HIERARCHICAL TRANSFORMER ARCHITECTURE FOR PIANO MUSIC GENERATION VIA PATCH-BASED ENCODING

Lekai Qian*, **Dehan Li***, **Haoyu Gu***
South China University of Technology, School of Future Technology
{ftqlk, 202364820081, 202364820071}@mail.scut.edu.cn

ABSTRACT

We present a hierarchical transformer architecture for piano music generation that employs patch-based encoding to efficiently model both local patterns and long-range dependencies. Our approach encodes 88-dimensional piano-roll representations into compact 8-bit patch tokens, reducing computational complexity by 75% while maintaining generation quality. The architecture consists of a Patch-Level Decoder for coarse temporal modeling and a Character-Level Decoder for fine-grained token generation within patches. The system supports musical conditioning including time signatures and sequence length control, with flexible generation strategies for both standard autoregressive and guided generation modes.

1. INTRODUCTION

Piano music generation faces significant challenges due to the high dimensionality of piano-roll representations (88 keys $\times$ time steps). Traditional autoregressive approaches struggle with quadratic complexity $\mathcal{O}(L^2)$ when modeling extended sequences, leading to memory constraints and training inefficiencies.

We propose a hierarchical transformer architecture that addresses these challenges through a patch-based encoding scheme inspired by vision transformers. Our approach treats piano-roll segments as patches, enabling efficient processing while preserving musical structure.

2. METHOD

2.1 Patch-Based Encoding

Given a binary piano-roll matrix $\mathbf{P} \in \{0,1\}^{88\times T}$, we divide it into non-overlapping patches of size 2×4 (2 pitches $\times$ 4 time steps). Each patch is encoded into an 8-bit integer token:

$$\text{token}_{i,j} = \sum_{k=0}^{7} b_k \cdot 2^{7-k} \tag{1}$$

where b_k represents the k-th bit in the flattened patch. This yields a token matrix $\mathbf{T} \in \{0,...,255\}^{44\times(T/4)}$, reducing sequence length by 75% while maintaining musical information.

Our vocabulary extends to special tokens: PAD=256 (padding), EOS=257 (end-of-sequence), and BOS=1 (beginning-of-sequence).

2.2 Model Architecture

2.2.1 *Patch-Level Decoder*

The Patch-Level Decoder models coarse temporal structure. Given patch tokens $\mathbf{X} \in Z^{B\times L\times 44}$ (batch size B, sequence length L):

$$\mathbf{E}_{\text{patch}} = \text{Linear}(\text{OneHot}(\mathbf{X})) \tag{2}$$

The model incorporates musical conditioning through learnable embeddings for time signature ($\tau \in \{0,...,4\}$) and sequence length ($\ell \in \{0,...,127\}$). The final input sequence $[\mathbf{e}_{\text{len}}; \mathbf{e}_{\text{ts}}; \mathbf{E}_{\text{patch}}]$ is processed through a GPT-2 transformer.

2.2.2 *Character-Level Decoder*

The Character-Level Decoder autoregressively generates 44 tokens within each patch. For each patch embedding $\mathbf{h} \in R^{d_{\text{model}}}$:

$$p(x_t|x_{<t},\mathbf{h}) = \text{Softmax}(\text{GPT2-LM}([\mathbf{h}; \mathbf{e}_{x_{<t}}])) \tag{3}$$

2.3 Generation Strategies

Standard Generation: Extends a prefix autoregressively until reaching maximum length or encountering three EOS rows. We employ temperature-controlled sampling with top-k filtering.

Guided Generation: Incorporates ground truth patches at specific intervals, enabling controlled generation with structural guidance:

$$\mathbf{x}_t = \begin{cases} \mathbf{x}_{\text{ground}}[i] & \text{if } t \bmod 5 = 0 \\ \mathcal{D}_c.\text{generate}(\mathbf{h}_t) & \text{otherwise} \end{cases} \tag{4}$$

Divergence Mechanism: Creates sparse representations by downsampling the temporal dimension: $\mathbf{T}_{\text{div}} =$

* Equal contribution

Late-Breaking / Demo Session Extended Abstract, ISMIR 2025 Conference

Parameter	Value
Patch size (H × W)	2×4
Vocabulary size	258
Max sequence length	128
Temperature default	0.7
Top-k default	10

Table 1. Key hyperparameters of the model.

PatchEncode($\mathbf{P}[:, :: 8]$), providing skeletal structure for long-range guidance.

3. IMPLEMENTATION DETAILS

3.1 Architecture Specifications

The model is trained end-to-end using teacher forcing with cross-entropy loss, masking PAD tokens (256) in loss computation. The data processing pipeline handles padding to ensure width divisibility by 4, patch extraction, and token encoding/decoding.

3.2 Training Protocol

4. DATASET AND TRAINING

Dataset: We train our models on a subset of the MuseScore dataset, containing approximately 15,000 piano pieces with durations ranging from 30 seconds to 2 minutes. The dataset covers diverse musical styles including classical, pop, and jazz compositions. Each piece is converted to piano roll format with 1/16 beat resolution, preserving temporal relationships across 88 piano keys.

Data Processing: Piano rolls are preprocessed by: (1) normalizing to binary format, (2) padding sequences to ensure temporal dimension divisibility by 4, and (3) extracting 2×4 patches for token encoding. This yields training sequences with manageable vocabulary size while maintaining musical structure integrity.

Training Configuration: Both decoders use GPT-2 architecture with 8 layers, 1024 hidden dimensions, and 8 attention heads. We employ AdamW optimizer with learning rate 1e-4, batch size 16, and 0.1 dropout. Training runs for 50 epochs on 2 NVIDIA RTX 4090 GPUs, requiring approximately 12 hours total training time.

5. KEY ADVANTAGES

Our hierarchical approach offers several benefits:

- **Computational Efficiency:** Reduces sequence length by 75% through patch encoding
- **Hierarchical Modeling:** Separates global structure from local detail generation
- **Flexible Control:** Supports time signature and length conditioning
- **Memory Efficiency:** Compact 8-bit representation per patch
- **Modular Design:** Independent optimization of coarse and fine decoders

6. DEMONSTRATION

Our demonstration will showcase:

1. Real-time music generation with adjustable temperature and top-k parameters
2. Comparison between standard and guided generation modes
3. Visualization of patch encoding/decoding process
4. Interactive control over time signature and sequence length

7. CONCLUSION

We presented a hierarchical transformer architecture that effectively balances computational efficiency with generation quality for piano music. The patch-based encoding scheme and two-level decoding strategy enable efficient modeling of both local patterns and long-range dependencies, making it suitable for real-time music generation applications.