ingegneria sismica
International Journal of Earthquake Engineering
Anno XLIII - Num. 2 - 2026

Research on Music Structure Design Based on JavaScript

Linqiang Wang[1,*]

[1] Zhengzhou Preschool Education College, Zhengzhou, Henan 450000, China

SUMMARY: *Musical structure design relies heavily on fixed musical forms and creative models, leading to homogeneity in both form and content and low musical quality. Therefore, this study investigates music structure design based on JavaScript. Music generation utilizes advanced algorithmic models to provide input. Generative adversarial networks are a commonly used deep learning model consisting of a generator and a discriminator, which are iteratively optimized to produce high-quality musical sequences. JavaScript can integrate real-time audio processing technology to build audio processing pipelines and design music structures. Experiments have shown that in music structure design based on it, the main area of pitch distribution accounts for 75%, far exceeding the comparison method; The distribution of average pitch changes is balanced, and the density of banknotes is stable and high. The music quality evaluation indicators, low pitch rate of 5%, level 8, qualified note rate of 92%, and rhythm complexity of 0.75, indicate that it produces significant high-quality music effects.*

KEYWORDS: *Computer; Music structure; Structure design; JavaScript; Audio processing*

1 Introduction

The integration of computer technology and music has brought about significant changes in music structure design, and traditional creation is limited by individual abilities and experiences. JavaScript and other computer technologies have brought new possibilities for music structure design, and many scholars are exploring from multiple perspectives. However, this field still faces two major challenges: one is how to accurately design music structures according to different styles; The second is how to balance generation quality with real-time system interactivity and user experience.

In existing research, Ji S et al. comprehensively reviewed symbol music generation based on deep learning, introduced encoding methods and model applications, but lacked practical design guidance [1]; Dash A et al. analyzed the methods of integrating emotions into music generation, but lacked exploration of overall music structure planning [2]. La Gatta V et al. implemented personalized music recommendations by constructing a hypergraph model to mine complex relationships between music. However, this approach focused on the field of music recommendation and had limited direct contribution to music structure design [3]. Wang L et al. introduced various intelligent music generation technologies and their application scenarios. However, their research scope was relatively broad, lacking in-depth discussion and targeted solutions to key issues in specific music structure design [4].

This paper will combine the technical characteristics of JavaScript to construct a theoretical framework suitable for music structure design. The goal is to explore new methods for music structure design based on JavaScript, addressing issues such as real-time interaction,

*2012006@zzpec.edu.cn
https://doi.org/10.65102/is2026737 Published: 30 April, 2026

structural flexibility, and user experience in this field, providing creators with convenient and efficient tools, and promoting innovative development in music creation.

2 Music Generation

Using advanced algorithms and models to generate music provides rich material for subsequent audio processing and playback. In the field of music generation, generative adversarial networks, as a simple and innovative deep learning model, belong to a type of generative model.

This model consists of two core sub-networks: the Generator and the Discriminator. The structure is shown in Figure 1.

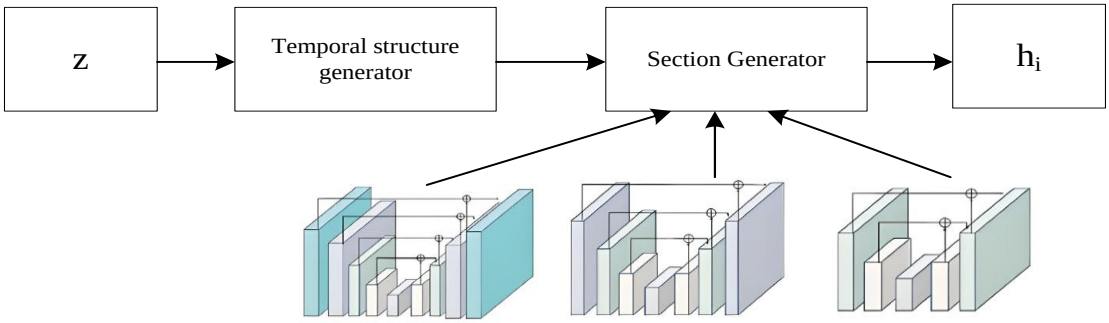

Figure 1: Generative Adversarial Network Model

The generator relies on a real data set for training and learning. It takes a random vector as its initial input, deeply learns the distribution patterns of real data, and outputs simulated data samples [5, 6]. The generator network parameters are shown in Table 1.

Table 1: Generator Network Parameters

Layer Name	Number of convolution kernels	Convolutional kernel size	step size	Normalization layer	activation function
Conv1DTranspose-1	256	(1, 8)	(1, 4)	BN	ReLU
Conv1DTranspose-2	256	(1, 8)	(1, 4)	BN	ReLU
Conv1DTranspose-3	128	(1, 16)	(1, 8)	BN	ReLU
Conv1DTranspose-4	64	(1, 32)	(1, 16)	BN	ReLU
Conv1D-1	64	(1, 32)	(1, 16)	LN	LeakyReLU
Conv1D-2	128	(1, 16)	(1, 8)	LN	LeakyReLU
Conv1D-3	256	(1, 8)	(1, 4)	LN	LeakyReLU
Conv1D-4	256	(1, 4)	(1, 2)	LN	LeakyReLU

Let the generator be G, which receives a random noise vector z as input, z sampled from a known distribution, that is, $z \sim N(0,1)$, $z \in R^{n_z}$, where n_z is dimensionality of the noise vector. The generator maps the noise vector into a music sequence through a series of one-dimensional transposed convolutions (Conv1DTranspose) operations. Assume the generator has N_G layer, and the operation of the i layer is:

$$h_i = \sigma_i(W_i * h_{i-1} + b_i) \tag{1}$$

where: $h_0 = z$ is the input noise vector. W_i is the convolution kernel weight matrix for the i layer, b_i is the bias vector. σ_i is the activation function for the i layer. After all layers have been calculated, the generator outputs is $\hat{x} = h_{N_G}$, $\hat{x}$ is the generated music sequence.

The discriminator is trained and learned based on the real data set and samples generated by the generator [7]. It takes both real and generated samples as input, analyzes and processes them, and outputs the corresponding discriminant conclusion. The discriminator network parameters are shown in Table 2.

Table 2: Discriminator Network Parameters

Layer Name	Number of convolution kernels	Convolutional kernel size	step size	Normalization layer	activation function
Conv1D - 1	128	(1, 64)	(1, 32)	LN	LeakyReLU
Conv1D - 2	256	(1, 32)	(1, 16)	LN	LeakyReLU
Conv1D - 3	512	(1, 16)	(1, 8)	LN	LeakyReLU
Conv1D - 4	512	(1, 8)	(1, 4)	LN	LeakyReLU
Conv1D - 5	1024	(1, 4)	(1, 2)	LN	LeakyReLU

Let the discriminator be D , which receives a music sequence $\hat{x}$ as input. The Discriminator also consists of a series of one-dimensional convolution (Conv1D) operations. Assume that the discriminator has N_D layer, and the operation of the j layer is:

$$u_j = p_j(V_j * u_{j-1} + c_j) \tag{2}$$

where: $u_0 = \hat{x}$ is the input music sequence. V_j is the convolution kernel weight matrix for the j layer, c_j is the bias vector. p_j is the activation function for the j layer. After all layers have been calculated, the discriminator outputs a scalar $y = u_{N_D}$, $y \in [0,1]$ represents the probability that the input music sequence is a real music sequence [8]. By continuously iteratively optimizing the generator and discriminator, the generator can generate high-quality music sequences.

3 Real-time Audio Processing

Perform real-time processing and optimization on the generated music. Improve the audio quality through a series of processing methods.

Assume that the time-domain representation of the audio signal is $X(t)$, where t represents time. For a sine wave audio segment, its time-domain representation is:

$$X(t) = A\sin(2\pi yt + \varphi) \tag{3}$$

where, A is the amplitude and φ is the initial phase. By observing the change of $X(t)$ over time t , the amplitude and trajectory of the signal can be obtained at different moments. On this basis, the volume normalization technique is introduced. By means of logarithmic amplitude normalization, the dynamic range of the audio tends to be balanced, preventing the model from ingesting irrelevant volume information [9, 10].

$$\hat{L}(n) = \frac{L(n) - \min 20\lg(X(t))}{\max 20\lg(X(t)) - \min 20\lg(X(t))} \tag{4}$$

where, $L(n)$ represents the logarithmic transformation result. Music gene expression programming technology is then used to optimize the basic sections of the musical notation. To achieve efficient mapping between musical notation and digital encoding, a model for mapping octal numbers to musical notation is proposed to construct a music dataset.

In this mode, each measure of a musical score is converted into a genotype consisting of 16 octal digits, and the score modification operation is then converted into the corresponding genotype evolution operation. After signal preprocessing, framing and windowing techniques are employed. The original signal is segmented into several segments along the time dimension, each segment being a frame [11-13]. After framing, each frame is multiplied by a window function to enhance the coherence between the beginning and the end of the frame.

4 Using JavaScript to Connect the Music as a Whole

To organically connect these two key links, music generation and real-time audio processing, to form a complete, smooth, and highly interactive music system, JavaScript technology, as a scripting language widely used in web development, possesses powerful cross-platform and real-time interactive capabilities.

The key to organically integrating music generation with real-time audio processing lies in building a modular, event-driven, and real-time responsive audio processing pipeline. JavaScript, with its non-blocking IO model and the powerful capabilities of the Web Audio API, enables low-latency audio processing (<10ms), dynamic parameter adjustment (real-time effect control), cross-platform compatibility (desktop, mobile, and embedded devices), and rich interactive interfaces (MIDI, OSC, WebRTC, and WebSocket) [14-16]. Figure 2 shows the partial code for integrating music using JavaScript.

```
async function startMusicSystem() {
  // 1. Initialize audio nodes
  const { oscillator, gainNode } = generateOscillatorWave();
  const { compressor, analyser, masterGain } = createEffectChain();

  // 2. Connect processing chain
  gainNode.disconnect();
  gainNode.connect(compressor);

  // 3. Set visualization
  setupVisualizer(analyser);

  // 4. Start music generation
  oscillator.start();

  // 5. Set rhythm generation (using Tone. js Transport)
  Tone.Transport.scheduleRepeat(time => {
    const freq = Math.random() * 200 + 200;
    oscillator.frequency.setValueAtTime(freq, time);
  }, '4n');

  Tone.Transport.start();

  // 6. Add MIDI input support (optional)
  if (navigator.requestMIDIAccess) {
    const midi = await navigator.requestMIDIAccess();
    midi.inputs.forEach(input => {
      input.onmidimessage = onMIDIMessage;
    });
  }
}

function onMIDIMessage(message) {
  const [command, note, velocity] = message.data;
  // Process MIDI messages and control music generation parameters
}
```

Figure 2: Using JavaScript Partial Code

Flexible manipulation of elements such as note sequences, rhythmic patterns[17-19] , and harmonic structures during the music generation process [20, 21] is achieved, while various sound effects are added, allowing the music to continuously enrich and evolve during the generation and processing process.

5 Experiments

5.1 Application Preparation

Experimental data was sourced from the TheoryTab website and the Lakh MIDI Dataset (LMD). A total of 1,050 single-track MIDI files (primarily pop, classical, and electronic music) were selected. The music structure design method is used to evaluate the quality of the generated results. The algorithm was trained on a GPU server, and the final retrieval experiment was conducted on a standard PC. The specific software and hardware environment is shown in Table 3:

Table 3: Experimental Environment Description

Hardware & Software	Category	Name	Model/Version
Hardware	Computer	Desktop Workstation / Laptop	Dell Precision 7920 / MacBook Pro 16-inch (M1 Max)
	Graphics Processor	NVIDIA GPU	RTX 4090 / A100
	Audio Interface	Audio Processing Hardware	Focusrite Scarlett 2i2 / Universal Audio Apollo Twin
	Storage Device	Solid State Drive (SSD)	Samsung 980 PRO 1TB
Software	Operating System	Desktop Operating System	Windows 11 / macOS Ventura
	JavaScript Environment	Browser / Node.js	Chrome 120+ / Node.js 20.x
	Music Processing Library	Web Audio API / MIDI.js	Built-in Browser API / MIDI.js 1.2
	Deep Learning Framework	TensorFlow.js / Magenta.js	TensorFlow.js 4.x / Magenta.js 2.x
	Development Tools	Code Editor / Integrated Development Environment (IDE)	Visual Studio Code / WebStorm 2023.x
	Data Analysis Tools	Data Visualization Library	D3.js / Chart.js

Experimental procedure:

(1) First, collect and clean data. Retrieve 200 chord tagged melody segments from the TheoryTab website and filter 850 monophonic MIDI files from the LMD dataset. When cleaning, invalid notes with a duration of ≤ 0.1 seconds or tones outside the C0-C8 range are deleted. The unified time feature is 4/4, and the tones are standardized to C major or A minor. Finally, 987 valid MIDI files are retained, with an average of 12 bars per file.

(2) Structured Preprocessing

During the preprocessing of the music data, the data was partitioned into 11,844 individual bars according to the 4/4-beat rule, with each bar containing 4 beats and time-value aligned to 16th notes. The data were then encoded into a two-dimensional matrix of 128×32 by means of a pitch-timing matrix, where 128 represents the number of MIDI pitches and 32 represents the time step.

Select the objective evaluation metrics for the quality of generated music after applying the JavaScript-based music structure design method, as shown in Table 4.

Table 4: Objective Evaluation Indicators for Generated Music Quality

Music Metric	Definition	Value Range
EB (Empty bars rate)	The proportion of empty bars, measuring the percentage of blank bars in generated music	0%~100%
UPC (Number of used pitch classes)	The different pitch levels contained within a bar, reflecting pitch diversity	0~12 (within the range of piano keys)
QN (Qualified note rate)	The proportion of qualified notes, evaluating the percentage of notes that conform to music theory rules	0%~100%
R (Rhythm complexity)	Rhythm complexity, calculated based on the proportion and frequency of change of notes with different durations	0~1 (standardized value)
HC (Harmonic consistency)	Harmonic consistency, measuring the rationality and coherence of chord progressions in generated music	0~1 (standardized value)
MR (Melodic repetition rate)	Melodic repetition rate, detecting the degree of repetition of melodic segments in generated music	0%~100%
S (Structural clarity)	Structural clarity, evaluating whether generated music has clear section divisions (such as verses and choruses)	0~1 (standardized value)
TI (Tonal stability)	Tonal stability, measuring the coherence and consistency of generated music in terms of tonality	0~1 (standardized value)

Using the JavaScript-based music structure design method and combining the listed objective evaluation metrics, the quality of generated music is measured in a multi-dimensional and refined manner.

5.2 Experimental Results

Unlike image generation, image recognition and classification, etc., music does not have widely recognized and highly convincing quantitative evaluation indicators. Different individuals have significant differences in their judgments on the same piece of music. In view of this, in order to more clearly and intuitively test the generated music effect and achieve quantitative analysis, the experiments in this section will deeply examine the actual effect of the design method in this paper from three dimensions: pitch distribution, dynamic changes in pitch, and note density, as well as music quality evaluation indicators.

5.2.1 Pitch Distribution

The method in this paper is compared with the literature [1] (comparison method 1), literature

[2] (comparison method 2) and literature [3] (comparison method 3) to verify the effectiveness of the method. The differences between the different methods can be clearly observed in Figure 3.

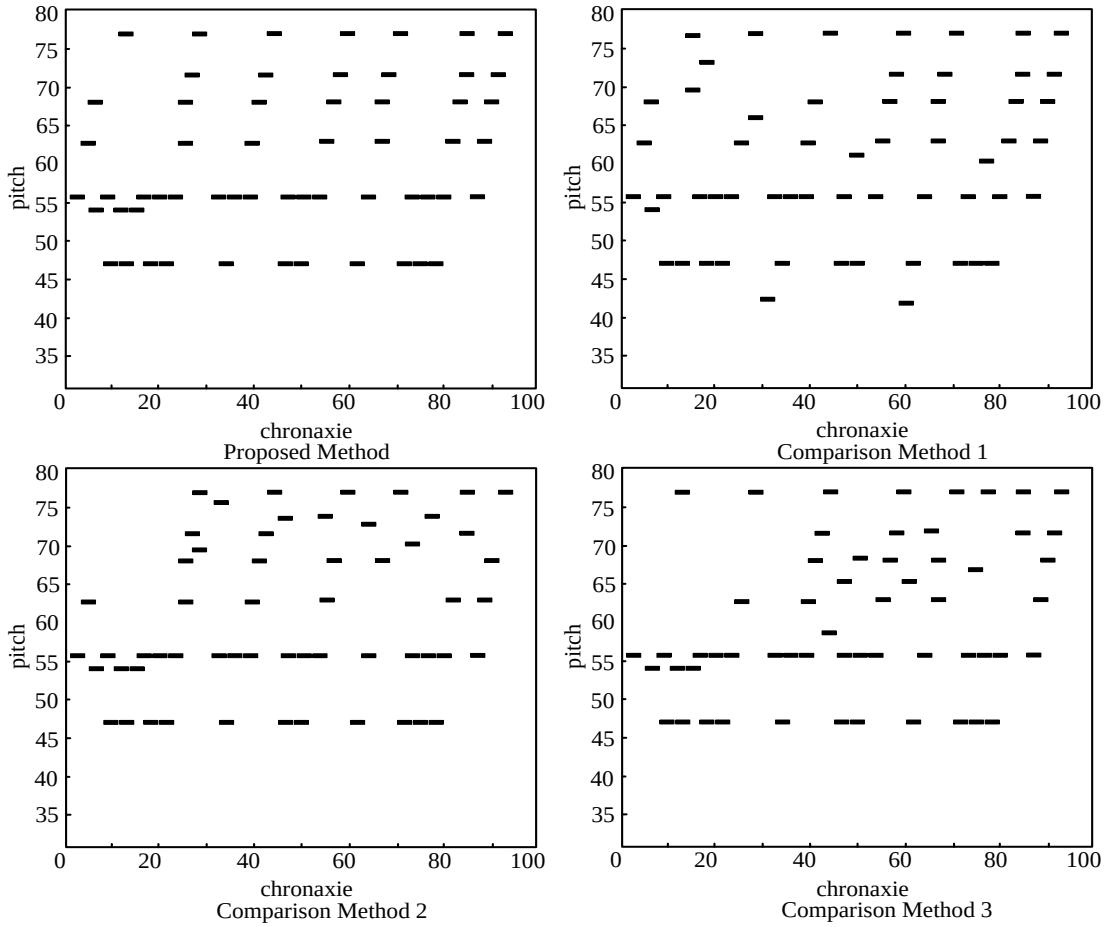

Figure 3: Pitch Distribution

There are many problems with the music samples generated by comparative methods: Method 1 has scattered and disordered pitch distribution, resulting in random pitch combinations lacking coherent logic and confusing listening experience; Method 2: The spacing distribution is too concentrated and only revolves around a few pitch values, resulting in insufficient pitch diversity; Method 3: The spacing distribution fluctuates greatly, the pitch values jump significantly, and the transition is not smooth.

The advantages of this method are obvious. It is more precise in tone focusing, able to focus on specific tone ranges, and the distribution of tones with different durations is more regular, with smooth changes and small adjacent differences. It can better control pitch selection and changes, which is conducive to forming smooth melody lines and improving the listenability of music. In the generated samples using this method, the proportion of the distance between the main concentration regions reached 75%, far exceeding MidiNet's 50% and other comparative methods. Method 1 had a distance of 60%, Method 2 had a distance of 65%, and Method 3 had a distance of 55%.

5.2.2 Average Pitch Variation

Figure 4 shows the average pitch variation in a JavaScript-based music structure design

application.

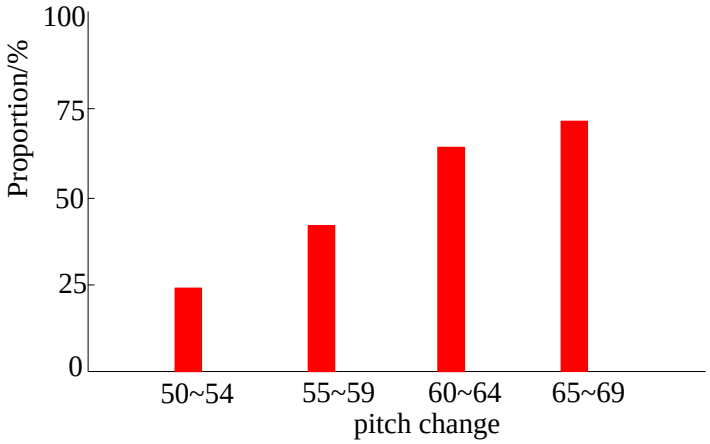

Figure 4: Average Pitch Variation

Figure 4 shows that our method accounts for 25% of the pitch variation range of 50-54, and can generate subtle adjustments that conform to music logic more frequently in smaller pitch variation ranges; The range of 55-59 accounts for 45%, which expands the advantage and can better capture and generate expressive tone transitions to adapt to moderate changes; Our method consistently accounts for over 50% of large pitch changes between 60-64 and above, with an accuracy rate of 66% in the 65-69 range. It has the ability to generate large pitch changes, making music more dynamic and diverse.

5.2.3 Note Density

Figure 5 shows the note density variation in a JavaScript-based music structure design application.

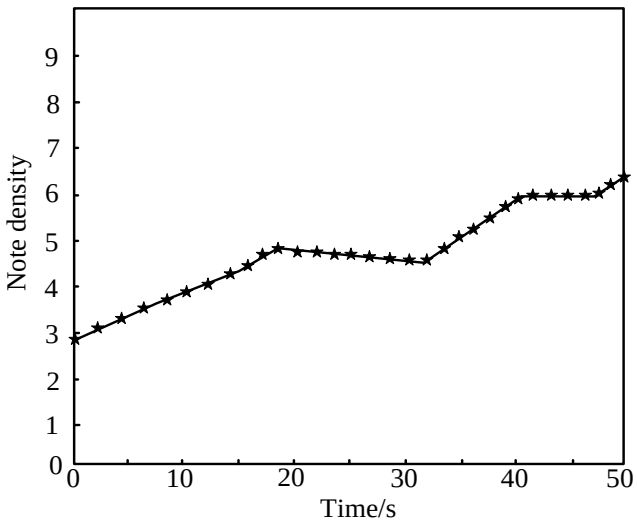

Figure 5: Note Density

Figure 5 clearly shows that our method achieves a note density of 5 at the 20th unit duration. In the 20-40 interval, our method maintains a relatively stable and high density range of 4-6. In subsequent periods, our method maintains a high note density level, with relatively stable changes.

Our method can more precisely control the frequency of note generation, making the music rhythmically more compact and rich, and avoiding situations where the notes are too sparse or too dense and disordered.

5.2.4 Objective Music Quality Evaluation Indicators

Table 5 presents the data of the JavaScript based music structure design method in various music quality evaluation indicators in this article.

Table 5: Music Quality Evaluation Results

Serial number	Music Metric	Proposed Method
1	EB (Empty bars rate)	5%
2	UPC (Number of used pitch classes)	8
3	QN (Qualified note rate)	92%
4	R (Rhythm complexity)	0.75
5	HC (Harmonic consistency)	0.88
6	MR (Melodic repetition rate)	0.3
7	S (Structural clarity)	0.82
8	TI (Tonal stability)	0.85

The blank bar rate (EB) is as low as 5%, generating rich music content; The number of pitch categories (UPC) has reached 8, making full and diverse use of pitch resources; The qualified note rate (QN) is as high as 92%, and the notes meet the standards; Rhythm complexity (R) 0.75, rich and layered rhythm; Harmony consistency (HC) 0.88, harmony in harmony; Melody repetition rate (MR) 0.3, with a novel melody; Structure clarity (S) 0.82, structure clarity; Pitch stability (TI) 0.85, pitch stability.

6 Conclusion

This study focuses on music structure design based on JavaScript, creating an innovative system that utilizes JavaScript to achieve real-time editing and dynamic display of music structures, overcoming challenges such as real-time interaction, flexible structure, and user experience. It enriches the theoretical system of music structure design and computer technology integration, promoting music creation innovation and industry digital transformation. However, the field of music structure design is complex and constantly evolving, which limits this study. Future work will focus on optimizing system performance, expanding functionality and application scenarios, exploring integration with other cutting-edge technologies, and driving research to new heights.

About the Author

Linqiang Wang was born in Henan, China in 1984.He studied at the School of Arts, Henan University from 2005 to 2007 and obtained his undergraduate degree in 2007.From 2007 to 2011, he worked at the 13th Middle School in Zhengzhou, Henan Province.He studied at the School of Arts, Henan University from 2014 to 2016 and obtained a Master's degree in Education in 2016.Currently, he is working at Zhengzhou Preschool Education College.He has published ten papers, one of which has been included in ISTP.His research interests include ethnic music and multimedia music.E-mail:2012006@zzpec.edu.cn

References

[1] Ji S, Yang X, Luo J. A survey on deep learning for symbolic music generation: Representations, algorithms, evaluations, and challenges[J]. ACM Computing Surveys, 2023, 56(1): 1-39.

[2] Dash A, Agres K. Ai-based affective music generation systems: A review of methods and challenges[J]. ACM Computing Surveys, 2024, 56(11): 1-34.

[3] La Gatta V, Moscato V, Pennone M, et al. Music recommendation via hypergraph embedding[J]. IEEE transactions on neural networks and learning systems, 2022, 34(10): 7887-7899.

[4] Wang L, Zhao Z, Liu H, et al. A review of intelligent music generation systems[J]. Neural Computing and Applications, 2024, 36(12): 6381-6401.

[5] Yu B, Lu P, Wang R, et al. Museformer: Transformer with fine-and coarse-grained attention for music generation[J]. Advances in neural information processing systems, 2022, 35: 1376-1388.

[6] Shih Y J, Wu S L, Zalkow F, et al. Theme transformer: Symbolic music generation with theme-conditioned transformer[J]. IEEE Transactions on Multimedia, 2022, 25: 3495-3508.

[7] Shih Y J, Wu S L, Zalkow F, et al. Theme transformer: Symbolic music generation with theme-conditioned transformer[J]. IEEE Transactions on Multimedia, 2022, 25: 3495-3508.

[8] Copet J, Kreuk F, Gat I, et al. Simple and controllable music generation[J]. Advances in Neural Information Processing Systems, 2023, 36: 47704-47720.

[9] Onyejelem T E, Aondover E M. Digital generative multimedia tool theory (DGMTT): A theoretical postulation in the era of artificial intelligence[J]. Adv Mach Lear Art Inte, 2024, 5(2): 01-09.

[10] Colafiglio T, Ardito C, Sorino P, et al. Neuralpmg: a neural polyphonic music generation system based on machine learning algorithms[J]. Cognitive Computation, 2024, 16(5): 2779-2802.

[11] Malavolta I, Nirghin K, Scoccia G L, et al. Javascript dead code identification, elimination, and empirical assessment[J]. IEEE Transactions on Software Engineering, 2023, 49(7): 3692-3714.

[12] Kuroki D, Pronk T. jsQuestPlus: A JavaScript implementation of the QUEST+ method for estimating psychometric function parameters in online experiments[J]. Behavior Research Methods, 2023, 55(6): 3179-3186.

[13] Jiang Y. Modular design on tile layout experiment with Javascript[J]. International Journal on Interactive Design and Manufacturing (IJIDeM), 2022, 16(3): 1163-1173.

[14] Kryk J, Plechawska-Wójcik M. Multi-aspect comparative analysis of JavaScript

programming frameworks–React. js and Solid. js[J]. Journal of Computer Sciences Institute, 2025, 34: 68-75.

[15] Meng J J, Chen X T, Li D C, et al. Pose estimation processing method of computer vision technology[J]. Computer Simulation, 2023, 40(05): 274-278.

[16] Peng Li, Zeng Fan. Application of improved K-means algorithm in the cultivation of creative music talents under the needs of sustainable development and transformation[J]. International Journal of Web Engineering and Technology,2024,19(1):4-19.

[17] Julian P. Merkofer, Guy Revach, Nir Shlezinger, et al. DA-MUSIC: Data-Driven DoA Estimation via Deep Augmented MUSIC Algorithm[J]. IEEE Transactions on Vehicular Technology,2024,73(2):2771-2785.

[18] Lanhui Liu, Menglin Kong, Cong Cao, et al. Personalized music recommendation algorithm based on machine learning[J]. Multimedia systems,2025,31(3):166.1-166.27.

[19] Lanqian Liu. Big data analysis for sustainable music teaching using collaborative filtering recommendation optimisation algorithm[J]. International journal of computational systems engineering,2024,8(3/4):263-272.

[20] Dong, Fang. An investigation of CNN-LSTM music recognition algorithm in ethnic vocal technique singing[J]. International Journal of Computational Science and Engineering, 2024,27(5):505-514.

[21] Hyeong Woo Ham, Hyuk-In Kwon, Joon-Ho Lee. Performance Analysis of the MUSIC Algorithm in the Presence of Correlated Noise[J]. IEEE Access,2025,1391957-91971.