	The total number of unique pitch classes used in the sample.
`overall_pitch_contour`	General shape or trend of the pitch over time, which can be ascending, descending, or static.
`average_pitch`	Mean pitch value across all notes measured in Hz.
`pitch_entropy`	Measure of the diversity of pitches in the sample. Higher values indicate greater diversity in pitch usage.
`pitch_class_entropy`	Measure of the diversity of pitch classes in the sample. Higher values indicate greater diversity in pitch classes usage.
`interval_range_min`	Minimum interval between consecutive notes. It provides insight into the sample's melodic intervals.
`interval_range_max`	Maximum interval between consecutive notes. It can indicate the presence of large leaps within the sample.
`groove`	Rhythmic feel or swing in the music. It helps to perceive style and mood of the sample.
`average_polyphony`	The average number of pitches played concurrently. It reflects the textural complexity of the sample.
`maximum_polyphony`	Maximum number of pitches played concurrently. It indicates the maximum textural density of the sample.
`note_density`	Density of notes measured as notes per unit of time. It indicates the sample's overall complexity.
`average_pitch_category`	Distinct categories of the average pitch of a sample based on the frequency range it falls into.

Melodic features such as pitch contour and interval ranges guide specific melodic trajectories. The `overall_pitch_contour` feature provides information about the general direction of the melody over time. The `interval_range_min` and `interval_range_max` features capture the smallest and largest intervals between consecutive notes within a sample. These features facilitate understanding the melodic range and the potential for expressive leaps within a piece.

Features that measure polyphony capture harmonic texture. They refer to the overall texture density of the sample. The `average_polyphony` feature measures the average number of notes played simultaneously throughout a piece. For instance, a lower value of this feature might indicate a piece with a simpler texture, such as a solo melody line or a melody with sparse accompaniment. On the other hand, the `maximum_polyphony` feature indicates the maximum number of notes played simultaneously at any point in the sample. It can reveal the peaks in textural density and the potential for harmonic fullness within the sample.

The `groove` feature maintains the rhythmic feel of the music. We obtain this feature by analyzing the microtiming deviations between the onset times of the notes. First, we extract the onset times of the notes. Then, we calculate the inter-onset intervals between the notes. Finally, we use the standard deviation of these intervals to measure the groove of the sample. We use the following equation to calculate the groove:

$$\texttt{groove} = \texttt{std}(\texttt{inter\_onset\_intervals}) \quad (1)$$

where `inter_onset_intervals` is an array of the time intervals between the onsets of each note in the sample, and `std` is the standard deviation of these intervals.

Features measuring pitch and pitch class entropy indicate tonal diversity and unpredictability. The `pitch_entropy` feature considers the specific octaves in which pitches are played and calculates their unpredictability throughout a piece. For instance, a high entropy value suggests that the composition uses a wide range of pitches in a less predictable manner. Meanwhile, a low pitch entropy indicates that the piece relies on a more limited set of pitches and may follow a more predictable pattern. However, `pitch_class_entropy` calculates the entropy for the pitch classes. This measure provides insight into the diversity of note choices in a more tonal sense. A high entropy value for this feature indicates the use of various notes, which can contribute to a richer harmonic language.

In addition to these features, the path to each sample on the storage is maintained for future processes such as generation and sampling.

4.2. Categorizing Average Pitch Frequency

The smooth integration of various musical elements is crucial in producing a compelling and well-balanced musical piece. Each instrument, vocal element, and percussive sound occupy a specific frequency range, which contributes to the overall sonic texture. When these frequency ranges are not carefully allocated, they can clash, resulting in an unclear or unpleasant sound.

The analysis and processing of musical data are simplified by categorizing the average sample's pitch frequency into distinct frequency ranges. This categorization reduces the complexity of the dataset and allows for a better understanding of the relationship between frequency ranges and the musical elements they represent. By categorizing the data, it is easier to identify the sample's tonal center on the frequency spectrum and recognize the typical instruments within these ranges. This categorization also leads to more homogeneous and musically coherent clusters. Following this step, the `average_pitch_category` feature is obtained. The frequency ranges used for the `average_pitch` categorization are provided in Table 3. For more information on audio frequency ranges and their corresponding musical elements, refer to [42,43].

Table 3. Frequency Ranges and their Corresponding Musical Elements.

Frequency Range	Musical Elements
Sub-bass (16–60 Hz)	The sample is characterized by very low frequencies, typical of instruments like the upright bass, tuba, and bass guitar.
Bass (60–250 Hz)	A value in this range suggests that the sample has a frequency range similar to the normal speaking voice and may be rich in bass elements.
Lower Midrange (250–500 Hz)	This category indicates that the sample's average pitch is within the range of brass instruments and mid-woodwinds, such as the alto saxophone.
Midrange (500 Hz to 2 kHz)	A value in this range captures the higher end of the fundamental frequencies of most musical instruments, including those like the violin and piccolo.
Higher Midrange (2 to 4 kHz)	This range is associated with the harmonics of instruments like the trumpet, which add brightness to the sound.
Presence (4 to 6 kHz)	A value in this range includes harmonics for instruments like the violin and piccolo.
Brilliance (6 to 20 kHz)	This category includes high-pitched sounds and harmonics of percussive elements like cymbals, which add sparkle and airiness to the music.

4.3. Normalizing Data

Data normalization is a process that ensures that each feature in a dataset has a similar range. This process aims to prevent any single feature from dominating the distance calculations performed by machine learning models during training. By normalizing the data, the model can consider the relative contributions of all features effectively.

Moreover, dimensionality reduction algorithms are sensitive to the scale of the data. If features in a dataset have different scales, the algorithm may give more weight to features with larger values, leading to inaccurate results. Normalizing the data to a specific range can reduce the influence of any outliers or extreme values present in the dataset. This can improve the quality of the low-dimensional representation of the data.

In this framework, the extracted feature vectors are scaled using MinMax scaling to fit a range of $(-1, 1)$.

4.4. Dimensionality Reduction

Dimensionality reduction is an important technique in machine learning and data analysis that transforms complex, high-dimensional data into a more manageable and interpretable form. It plays an essential role in addressing several challenges and enhancing the overall effectiveness of data-driven approaches. This technique can mitigate overfitting, improve computational efficiency, reduce noise, enable data visualization and exploration, extract relevant features, optimize data storage and transmission, and enhance model interpretability and explainability.

In the designed framework, the t-SNE algorithm is used to transform the extracted feature vectors into two dimensions to facilitate analysis, visualization, and training processes.

5. Agent Roles and Interactions

The designed framework involves two agents: the perceiving agent and the generative agent. As an entity, these agents are involved in the generation process using the methods described in Section 3.

The perceiving agent has two main roles: learning and evaluating. It learns the underlying musical characteristics based on the feature vectors extracted from the samples in the dataset. It evaluates and provides feedback on the musical characteristics of the generative agent's output. On the other hand, the generative agent carries out the output and employs reinforcement learning to optimize its outcomes. These agents work collaboratively, similar to the collaboration of a composer with a music theorist. The perceiving agent gives the generative agent a better understanding of the environment by encoding the musical samples in a higher level of abstraction. Consequently, the generative agent concentrates more on its action improvement and interaction with the user.

Here, the framework assigns an active role to the user. Therefore, the user is considered an agent who interacts with other agents within the musical environment. The user's role is to evaluate the model's output and provide new inputs to guide the generations. User feedback enables the generative agent to adapt and learn the user's preferences, potentially introduce novelty and creativity, and navigate complex environments such as musical structure. Indeed, the perceiving agent provides objective feedback based on its musical knowledge, and the user offers subjective feedback. The user considers aspects that are difficult to quantify, such as emotional impact, stylistic relevance, and overall musicality. Note that the perceiving agent also observes the changes in the environment, such as the human feedback given to the generative agent. Hence, it can provide input according to the related musical context of the generative agent.

In the following, we delve deeper into the details of the agents and their underlying processes.

5.1. Perceiving Agent

The perceiving agent is designed to understand and interpret the characteristics of musical samples. Its primary function is to learn the patterns and similarities among

musical samples using the extracted feature vectors. To achieve this, the perceiving agent employs a growing hierarchical self-organizing map (GHSOM) to capture the hierarchical and topological structure of data.

As described in Section 3, various forms of the hierarchical structure of data can be obtained using the GHSOM model. These forms are primarily determined by the choice of parameters that control the GHSOM's growth, resulting in lower or deeper hierarchies. The choice of the hierarchy impacts the perceiving agent's understanding of feature vectors. Lower hierarchies with detailed refinements provide a broad view with granular analysis, while deeper hierarchies offer a fine-grained understanding of nested feature vectors. For instance, the agent focuses on specific subclusters in deeper hierarchies, which may correspond to more nuanced musical styles or genres. The optimal choice can be a balance between the breadth and depth of the hierarchical structure. This balance lets the agent capture the sample's broad characteristics and intricate hierarchical relationships. Figure 3 demonstrates lower and deeper GHSOM structure examples.

t-SNE is a useful tool for determining the optimal hierarchical structure parameters in GHSOM. This is achieved by visually inspecting the resulting clusters or patterns in the feature vectors. The hierarchical structure depth can be chosen based on the number of clusters within the t-SNE embedding. The size of the maps is defined based on the data's local density in the lower-dimensional space. It is important to note that t-SNE is used for exploration and interpretation purposes, not as a definitive guide for GHSOM parameter selection. Once the parameters are determined, the GHSOM model is trained on the feature vectors. The quality of the GHSOM map is then manually inspected using its latent space. Figure 1 illustrates the GHSOM training process.

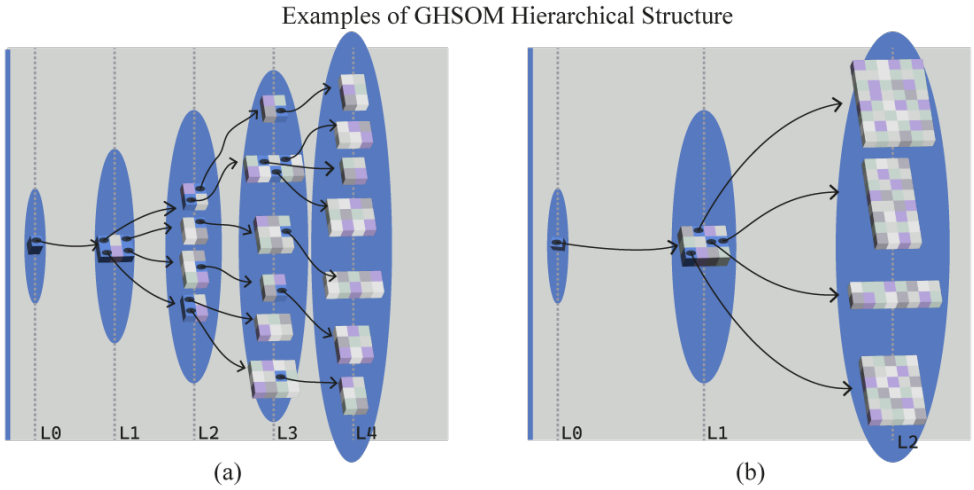


Figure 3. Illustrative comparison of the GHSOM model configurations showcasing (**a**) a hierarchical structure with multiple levels, where each subsequent layer represents increasingly fine-grained clusters of the data, and (**b**) a hierarchical structure with fewer levels, resulting in a broader and more generalized representation of the data across larger maps with less emphasis on depth.

Once an optimal GHSOM model is obtained, the hierarchical structure serves as a knowledge base. In this knowledge base, each neuron represents a cluster of similar data points corresponding to similar samples. The perceiving agent utilizes this knowledge base in two ways: clustering similarity and feature vector analysis.

5.1.1. Clustering Similarity

The agent receives new input samples and identifies which neurons in the GHSOM are activated by each sample. Samples that activate the same or neighboring neurons maintain a degree of similarity. Furthermore, the agent can assess the proximity of the activated neurons. Neurons situated close to each other typically signify that the samples they represent share more features in common. On the other hand, the agent can consider

the hierarchical organization of the clusters to identify samples with broader similarities. For instance, samples that belong to the same higher-level cluster or parent map but are distributed among different subclusters or child maps may exhibit similarities with nuanced distinctions.

5.1.2. Feature Vector Analysis

The agent can compare the feature vectors by calculating their distances and angles. A feature vector associated with a neuron represents the cluster centroid of that neuron. By calculating the distance and angle, the agent can assess the similarity between clusters. A shorter distance or smaller angle suggests that the samples within the clusters share features that are oriented similarly within the feature space. For instance, the Euclidean distance might reveal similar samples in overall musical content, while cosine similarity could identify patterns that share a common structure but differ in scale.

These methods can provide a basis for the perceiving agent's analysis, comparison, and exploration. By combining these methods, the perceiving agent can effectively determine the similarity of patterns based on the GHSOM clusters and feature vectors. Moreover, the results of the analysis can help the agent evaluate and comprehend the structure and coherence of the sequences created by the generative agent. Consequently, the agent can provide feedback to the generative agent and suggest further generative processes to generate music that is varied and thematically consistent.

5.2. Generative Agent

The generative agent is designed to learn and actively engage in the creative process of constructing a melodic sequence that is both structurally sound and aesthetically pleasing. The agent's objective is to understand the musical structure and develop new melodic possibilities by sequencing different samples. At each step of the music generation process, the agent selects the best-matching neuron in the GHSOM hierarchical structure. The selected neuron represents a cluster of samples suitable for the corresponding time step in the musical sequence. To guide the decision-making process, the generative agent uses a policy that balances the likelihood of a neuron's occurrence based on its learned knowledge and the potential for discovering new, rewarding clusters. This represents reinforcement learning, where the agent's goal is to maximize a reward reflecting the quality of the generated music.

Using the DQN algorithm, the agent learns the optimal policies through Q-learning, which seeks to optimize the cumulative reward after each iteration. Policies are chosen based on a minibatch of random samples from the main network. The main network generates actions, while the target network generates a consistent target for calculating the loss associated with the chosen action. At each time step, the agent generates an action, a_t, following a policy, π, and based on the current state, s_t. The environment then generates a reward, r_{t+1}, and a new state, s_{t+1}. This process continues until a satisfactory result is achieved.

The goal of training the main network is to approximate the optimal action-value function $Q(st, at)$. It denotes the expected total reward for taking an action, a_t, in a given state, s_t, and following the optimal policy afterward. The network receives the current state as input and outputs a vector of Q-values for each potential action in that state. The network's training involves a modified Q-learning algorithm that reduces the discrepancy between the predicted and actual Q-values derived from the Bellman equation. The target network is not updated during the Q-learning update step but is periodically updated to match the weights of the main network. This helps to stabilize the training process by preventing the Q-values from oscillating or diverging during training.

During the training process, the agent predicts a sequence of tokens. Each token represents a neuron within the GHSOM's hierarchical structure. The agent is trained from scratch to learn the musical structure and capture the transitions between the samples. Initially, the agent lacks related musical knowledge. During training, the agent is exposed

to the input data and iteratively improves its understanding of how these musical elements combine to create a coherent piece. The agent's learning is driven by feedback mechanisms, and over time, it refines its internal representations of music. This way, the agent learns to generate new sequences considering the desired musical characteristics.

Furthermore, NoisyNet [44] is used to promote exploration within the action space. NoisyNet tackles the exploration-exploitation dilemma by introducing noise into the network's weights, which are responsible for estimating Q-values or policies. The noise is integrated to maintain the network's differentiability. This method allows for continuous training through gradient descent. NoisyNet's primary benefit is its systematic approach to managing exploration and exploitation. It eliminates the need to add extra exploration noise to the actions.

The designed framework uses the LSTM model for the DQN network. The LSTM model captures the temporal dependencies to predict the next token in the sequence. The model architecture includes an LSTM layer with 128 units with a 0.2 drop-out rate, followed by a densely connected layer to carry out the predictions. The exponential linear unit (ELU) is used as an activation function for the LSTM layer, softmax for the dense layer, and adaptive moment estimation (Adam) as an optimizer to minimize the cross-entropy function.

5.3. Human Agent

In the designed framework, the user is considered an agent and has the role of evaluator and collaborator. The primary responsibility involves observing the outputs generated by the model and providing feedback accordingly. The user can also collaborate with the agents by providing inputs after each iteration. Therefore, it can effectively guide the generative agent's future outputs to align better with the desired musical preferences.

Such interactions contribute to the agent's learning, in which the agent incorporates the feedback into its generative process. This interaction allows the user to influence the agent's creative direction, potentially leading to innovative results. The user's feedback is precious in complex creative environments, such as music composition, where the agent requires a nuanced understanding that it may not possess inherently. The feedback may encompass the music's emotional impact, which can vary from user to user, depending on personal perception of music. Moreover, the user can also consider stylistic relevance so that the output is technically sound and appropriate for the intended purpose.

In addition to providing feedback, the user can guide the sequence generation by passing inputs. The agents would incorporate the inputs to optimize the outputs and refine the decision-making process. The inputs can come in two forms: complete songs or samples. The complete songs serve as material for the agents to understand and learn the desired musical structure. The perceiving agent creates vectors of best-matching neurons for each segment within the given songs. These vectors serve as a ground truth for the generative agent to guide the agent toward understanding the preferred musical structure.

On the other hand, the second form of input, the samples, is more related to the perceiving agent. This input signals the agent to look for alternative directions within the GHSOM hierarchical structure. We will explain the complete procedure in Section 6.

5.4. Definition of Reward Function

The designed framework combines and incorporates objective and subjective methods as a reward function for the RL agent. This method helps the generative agent to optimize its actions and produce music that aligns with human aesthetic preferences and adheres to musical principles. This method needs to consider different aspects of music composition, including melodic structure, hierarchical organization, transition between patterns, repetition, and variation. Indeed, these metrics and the human evaluation can form a composite reward function. This reward function facilitates the agent's understanding of its actions to find optimal strategies and capture users' preferences. Here, the definition of the reward policy is based on three criteria:

- Ground truth reward based on the target structure vectors denoted as r_{ground_truth}.
- Structure reward using GHSOM hierarchical structure denoted as $r_{structure}$.
- Human feedback reward denoted as $r_{human\_feedback}$.

5.4.1. Ground Truth Reward

The purpose of the ground truth reward is to encourage the generative agent to produce sequences similar to the ideal sequence of neurons' tokens. This feedback mechanism enables the agent to learn structural elements that characterize the given songs by the user. Indeed, it ensures that the agent maintains a level of fidelity to the desired musical structure.

The similarity between the generated sequence and the ground truth is quantified using the negative log-likelihood (NLL) loss. NLL is a statistical measure that calculates the performance of a probabilistic model. Here, it assesses how well the model predicts the sequence of neurons' tokens. The NLL loss function evaluates the probability the model assigns to the actual, observed sequence of tokens (the ground truth). The NLL loss will be low if the model gives a high likelihood to the correct sequence, meaning a higher ground truth reward. Conversely, The NLL loss increases if the model assigns a low probability, resulting in a lower reward. Therefore, the objective is to decrease the loss as the agent continues learning.

5.4.2. Structure Reward

The structure reward evaluates the transitions or changes between the samples within the generated sequence. The main objective is to prevent the model from abrupt changes that are relatively quick or completely irrelevant. This feedback mechanism uses the key aspects of the perceiving agent's analysis: clustering similarity and feature vector analysis. The agent is responsible for evaluating the structural coherence of generated musical sequences and providing feedback to the generative agent accordingly.

The agent utilizes clustering similarity to identify similarity between clusters of predicted neurons. Similarly, the agent analyzes the feature vectors associated with each neuron using feature vector analysis. These feature vectors represent the centroid of the clusters that the neurons belong to. This way, the agent assesses cluster similarity by calculating the distance between feature vectors. This feedback mechanism encourages smoother transitions between similar clusters, resulting in more musically coherent sequences and contextually related samples. The following shows how the perceiving agent's analysis results are incorporated into the structure reward.

The Euclidean distance between the activated neurons' weight vectors is calculated for clustering similarity analysis. Hence, the clustering reward is defined as follows:

$$r_{clustering} = \exp(-\alpha \cdot d(n_t, n_{t-1})) \tag{2}$$

where $d(n_t, n_{t-1})$ is the Euclidean distance between the weight vectors of the neurons activated by consecutive samples at times t and $t-1$. α is a scaling factor that adjusts the sensitivity of the reward to the distance. The exponential function is used to ensure a smooth decrease in reward as the distance increases.

Similarly, using the Euclidean distance, the reward based on the feature vector analysis is defined as follows:

$$r_{feature} = \exp(-\gamma \cdot d(f_t, f_{t-1})) \tag{3}$$

where γ is a scaling factor that determines the sensitivity of the reward to the distance. The overall structure reward is a composite of these aspects:

$$r_{structure} = \beta_1 \cdot r_{clustering} + \beta_2 \cdot r_{feature} \tag{4}$$

The weights β_1 and β_2 balance the contribution of each aspect to the total reward. These weights can be tuned based on the importance of each aspect in the context of the musical generation task. To prevent any single component from dominating the reward

function and to keep the reward values manageable, each component is normalized in the following way:

$$r_{structure} = \frac{\beta_1 \cdot r_{clustering}}{Z_1} + \frac{\beta_2 \cdot r_{feature}}{Z_2} \tag{5}$$

where Z_1 and Z_2 are normalization factors for each reward component, which can be defined as the maximum possible rewards for each component.

5.4.3. Human Feedback Reward

The human feedback reward integrates human judgment into the evaluation framework. This aspect of the reward system is essential because it allows the model to incorporate subjective evaluations that are not easily captured by objective measures such as ground truth or structure rewards. Human feedback is gathered by presenting the generated sequences to human evaluators as system users. The users then provide assessments based on their experience, expertise, and taste. This feedback can highlight successful outcomes that might be infrequent or difficult for the agent to discover independently. Moreover, it can encourage the agent to explore areas of the solution space that the initial reward functions might not cover. Here, the user explicitly evaluates the generation with +1 as a positive reward and -1 as a negative reward.

Finally, the definition of the final reward r_t for the action to be taken at time t is as follows:

$$r_t = \lambda * r_t^{ground\_truth} + \mu * r_t^{structure} + \nu * r_t^{human\_feedback} \tag{6}$$

where λ, μ, and ν are the weight factors that control the impact of each component of the reward function on the agent's learning process. Note that the weight factors are experimental values and can be defined based on the specific goals or stages of development.

5.5. Agent Communication

Communication is an essential aspect of the multi-agent systems (MAS). It enables agents to exchange information and coordinate their actions. The designed framework utilizes the DQN replay buffer. The replay buffer is a fundamental component of the DQN algorithm [45]. It serves as a repository for storing the generative agent's experiences as a tuple, including the current state, the action taken, the reward received, and the subsequent state encountered. The replay buffer also plays an essential role in stabilizing and enhancing the learning trajectory of the agent. It enables the agent to recall and learn from past experiences and prevents short-term memory pitfalls and the volatility of online learning [46].

The DQN replay buffer can be transformed into a communication repository accessible to both the generative and perceiving agents by extending its functionality. After each iteration, the generative agent stores its experiences in the replay buffer, which the perceiving agent then uses to enhance its actions and comprehension of the environment. This communication framework encourages collaborative learning among the agents, allowing them to coordinate their actions with a stronger sense of context and anticipation to operate within the musical environment. Figure 4 illustrates the interaction of agents using the replay buffer.

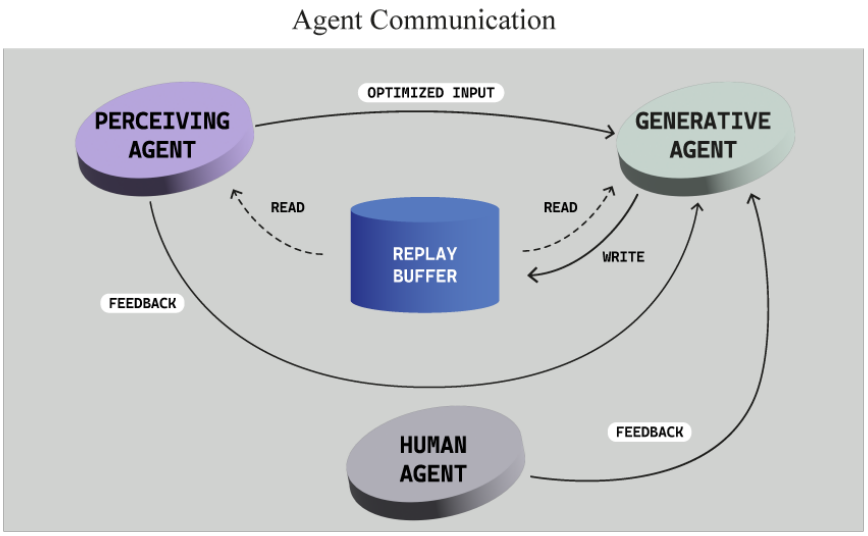


Figure 4. This figure illustrates the communication process within the framework utilizing the DQN replay buffer. The generative agent saves an experience tuple, which includes the state, action, received reward, and subsequent state after receiving feedback from both the perceiving agent and the user. The perceiving agent then reads the replay buffer to analyze current and past experiences and enhance its input for future iterations.

6. Generation Process

The generation process of the designed framework is where the agents evolve and their collaborative efforts come to fruition. It is an evolutionary procedure that hinges on the dynamic interplay between the perceiving agent's analytical capabilities and the generative agent's creative functions. During this stage, the agents engage in a continuous loop of action, observation, and refinement, which allows them to adapt and improve over time. Figure 1 depicts the generation process.

The process starts with the perceiving agent using the GHSOM to determine the best-matching neuron or fitting cluster within the hierarchical structure of GHSOM for a given input. Once the cluster is identified, the agent prepares an input vector for the generative agent. This vector contains a sequence of neurons represented as tokens, and its size varies depending on the input type provided by the user.

For instance, if the input is a sample, the vector size is one. However, if the input is a complete song, the framework first performs a segmentation algorithm using the similarity matrix described in [5]. This algorithm identifies each segment or pattern and annotates the given song. The perceiving agent then receives the annotated song and determines the best-matching neuron for each segment. The result is a vector containing the tokens for the corresponding segments. The complete songs can be an example from the Clean MIDI subset of the Lakh MIDI dataset (https://colinraffel.com/projects/lmd/, accessed on 12 March 2024) or any other sources. It is worth noting that the input vectors obtained from complete songs are used as ground truth in Section 5.4. Figure 5 illustrates the perceiving agent during the generation process. The generation process of the designed framework is where the agents evolve and their collaborative efforts come to fruition. It is an evolutionary procedure that hinges on the dynamic interplay between the perceiving agent's analytical capabilities and the generative agent's creative functions. During this stage, the agents engage in a continuous loop of action, observation, and refinement, which allows them to adapt and improve over time. Figure 1 depicts the generation process.

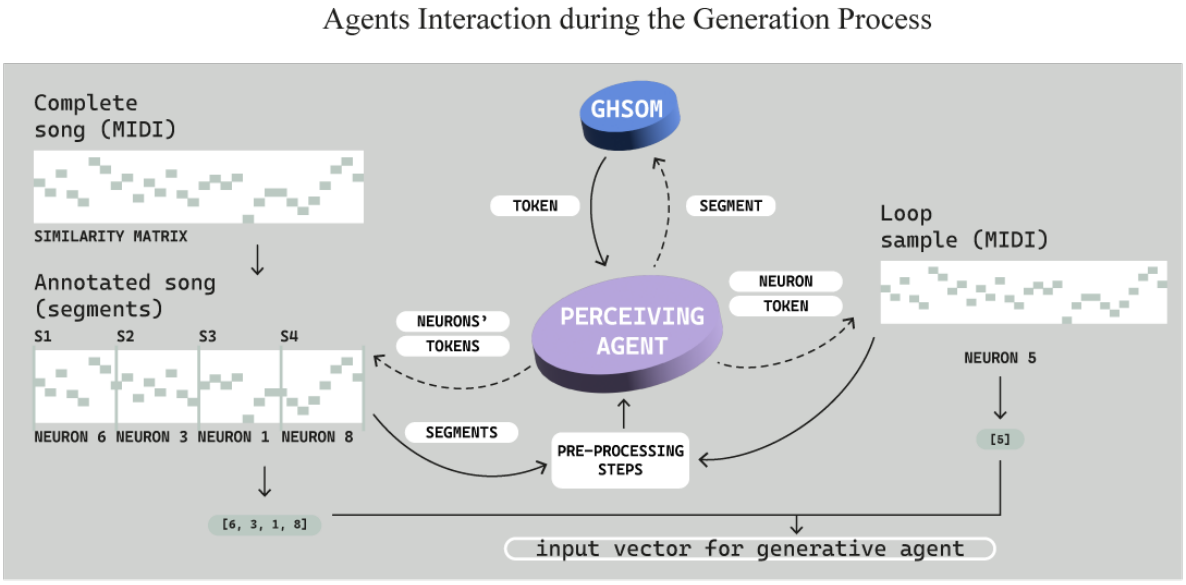


Figure 5. This figure illustrates how the perceiving agent handles different input types during generation. The process starts with an input, a single sample resulting in a vector of size one or a complete song segmented using a similarity matrix algorithm. After this, the perceiving agent assigns the best-matching neuron token to each annotated segment, thus creating an input vector. These vectors serve as input for the generative agent and the ground truth when defining rewards. The complete process can be viewed in Figure 1.

The generative agent uses the vectors prepared by the perceiving agent as a basis to learn the musical structure and improve its actions. It creates a sequence of segments, ensuring the transitions between segments are coherent and musically pleasing. The perceiving agent and human agent observe the environment and the generative agent's actions. They evaluate the generated sequence using the methods described in Section 5.4. Using Equation (6), the generative agent receives feedback to refine its action. Using the feedback, it aims to capture the user's preferences and the nuances of music composition. Figure 6 illustrates the interactions between the agents during the generation process.

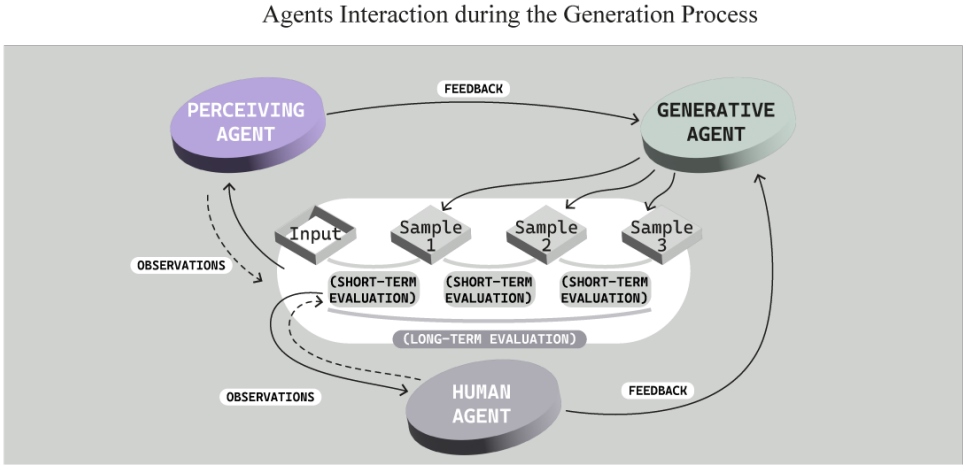


Figure 6. This figure depicts how agents interact with each other during the music generation process. The generative agent generates a sequence of samples to produce smooth and musically pleasing transitions. The perceiving agent and human agent are responsible for monitoring the environment and the output generated by the generative agent. They assess the music sequence using different criteria and provide feedback to the generative agent. The feedback is intended to guide the generative agent to align better with user preferences and the intricacies of musical composition.

The perceiving agent uses the communication method to assess if its input is optimal. It performs this by analyzing the experiences in a replay buffer. It clusters similar experiences to identify patterns for better decision-making and reinforce optimal actions. The agent extracts reward and action pairs and applies a clustering algorithm to group experiences. It can perform clustering based on the experiences that lead to similar rewards or those that involve similar actions. Therefore, the perceiving agent can identify suitable samples by examining the clusters associating specific actions with successful outcomes. For instance, a consistent cluster of experiences leading to high rewards may indicate an effective strategy.

6.1. Iterative Refinement

The user has the option to provide either a complete song or a sample with each iteration. The complete songs enable the agents to refine their understanding of the preferred structure and composition, making this input more focused on the generative agent. On the other hand, the perceiving agent uses the samples to identify clusters or neurons within the GHSOM that are closer to the user's musical preferences. If the user does not provide any input, the perceiving agent uses the previous output as new input. This way, the agents assume that the previous generation aligns with the user's expectations, and the generation process continues until the user signals a new input or termination of the process.

Furthermore, this iterative process allows the refinement of the clusters within the GHOSM model with every round of feedback and generation. The framework discovers and updates the clusters as the user provides more input. This continuous learning process makes the system proficient at understanding the musical characteristics and meeting the user's musical expectations.

6.2. Content Generation

The framework's output is a sequence of tokens, each representing a specific neuron or cluster within the GHSOM hierarchical structure. Each cluster contains a collection of musical samples that share similar features. When the framework generates content, it predicts a token corresponding to one of these clusters. The token acts as a reference point that indicates a specific location where the musical samples align with the user's preferences. The token selection is not the end of the generative process but rather a step toward the music composition.

Once a token is predicted, the framework randomly samples from the corresponding cluster within the GHSOM model to create the musical output. This random sampling can introduce variation and creativity within the constraints of the user's preferences. It also allows for exploring new combinations and sequences of samples that may not have been selected before but are still coherent within the cluster's characteristics.

7. Interface

An interface is an intermediary that enables users to interact with and control such frameworks effectively. In the proposed framework, the human agent listens to the music as it is generated and provides feedback. This feedback translates into reward signals for the agents. The ability of a framework to learn and adapt to user preferences largely depends on the quality and consistency of the feedback it receives. Hence, it is crucial to have an interface that encourages sustained engagement, as it is essential for gathering feedback over time.

PureData (PD) (https://puredata.info/, accessed on 12 March 2024) is a programming language that uses a visual interface to create interactive computer music and multimedia content. PD employs functional blocks called "objects" that can be manipulated and connected to process and generate audio, video, and visual data. The real-time processing capabilities of PD are crucial for creating a responsive and engaging feedback loop. Moreover, it is particularly advantageous for MAS interaction due to its capability to operate in real time, which is essential for dynamic and responsive environments. With PD, users

can hear the immediate impact of their feedback on the music, which makes the experience more satisfying and helps them understand the relationship between their input and the framework's output. PD's visual programming paradigm allows users to intuitively construct and modify the framework's behavior by connecting objects to represent the data flow.

Furthermore, PD can connect the proposed framework and different digital audio workstations (DAWs). It allows the transmission and reception of MIDI data, which enables the MIDI output created by the framework to be sent through PD and straight into the preferred DAW. This, in turn, allows for additional processing, mixing, and mastering using the advanced features of the DAW.

PD (Pure Data) supports various communication protocols, including Open Sound Control (OSC). OSC is a protocol for communication between computers, sound synthesizers, and other multimedia devices, especially for live performances and production. OSC facilitates efficient communication between the PD interface and the underlying functionalities. It can handle complex data structures and support high-speed messaging, making it an ideal choice for interactive systems. Using OSC, the interface can transmit various data types, from simple messages to control structures, enabling a robust and flexible interaction between the human agent and the MAS.

Moreover, PD offers a range of internal objects and external plugins that can be used to render MIDI files. This library of objects provides users with different options for audio synthesis and processing, which can be used to render the generated music within the PD environment. This feature allows users to listen to and modify the framework's output per their requirements. This is particularly useful for users who do not have access to a full-featured DAW, as it offers a standalone solution to preview and manipulate the generated music. Moreover, they can save each generation process (experiment) with its corresponding model parameters as a session, allowing them to resume their work for further experimentation or development.

8. Discussion

Our framework addresses RQ1 by prioritizing user experience. It is designed to accommodate users with varying musical backgrounds by emphasizing personalized and adaptive interactions. The framework maintains a balance between autonomy and user involvement to encourage creative exploration and remain responsive to user feedback. Through a collaborative process, the user actively participates in the music generation and development process by providing feedback and engaging in iterative interactions. As a result, the framework can better understand and adapt to the user's musical tastes and creative expression. This implies a trade-off where user involvement is prioritized to enhance the music creation experience.

8.1. Musical Pattern Recognition and Hierarchical Representation

The framework aims to balance structured musical repetition and creative variations, capturing short and long-term musical structures. Instead of the notes, the framework incorporates samples, each representing a melodic pattern. Using a dataset of samples, a set of musical features is extracted from the MIDI samples, capturing the melodic and rhythmic dimensions of music. This quantification forms a knowledge base for the agents to draw comparisons, identify patterns, and potentially uncover the underlying relationships between the musical samples. To enhance the agent's perception and understanding of music, the GHSOM hierarchical structure is employed. This structure offers a dynamic representation of samples that can be adjusted or updated to meet specific objectives, thereby facilitating a deeper understanding of music.

Regarding the RQ2 and RQ3, the perceiving agent uses the GHSOM model to discern relationships between various rhythmic and melodic elements and match clusters of comparable musical samples. The GHSOM model can scale and adapt its structure dynamically, making it ideal for representing music features at different levels of abstraction.

This adaptability is especially useful because it allows for a wide range of musical data without requiring the definition of cluster dimensions. Moreover, it maintains topological relationships, ensuring that similar features are mapped close together. This topological accuracy and the model's hierarchical nature allow contextual analysis of the musical samples. This demonstrates the framework's ability to organize and adapt musical samples and recognize and use them in generating and evaluating musical sequences.

8.2. Maintaining Consistency in Musical Styles and Structures

Section 5 described two approaches of interactive exploration for the perceiving agent, including clustering similarity and feature vector analysis. These methods offer advantages for the perceiving agent in evaluating generated sequences by the generative agent. Regarding the RQ4, the perceiving agent can quantitatively compare generated sequences with a set of reference compositions or with each other. Metrics such as cosine and Euclidean distances help evaluate the similarity and diversity among the generated sequences. These metrics ensure that the variations produced by the generative agent maintain a desired level of consistency with certain musical styles or structures. Grouping the generated sequences into clusters based on their similarities further assists in identifying common themes or outlier pieces that deviate from the expected patterns. This structured approach facilitates the generative agent to refine its generative processes and provides a clear framework for feedback. The agent can iteratively improve the generated music's overall quality and creative value through this framework.

8.3. Effectiveness of User Involvement

The adaptive nature of the framework allows the agents to adjust their behavior based on the user's feedback and input. Concerning the RQ5, the framework includes user qualitative evaluations as a metric to guide the process of generating music and ensure consistency in the output. Although the system does not solely depend on traditional quantitative metrics, user feedback is a dynamic and adaptable metric that reflects the desired musical styles or structures. The agents can create complete musical pieces that reflect the user's creative intent. This indicates a symbiotic relationship between the system and the user. During the interaction, the user provides qualitative assessments, which are valuable in guiding the music's structure, style, and emotional content. The user can also provide specific examples of the desired music. The agents analyze these examples to discern the underlying patterns and stylistic features defining user preferences.

Additionally, the user engages with the perceiving agent's objective analysis through this interaction. The perceiving agent analyzes the samples based on their structural coherence and technical aspects. This objective feedback is balanced by the user's subjective impressions, reflected in the input loop samples. Therefore, the framework forms a response based on this balance of objective and subjective feedback. This ensures that the framework can adapt to both the technical aspects of music and the emotional content desired by the user and remain sensitive to the evolving musical context and user preferences, providing a nuanced response to RQ3.

Indeed, it shapes the subsequent actions taken in the music creation process. Similarly, the user's feedback serves as a source of environmental context for the perceiving agent. The agent can better understand the human role in the creative process by observing the user's reactions and feedback through the replay buffer. This understanding helps the perceiving agent adjust its input to the generative agent. It ensures that the framework remains sensitive to the evolving musical context influenced by human interaction.

8.4. Incorporating User Feedback in Learning

The proposed framework allows the agents to learn and fine-tune their behavior based on the feedback they receive from the user. The user's active participation in providing continuous feedback facilitates the refinement of the agents' understanding of musical structure and preferences. To answer RQ6, the framework allows users to provide feedback

at various stages. Users can provide a complete song or a sample, which the agents use to refine their understanding and adjust their output. This iterative process ensures the framework adapts to the user's evolving musical preferences. Additionally, the perceiving agent uses the previous output as new input, allowing the generation process to continue as long as it aligns with the user's expectations. The continuous adaptation and refinement method through user feedback is central to the framework's design.

However, several challenges involve incorporating human feedback mechanisms in such an approach. One major challenge is to ensure that the evaluations are consistent and reliable. Additionally, collecting human feedback can take time and effort to obtain at a large scale. It may not always provide the exact reasons for success or failure. Moreover, relying solely on human feedback can lead to conservative strategies where the agent might only exploit known favorable actions to maximize the reward. All of these challenges are directly related to RQ4 and RQ5.

To tackle such challenges, a composite reward function with additional objective reward components is used. These objective measures help to stabilize the learning process by providing consistent signals that counteract the variability of human feedback. This leads to a more reliable and predictable training outcome. They also offer more frequent updates, which guide the agent's learning in between the less frequent instances of human feedback. This helps the agent maintain a steady learning trajectory and prevents long periods without meaningful reward signals. Additionally, these objective reward components encourage exploration by rewarding novel or diverse behaviors that may not receive immediate positive human feedback but are valuable for discovering new strategies or solutions. Furthermore, they can establish a baseline level of performance that ensures the agent meets certain minimum standards or criteria. This approach answers RQ4 by utilizing objective reward components to enhance the learning process and encourage exploration.

To control the influence of these components, weighting factors described in Section 5.4 are included. The weighting factors are important as they help the agent learn from objective measures and subjective human input. By using weight factors to control the influence of each reward component, the learning process can be customized to suit specific goals or developmental stages. This flexibility enables the agent to produce outputs that range from strictly following the objective components to highly personalized based on human feedback. In response to RQ5, users can experiment with different weight configurations to explore the effects on the agent's learning and the quality of the generated content. This adaptability is particularly useful in music composition domains with diverse requirements, where the desired balance between creativity, theory, and personal taste can vary greatly depending on the context.

8.5. Interaction among Agents

Our framework is designed to maintain coherence without excessive central control by providing autonomy to each agent. It features a collaborative environment where perceiving and generative agents interact using a shared DQN replay buffer, resulting in a more flexible and creative approach toward music generation. In relation to RQ7, the replay buffer serves as a communication repository, which allows the perceiving agent to access the generative agent's experiences and provide feedback that informs future actions. The replay buffer serves as an effective communication tool, facilitating knowledge transfer and coordination between agents. It enables sharing of insights, learning from each other's actions, and collectively developing a more nuanced understanding of musical experiences. This approach offers several advantages:

- It enables the collection of a comprehensive dataset that mirrors the user's behavior over time, allowing for the detection of subtle patterns and preferences.
- This rich data collection can be further utilized for pattern recognition, where perceiving agents analyze the data to identify trends that inform the generative agents about the most effective ways to engage with the user.

- This method of learning user preferences is non-invasive, as it does not require explicit feedback from the user, thus enhancing the user experience by adapting to their needs unobtrusively.
- Communication efficiency is another advantage, as the replay buffer provides a less resource-intensive method for agents to share information compared to real-time communication.

This collaborative and communicative approach within the multi-agent system is key to the framework's ability to generate innovative music that reflects user preferences. Furthermore, it allows the perceiving agent to gather information about user preferences indirectly and continuously throughout the generation process.

8.6. Framework Expansion

The MAS architecture allows for modularity and the use of various computational methods. It promotes distributed problem solving, which involves breaking down complex issues into smaller, more 9396 manageable sub-problems that can be solved by multiple agents working together. The system's capabilities can be expanded by introducing several RL agents with diverse behaviors, each assigned to a specific task [5]. However, adding more RL agents to the system can create challenges in effectively coordinating them as system complexity increases. The hierarchical organization of agents can be used to simplify the complexity of managing multiple agents and their interactions. In this architecture, specialized agents work under a higher-level coordinating agent.

For instance, the perceiving agent may have specialized agents working on tasks such as action evaluation, policy refinement, temporal analysis, contextual adaptation, and anomaly detection. Each agent can analyze specific aspects of the generative agent's performance using the replay buffer.

- The action evaluation agent can examine the outcomes of actions taken within their designated clusters. By identifying which actions yield higher rewards, this agent may determine the effectiveness of various actions and signal which behaviors are most beneficial for the framework.
- Once successful actions are identified, the policy refinement agent can adjust decision-making policies to favor these actions. This may involve algorithmic modifications that increase the selection probability of high-reward actions, thus optimizing the generative agent's behavior patterns.
- Similarly, the temporal analysis agent can focus on actions' temporal aspects and consequences. It can cluster experiences by the outcome and the sequence in which events occur. This may allow for a deeper understanding of the long-term effects of certain behaviors. Such insights can help to develop strategies that optimize rewards over extended periods.
- The contextual adaptation agent can be proficient at recognizing the nuances of different environmental states and adjusting their actions accordingly. It can identify which clusters of actions are most successful in particular musical contexts and tailor their policies to be context-sensitive.
- The anomaly detection agent determines and analyzes deviations from typical patterns. This agent can prepare the framework for unexpected events and can propose policy adjustments to handle rare but impactful situations better.

As we explore the possibility of adding specialized agents to the framework, it is crucial to have an effective communication method in place. This is particularly relevant to RQ7. The shared replay buffer, which was initially meant to serve as a memory tool for the generative agent, is now being repurposed as a communication channel.

However, the framework's interactions using replay buffer communication may face scalability, conflict management, and agent negotiation limitations. To tackle this problem, decentralized learning can be used. This approach involves agents learning from the actions of other agents within their environment. By doing so, each agent can develop a range of

experiences that enable them to operate effectively. However, while this method helps build robustness and adaptability, it may not be sufficient in ensuring that agents can achieve high levels of coordination and share knowledge effectively. Therefore, decentralized learning can be integrated with a shared replay buffer to address this. The shared replay buffer enhances coordination among agents by providing a joint knowledge base. Furthermore, the agents can effectively align their decision-making processes with human preferences and intentions.

9. Conclusions

In this study, we proposed a framework design that combines multi-agent systems and reinforcement learning algorithms. This framework utilizes a user-centric approach to address the structural and creative aspects of symbolic music generation. It aims to balance the autonomy of individual agents and the overall coherence of the music generated. The main goal of the musical agents within the framework is to develop their skills in creating structured music by sequencing samples. Each agent is assigned a specific task in the composition process. The perceiving agent serves as the music theorist responsible for perceiving and understanding the musical elements, while the generative agent is the composer responsible for generating new musical sequences.

One notable aspect of this framework is its user-centric approach. This means that human users play an active role in the learning process of the agents. Users can guide the agents to produce compositions specific to their tastes and requirements by specifying their musical preferences. This approach increases the agents' flexibility and ensures that the generated music resonates with human listeners on a personal level. Indeed, the framework aims to balance structured musical repetition and creative variations, capturing short and long-term musical structures.

Another aspect is the perceiving agent's ability to internalize musical knowledge by analyzing loop samples using the growing hierarchical self-organizing map (GHSOM) structure. Similarly, another aspect is the composer agent's capacity to generate novel sequences by learning the smooth transition between the loops. This indicates introducing musical variations to maintain the audience's interest and determining the ideal repetition count to strengthen the musical narrative.

Future work will focus on the practical implementation of the framework and its empirical evaluation. We plan to explore the framework's potential as a music creation tool and assess how it aids users' creative process and influences their workflow and output. Such a study also facilitates the improvement of the learning framework for various musical contexts.

Author Contributions: Conceptualization, S.D. and B.A.B.; Data curation, S.D.; Formal analysis, S.D. and B.A.B.; Funding acquisition, B.A.B.; Investigation, S.D. and B.A.B.; Methodology, S.D. and B.A.B.; Project administration, S.D. and B.A.B.; Resources, S.D. and B.A.B.; Supervision, B.A.B.; Validation, S.D. and B.A.B.; Visualization, S.D.; Writing—original draft, S.D.; Writing—review and editing, S.D. and B.A.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was performed as part of UiT The Arctic University of Norway Ph.D. scholarship program funded by Norges Forskningsråd: 107A.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Ji, S.; Yang, X.; Luo, J. A Survey on Deep Learning for Symbolic Music Generation: Representations, Algorithms, Evaluations, and Challenges. *ACM Comput. Surv.* **2023**, *56*, 7. [CrossRef]
2. Dadman, S.; Bremdal, B.A.; Bang, B.; Dalmo, R. Toward Interactive Music Generation: A Position Paper. *IEEE Access* **2022**, *10*, 125679–125695. [CrossRef]
3. Jaques, N.; Gu, S.; Turner, R.E.; Eck, D. Tuning Recurrent Neural Networks with Reinforcement Learning. 2017. Available online: https://openreview.net/forum?id=Syyv2e-Kx (accessed on 12 March 2024).

4. Roberts, A.; Engel, J.; Raffel, C.; Hawthorne, C.; Eck, D. A hierarchical latent vector model for learning long-term structure in music. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; PMLR: Birmingham, UK, 2018; pp. 4364–4373.
5. Dadman, S.; Bremdal, B.A. Multi-agent Reinforcement Learning for Structured Symbolic Music Generation. In Proceedings of the International Conference on Practical Applications of Agents and Multi-Agent Systems, Guimaraes, Portugal, 12–14 July 2023; Springer: Berlin/Heidelberg, Germany, 2023; pp. 52–63.
6. Gould, E. *Behind Bars: The Definitive Guide to Music Notation*; Faber Music Ltd.: London, UK, 2016.
7. Holland, S.; Mudd, T.; Wilkie-McKenna, K.; McPherson, A.; Wanderley, M.M. *New Directions in Music and Human-Computer Interaction*; Springer: Berlin/Heidelberg, Germany, 2019.
8. Wu, S.L.; Yang, Y.H. MuseMorphose: Full-song and fine-grained music style transfer with one transformer VAE. *arXiv* **2021**, arXiv:2105.04090.
9. Pesce, E.; Montana, G. Improving coordination in small-scale multi-agent deep reinforcement learning through memory-driven communication. *Mach. Learn.* **2020**, *109*, 1727–1747. [CrossRef]
10. Huang, C.Z.A.; Vaswani, A.; Uszkoreit, J.; Shazeer, N.; Simon, I.; Hawthorne, C.; Dai, A.M.; Hoffman, M.D.; Dinculescu, M.; Eck, D. Music transformer. *arXiv* **2018**, arXiv:1809.04281.
11. Dong, H.W.; Hsiao, W.Y.; Yang, L.C.; Yang, Y.H. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
12. Briot, J.P.; Pachet, F. Deep learning for music generation: Challenges and directions. *Neural Comput. Appl.* **2020**, *32*, 981–993. [CrossRef]
13. Yu, B.; Lu, P.; Wang, R.; Hu, W.; Tan, X.; Ye, W.; Zhang, S.; Qin, T.; Liu, T.Y. Museformer: Transformer with fine-and coarse-grained attention for music generation. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 1376–1388.
14. Lv, A.; Tan, X.; Lu, P.; Ye, W.; Zhang, S.; Bian, J.; Yan, R. GETMusic: Generating Any Music Tracks with a Unified Representation and Diffusion Framework. *arXiv* **2023**, arXiv:2305.10841.
15. Lu, P.; Tan, X.; Yu, B.; Qin, T.; Zhao, S.; Liu, T.Y. MeloForm: Generating melody with musical form based on expert systems and neural networks. *arXiv* **2022**, arXiv:2208.14345.
16. Liu, H.; Xie, X.; Ruzi, R.; Wang, L.; Yan, N. RE-RLTuner: A topic-based music generation method. In Proceedings of the 2021 IEEE International Conference on Real-Time Computing and Robotics (RCAR), Xining, China, 15–19 July 2021; IEEE: New York, NY, USA, 2021; pp. 1139–1142.
17. Kumar, H.; Ravindran, B. Polyphonic music composition with LSTM neural networks and reinforcement learning. *arXiv* **2019**, arXiv:1902.01973.
18. Jiang, N.; Jin, S.; Duan, Z.; Zhang, C. Rl-duet: Online music accompaniment generation using deep reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 710–718.
19. Ji, S.; Yang, X.; Luo, J.; Li, J. RL-Chord: CLSTM-Based Melody Harmonization Using Deep Reinforcement Learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**. [CrossRef]
20. Chen, S.; Zhong, Y.; Du, R. Automatic composition of Guzheng (Chinese Zither) music using long short-term memory network (LSTM) and reinforcement learning (RL). *Sci. Rep.* **2022**, *12*, 15829. [CrossRef]
21. Cideron, G.; Girgin, S.; Verzetti, M.; Vincent, D.; Kastelic, M.; Borsos, Z.; McWilliams, B.; Ungureanu, V.; Bachem, O.; Pietquin, O.; et al. MusicRL: Aligning Music Generation to Human Preferences. *arXiv* **2024**, arXiv:2402.04229v1.
22. Mo, F.; Ji, X.; Qian, H.; Xu, Y. A User-customized Automatic Music Composition System. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; IEEE: New York, NY, USA, 2022; pp. 640–645.
23. Dai, S.; Ma, X.; Wang, Y.; Dannenberg, R.B. Personalized Popular Music Generation Using Imitation and Structure. *J. New Music Res.* **2022**, *51*, 69–85. [CrossRef]
24. Mysliwiec, D. AI-Composed Music for User Preference Using Reinforcement Learning. Bachelor's Thesis, University of Twente, Enschede, The Netherlands, 2023.
25. Ma, X.; Wang, Y.; Wang, Y. Content based User Preference Modeling in Music Generation. In Proceedings of the 30th ACM International Conference on Multimedia, Lisboa, Portugal, 10–14 October 2022; pp. 2473–2482.
26. Young, M. NN music: Improvising with a 'living'computer. In Proceedings of the International Symposium on Computer Music Modeling and Retrieval, Copenhagen, Denmark, 27–31 August 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 337–350.
27. Smith, B.D.; Garnett, G.E. Reinforcement learning and the creative, automated music improviser. In Proceedings of the International Conference on Evolutionary and Biologically Inspired Music and Art, Málaga, Spain, 11–13 April 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 223–234.
28. Collins, N. Reinforcement Learning for Live Musical Agents. In *ICMC*; 2008. Available online: https://composerprogrammer.com/research/rlforlivemusicalagents.pdf (accessed on 12 March 2024).
29. Bown, O. Experiments in modular design for the creative composition of live algorithms. *Comput. Music J.* **2011**, *35*, 73–85. [CrossRef]

30. Hutchings, P.; McCormack, J. Using autonomous agents to improvise music compositions in real-time. In Proceedings of the Computational Intelligence in Music, Sound, Art and Design: 6th International Conference, EvoMUSART 2017, Amsterdam, The Netherlands, 19–21 April 2017; Proceedings 6; Springer: Berlin/Heidelberg, Germany, 2017; pp. 114–127.
31. Wooldridge, M.J. *An Introduction to Multiagent Systems*, 2nd ed.; Wiley: Chichester, UK, 2009.
32. Blackwell, T.; Bown, O.; Young, M. Live Algorithms: Towards autonomous computer improvisers. In *Computers and Creativity*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 147–174.
33. Carpenter, G.; Grossberg, S. *Adaptive Resonance Theory*; Technical Report; Boston University Center for Adaptive Systems and Department of Cognitive: Boston, MA, USA, 1998.
34. Rauber, A.; Merkl, D.; Dittenbach, M. The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data. *IEEE Trans. Neural Netw.* **2002**, *13*, 1331–1341. [CrossRef] [PubMed]
35. Kohonen, T. The self-organizing map. *Proc. IEEE* **1990**, *78*, 1464–1480. [CrossRef]
36. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.
37. van der Maaten, L.; Hinton, G. Visualizing Data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
38. Pál, T.; Várkonyi, D.T. Comparison of Dimensionality Reduction Techniques on Audio Signals. In *ITAT*; 2020; pp. 161–168. Available online: https://www.researchgate.net/profile/Tamas-Pal-2/publication/348416323_Comparison_of_Dimensionality_Reduction_Techniques_on_Audio_Signals/links/5ffdbba5299bf140888cf2d0/Comparison-of-Dimensionality-Reduction-Techniques-on-Audio-Signals.pdf (accessed on 12 March 2024).
39. Dupont, S.; Ravet, T.; Picard-Limpens, C.; Frisson, C. Nonlinear dimensionality reduction approaches applied to music and textural sounds. In Proceedings of the 2013 IEEE International Conference on Multimedia and Expo (ICME), San Jose, CA, USA, 15–19 July 2013; IEEE: New York, NY, USA, 2013; pp. 1–6.
40. Raffel, C.; Ellis, D.P. Intuitive analysis, creation and manipulation of MIDI data with pretty_midi. In Proceedings of the 15th International Society for Music Information Retrieval Conference Late Breaking and Demo Papers, Taipei, Taiwan, 27–31 October 2014; pp. 84–93.
41. Dong, H.W.; Chen, K.; McAuley, J.; Berg-Kirkpatrick, T. MusPy: A Toolkit for Symbolic Music Generation. *arXiv* **2020**, arXiv:2008.01951.
42. Zytrax. Frequency Ranges. 2022. Available online: https://www.zytrax.com/tech/audio/audio.html (accessed on 13 December 2023).
43. Young, C. Audio Frequency Ranges. 2023. Available online: https://www.gear4music.com/blog/audio-frequency-range/ (accessed on 13 December 2023).
44. Fortunato, M.; Azar, M.G.; Piot, B.; Menick, J.; Osband, I.; Graves, A.; Mnih, V.; Munos, R.; Hassabis, D.; Pietquin, O.; et al. Noisy networks for exploration. *arXiv* **2017**, arXiv:1706.10295.
45. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. *arXiv***2015**, arXiv:1511.05952.
46. Fedus, W.; Ramachandran, P.; Agarwal, R.; Bengio, Y.; Larochelle, H.; Rowland, M.; Dabney, W. Revisiting fundamentals of experience replay. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual Event, 13–18 July 2020; pp. 3061–3071.