Automated Expressive MIDI Rendering System for MIREX 2025 RenCon Competition

Leduo Chen
CSE, UCSD
San Diego, US
lec015@ucsd.edu

Xinrui Su
Physics and astronomy, UCL
london, UK
ucapxsu@ucl.ac.uk

Yuqiang Li
EECS, Queen Mary Univ. of London
London, UK
y.j.li@se24.qmul.ac.uk

Honyu Andy Shing
Science Dept., Jincai High Sch. Int. Div.
Shanghai, China
andy_shing@jcid.cn

Junchuan Zhao
School of Computing, NUS
Singapore, Singapore
junchuan@u.nus.edu

Zihan Chai, Kunyang Zhang, Shengchen Li
School of Adv. Tech., Xi'an Jiaotong-Liverpool Univ.
Suzhou, China
Zihan.Chai23@student.xjtlu.edu.cn
Kunyang.Zhang23@student.xjtlu.edu.cn
Shengchen.Li@xjtlu.edu.cn

***Abstract*—This report presents a system for the MIREX 2025 RenCon Expressive Piano Performance Rendering Competition. It transforms MusicXML scores into expressive MIDI performances using a pipeline of tokenization, detokenization, and post-processing. Leveraging the note-aligned (n)-ASAP dataset, the system captures fine-grained timing, dynamics, and articulation, achieving high-quality expressive output with adaptive token processing and robust audio rendering.**

I. Introduction

The MIREX 2025 RenCon Expressive Piano Performance Rendering Competition challenges systems to render expressive piano performances from MusicXML scores. Generate a realistic performance through performance information such as rhythm and pedals. This task advances automated music rendering, with technical challenges including mapping symbolic scores to human performances and modeling tokens with expressive timing.

Our system addresses these challenges through a pipeline of tokenizers, model training, detokenizers, and audio rendering. To generate refined and expressive audio, we utilize the note-aligned (n)-ASAP dataset [3], processing features at note-level granularity throughout the pipeline, which is more precise than beat-based approaches, capturing fine-grained timing, dynamics, and articulation. This report reviews related work, details methodology of the generation process.

II. Methodology

Our system for the RenCon Expressive Piano Performance Rendering Competition processes MusicXML scores and MIDI performance data to generate expressive piano performances. The architecture comprises four main components: (1) a tokenizer to convert input scores and performances into aligned tokens, (2) a model to learn and predict expressive parameters, and(3) a detokenizer to transform predicted tokens back into performance events with the score tokens. Below, we describe each component in detail.

A. Tokenizer

In expressive piano performance rendering, a tokenizer is essential to convert note-level MusicXML scores and MIDI performances into structured tokens with derived features, such as onset deviations and local tempo. Unlike basic preprocessing, it enables models to learn expressive nuances from aligned data.

Our system processes MusicXML scores, MIDI performances, and match files. We patched music21 Python library with a `NoteIdPreservingParser` to extract note IDs, ensuring robust alignment. The `MusicXMLTokenizer` produces `ScoreNoteToken` (pitch, e.g., C4; duration; articulations) and `ScoreMetadata`(e.g., composer, performer, genre). The `MIDITokenizer` `PerformanceNoteToken` (pitch, velocity, onset/duration in seconds) and `PerformanceControl` (sustain) converting tokens from ticks to seconds. The `PairedTokenizer` aggregates these into aligned token pairs using note IDs, with suffix disambiguation (e.g., `n1-1`). The `ScoreExplainer` processes these pairs to derive features like onset deviation ($\delta_{\text{onset}} = b_{\text{est}} - b_{\text{score}}$) and local tempo, estimated using a 5-second sliding window to fit a smoothing spline.

B. Data Representation and Feature Embedding

1) Score and Performance Features: Score features include **pitch** (discrete tokens like 'A##1'), **position** (beat location), **duration** (note values), performance directives (**staccato**, **accent**), and **part identifiers** for voice separation. Performance features capture expressive interpretation through **onset deviation in seconds** and **duration deviations in seconds** for timing adjustments, **local tempo** for tempo fluctuations, **velocity** for dynamics, and **sustain level** for pedal usage.

2) Hybrid Embedding Mechanism: Different feature types require tailored embedding strategies. Discrete features use standard learnable embedding tables mapping each value to a dense vector. Continuous features employ learned projection networks with normalization and non-linearities to preserve numerical relationships while mitigating gradient instability.

For pitch representation, we decompose traditional discrete encoding into note class (C–B), accidental, and octave components, mirroring Western tonal hierarchy. Unlike conventional methods that treat C4 and C5 as unrelated categories, this shares note embeddings across octaves, encoding octave equivalence and improving generalization.

3) Musical Position Embedding: We replace the standard absolute positional encoding with a hybrid `MusicalPositionalEmbedding`. It integrates (1) learned embeddings interpolated over continuous beat positions, capturing fine-grained rhythmic nuances (e.g., syncopation at beat 1.5), and (2) sinusoidal encodings to model periodic structures such as meter and phrase cycles. The concatenation of these representations provides a unified embedding that jointly encodes local rhythmic detail and global temporal organization.

C. Model Architecture

Our transformer encoder-decoder architecture uses a 4-layer bidirectional encoder for score context and 6-layer causal decoder for autoregressive performance generation [4]. A tempo predictor extracts chunk-level global tempo from encoded scores via attention pooling, serving dual purposes: conditioning the decoder for temporal coherence and enabling beat-to-time conversion. The decoder incorporates predicted average tempo and composer embeddings through Style-Adaptive Layer Normalization (SALN) [1]. Rather than concatenating conditioning signals with input token embeddings, the joint tempo–composer vector is injected into the decoder's normalization layers, allowing each transformer layer to emphasize composer-specific style while preserving global tempo consistency.

1) Output Prediction Strategy: The output layer employs three specialized strategies. Discrete features use classification heads, with sustain pedal as binary classification due to skewed distribution. Following prior work [2], pitch prediction employs auxiliary losses by simultaneously predicting the complete pitch number, pitch class (pitch mod 12, representing note identity), and octave (pitch div 12, representing register). This multi-task approach helps the model learn hierarchical pitch structure beyond individual pitch tokens. Continuous features with perceptual quantization use binned regression with custom boundaries for local tempo and velocity, while onset and duration deviations use direct regression with Huber loss for millisecond precision.

D. Generation

Due to Transformer sequence length constraints, we employ an overlapping window strategy segmenting complete scores into 512-length windows with 256-token overlap. Using cumulative overlap approach, the first window generates autoregressively while subsequent windows use previous outputs as ground truth in overlap regions and generate only non-overlapping sections. To ensure tempo continuity, the model predicts tempo distribution parameters for each window then applies exponential smoothing: $\text{tempo}^{(t)}_{\text{smooth}} = 0.3 \cdot \text{tempo}^{(t)}_{\text{raw}} + 0.7 \cdot \text{tempo}^{(t-1)}_{\text{smooth}}$. During generation, score-related features use ground truth from input while performance-specific parameters are generated autoregressively with nucleus sampling.

E. Detokenizer

The detokenizer obtains data from model outputs, consisting of performance information in `FullPerformanceToken` sequences and runtime configuration. The `FullPerformanceToken` includes five types of information: score note data (pitch and duration), performance note data, onset deviations, duration deviations, and sustain levels. These components enhance the expressiveness of the score by incorporating timing variations, dynamic adjustments, and pedal effects, enabling musically coherent MIDI output.

Runtime configuration includes `composer_id`, which specifies the arrangement style, and sequence processing parameters: `sequence_length` , `overlap_length` , and `stride` , which control the segmentation and continuity of token sequences. Additional metadata, such as `use_decomposed_pitch=true`, improves pitch accuracy by separating pitch components, ensuring precise MIDI event generation

III. Post-processing

Post-processing refines the generated MIDI events to ensure musical coherence. In Detokenizer part, The `remove_overlaps` function eliminates note overlaps by limiting duration reductions to 50%, preventing simultaneous notes of the same pitch. The `smooth_tempo` function applies cubic spline interpolation to create natural tempo transitions, avoiding abrupt changes.

The MIDI postprocessing and rendering process uses Logic Pro. The MIDI information is imported to the application and a "Velocity processor" is applied to control the distribution of note velocities. The rendering soundfont is based on the "Steinway Grand Piano" patch available in Logic Pro with additional reverb effect and audio dynamic compressor added to ensure the rendered audio resembles an authentic recording.

References

[1] Chung-Ming Chien, Jheng-Hao Lin, Chien-Yu Huang, Po-Chun Hsu, and Hung-Yi Lee. *Investigating on Incorporating Pretrained and Learnable Speaker Representations for Multi-Speaker Multi-Style Text-to-Speech*, pages 7748–7759. IEEE, 6 2021.

[2] Yuqiang Li, Shengchen Li, and György Fazekas. Pitch class and octave-based pitch embedding training strategies of symbolic music generation. *Proceeding of 16th International Symposium on Computer Music Multidisciplinary Research*, November 2023.

[3] S. D. Peter, C. E. Cancino-Chacón, F. Foscarin, A. McLeod, E. Karystinaios, F. Henkel, M. Neuwirth, and G. Widmer. Automatic note-level score-to-performance alignments in the ASAP dataset. *Transactions of the International Society for Music Information Retrieval*, 6(1):27–39, 2023.

[4] A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A Gomez, L Kaiser, and I Polosukhin. *Attention is All you Need*, volume 30, pages 5998–6008. Curran Associates, Inc, 2017.