PIANOROLL-EVENT: A NOVEL SCORE REPRESENTATION FOR SYMBOLIC MUSIC

Lekai Qian, Haoyu Gu*, Dehan Li*, Boyu Cao, and Qi Liu*†

School of Future Technology, South China University of Technology

ABSTRACT

Symbolic music representation is a fundamental challenge in computational musicology. While grid-based representations effectively preserve pitch-time spatial correspondence, their inherent data sparsity leads to low encoding efficiency. Discrete-event representations achieve compact encoding but fail to adequately capture structural invariance and spatial locality. To address these complementary limitations, we propose Pianoroll-Event, a novel encoding scheme that describes pianoroll representations through events, combining structural properties with encoding efficiency while maintaining temporal dependencies and local spatial patterns. Specifically, we design four complementary event types: Frame Events for temporal boundaries, Gap Events for sparse regions, Pattern Events for note patterns, and Musical Structure Events for musical metadata. Pianoroll-Event strikes an effective balance between sequence length and vocabulary size, improving encoding efficiency by 1.36× to 7.16× over representative discrete sequence methods. Experiments across multiple autoregressive architectures show models using our representation consistently outperform baselines in both quantitative and human evaluations.

Index Terms— Symbolic music representation, Music encoding, Music generation, Sequence modeling

1. INTRODUCTION

Symbolic music representation provides the foundation for computational musicology, yet the inherent complexity of musical structures poses significant challenges for establishing efficient encoding schemes. In contrast to natural language processing and computer vision with their standardized representations, the symbolic music community lacks consensus on optimal encoding strategies. This fragmentation impedes both fair comparison between methods and knowledge transfer across representations.

Current research in symbolic music representation predominantly follows two paradigms: continuous-time representation and discrete-event representation. Continuous-time approaches (e.g., pianoroll [1]) preserve the spatial relationship between pitch and temporal positions by modeling symbolic music as 2D matrices. However, the inherent sparsity of musical events typically results in matrices with substantial empty values, which may compromise the efficiency of information encoding. Discrete-event representations encode music as token sequences (e.g., ABC notation [2], MIDI Events [3]), facilitating efficient modeling. While generally offering significantly improved information efficiency, such approaches may not adequately capture inherent musical invariances. Critical properties including relative pitch intervals and timing patterns are frequently encoded as absolute values, potentially obscuring fundamental musical relationships and constraining representation learning capacity.

To address the limitations of existing representation paradigms, we propose Pianoroll-Event, a novel encoding scheme that combines the spatial structure preservation of pianoroll with the efficiency of discrete-event sequential representations. Our method first discretizes pianoroll into temporal frames to preserve time-dependent relationships, then transforms these frames into structured event sequences through four complementary event types: Frame Events that mark frame boundaries while compressing consecutive empty pitch regions, Pattern Events that encode local note activation patterns, Gap Events that efficiently represent sparse pitch intervals within frames, and Musical Structure Events that capture essential musical context including bar positions and time signature changes. Through this design, Pianoroll-Event achieves effective music information compression while ensuring each tokenized element maintains clear semantic meaning. In summary, the contributions of this paper are as follows:

- We propose Pianoroll-Event, a novel symbolic music encoding that bridges spatial and sequential representations by designing four complementary event types to transform pianoroll into efficient token sequences.

- We analyze encoding efficiency in terms of sequence length and vocabulary size, demonstrating significant computational advantages with compression ratios ranging from 1.36× to 7.16× over existing approaches.

- We conduct comprehensive experiments showing that Pianoroll-Event achieves state-of-the-art performance with improvements ranging from 3.43% to 47.00% in objective metrics and 30.61% to 66.56% in subjective evaluations across mainstream sequence modeling ar-

*Equal contribution. †Corresponding author: drliuqi@scut.edu.cn chitectures, with systematic ablation studies validating the effectiveness of each event component.

2. RELATED WORK

2.1. Symbolic Music Representation Methods

Symbolic music representation, as the foundation of music information processing, requires balancing information density, computational efficiency and expressive capability. **Pianoroll representation** [1] encodes music as a two-dimensional matrix, intuitively displaying the spatiotemporal distribution of notes, yet its sparsity leads to inefficiency and fixed time resolution limits complex rhythm expression. **ABC Notation** [2] uses letters to represent music with a compact format but is primarily designed for monophonic melodies. **MIDI event sequences** [3]can precisely record note attributes, though early direct conversion methods [4] [5]preserve complete information at the cost of excessive sequence length. REMI [6] effectively reduces sequence length by introducing relative position markers. Compound Word [7][8] combines related tokens into compound words, while OctupleMIDI [9][10] encodes note attributes as octuple tokens, both achieving more efficient representations. Byte Pair Encoding [11] automatically constructs vocabularies by learning frequent patterns to compress sequences.

2.2. Sequence Modeling and Autoregressive Generation

Sequence modeling represents a crucial approach in symbolic music generation. Early LSTM [12] mitigated long-term dependencies through gating mechanisms, though recurrent architectures limited parallel efficiency. Transformer[13] achieves global dependency modeling via self-attention, with Music Transformer [14] integrating relative position encoding for coherent long-form generation. Pop Music Transformer [6] extends context through segment-level recurrence, while MuseNet [15] demonstrates large-scale pretraining potential. These advances echo language model developments like GPT-2 [16] and LLaMA [17][18], where discrete event representations naturally adapt to autoregressive frameworks.

3. METHOD

This section describes the Pianoroll-Event conversion process. First, we partition the pianoroll into fixed-length frames. Then, we convert each frame into a sequence of four events that efficiently encode the sparse pitch information. Finally, we tokenize these events to produce sequences compatible with standard sequence modeling architectures.

3.1. Temporal Framing

We apply temporal framing to the pianoroll to generate fundamental units for subsequent transformation. Let $\mathbf{P} \in \{0,1\}^{T\times H}$ denote a pianoroll, where T is the number of time steps and $H = 88$ represents the standard piano pitch range. The framing process partitions $\mathbf{P}$ into an ordered sequence $\{\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_N\}$, with $N = \lceil T/L \rceil$ and each frame $\mathbf{F}_i \in \{0,1\}^{L\times H}$. Specifically, each frame is obtained as:

$$\mathbf{F}_i = \mathbf{P}[(i-1)L : \min(iL, T), :] \tag{1}$$

where the notation $\mathbf{P}[a : b, :]$ denotes the slice of $\mathbf{P}$ from time step a to $b-1$. This procedure preserves local musical structures like chordal verticality and melodic continuity, and maintains strict temporal dependencies across frames, enabling downstream models to capture music's temporal logic.

3.2. Pianoroll-Event Generation

A fundamental characteristic of pianoroll representations is their inherent sparsity—most entries are zeros, with active notes typically concentrated within limited pitch ranges. This sparsity motivates our event-based encoding strategy, which transforms dense frame representations into compact event sequences. We uniformly partition each frame $\mathbf{F}_i$ along the pitch dimension into fixed-size blocks, then encode these blocks as typed events based on their content and position.

Specifically, we partition each frame $\mathbf{F}_i \in \{0,1\}^{L\times H}$ into a sequence of Event Blocks $\{\mathbf{B}_{i,1}, \mathbf{B}_{i,2}, \ldots, \mathbf{B}_{i,K}\}$, where each block $\mathbf{B}_{i,j} \in \{0,1\}^{h\times w}$ has fixed size. The encoding process examines these blocks sequentially to generate typed events. The consecutive empty blocks at the beginning of a frame are merged into a single Frame Event, which marks the frame boundary and encodes the start position. Conversely, the consecutive empty blocks at the end are discarded, as they can be reconstructed during decoding given the fixed frame length. For the intermediate blocks, each non-empty block $\mathbf{B}_{i,j}$ is mapped to a unique Pattern Event: Pattern$(\mathbf{B}_{i,j})$ that preserves its note configuration, while a sequence of r consecutive empty blocks is compressed into a single Gap Event: Gap(r). Additionally, Musical Structure Events are inserted at measure boundaries to encode time signatures, bar lines, and other symbolic information.

These events are then mapped to discrete tokens for sequence modeling. Each event type corresponds to a distinct token vocabulary: Frame tokens encode starting positions, Pattern tokens represent unique note configurations, Gap tokens specify run lengths, and Musical Structure tokens encode symbolic elements. Given a pianoroll $\mathbf{P} \in \{0,1\}^{T\times H}$ partitioned into frames $\{\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_N\}$, the final event sequence is obtained as:

$$\mathbf{S} = \bigoplus_{i=1}^{N} [\text{Encode}(\mathbf{F}_i) \oplus \text{MSE}_i] \tag{2}$$

where Encode$(\cdot)$ denotes the frame encoding process described in Algorithm 1, MSE$_i$ represents optional Musical

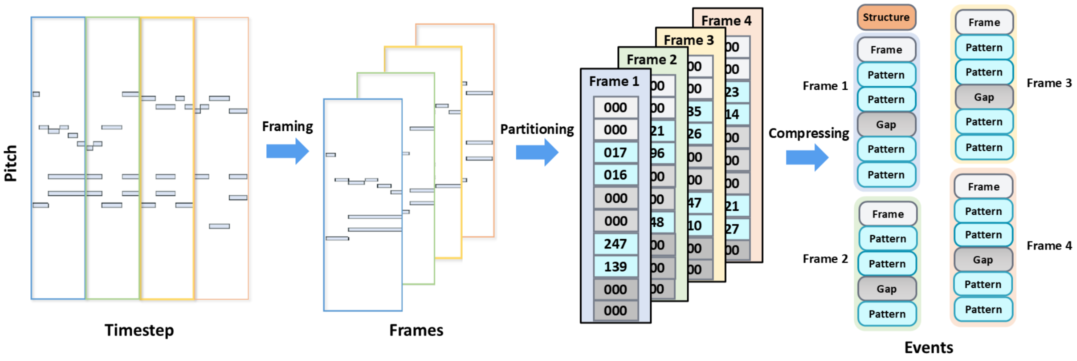

Fig. 1. The process of converting pianoroll representation into pianoroll-events. Through frame segmentation, partitioning, and compression operations, the pianoroll is transformed into a sequence of pianoroll-events containing diverse event types.

Structure Events at measure boundaries, and $\oplus$ denotes concatenation. This semantic mapping provides downstream models with meaningful token-level interpretations while achieving substantial compression. The resulting sequences preserve temporal structure and pitch patterns, maintaining compatibility with standard sequence modeling architectures.

Algorithm 1 Pianoroll-Event Encoding Process

```
Require: Pianoroll frame $\mathbf{F} \in \{0,1\}^{L \times H}$
Ensure: Event sequence $\mathbf{S}$
Partition $\mathbf{F}$ into blocks $\{\mathbf{B}_1, \mathbf{B}_2, \dots, \mathbf{B}_K\}$ of size $h \times w$
Merge leading empty blocks into Frame_Event
$\mathbf{S} \leftarrow$ [Frame_Event]; $r \leftarrow 0$
for each block $\mathbf{B}_i$ in the middle region do
    if $\mathbf{B}_i$ is empty then
        $r \leftarrow r + 1$
    else
        if $r > 0$: $\mathbf{S}$.append(Gap_Event($r$)); $r \leftarrow 0$
        $\mathbf{S}$.append(Pattern_Event($\mathbf{B}_i$))
    end if
end for
if $r > 0$: $\mathbf{S}$.append(Gap_Event($r$))
Drop trailing consecutive empty blocks
return $\mathbf{S}$
```

4. EXPERIMENTS AND ANALYSIS

4.1. Dataset

We train our models on the MuseScore dataset containing 140,000 two-track piano scores with durations ranging from 1 to 5 minutes. We convert these scores into multi-hot array piano roll representations with a temporal resolution 1/16 of a beat. Each time step preserves 88 pitch values corresponding to the standard piano keyboard. We then encode the piano rolls into different token representations for model training.

4.2. Encoding Efficiency Analysis

We evaluate encoding efficiency using the Budget-Aware Difficulty Index (BDI), defined as:

$$\text{BDI} = L^2 \times \sqrt{V} \tag{3}$$

where L is the average sequence length and V is the vocabulary size. This metric captures both the quadratic computational complexity of self-attention mechanisms and the capacity dilution effect of large vocabularies on model parameters.

Table 1 compares five encoding methods. Our approach achieves the lowest BDI (1.048×10^7), showing an optimal balance between sequence compression and vocabulary size. For fair comparison, we exclude velocity tokens from all sequence-based methods. Yet our approach still maintains its efficiency advantage over both traditional long-sequence methods and vocabulary-heavy BPE approaches.

Table 1. Encoding efficiency comparison

Method	L	V	**BDI** ↓	**Efficiency**
Ours	749.8	347	**1.048**	**1.00**×
REMI	1339.7	330	3.261	3.11×
MIDILike	1398.9	448	4.143	3.96×
REMI-BPE	317.8	20,000	1.429	1.36×
ABC Notation	2575.0	128	7.504	7.16×

4.3. Generation Quality Evaluation

We train all models for 20 epochs on the MuseScore dataset using an NVIDIA RTX 4090, with learning rate 1e-4 and batch size 256. The model configurations are: GPT-2-Small (4 layers, 512 hidden size), GPT-2-Large (8 layers, 768 hidden size), Llama (6 layers, 768 hidden size), and LSTM (4 layers, 512 hidden size). For evaluation, we generate 50 samples per method targeting 40-90 second durations for statistical analysis. Subjective assessment involves 10 samples per model evaluated by 30 musically trained listeners in a double-blind setup, with scores averaged to obtain the final **MOS**.

We employ objective metrics from MusPy [19]: **Polyphony Rate (PR)** for harmonic richness, **Groove Consistency (GC)** for rhythmic stability, and **Scale Consistency (SC)** for tonal coherence. The **JS Divergence Similarity** is computed as $\text{JS} = 100 \times \exp(-2 \times \overline{\text{JS}})$, where $\overline{\text{JS}}$ represents the average Jensen-Shannon divergence calculated using mean and variance of the distributions across all three metrics. Tables 2, 3, 4, and 5 show the evaluation results with mean values.

Table 2. GPT-2-Small Model Experimental Results

Method	**PR**	**GC**	**SC**	**JS↑**	**MOS↑**
REMI	0.735	0.844	0.698	34.86	1.97
REMI-BPE	**0.612**	0.847	0.844	54.36	2.50
MIDI-Event	0.773	0.869	0.692	39.33	2.07
CP	0.634	0.773	0.807	51.08	1.83
Octuple	0.041	0.928	0.905	43.31	1.67
ABC	0.398	**0.997**	0.977	63.29	2.63
Ours	0.653	0.927	**0.917**	**67.95**	**3.80**
GT	0.583	0.980	0.943	-	4.83

Table 3. GPT-2-Large Model Experimental Results

Method	**PR**	**GC**	**SC**	**JS↑**	**MOS↑**
REMI	0.751	**0.992**	0.710	35.85	1.07
REMI-BPE	0.286	0.815	0.878	55.27	2.93
MIDI-Event	0.748	0.855	0.709	40.53	2.03
CP	**0.719**	0.726	0.799	49.93	3.00
Octuple	0.078	0.916	0.909	50.61	2.33
ABC	0.261	0.997	0.966	65.18	2.00
Ours	0.742	0.936	**0.962**	**68.86**	**4.27**
GT	0.583	0.980	0.943	-	4.83

Table 4. Llama model experimental results

Method	**PR**	**GC**	**SC**	**JS↑**	**MOS↑**
REMI	0.805	0.804	0.695	34.87	1.13
REMI-BPE	0.346	0.860	0.832	49.94	2.07
MIDI-Event	0.955	0.835	0.719	31.56	2.07
CP	0.715	0.749	0.782	45.46	1.20
Octuple	0.111	0.918	0.853	45.58	1.36
ABC	0.368	**0.997**	0.960	64.77	4.33
Ours	**0.668**	0.912	**0.959**	**64.94**	**4.67**
GT	0.583	0.980	0.943	-	4.83

Table 5. LSTM model experimental results

Method	**PR**	**GC**	**SC**	**JS↑**	**MOS↑**
REMI	0.424	0.991	0.709	34.48	1.27
REMI-BPE	0.366	0.811	0.857	39.90	3.00
MIDI-Event	0.770	0.821	0.878	56.25	1.73
CP	0.750	0.806	0.689	35.63	1.90
Octuple	0.167	0.912	**0.948**	56.61	2.43
ABC	0.281	**0.975**	0.957	61.97	2.33
Ours	**0.601**	0.937	0.832	**62.53**	**3.53**
GT	0.583	0.980	0.943	-	4.83

Our method consistently achieves the highest JS scores across all architectures, with GC and SC metrics maintaining values above 0.93 and 0.92 respectively. This demonstrates superior alignment with ground truth distributions compared to all baseline methods.

4.4. Ablation Study

We conduct ablation experiments on GPT-2-Large to evaluate the contribution of each encoding component. Starting from pattern events only (P), we progressively add frame compression for leading zeros (PF+), remove trailing zeros (PF), and finally incorporate gap tokens for internal zeros (Full). As shown in Table 6, both objective metrics and subjective evaluations demonstrate consistent improvement across variants. The JS similarity increases from 50.16 (P) to 68.86 (Full), while MOS improves from 2.20 to 4.07, confirming that each component contributes meaningfully to the encoding's effectiveness.

Table 6. Ablation study on GPT-2-Large model

Method	**PR**	**GC**	**SC**	**JS↑**	**MOS↑**
P	0.370	0.723	0.714	50.16	2.20
PF+	0.683	0.905	0.962	60.92	3.20
PF	0.716	0.900	0.945	62.96	3.67
Full	**0.742**	**0.936**	**0.962**	**68.86**	**4.07**
GT	0.583	0.980	0.943	-	4.8

5. CONCLUSION

We presented Pianoroll-Event, a novel symbolic music encoding that bridges grid-based and discrete-event representations through four complementary event types. Our approach preserves spatial-temporal relationships while achieving superior encoding efficiency compared to existing methods. Experiments demonstrate state-of-the-art generation quality across various model architectures in both objective and subjective evaluations. This work provides a promising unified representation framework for advancing computational musicology and music generation tasks.

6. REFERENCES

[1] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," *arXiv preprint arXiv:1206.6392*, 2012.

[2] Chris Walshaw, "Abc2mtex: An easy way of transcribing folk and traditional music, version 1.6 user guide," Tech. Rep., University of Greenwich, 1996.

[3] MIDI Manufacturers Association, "The complete midi 1.0 detailed specification," Tech. Rep., MIDI Manufacturers Association, 1996.

[4] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan, "This time with feeling: Learning expressive musical performance," in *Neural Computing and Applications*, 2018.

[5] Jeff Ens and Philippe Pasquier, "Mmm : Exploring conditional multi-track music generation with the transformer," 2020.

[6] Yu-Siang Huang and Yi-Hsuan Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1180–1188.

[7] Wen-Yi Hsiao, Jen-Yu Liu, Yin-Cheng Yeh, and Yi-Hsuan Yang, "Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, vol. 35, pp. 178–186.

[8] Sida Tian, Can Zhang, Wei Yuan, Wei Tan, and Wenjie Zhu, "Xmusic: Towards a generalized and controllable symbolic music generation framework," 2025.

[9] Mingliang Zeng, Xu Tan, Rui Wang, Zeqian Ju, Tao Qin, and Tie-Yan Liu, "Musicbert: Symbolic music understanding with large-scale pre-training," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021, pp. 791–800.

[10] Zhiwei Lin, Jun Chen, Boshi Tang, Binzhu Sha, Jing Yang, Yaolong Ju, Fan Fan, Shiyin Kang, Zhiyong Wu, and Helen Meng, "Multi-view midivae: Fusing track- and bar-view representations for long multi-track symbolic music generation," 2024.

[11] Nathan Fradet, Nicolas Gutowski, Fabien Chhel, and Jean-Pierre Briot, "Byte pair encoding for symbolic music," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 2001–2020.

[12] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," 2023.

[14] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck, "Music transformer: Generating music with long-term structure," *arXiv preprint arXiv:1809.04281*, 2018.

[15] Christine Payne, "Musenet," *OpenAI Blog*, 2019.

[16] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, pp. 9, 2019.

[17] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al., "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[18] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom, "Llama 2: Open foundation and fine-tuned chat models," 2023.

[19] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsien Yang, "Muspy: A toolkit for symbolic music generation," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 101–108.