Multi-Agent Reinforcement Learning for Structured Symbolic Music Generation

Shayan Dadman[1 [0000−0003−1970−5353]] and Bernt Arild Bremdal[1 [0000−0002−0042−3949]]

Department of Computer Science, Arctic University of Tromsø, Lodve Langesgate 2, 8514 Narvik, Norway shayan.dadman@uit.no

Abstract. Generating structured music using deep learning methods with symbolic representation is a challenging task due to the complex relationships between musical elements that define a musical composition. Symbolic representation of music, such as MIDI or sheet music, can help overcome some of these challenges by encoding the music in a format that allows manipulation and analysis. However, the symbolic representation of music still requires interpretation and understanding of musical concepts and theory. In this paper, we propose an approach that utilizes Multi-Agent Systems (MAS) and Reinforcement Learning (RL) for symbolic music generation. Our model primarily focuses on music structure. It operates at a higher level of abstraction, enabling it to capture longer-term musical structure and dependency. We utilize RL as a learning paradigm and the human user as a musical expert to facilitate the agent's learning of global dependency and musical characteristics. We show how the RL agent can learn and adapt to the user's preferences and musical style. Furthermore, we present and discuss the potential of our approach for agent learning and adaptation and distributed problem-solving in the field of music generation.

Keywords: Adaptive learning · distributed problem solving · deep learning (DL) · deep Q-network (DQN) · multi-agent systems (MAS) · reinforcement learning (RL) · music generation.

1 Introduction

Music has a clear, well-defined structure that provides the foundation for creating a piece. It can sound disorganized, disjointed, and lacking musical coherence without a clear structure. The structure challenge in symbolic music generation involves generating a musical piece that follows the rules of music theory while maintaining a coherent structure throughout the piece. These rules can include adhering to a consistent key signature, following chord progressions, and maintaining a steady rhythm and tempo. Additionally, the generated piece must have a clear structure that captures the listener's attention. This requires creating a sense of tension and release throughout the piece, as well as varying the melody, harmony, and rhythm to add contrast and variety.

Deep learning models can address the structure challenge in symbolic music generation in various ways, as highlighted by [3]. Models such as MusicVAE [17], Music Transformer [8], and MuseGAN [5] generate music that is musically coherent and stylistically consistent with the input dataset. In some cases, the music generated by these programs can be tedious or repetitive, particularly in longer pieces where there is a lack of variation over time. These note-based models struggle to capture the complexity of musical expressions, such as rhythm, dynamics, and articulation, as they focus primarily on the immediate context of the preceding notes [20]. Furthermore, the optimization objective of these models is often based on minimizing a loss function that measures the discrepancy between the generated music and the training data.

In contrast, reinforcement learning (RL) models learn through an iterative trial and error process, providing flexibility and adaptability to changes in the task or environment. RL-Tuner [9] utilizes two DQN and two RNN models to generate melodies using user-defined constraints. Later, [13] proposed an extension to RL-Tuner that uses the Latent Dirichlet Allocation (LDA) called RE-RLTuner. [12] used LSTM RNN to compose melody and chords, where the agent's objective is to find a suitable combination of sequences. RL-Duet [11] can generate melodic and harmonic parts in an online accompaniment framework using actor-critic with a generalized advantage estimator (GAE). RL-Chord [10] is a melody harmonization system using RL and conditional LSTM (CLSTM) to generate chord progression. [1] 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. Nevertheless, despite the RL advantage in music generation, defining a reward function for musical structure remains challenging [3]. Therefore, the generated music by RL models may still lack coherency and structure.

Another potential approach to addressing symbolic music generation is using multi-agent systems (MAS) in combination with reinforcement learning (RL) [3]. MAS are systems composed of multiple agents that interact with each other to achieve a common goal [19]. Similar to how musicians in a band collaborate and coordinate, agents in MAS architecture can work together, each focusing on specific aspects of the musical structure. Despite its potential, MAS has limitations, including the challenge of coordinating multiple agents, which can lead to high computational complexity and difficulties in balancing agents' autonomy with system coherence. By utilizing RL in MAS, the agents learn by trial and error, thereby refining their behaviors through interaction with the environment. This approach allows the agents to adapt to the musical context and each other, resulting in more harmonious and engaging compositions.

Smith and Garnett [18] propose a musical agent with adaptive resonance theory (ART) and reinforcement learning (RL) to generate monophonic melody. ART is similar to Self-Organizing Maps (SOMs), used to classify and categorize the data vectors. Improvagent [2] is a musical agent that utilizes Sarsa reinforcement learning algorithm. 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.

Moreover, we can incorporate human expertise and creativity into the music generation process through MAS. Communication between agents and a human agent allows for guidance on the overall direction of the music generation, while RL agents handle the low-level details of generating individual musical elements. Indeed, by orchestrating agents similar to musicians in a band and facilitating communication with a human agent, MAS can capture and model the complex interactions and dependencies between musical elements.

Here, we propose a model based on MAS to tackle the structure challenge of symbolic music generation. Our model works directly with musical patterns. It operates at a higher level of abstraction than note-based models. The idea is to capture long-term musical structure and dependency by learning to identify and manipulate patterns. In this manner, the model can generate more complex and interesting musical pieces. Besides, our model utilizes MAS architecture by incorporating RL deep Q-network (DQN) as a learning paradigm and the human agent as a musical expert. Through interaction with the environment, the RL agent receives input and feedback for the generated music from the music-related reward functions and the human agent. This allows the RL agent to learn and adapt to the user's preferences and musical style. Furthermore, we introduce a method utilizing the DQN replay buffer as the MAS communication method. This method represents a collaborative learning process, allowing the agents to coordinate their actions more effectively and achieve better results. Indeed, this framework offers interactivity, flexibility, and adaptability throughout the generation process.

2 Background

Growing Hierarchical Self-Organizing Maps (GHSOM) is an unsupervised machine learning algorithm that learns to represent high-dimensional input data in a lower-dimensional space [4]. GHSOM is useful for clustering and visualization of data and able to grow a hierarchical structure of self-organizing maps (SOMs). It can capture the input data's local and global structure by recursively splitting a SOM into smaller SOMs. At each level of the hierarchy, GHSOM learns a codebook of prototype vectors representing the input data through a process known as competitive learning.

Recurrent Neural Networks (RNNs) are a class of neural networks that can process sequential data by allowing information to persist over time. 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. Long Short-Term Memory (LSTM) is a type of RNN that effectively addresses this problem with gating mechanisms, consisting of three sigmoidal units and one hyperbolic tangent unit, that selectively update, forget, and output in-

formation. By using a cell state as "memory," LSTMs can effectively capture long-term dependencies in sequential data.

Reinforcement Learning (RL) is a machine learning subfield that teaches agents to make decisions based on rewards and punishments. The agent interacts with an environment, learns from feedback, and adapts its behavior to achieve a specific goal. The agent's objective is 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, and policy-based methods aim to learn the optimal policy directly. Additionally, there are hybrid approaches, such as deep Q-networks (DQN) [14], which combine Q-learning with deep neural networks to approximate the Q-value function and handle large and continuous state spaces. RL is useful in developing autonomous agents that can learn from experience, improve their decision-making processes, and optimize their behavior over time.

Dimensionality Reduction is a technique that involves reducing the number of features or variables in a dataset while maintaining as much information as possible. This is typically achieved by projecting high-dimensional data onto a lower-dimensional space. There are two main categories of dimensionality reduction techniques: linear and nonlinear. Linear techniques, such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), are commonly used for datasets with a linear structure. Nonlinear techniques, such as t-Distributed Stochastic Neighbor Embedding (t-SNE), UMAP (Uniform Manifold Approximation and Projection), and ISOMAP (Isometric Feature Mapping), are used when the underlying structure of the data is nonlinear.

For music, [6] and [15] performed a comparative analysis between PCA, t-SNE, ISOMAP, and SOMs 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.

3 System Design

In Section 1, we presented the approaches for symbolic music generation using RL algorithms. RL algorithms are designed to adjust the behavior of a single agent and learn based on the rewards received from the environment. One of the main challenges of RL is the trade-off between exploration and exploitation. Besides, defining a suitable reward function that addresses musical characteristics according to human user preferences is hard. We address these challenges in our model by using MAS architecture and involving the human user in the agents' learning process and providing information about the system's goals.

In the following, we explain different aspects of our model. We first explain the data processing approach and then continue with training and generation

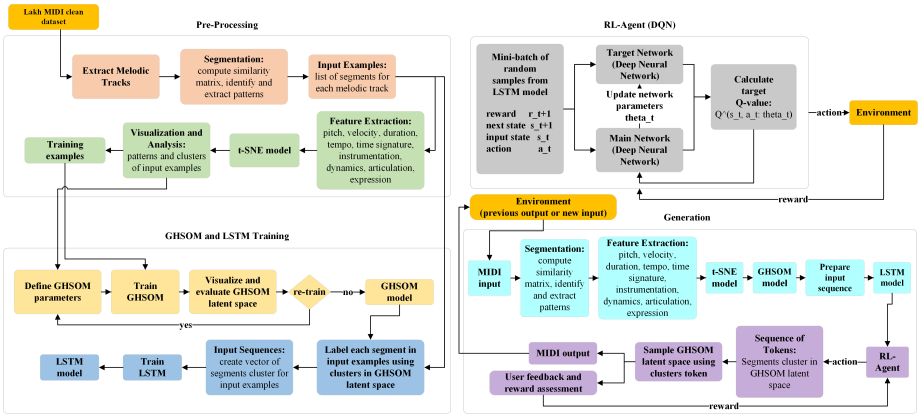

Fig. 1. The architecture of the proposed model

processes. Figure 1 illustrates the components of our model and depicts the training and generation processes. Overall, our model involves a perceiving agent, a generative (decision-making) agent, and a human agent (user). The perceiving agent organizes and processes the inputs and passes them to the generative agent. The generative agent carries out the system's output by considering the previous and current state. Indeed, the perceiving agent provides the generative agent with a better understanding of the environment by encoding the inputs in a higher level of abstraction. In this manner, the generative agent concentrates more on its action improvement. The user evaluates the model's output and provides new inputs to guide the generations. User feedback enables the generative agent to adapt and learn the user's preferences, introduce novelty and creativity, and navigate through complex environments such as music structure. Furthermore, the perceiving agent observes the changes in the environment, such as the human feedback given to the RL agent to provide input according to the related musical context.

3.1 Data Processing

Symbolic representation of music refers to the process of encoding musical information using a set of symbols, typically in the form of a digital score or MIDI file. Each musical event is represented in this format by a combination of symbols that encode its pitch, duration, timing, and other relevant attributes.

Here, we consider Clean MIDI subset of the Lakh MIDI dataset [16]. This subset consists of 45,129 MIDI files that have been filtered and cleaned to remove any duplicates, corrupt files, or files that do not meet specific quality standards. These quality standards include having a minimum number of tracks, a minimum duration, and being free of obvious errors such as missing or extraneous notes. We process each MIDI example to identify and maintain only melodic tracks. We identify the short and long patterns within the melodic tracks using

the similarity matrix, similar to [1] approach. We mark and segment the melodic tracks based on identified patterns and extract each segment as a separate MIDI file. We also maintain the order of segments in original melodic tracks. We extract relevant musical features as a feature vector for each extracted segment. The musical features include pitch, duration, velocity, tempo, time signature, instrumentation, dynamics, articulation, and expression. After creating the feature vectors, we normalize them so that each feature is scaled to the same range. We use t-SNE to reduce the dimensionality of the feature vectors before passing them to GHSOM for training.

3.2 Perceiving Agent - GHSOM and LSTM

GHSOM We follow the instructions given by [4] to implement the model and train it after processing and preparing the training examples as described in Section 3.1. Furthermore, to define the model parameter, we utilize the t-SNE algorithm. In particular, we use t-SNE to determine the number of levels in the GHSOM hierarchy, the size of the maps at each level, and the learning rate and neighborhood function parameters.

To determine the number of levels in the GHSOM hierarchy, we use t-SNE to visualize and inspect the resulting clusters or patterns in the data. The number of levels in the GHSOM hierarchy can be chosen to correspond to the level of abstraction suggested by the t-SNE visualization. To determine the size of the maps at each level, we use t-SNE to estimate the local density of the data in the lower-dimensional space. In GHSOM, maps at each level should be large enough to capture the local structure of the data but not so large as to lose the resolution needed to distinguish between neighboring clusters. The size of the maps can be chosen based on the local density estimated from the t-SNE visualization. We choose the learning rate and neighborhood function parameters using the identified clusters or patterns in the data by t-SNE. These parameters maintain balance in the exploration of the high-dimensional space with the exploitation of the clusters or patterns suggested by the t-SNE visualization.

Note that we use t-SNE as a tool for exploration and interpretation rather than as a definitive guide to the GHSOM parameter selection.

LSTM Following the same order of the segments in the original melodic track, we use the trained GHSOM to label each segment with the corresponding cluster number. In this manner, we create a vector of numbers and prepare the training examples for the LSTM model. Essentially, we train the LSTM model to capture the temporal dependencies to predict the next token. 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. We used `ELU` (Exponential Linear Unit) as an activation function for the LSTM layer and `softmax` for the dense layer, and Adam as an optimizer to minimize the cross-entropy function.

3.3 Generative Agent - Optimization with RL

Model Architecture While the GHSOM captures the topological structure and the LSTM model learns the dependency among segments from the original melodic tracks, the model may get stuck with a specific order of segments and need help to explore new variations. We use Reinforcement Learning DQN algorithm to further optimize the model's performance. DQN [14] is a model-free RL algorithm that uses Q-learning to learn optimal policies. It maximizes the total reward at the end of each epoch by selecting policies based on a minibatch of random samples from the LSTM model. The main network generates actions, and the target network produces a stable target to compute the loss of the selected 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.

We train the main network, to approximate the optimal action-value function $Q(st, at)$. $Q(st, at)$ represents the expected cumulative reward for taking action, a_t, in state, s_t, and following the optimal policy after that. The input to the network is the current state, s_t, and the output is a vector of Q-values for each possible action, a_t, in that state. During training, the network is updated using a variant of the Q-learning algorithm that minimizes the difference between the predicted Q-values and the true Q-values obtained from the Bellman equation. The target network is a separate copy of the main network that is used to generate the target Q-values used in the Q-learning update. 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 predict a sequence of tokens, where it learns the structure and capture the transitions between the segments. Therefore, the model parameters are updated after generating a complete sequence rather than a single token.

Moreover, to encourage the model to explore action space, we use NoisyNet [7]. NoisyNet addresses the exploration-exploitation tradeoff by adding noise to the weights of the neural network used to estimate the Q-values or policy. The noise is added in a way that preserves the differentiability of the network. In this manner, it can still be trained using gradient descent. The main advantage of NoisyNet is that it provides a principled way of balancing exploration and exploitation without requiring additional exploration noise to be added to the actions. We use Python and Tensorflow library with Keras to implement functionalities and agents.

Reward Definition We define the reward policy based on three criteria:

- r_{ground_truth}: Ground truth reward based on the original melodic tracks
- $r_{structure}$: Sequence structure reward based on manual rules
- r_{hil}: Human feedback reward

The ground truth reward, r_{ground_truth}, evaluates the generated sequence based on the original melodic tracks. As described in Section 3.1, each melodic track in the dataset are segmented based on the variation of patterns. We measure the gap between the model's output and the ground truth using the negative log-likelihood (NLL) loss. NLL penalizes the model for assigning a low probability to the observed data and rewards it for assigning a high probability to the observed data. The objective is to decrease the loss as the agent continues learning.

The sequence structure reward, $r_{structure}$, is proposed to evaluate the transition between the segments within the generated sequence. The main objective is to prevent the model from sudden transitions that are relatively quick or completely irrelevant. To do so, we train a smaller GHSOM model using only the vector of segments within each melodic track. Then we use the topological latent space of GHSOM to assess the transitions based on the closeness of the segment at step `t` to the `t-1` segment within the predicted sequence. To measure the closeness, we use Euclidean (L2) distance. Given the calculated distance, the definition of the reward is

$$r_{structure}(d) = \begin{cases} -1, & \text{if } d > threshold \\ 1, & \text{if } d < threshold \end{cases} \tag{1}$$

where the *threshold* is an experimental value calculated by taking a percentage of the average distance between all of the data points.

The human feedback reward, r_{hil}, incorporates the human-in-the-loop (HIL). The basic idea behind HIL is to use feedback from a human expert to shape the reward function of the reinforcement learning agent. The user provides feedback in the form of evaluations of the agent's actions, which are then used to adjust the reward function to better align with the user's preferences. Specifically, the reward function is augmented with a term that captures the feedback from the user. The user provides explicit evaluations of the generation with $+1$ as positive reward and -1 as negative reward. The human feedback reward is as follows:

$$r_{hill}(s_t, a_t) = w * e(s_t, a_t) \tag{2}$$

where w is a weight that controls the influence of the expert evaluations on the reward function, and $e(s_t, a_t)$ is the expert evaluation of the agent's action in state s_t and action a_t.

The instant reward r_t for the action to be taken at time t is

$$r_t = \alpha * r_t^{ground\_truth} + \beta * r_t^{structure} + \gamma * r_t^{hil} \tag{3}$$

where α, β, and γ are the weight factors that are experimental values. They controls the impact of each reward function in guiding the agents behavior and can be adjusted during training.

3.4 Agents Communication

Communication is an essential aspect of MAS, as agents need to exchange information and coordinate their actions. Various communication methods have been proposed and implemented in MAS, ranging from message-passing and negotiation protocols to the use of shared memory spaces [19]. Our communication method uses the DQN replay buffer that stores the agent's experiences as tuples (state, action, reward, next state) that stabilizes and improves the learning process. By extending the replay buffer to serve as a communication repository, agents can access a shared replay buffer to not only learn from their experiences but also benefit from the experiences of other agents. This collaborative learning process allows the agents to coordinate their actions more effectively and achieve better results in complex environments.

3.5 Generation

During the generation process, the input of the model consists of the outputs in the previous step and the human inputs. In the first step, the perceiving agent processes the given input as described in Section 3.1. Then it uses GHSOM to identify the input cluster within the GHSOM latent space. Using the output of GHSOM, it creates the input vector for the generative agent to generate the next sequence of tokens. At this stage, the system provides the human user with the generated sequence for evaluation. It incorporates the evaluations obtained from the human user and other reward functions using Equation 3 to guide the generative agent. To generate content, the system takes the tokens in the generated sequence and uses the GHSOM latent space to randomly sample the corresponding cluster of segments. Figure 1 illustreates the generation process of the system.

4 Discussion

Here, we proposed an approach that works directly with musical patterns to capture long-term dependency between musical elements. This approach can capture the common patterns and structures that define a musical style and provide the RL agent with a more structured and meaningful input. In this manner, the agent obtains a better understanding of dependency among musical elements and the overall flow of the music. Consequently, the agent can generate new pieces of music by combining, manipulating, and rearranging the patterns creatively.

In our approach, the agent can also learn from the human user's feedback to generate music. In this way, the generated music is more aligned with the user's preferences. As an expert, the human user can provide valuable feedback on the quality and guide the desired structure, style, and emotional content of the music. The human user also provides input throughout the interaction. The given input can include examples of music in a particular structure or style.

The agent uses the input to improve the quality of generated music and adjusts its behavior regarding musical style and structure. Similarly, it can generate a complete piece based on that input.

MAS architecture allows modularity and the use of various computational methods. It promotes distributed problem solving, which is solving complex problems by breaking them down into smaller, more manageable sub-problems that can be solved by multiple agents working together. Our system consists of three agents: perceiving, generative, and human agents. The perceiving and generative agents use the replay buffer as a communication repository to coordinate their actions collaboratively. During their interaction, both agents carefully observe the feedback from the human and use it to improve their performance. The generative agent learns and adapts to the user's preferences based on the feedback received, while the perceiving agent uses the feedback to provide relevant input to the generative agent. The perceiving agent interacts with the replay buffer by grouping and selecting the experiences based on their similarity to human feedback.

Additionally, we can incorporate human expertise and preferences into the communication process by observing and providing feedback on the experiences stored in the shared replay buffer. We can add human feedback as metadata in the replay buffer. This input could be suggestions for alternative actions, additional context information, or other guidance based on human feedback. In this manner, the human agent can guide the agents to focus on specific experiences or suggest alternative actions. Additionally, we can access the agents' evaluations and suggestions in the shared replay buffer to better understand the MAS's current state, decision process, and performance.

One approach to creating an interactive interface for human agents to interact with MAS is the use of PureData (PD) and Open Sound Control (OSC). PD's visual programming environment ensures that interaction is intuitive and user-friendly, while OSC facilitates efficient communication between the PD interface and the Python-based MAS. This approach enables real-time adjustments and feedback integration. Additionally, we can integrate PD with a Digital Audio Workstation (DAW) to enhance the user experience further.

We can expand the system by introducing several RL agents with diverse behaviors, each assigned to a specific task, such as melody, harmony, and rhythm generation. Nonetheless, adding more RL agents introduces challenges in effectively coordinating them as system complexity rises. Indeed, maintaining a shared replay buffer for many agents can lead to increased memory and computational requirements. We can explore techniques such as data compression, prioritized experience replay, and hierarchical organization of agents to optimize the system for several agents. Data compression techniques can help reduce the replay buffer's memory footprint, while prioritized experience replay can enhance the learning process by focusing on the most informative experiences. Hierarchical organization of agents, where a group of specialized agents works under a higher-level coordinating agent, can simplify the complexity of managing multiple agents and their interactions. However, these optimizations might introduce

trade-offs in system performance, learning speed, and resource consumption. The replay buffer communication is also limited in its applicability in heterogeneous MAS. This poses a challenge to managing possible conflicts and the agent's ability to negotiate or compromise to find a mutually acceptable solution.

One strategy is to train agents to learn from the actions of other agents in their local environment using decentralized training. It can lead to robust and adaptive agents, as each agent can learn from its own experiences and adapt to environmental changes. We can integrate decentralized learning with a shared replay buffer to form a hybrid approach for MAS. This approach combines decentralized learning's scalability, robustness, and flexibility with shared replay buffer's enhanced coordination, knowledge transfer, and human-agent interaction, resulting in more effective learning and collaboration among diverse agents. Consequently, the integrated approach can lead to improved overall performance, alignment with human preferences, and efficient achievement of shared goals in complex multi-agent environments.

5 Conclusion

In this study, we discussed the current approaches in symbolic music generation and their shortcomings. Notably, we emphasized the challenges involved with these models to address musical structure. Mainly, these models struggle to generate innovative and expressive music with long-term structure. In a way, the note-based approach of these models limits their ability to capture the complexity of musical expressions, such as rhythm, dynamics, and articulation. On the other hand, reinforcement learning models offer more flexibility and adaptability but face challenges in navigating complex environments and definition of music related reward functions.

To alleviate these challenges, we proposed an approach based on multi-agent systems and reinforcement learning algorithms that works directly with musical patterns. The proposed approach particularly improves the agent's adaptability to human preferences and learning of musical elements dependency to capture the structure and overall flow of the music. Additionally, we discussed that the modularity of MAS architecture allows for distributed problem-solving by utilizing multiple agents, each specializing in specific musical tasks. However, challenges exist in programming agents to coordinate effectively, particularly when introducing a diverse range of agents. Therefore, we discussed how combination of decentralized training and replay buffer could be a suitable strategy to alleviate this challenge. Overall, the proposed model represents an interactive, adaptable, and flexible framework for music generation.

References

1. Chen, S., Zhong, Y., Du, R.: Automatic composition of guzheng (chinese zither) music using long short-term memory network (lstm) and reinforcement learning (rl). Scientific Reports **12**(1), 15829 (2022)

2. Collins, N.: Reinforcement learning for live musical agents. In: ICMC (2008)
3. Dadman, S., Bremdal, B.A., Bang, B., Dalmo, R.: Toward interactive music generation: A position paper. IEEE Access **10**, 125679–125695 (2022)
4. Dittenbach, M., Merkl, D., Rauber, A.: The growing hierarchical self-organizing map. In: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium. vol. 6, pp. 15–19. IEEE (2000)
5. 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. vol. 32 (2018)
6. Dupont, S., Ravet, T., Picard-Limpens, C., Frisson, C.: Nonlinear dimensionality reduction approaches applied to music and textural sounds. In: 2013 IEEE International Conference on Multimedia and Expo (ICME). pp. 1–6. IEEE (2013)
7. 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 preprint arXiv:1706.10295 (2017)
8. 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 preprint arXiv:1809.04281 (2018)
9. Jaques, N., Gu, S., Turner, R.E., Eck, D.: Tuning recurrent neural networks with reinforcement learning (2017)
10. Ji, S., Yang, X., Luo, J., Li, J.: Rl-chord: Clstm-based melody harmonization using deep reinforcement learning. IEEE Transactions on Neural Networks and Learning Systems (2023)
11. 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. vol. 34, pp. 710–718 (2020)
12. Kumar, H., Ravindran, B.: Polyphonic music composition with lstm neural networks and reinforcement learning. arXiv preprint arXiv:1902.01973 (2019)
13. Liu, H., Xie, X., Ruzi, R., Wang, L., Yan, N.: Re-rltuner: A topic-based music generation method. In: 2021 IEEE International Conference on Real-time Computing and Robotics (RCAR). pp. 1139–1142. IEEE (2021)
14. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
15. Pál, T., Várkonyi, D.T.: Comparison of dimensionality reduction techniques on audio signals. In: ITAT. pp. 161–168 (2020)
16. Raffel, C.: The lakh midi dataset v1.0. https://colinraffel.com/projects/lmd/ (2016)
17. Roberts, A., Engel, J., Raffel, C., Hawthorne, C., Eck, D.: A hierarchical latent vector model for learning long-term structure in music. In: International conference on machine learning. pp. 4364–4373. PMLR (2018)
18. Smith, B.D., Garnett, G.E.: Reinforcement learning and the creative, automated music improviser. In: International Conference on Evolutionary and Biologically Inspired Music and Art. pp. 223–234. Springer (2012)
19. Wooldridge, M.J.: An introduction to multiagent systems. Wiley, Chichester, 2nd ed. edn. (2009)
20. Wu, S.L., Yang, Y.H.: Musemorphose: Full-song and fine-grained music style transfer with one transformer vae. arXiv preprint arXiv:2105.04090 (2021)