An Internet-based Music Generation System using Reinforcement Learning

Cheng-Han Wu[1], Timothy K. Shih[1], Chien-Hao Huang[1], Yu-Cheng Lin[2]*

[1] Department of Computer Science and Information Engineering, National Central University, Taiwan
[2] Department of Computer Science and Engineering, Yuan-Ze University, Taiwan
whoshank@gmail.com, timothykshih@gmail.com, chhuang8869@gmail.com, linyu@saturn.yzu.edu.tw

Abstract

Deep learning has been widely applied to digital art and music creation. However, producing melodies that follow music theory and match human compositional patterns remains challenging. This study proposes a symbolic music generation system that integrates supervised learning and reinforcement learning. The core framework employs recurrent neural networks for sequence modeling, while a reinforcement learning module formulates music rules as reward functions to guide pitch, duration, and rhythm. This hybrid approach helps produce outputs that better match human compositional patterns. We deploy the proposed framework as an Internet-based system that supports five distinct music styles and is publicly accessible online.

Keywords: Symbolic music generation, Reinforcement learning, Long Short-Term Memory (LSTM), Asian music styles, Reward function design

1 Introduction

Deep learning has advanced generative modeling, including music generation, through improved training methods, large-scale datasets, and increased computational resources. While commercial systems can generate and edit music under style or user-specified constraints, generating melodies that follow music theory and align with human preferences remains challenging.

Music generation tasks are commonly categorized into audio and symbolic types. Audio methods operate on waveforms and focus on sound quality. Symbolic methods use representations such as MIDI and focus on musical structure with lower computational cost. This work focuses on symbolic music generation. We target monophonic melodies that follow distinct style patterns, with a primary emphasis on traditional Chinese music.

Model training for specific styles is often constrained by data scarcity. In this study, only about 200 original scores were collected per style. We therefore apply supervised learning with a Markov model guided by music theory to generate additional training data. The augmented scores are selected by experts. The final dataset for each style is about ten times larger than the original set.

Music composed by humans follows structural and theoretical rules that are difficult to capture using supervised training alone. The learning framework adopted in this work builds upon a previously established approach that combines supervised learning and reinforcement learning for symbolic melody generation [1]. We formulate music rules as reward functions to guide pitch, duration, and rhythm. This hybrid approach helps the system learn both data patterns and music theory expectations. This study extends the framework to an Internet-based system that supports five styles.

Music evaluation is challenging. Therefore, we provide an Internet-based system for public access and feedback. The tool is available online at https://sites.google.com/g.ncu.edu.tw/minelabaimusicvideo. It generates music in five distinct styles. Feedback from users and musicians has been positive, and we are collaborating with a local music company to further improve the system.

2 Related Work

Early studies on music generation relied on rule-based systems, Markov models, and neural networks. Todd [2] and Mozer [3] introduced early recurrent approaches that are related to modern RNN formulations [4]. With improved computation, deep learning became practical for sequence generation, and LSTM networks [5] were widely adopted to model long-term dependencies.

Symbolic music generation has been studied with LSTM-based architectures in various styles. Eck and Schmidhuber [6] applied LSTM to generate blues music, while DeepBach [7] used stacked LSTM layers for chorale generation. Time with Feeling [8] extended LSTM-based modeling to predict both notes and dynamics. These studies demonstrate the effectiveness of recurrent sequence modeling for musical structure.

Beyond general sequence architectures, cultural characteristics have also been considered. MG-VAE [9] models Chinese folk songs with regional style factors using a variational autoencoder and adversarial training. However, it does not explicitly incorporate reinforcement learning or rule-based music theory constraints.

To improve realism and diversity, generative adversarial networks (GANs) [10] and other neural generative models have been explored. C-RNN-GAN [11] combines GAN training with LSTM generation, and

*Corresponding Author: Timothy K. Shih; Email: timothykshih@gmail.com
DOI: https://doi.org/10.70003/160792642026072704014

MuseGAN [12] targets multi-track polyphonic music. Symbolic Music GAN [13] generates structured symbolic melodies. Museformer [14] applies a Transformer model with hierarchical attention for long sequences, and Transformers have broadly influenced both text and music generation [15]. Music Transformer [16], developed in the Magenta project [17], uses relative attention for melody generation. More recently, MIDI-DDPM [18] applies diffusion modeling with a variational autoencoder in latent space, and discrete diffusion models [19] provide symbolic-level control. CP Transformer [20] supports interactive generation with structured MIDI input. Many of these methods focus on data-driven generation and often do not explicitly encode music theory rules.

Reinforcement learning has been applied to symbolic music generation to improve controllability and enforce constraints. RL-Tuner [21] combines a melody RNN with double DQN [22] to guide generation toward user preferences. RL-Duet [23] supports real-time accompaniment with human and system input. RL-Tuner-CP [24] integrates reinforcement learning with constraint programming, and Amadeus [25] combines supervised training with planning for polyphonic generation. RL-Chord [26] applies reward-guided generation for chord sequences, while MuseBERT [27] uses reinforcement learning in a BERT-style model for long-term coherence. Polyffusion [28] addresses rhythm generation under low-data conditions. A hybrid supervised and reinforcement learning framework for symbolic melody generation was previously validated in an academic thesis [1]. The present work extends this line of research by adopting PPO-based reinforcement learning with rule-defined rewards. It also implements an Internet-based system for multi-style melody generation.

3 The Proposed Models

Our system includes three models that follow the same design strategy. Each model is an LSTM-based sequence generator. The three models are the pitch model, the duration model, and the bar profile model. At each time step, each model generates one value: pitch, duration, or bar profile.

The bar profile is a high-level feature that represents the rhythm pattern within a bar. More details are given in the next section. Because this work uses symbolic music generation, we define each note using two basic attributes: pitch and duration. We represent durations using the thirty-second note as the smallest unit. For example, an eighth note equals four thirty-second notes, and a quarter note equals eight thirty-second notes. We do not model note velocity because our focus is melody.

To use both reinforcement learning rewards and dataset features, we train the models with supervised learning and reinforcement learning in an interleaved manner. Both training stages use the same network architecture. During supervised training, we train the policy network that is later used in reinforcement learning. The value network is not used in supervised training because only the policy network selects actions. This strategy lets the model learn from the dataset while following constraints. All three models are trained using the same procedure.

The pitch, duration, and bar profile models are trained separately with the same architecture, as shown in Figure 1. In this figure, the term "symbol" refers to pitch, duration, or bar profile, depending on the model. At each time step, the current symbol is represented by a one-hot vector, as in standard LSTM training. We also include additional input features to improve learning. These features are described in the next section. Although the explicit input at each time step is one symbol, the LSTM also maintains a hidden state. The hidden state stores the sequence history. Therefore, generation depends on both the current symbol and the sequence context captured by the hidden state.

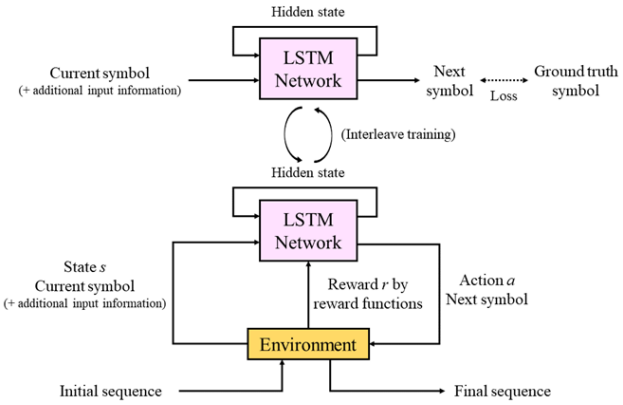


Figure 1. Overall architecture of the proposed framework with interleaved supervised learning and reinforcement learning

During both training and generation, the bar profile model outputs one bar profile for each bar. This bar profile is used as an input to the pitch model and the duration model for the same bar. In addition, both models take the current pitch and the current duration as inputs. In Figure 2, b, p, and d denote the bar profile, pitch, and duration, respectively. M_b, M_p, and M_d denote the bar profile, pitch, and duration models, respectively. For clarity, Figure 2 omits the hidden state, additional input features, and positional encoding. These components are described in the following section.

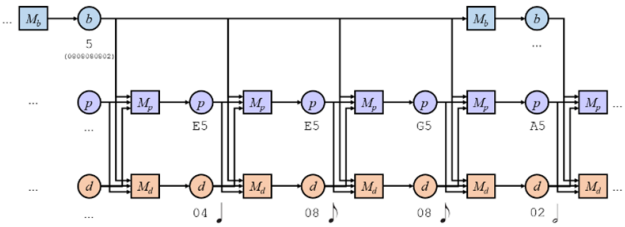
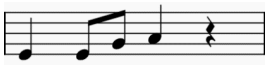


Figure 2. Architecture for generating a single bar, where b, p, and d denote the bar profile, pitch, and duration, respectively

3.1 Hierarchical Recurrent Neural Network (Bar Profile)

Inspired by [29], one model in our system generates a sequence of bar profiles. A bar profile is a high-level representation of the rhythm pattern within a bar. We define 16 bar profiles, and each profile is represented by a one-hot vector of length 16. This representation is consistent with those used for pitch and duration. Following [29], we construct bar profiles as shown in Figure 3. We first segment each melody in the dataset into bars. We then convert the note durations in each bar into a binary sequence of `0`s and `1`s, using the thirty-second note as the smallest unit. In this step, pitch information is not used.



`[10000000100010001000000010000000]`

Figure 3. Converting note durations in a single bar into a binary sequence of `0`s and `1`s

We then apply K-modes clustering, a discrete variant of K-means in which cluster centers are integer vectors. We cluster all bar clips and select 16 cluster centers as the main bar rhythm patterns. If clustering performs well, these 16 bar profiles should match most bar clips in the dataset. They can therefore represent the rhythmic structure of the dataset.

We choose K=16 based on practical experience. Our goal is to use a reasonable number of rhythm patterns that cover the dataset well. If K is too large, the generated music may sound overly random and show less repetition. A larger K also increases the input dimension, which can reduce model performance. Figure 4 shows the average distance from each bar rhythm in the dataset to the K cluster centers for different values of K. We set K=16 to balance computational efficiency with the ability to capture recurring rhythm patterns.

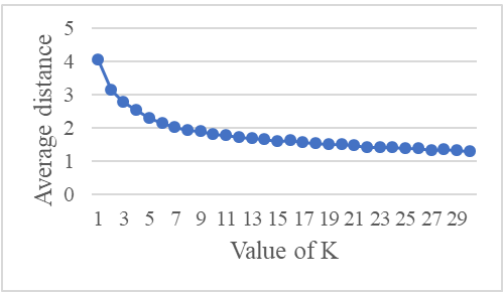


Figure 4. Average distance from each bar rhythm to the K cluster centers for different K values

We run the clustering method multiple times and select the set of profiles with the lowest average distance to the cluster centers. Because the initial cluster centers are randomly chosen, the results vary slightly across runs. The final rhythm patterns for the 16 bar profiles are shown in Figure 5. Each line in the figure shows two bars. As noted in [16], the bar profile sequence is much shorter than the pitch and duration sequences because the model generates one bar profile per bar. This shorter sequence makes it easier for the model to learn the overall structure of high-level rhythm patterns.

To reduce the input dimension, we select only the 22 pitches that are most common in our dataset. For durations, we select the 11 duration types observed in the dataset. These 22 pitches and 11 durations cover most of the content in Taiwanese Opera and Hakka Folk Song, which are the main styles in this study. If other researchers apply our modules to other styles, these values can be adjusted accordingly.

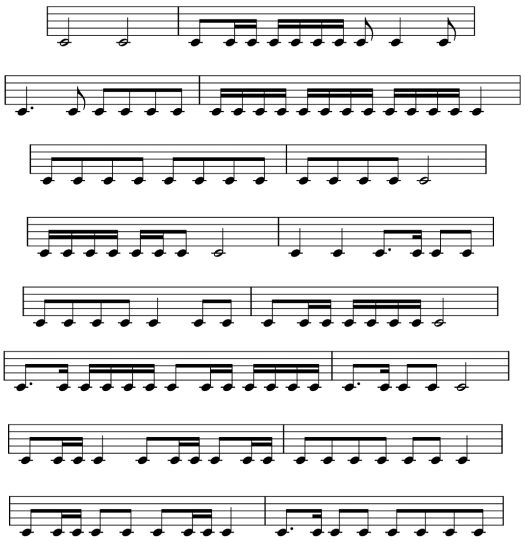
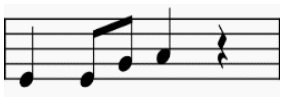


Figure 5. Rhythm patterns of the 16 bar profiles

Each time the duration model generates a value, it outputs an explicit duration. This encoding follows the note-level representation described in [30], where each duration is represented by a one-hot vector. This format is close to how humans describe duration during composition. Because each note has both a pitch and a duration, the pitch and duration sequences always have the same length. Another common approach is a frame-based sequence. In this approach, a melody clip is divided into fixed-length frames, and each frame represents an event. Events can include pitch onset, no event, or pitch offset. Both encoding methods can represent any monophonic melody. In our experiments, the note-level one-hot encoding gives better results. Figure 6 compares the two encoding methods.



Explicit encoding

```
Pitch:    [E5  E5  G5  A5  Rest]
Duration: [4   2   2   4   4   ]
```

Implicit encoding

`[E5 N N N E5 N G5 N A5 N N N Off N N N]`

Figure 6. Two methods for encoding the duration of the same melody clip (In the implicit encoding, `N` denotes "no event" and `Off` denotes "pitch offset." Both methods use the sixteenth note as the smallest time unit.)

For the pitch model, we use the current pitch, the current duration, and extra structural information. These features help the model learn rhythm and timing more effectively. The input to the pitch model includes the following components:

1. Current pitch: represented by a one-hot vector of length 22.
2. Current duration: represented by a one-hot vector of length 11.
3. Beat position within the bar: represented by a one-hot vector of length 4 (since there are 4 beats per bar).
4. Frame position within the beat: represented by a one-hot vector of length 8 (since there are 8 frames per beat).
5. Bar parity: represented by a binary value (0 for odd-numbered bars and 1 for even-numbered bars).
6. Relative position in the melody: represented by a floating-point value between 0 and 1 (0 indicates the beginning and 1 indicates the end).
7. Current bar profile: represented by a one-hot vector of length 16 (output from the bar profile model).

We combine these seven components to form an input vector of length 63. The shortest duration in our system is a thirty-second note. Therefore, each bar contains 32 frames. Components 3 and 4 specify the exact note position within a bar because 32 frames equal 4 beats with 8 frames each. We include component 5 because the dataset shows a clear difference between odd and even bars. Component 6 indicates the relative position of the note in the full melody. These features help the model learn timing and rhythmic structure. The duration model uses the same input format as the pitch model. For the bar profile model, the input consists of the bar profile itself (length 16), plus the bar parity and relative position features.

In our implementation, we use the same network for supervised learning and reinforcement learning. We adopt PPO with an LSTM architecture, which is often called Recurrent PPO [31]. To implement this setup, we include the LSTM hidden state in the input. When the environment transitions to the next state, we pass the hidden state to the next time step. During data collection, we also store the hidden state for training. Figure 7 shows this process. PPO captures long-term dependencies by maximizing reward over an episode. The LSTM further improves the memory of past information through its hidden state.

In our input representation, we use a single floating-point value to indicate the relative position of a note in the full melody. For example, the value 0.254 in Figure 8 indicates that the note occurs at about one-quarter of the melody. However, our experiments show that this feature is not effective for learning rules related to global structure, including the pitch and duration of the final note. One reason is that relative position does not reflect differences in melody length. For instance, the ending note of an 8-bar melody might occur at a relative position of 0.9. In a 32-bar melody, the same value appears too early to indicate the ending note.

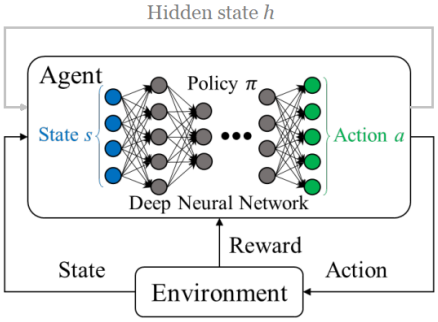
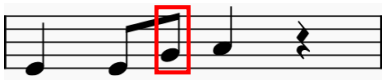


Figure 7. Recurrent PPO architecture with an LSTM hidden state passed across time steps



[0..1..0 0..1..0 0100 00001000 0 0.254 0..1..0]
(pitch duration beat frame odd/even pos profile)

Figure 8. Pitch model input for a single note, assuming the note in the red frame is in an odd-numbered bar with a relative position of 0.254

To address this limitation, we add absolute position information to the model. We adopt Length-Difference Positional Encoding (LDPE), which is a modified version of the Transformer positional encoding proposed in [32]. In the Transformer, the attention mechanism lacks built-in knowledge of sequence order. To address this issue, [30] introduced positional encoding, which maps each position to a vector of a chosen dimension. For example, if the input embedding has 16 dimensions, each position is mapped to a 16-dimensional vector. This approach is more efficient than using large one-hot vectors, especially for long sequences. The original positional encoding formula, $PE_{(pos,\ i)}$, is shown below. Here, i is the dimension index, pos is the position, and d is the total dimension.

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

(Source: [15, 32])

The method in [32] aims to control the length of the generated sequence. It treats the position relative to the end of the sequence as more important than the position relative to the beginning. This idea aligns with our melody generation task, where the ending position is more critical than the starting point. If the model does not learn this well, the melody may sound incomplete. The key difference between the original positional encoding and LDPE is how position is measured. LDPE encodes position based on the

distance to the end of the sequence. Therefore, positions with the same distance to the end share the same encoding. The LDPE formula is written as $LDPE_{(pos,\ len,\ i)}$, where *len* is the target sequence length.

$$LDPE_{(pos,2i)} = \sin\left(\frac{len - pos}{10000^{\frac{2i}{d}}}\right)$$

$$LDPE_{(pos,2i+1)} = \cos\left(\frac{len - pos}{10000^{\frac{2i}{d}}}\right)$$

(Source: [32])

In the original Transformer, positional encoding is added to the output of the embedding layer, and the result is then fed into the attention blocks, as shown in Figure 9. In our implementation, we do not use an embedding layer or attention blocks. Instead, we add the LDPE vector to the output of the first linear (fully connected) layer, and then pass the result to the second linear layer, as shown in Figure 10. In the rest of this paper, we use the term "positional encoding" to refer to LDPE as proposed in [32], which is the method used in our system.

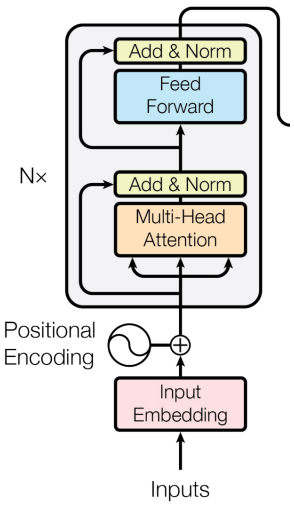


Figure 9. Positional encoding is added to the embedding output before the attention blocks in the Transformer encoder (Adapted from [15])

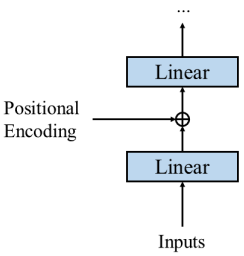


Figure 10. Data flow of adding LDPE to the output of the first linear layer in our network

Reward functions for melody generation are highly subjective. In our system, we design these functions based on input from music experts. Different music styles often require different reward rules. For example, although Taiwanese Opera and Hakka Folk Song are both influenced by traditional Chinese music, their reward rules are different. Designing these functions takes time and repeated adjustment. Frequent discussions between music experts and engineers are needed to ensure that the generated melodies are acceptable.

In traditional Chinese music, many melodies use hexatonic or heptatonic scales derived from the pentatonic scale. In the hexatonic scale, B is commonly used, while F is rarely used. The melody contour often moves up and down around central notes, forming a wave-like shape. Rhythm also changes with musical imagery and emotional expression, and it often shifts between simple and complex patterns.

The pitch, duration, and bar profile models each use their own reward functions. Some rewards are applied at every time step when a symbol is generated. Other rewards are applied at the end of a bar or at the end of the full melody. The following sections describe the reward rules for each model. A rule marked with (+) gives a positive score when it is satisfied, while a rule marked with (-) gives a negative score when it is violated. The reward values depend on the selected music style.

3.2 Reward Function for Pitch Model

The pitch model generates one pitch at each time step. To improve musical coherence and reflect features of traditional Chinese music, we define a set of rule-based rewards. Each rule gives a positive or negative score based on the pitch value, the interval between consecutive pitches, and the structure within each bar. We define:

- p_t: the pitch generated at time step t.
- i_t: the interval between the pitch at time step t and the pitch at time step t-1.
- *HR*: the set of all (first pitch, last pitch) pairs for each bar in the dataset.
- *RH*: the set of all (last pitch of a bar, first pitch of the next bar) pairs in the dataset.

These rules guide pitch generation so that the output follows the style patterns of traditional Chinese melodies. Most rules are derived from statistical trends in the dataset, while others are based on Chinese music theory. The rules focus on scale adherence, interval control, and phrase structure. Table 1 lists the reward rules for the pitch model. Each rule is evaluated when a pitch is generated. A positive score is given when the condition matches a desired pattern, and a negative score is given when the condition is violated. The "Condition" column describes each rule, and the "Reward" column indicates whether the behavior is encouraged or discouraged.

These rules are based on both music theory and empirical observations. First, most traditional Chinese music follows the pentatonic scale. Therefore, most pitches should stay within the five pentatonic tones, as stated in Rule 1. Second, to avoid unnatural pitch movement, we

limit interval size. Within a bar, the interval must not exceed 12 semitones (Rule 2). Across bar boundaries, the first pitch of a new bar can differ by up to 14 semitones from the last pitch of the previous bar (Rule 3). Third, the relationship between the first and last pitch in a bar is important for phrase structure. We define *HR* from the dataset to ensure that these pitch pairs resemble real phrases (Rule 8). Likewise, we use *RH* to encourage natural transitions between consecutive bars (Rule 4). Fourth, some interval patterns are uncommon in traditional Chinese music and are penalized. These include perfect fourths (5 semitones) and two consecutive third intervals (3 or 4 semitones) (Rules 5 and 6). In contrast, alternating major seconds and minor thirds is a common melodic pattern and is rewarded (Rule 7). Finally, because all melodies in our dataset are in the key of C, endings on pitches other than C, G, or A are discouraged (Rule 9).

Table 1. Reward rules for the pitch model

No.	Condition	Reward
1	p_t is not in the Chinese pentatonic scale (C, D, E, G, A)	(-)
2	p_t is not the first pitch of a bar, and interval $i_t > 12$ semitones	(-)
3	p_t is the first pitch of a bar, and interval from $p_{t-1} > 14$ semitones	(-)
4	p_t is the first pitch of a bar, and pair $(p_{t-1}, p_t) \notin RH$	(-)
5	Interval i_t is a perfect fourth (5 semitones)	(-)
6	Consecutive intervals i_t and i_{t-1} are both thirds (3 or 4 semitones)	(-)
7	The pair of intervals (i_{t-1}, i_t) is (2, 3) or (3, 2)	(+)
8	p_t is the last pitch of a bar, and pair (first pitch, p_t) $\notin HR$	(-)
9	The melody ends on a pitch other than C, G, or A	(-)

3.3 Reward Function for Duration Model

The duration model is designed to capture rhythmic structure and bar-level alignment. To evaluate the generated durations, we define a set of rule-based rewards. These rules guide the model to produce rhythm patterns that are musically appropriate and stylistically consistent. Table 2 lists the reward rules for the duration model.

These rules are based on musical intuition and statistical patterns in the dataset. Rule 1 ensures that the final note has enough duration to signal closure. A short ending note may not provide a clear sense of completion. This rule also supports positional encoding, which helps the model identify a note's relative position in the melody. Rule 2 maintains rhythmic alignment across bar pairs. In traditional Chinese music, rhythms are often simple and regular, and notes rarely extend across a bar line. Rules 3 and 4 control rhythmic repetition. Moderate repetition supports structure and familiarity, but excessive repetition reduces variation and can make the melody monotonous. Rule 5 discourages long durations for non-pentatonic notes, because they often serve as short passing tones and are not meant to be sustained [33]. Rule 6 prevents several long notes from appearing consecutively, which improves rhythmic variety and avoids overly static patterns.

Table 2. Reward rules for the duration model

No.	Condition	Reward
1	The ending note duration is shorter than a half note (16 frames)	(-)
2	The total duration of an odd bar and the next even bar is not exactly two whole notes (64 frames)	(-)
3	The current bar rhythm matches the previous bar, and the pattern has repeated no more than four times	(+)
4	The current bar rhythm matches the previous bar, and the pattern has repeated more than four times	(-)
5	The current pitch is not in the Chinese pentatonic scale, and its duration exceeds an eighth note (4 frames)	(-)
6	Three consecutive notes have durations equal to or longer than a half note (16 frames)	(-)

3.4 Reward Function for Bar Profile Model

The reward rules for the bar profile model are similar to those for the duration model. These rules complement the duration rules because bar profiles guide the high-level rhythm structure of each bar. Table 3 lists the reward rules for the bar profile model. These rules regulate repetition at the bar level. They allow moderate repetition while discouraging excessive reuse of the same bar profile.

Table 3. Reward rules for the bar profile model

No.	Condition	Reward
1	The ending note duration of the previous profile is shorter than a half note	(+)
2	The current profile matches the previous one, and the pattern has repeated no more than four times	(+)
3	The current profile matches the previous one, and the pattern has repeated more than four times	(-)

All models and experiments are implemented using the PyTorch library [34]. In our network, the input first passes through three fully connected layers, each with 128 neurons. We insert a positional encoding vector between the first and second layers. The positional encoding vector encodes the global position of the current note. The data then enters a stacked LSTM with three recurrent layers. Each LSTM layer has a hidden size of 128, so the hidden state tensor has dimensions 3 × 128. In this multi-layer LSTM, the output of each layer is fed into the next layer [35]. The LSTM output then passes through three additional fully connected layers with 128, 128, and 64 neurons, respectively. This produces the final model output. The input and output dimensions depend on each model, but the hidden-layer configuration is the same for all three models. Figure 11 shows the complete data flow. We use this architecture for both supervised learning and reinforcement learning.

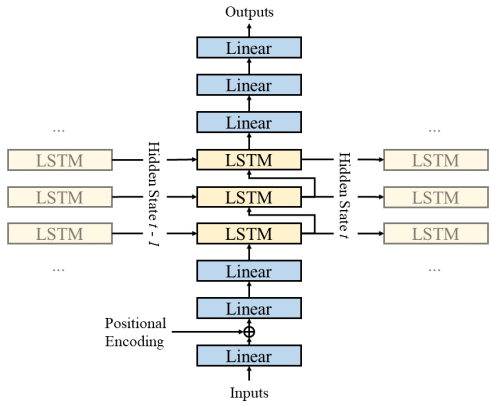


Figure 11. Data flow of the proposed network, including fully connected layers, positional encoding, and a three-layer LSTM

4 The Internet-based Music Generation System

This section describes the Internet-based tool that we developed for music generation. The tool includes the melody generation system and also supports accompaniment, harmonic structure, and percussion with various instruments. The system runs on a machine equipped with an RTX 4090 GPU and 24 GB of RAM. To balance computational cost and model performance, we combine deep learning with traditional algorithmic methods. The tool is designed for Asian music and currently supports five music styles:

1. Taiwanese Opera (also known as koa-a-hi in Taiwan)
2. Hakka Folk Song (from the Hakka ethnic group)
3. Traditional Jiangnan Style (from southeastern China)
4. Traditional Qin Style (from northwestern China)
5. Japanese Nakashi (a Taiwan-localized style influenced by Japanese music)

Figure 12 illustrates the system structure and the workflow for online music generation. The tool provides both English and Chinese user interfaces. It allows users to generate music online and to download music scores, MIDI files, and audio files.

On the left side of Figure 12, we show the datasets we collected. In the music database, we store music scores in an internal representation format. We also developed a tool that converts MusicXML files to this internal format and vice versa. The internal representation includes basic musical elements such as pitch and duration.

Since the number of original songs is limited, we apply data augmentation using Markov models to generate additional training data for the neural networks. To generate music notes, we first use a standard Hidden Markov Model. However, the melodies produced by this method do not always meet the desired musical quality. Therefore, the melody generation module using deep learning, as described in the previous section, is employed to improve musical coherence. We also use simulated annealing as an optimization method to produce skeletal melodies. For harmonic progression, which follows relatively simple rules, we apply a finite state machine.

On the right side of Figure 12, the system includes the deep learning modules. Each module is responsible for a different part of the music composition process. Because each component requires different training data and modeling strategies, the modules are trained separately. Together, the four modules form a complete music score.

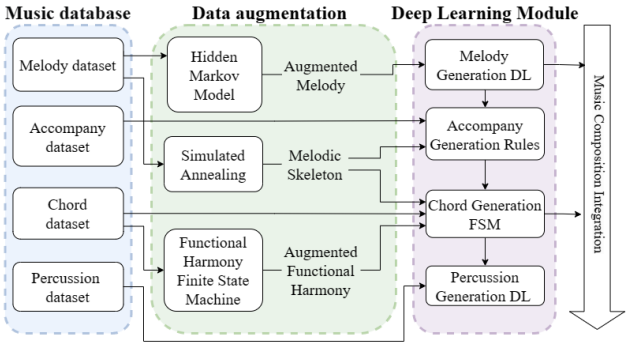


Figure 12. System components and workflow for online music generation

In the Proximal Policy Optimization (PPO) network, a key challenge is how to encode music rules in the reward function, especially rules for Asian music. Since the system supports five music styles, we define a separate set of reward functions for each style. The scale tones used in each style are as follows:

1. Taiwanese Opera: C, D, E, G, A
2. Hakka Folk Song: C, E, F, A, B
3. Traditional Jiangnan Style: C, D, E, G, A
4. Traditional Qin Style: F, A
5. Japanese Nakashi: C, E, F, A, B

We determine these scale tones through discussions with musicians for each style. We use them in our experiments, and users generally accept the generated results. Different musicians may suggest additional changes, but this is beyond the scope of this study.

For the Jiangnan and Taiwanese Opera styles, we apply additional rules. These rules prefer intervals such as major seconds and minor thirds. They also restrict large leaps, such as jumps greater than one octave. At the phrase level, traditional Asian music often uses specific starting and ending notes. The relation between these notes is important, but no fixed rule exists. Therefore, we use statistical patterns from our dataset. If the starting or ending notes deviate from common patterns, we apply a penalty. Other rules control the use frequency of scale tones and the number of repeated scale tones. We also add constraints on beat length and beat patterns within each bar, and we implement them as reward terms.

Note that scale, beat, and bar profile each have their own reward functions. We also observe dependencies between scale and bar profile, or between scale and sequence patterns, especially in the Traditional Qin Style. Since there are no strict universal rules, all rules must be tested and validated by experienced musicians. The deep learning module for melody generation includes a sequence generation component and a reinforcement learning component.

Our system provides Chinese dulcimer (yangqin) accompaniment based on traditional performance rules. The dulcimer part follows the rhythm and pitch of the main melody. Chord generation is also important. We use functional harmony to produce chord progressions that follow the melody and its structure. We assign a chord to each bar, or to a segment within a bar. The chord sequence follows a set of defined rules.

We propose a chord generation method based on a finite-state machine (FSM), which controls transitions between harmonic states. The FSM, shown in Figure 13, includes states that represent basic harmonic functions such as tonic, dominant, and subdominant. For simplicity, Figure 13 includes only eight basic chords. More complex FSMs can be built for advanced harmonic generation. In theory, the FSM can be non-deterministic because several chords may be suitable at a given point in the melody. We can prioritize the next chord using statistical analysis based on the characteristics of a given style. More advanced FSMs can also reflect different emotional expressions and phrase structures.

Asian percussion instruments differ from those in Western music. They include drums, cymbals, gongs, wooden fish, bangzi, and other instruments. Their timbre is important in styles such as Jiangnan and Taiwanese Opera. Because standard MIDI does not include these sounds, we worked with musicians to record percussion samples and built custom soundfonts for our system. Each music style has its own rhythm patterns. Based on suggestions from musicians, we use a probability function to select patterns that match each generated melody. To improve consistency, we trained a deep learning model with a temporal convolutional network and attention modules. Percussion data design remains subjective and is guided by expert input.

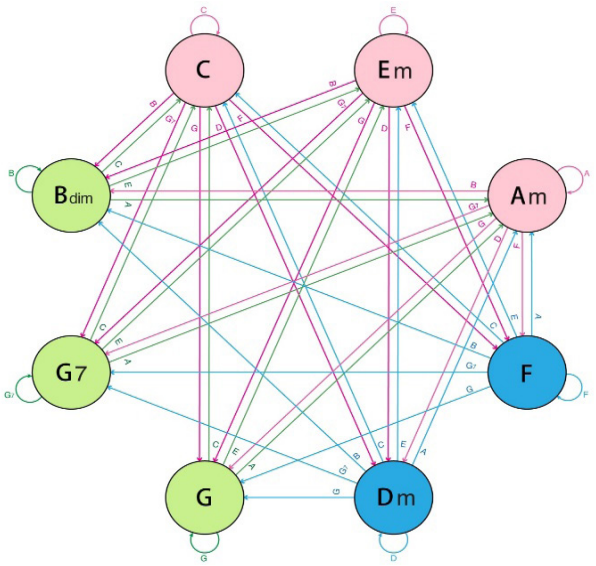


Figure 13. Proposed chord-generation finite-state machine that currently uses eight chords and selects the next state at random based on a probability model

We developed a web-based tool for public use, as shown in Figure 14. It offers three music creation options. First, users can select a sample melody. Each style includes 200 to 300 songs. Musicians selected these songs, and music students entered them. Second, users can generate music with a deep learning model by setting the tempo and length. The model generates one bar at a time, so the output has no fixed length. Third, users can generate music from text. This option targets users without prior music training. The tool supports five Asian styles and provides both English and Chinese interfaces. Users can explore and compare these styles. Figure 15 shows an example output.

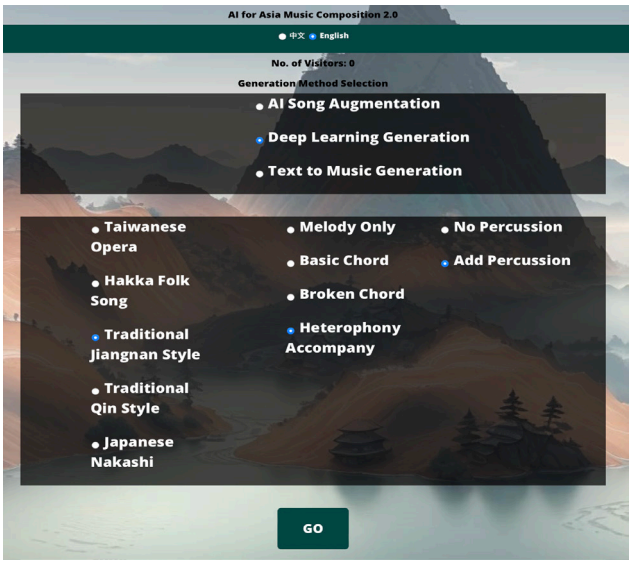


Figure 14. User interface of the web-based tool for Asian music generation

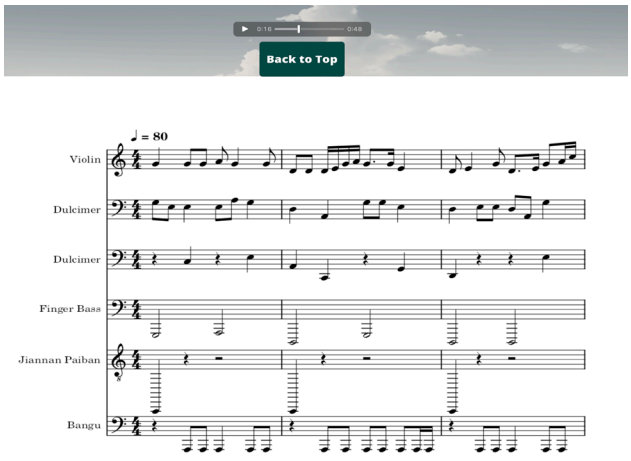


Figure 15. Example of a music score generated by the system

After the melody is generated, the system provides either functional harmonic accompaniment or heterophonic accompaniment. These two types of accompaniment cannot be used together. Functional harmony can produce either basic chords or broken chords. For percussion, we use both standard MIDI-based synthesized instruments and sampled percussion sounds.

5 Conclusion and Future Work

In this paper, we presented a deep learning-based music generation system. The system combines supervised learning and reinforcement learning to generate melodies in five Asian music styles. Inspired by prior work on Jiangnan music, we extended the system with style-specific reward functions and selected training data. The generated music reflects both statistical patterns in the dataset and human-defined musical rules.

We developed a web-based tool that allows users to generate music online. The tool supports score download, MIDI export, and audio playback. The system currently supports five styles: Taiwanese Opera, Hakka Folk Song, Jiangnan Style, Qin Style, and Japanese Nakashi. We selected these styles for their cultural significance and distinct musical features. We implemented the system in PyTorch on a machine with an RTX 4090 GPU. The model comprises three neural modules for pitch, duration, and bar profile generation. Each module follows style-specific reward rules designed with music experts. We apply reinforcement learning using PPO with an LSTM to ensure that the melody respects musical structure.

Experimental results show that the generated melodies are stylistically consistent. They are also well received by both general users and professional musicians. The system is publicly accessible. It demonstrates how symbolic music generation can combine AI methods with cultural knowledge.

In future work, we aim to improve timbre quality beyond the standard MIDI set. We plan to enhance rhythm and accompaniment features. We also intend to support additional music styles. Furthermore, we will integrate Chinese percussion instruments and more advanced harmonic structures. We continue to work with the music industry and welcome further academic and industrial collaboration. The system is available at http://140.115.51.218:8888.

References

[1] C.-H. Huang, *Combining Deep Supervised Learning and Reinforcement Learning for Music Melody Generation*, M.S. Thesis, National Central University, Taoyuan, Taiwan, 2023.

[2] P. M. Todd, A connectionist approach to algorithmic composition, *Computer Music Journal*, Vol. 13, No. 4, pp. 27-43, Winter, 1989. https://doi.org/10.2307/3679551

[3] M. C. Mozer, Neural Network Music Composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing, *Connection Science*, Vol. 6, No. 2-3, pp. 247-280, January, 1994. https://doi.org/10.1080/09540099408915726

[4] J.-P. Briot, G. Hadjeres, F. Pachet, *Deep Learning Techniques for Music Generation*, Springer International Publishing, 2019.

[5] A. Graves, Generating sequences with recurrent neural networks, *arXiv preprint*, arXiv:1308.0850, August, 2013. https://arxiv.org/abs/1308.0850

[6] D. Eck, J. Schmidhuber, *A First Look at Music Composition Using LSTM Recurrent Neural Networks*, Technical Report No. IDSIA-07-02, January, 2002.

[7] G. Hadjeres, F. Pachet, F. Nielsen, DeepBach: A Steerable Model for Bach Chorales Generation, *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, Sydney, NSW, Australia, 2017, pp. 1362-1371.

[8] S. Oore, I. Simon, S. Dieleman, D. Eck, K. Simonyan, This Time with Feeling: Learning Expressive Musical Performance, *Neural Computing and Applications*, Vol. 32, No. 4, pp. 955-967, February, 2020. https://doi.org/10.1007/s00521-018-3758-9

[9] J. Luo, X. Yang, S. Ji, J. Li, MG-VAE: Deep Chinese Folk Songs Generation with Specific Regional Style, *Proceedings of the 7th Conference on Sound and Music Technology (CSMT)*, Harbin, China, 2019, pp. 93-106. https://doi.org/10.1007/978-981-15-2756-2_8

[10] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, *Proceedings of the 27th International Conference on Neural Information Processing Systems (NeurIPS 2014)*, Montreal, Quebec, Canada, 2014, pp. 2672-2680.

[11] O. Mogren, C-RNN-GAN: Continuous Recurrent Neural Networks with Adversarial Training, *Constructive Machine Learning Workshop at Neural Information Processing Systems (NIPS 2016)*, Barcelona, Spain, 2016, pp. 1-8.

[12] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, Y.-H. Yang, Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment, *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-18)*, New Orleans, LA, USA, 2018, pp. 34-41.

[13] G. Lavrentiadis, N. A. Abrahamson, N. M. Kuehn, A non-ergodic effective amplitude ground-motion model for California, *Bulletin of Earthquake Engineering*, Vol. 21,

No. 11, pp. 5233-5264, September, 2023.
https://doi.org/10.1007/s10518-021-01206-w

[14] B. Yu, P. Lu, R. Wang, W. Hu, X. Tan, W. Ye, S. Zhang, T. Qin, T.-Y. Liu, Museformer: Transformer with Fine- and Coarse-Grained Attention for Music Generation, *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022)*, New Orleans, LA, USA, 2022, pp. 1376-1388.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention Is All You Need, *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS 2017)*, Long Beach, CA, USA, 2017, pp. 6000-6010.

[16] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, D. Eck, Music Transformer: Generating Music with Long-Term Structure, *International Conference on Learning Representations (ICLR 2019)*, New Orleans, LA, USA, 2019, pp. 1-15.

[17] Magenta, https://magenta.tensorflow.org/ (accessed Jan. 5, 2026).

[18] G. Mittal, J. Engel, C. Hawthorne, I. Simon, Symbolic Music Generation with Diffusion Models, *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR 2021)*, Online, 2021, pp. 468-475.

[19] M. Plasser, S. Peter, G. Widmer, Discrete Diffusion Probabilistic Models for Symbolic Music Generation, *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI-23)*, Macao, China, 2023, pp. 5842-5850.
https://doi.org/10.24963/ijcai.2023/648

[20] G. Hadjeres, L. Crestel, The Piano Inpainting Application, *arXiv preprint*, arXiv:2107.05944, July, 2021.
https://arxiv.org/abs/2107.05944

[21] N. Jaques, S. Gu, R. E. Turner, D. Eck, Tuning Recurrent Neural Networks with Reinforcement Learning, *International Conference on Learning Representations (ICLR 2017) Workshop*, Toulon, France, 2017, pp. 1-13.

[22] H. van Hasselt, A. Guez, D. Silver, Deep Reinforcement Learning with Double Q-learning, *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-16)*, Phoenix, AZ, USA, 2016, pp. 2094-2100.

[23] N. Jiang, S. Jin, Z. Duan, C. Zhang, RL-Duet: Online Music Accompaniment Generation Using Deep Reinforcement Learning, *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-20)*, New York, NY, USA, 2020, pp. 710-718.

[24] D. Lafleur, S. Chandar, G. Pesant, Combining Reinforcement Learning and Constraint Programming for Sequence-Generation Tasks with Hard Constraints, *Proceedings of the 28th International Conference on Principles and Practice of Constraint Programming (CP 2022)*, Haifa, Israel, 2022, pp. 30:1-30:16.

[25] H. Kumar, B. Ravindran, Polyphonic Music Composition with LSTM Neural Networks and Reinforcement Learning, *arXiv preprint*, arXiv:1902.01973, March, 2019.
https://arxiv.org/abs/1902.01973

[26] S. Ji, X. Yang, J. Luo, J. Li, RL-Chord: CLSTM-Based Melody Harmonization Using Deep Reinforcement Learning, *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 35, No. 8, pp. 11128-11141, August, 2024.
https://doi.org/10.1109/TNNLS.2023.3248793

[27] Y. Quan, S. Yue, B. Liao, Impact of electron-phonon interaction on thermal transport: A review, *Nanoscale and Microscale Thermophysical Engineering*, Vol. 25, No. 2, pp. 73-90, April, 2021.
https://doi.org/10.1080/15567265.2021.1902441

[28] L. Zhang, C. Callison-Burch, Language models are drummers: Drum composition with natural language pre-training, *Proceedings of the AAAI 2023 Workshop on Creative AI Across Modalities*, Washington, DC, USA, 2023, pp. 1-8.

[29] J. Wu, C. Hu, Y. Wang, X. Hu, J. Zhu, A Hierarchical Recurrent Neural Network for Symbolic Melody Generation, *IEEE Transactions on Cybernetics*, Vol. 50, No. 6, pp. 2749-2757, June, 2020.
https://doi.org/10.1109/TCYB.2019.2953194

[30] Z. Sun, J. Liu, Z. Zhang, J. Chen, Z. Huo, C. H. Lee, X. Zhang, Composing Music with Grammar Argumented Neural Networks and Note-Level Encoding, *Proceedings of the APSIPA Annual Summit and Conference (APSIPA ASC)*, Honolulu, HI, USA, 2018, pp. 1864-1867.

[31] Recurrent PPO - Stable Baselines3 Contrib Documentation, https://sb3-contrib.readthedocs.io/en/master/modules/ppo_recurrent.html (accessed Jan. 5, 2026).

[32] S. Takase, N. Okazaki, Positional Encoding to Control Output Sequence Length, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Minneapolis, MN, USA, 2019, pp. 3999-4004.
https://doi.org/10.18653/v1/N19-1401

[33] B. Benward, M. N. Saker, *Music in Theory and Practice, Vol. 2*, McGraw-Hill Higher Education, 2008.

[34] PyTorch Foundation, https://pytorch.org/ (accessed Jan. 5, 2026).

[35] PyTorch Foundation, LSTM - PyTorch Documentation, https://docs.pytorch.org/docs/stable/generated/torch.nn.LSTM.html (accessed Jan. 5, 2026).

Biographies

Cheng-Han Wu holds dual B.S. degrees in Computer Science and Chinese Music from Chinese Culture University and an M.S. in Business Management from National Central University. He is a Ph.D. student in Computer Science and Information Engineering at National Central University, focusing on AI techniques for computer music.

Timothy K. Shih is a Distinguished Professor of Computer Science and Information Engineering at National Central University, Taiwan. He is founding co-editor-in-chief of the International Journal of Distance Education Technologies and has served on several IEEE and ACM editorial boards. His honors include research awards and best paper prizes.

Chien-Hao Huang received a B.S. in Computer Science and Information Engineering from National Chung Cheng University and an M.S. in Computer Science and Information Engineering from National Central University, Taiwan. His M.S. thesis focused on music generation using deep learning techniques.

Yu-Cheng Lin received his M.S. in Information Computer Engineering (1997) and Ph.D. in Electronic Engineering (2001) from Chung Yuan Christian University, Taoyuan, Taiwan. He is an assistant professor in Computer Science and Engineering at Yuan Ze University. His research interests include EDA, physical design, VLSI CAD algorithms, and AI music.