VirtuosaAI: Music Generation with Transformers

Rishikesh Vishnu Sivakumar
MPSYS
Chalmers University of Technology
Göteborg, Sweden
vishnuri@chalmers.se

Xiaoying Liu
MPCAS
Chalmers University of Technology
Göteborg, Sweden
liuxiao@chalmers.se

***Abstract*—This project explores an AI-driven approach to generating classical piano music using a Transformer-based architecture and a relative self-attention mechanism, focusing on capturing long-term dependencies and nuanced musical expressions essential for coherent and emotive piano compositions. By training on the MAESTRO dataset and implementing relative self-attention, the model produces structured and expressive music pieces that emulate the qualities of virtuoso piano performances. For the same data, an LSTM-RNN model is also trained, and the performances of both models are compared. The results show that Transformers perform better and demonstrate their efficacy in generating classical compositions that retain thematic consistency, offering potential applications in automated music production and interactive media.**

***Index Terms*—music generation, Transformer, LSTM-RNN, piano music**

I. Introduction

Music generation using deep learning has become a prominent research area, especially with advancements in deep learning models such as Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) and Transformers. For classical piano music, maintaining emotional depth and structural coherence presents unique challenges. Traditional Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, while effective for sequence data, often struggle with long-term dependencies intrinsic to complex compositions. Transformers, known for their capacity to model extended dependencies through attention mechanisms, are more suited for this task.

Building on advancements in symbolic music generation, this project implements both a Transformer model with relative attention mechanisms and an LSTM-RNN model for classical piano music generation. Both approaches are designed to capture the sequential and hierarchical nuances essential to classical music, leveraging tokenization schemes to map MIDI files into a structure the models can effectively process. Through comparative analysis of both architectures, we evaluate their ability to generate authentic, structured compositions that emulate the thematic and expressive qualities found in the works of classical pianists. The results demonstrate the superior performance of the Transformer architecture in maintaining long-term musical coherence while preserving the intricate details of classical compositions.

II. Related Work

Recent advancements in symbolic music generation have highlighted the effectiveness of Transformer architectures. These models have gained prominence for their exceptional capability to capture long-range dependencies within musical sequences, enabling the generation of more coherent and structurally rich compositions. Their unique self-attention mechanism allows for the processing of vast amounts of musical information, leading to innovative approaches in composing rhythmically and harmonically complex pieces.

A crucial concept for this project is relative self-attention, as described in Shaw et al.'s work on Self-Attention with Relative Position Representations [1]. This paper introduces a generalizable approach for encoding relative positional information within the self-attention mechanism of Transformers. Their method extends the self-attention mechanism to efficiently consider representations of the relative positions, or distances, between sequence elements. By incorporating relative position representations directly into the self-attention calculation, the model enhances its ability to capture sequential dependencies without relying on absolute position encodings.

Building on this foundation, Huang et al.'s Music Transformer [2] applies the concept of relative self-attention to Transformer models for symbolic music generation. A key innovation in their work is the introduction of an efficient relative attention algorithm, which significantly reduces the memory complexity of relative self-attention from quadratic to linear in sequence length. By implementing a "skewing" procedure that reorganizes the relative attention computation, the Music Transformer can handle much longer sequences, enabling the generation of minute-long compositions with thousands of steps. This improvement is crucial for music generation, as it allows the model to capture and maintain long-term musical structures over extended sequences, resulting in more coherent and structurally sound compositions.

Further building upon these advancements, our project particularly benefited from the work of Xiong et al. on On Layer Normalization in the Transformer Architecture [3]. This paper provides crucial insights into the efficiency of using pre-layer normalization (Pre-LN) for Transformer models in generative tasks.

Additionally, the Gaussian Error Linear Unit (GELU) activation function has been recognized for its effectiveness

in deep learning models, especially Transformers. "Gaussian Error Linear Units (GELUs)" [4] by Hendrycks and Gimpel offers a thorough mathematical analysis, showcasing its differentiability and smoothness, and demonstrates its superior performance over traditional functions like ReLU in various applications.

In summary, the integration of advanced Transformer architectures and innovative activation functions like GELU underscores the ongoing evolution of symbolic music generation, paving the way for more sophisticated and expressive musical compositions.

III. Method

A. Dataset, Data Preprocessing, and Tokenization

For this project, we utilize the MAESTRO dataset (V3.0.0), which offers over 200 hours of paired audio and MIDI files from professional piano performances. This high-quality dataset includes note-on/off events, velocity, and pedal usage, making it ideal for symbolic music modeling. The MAESTRO dataset comprises 1,276 performances, with a total of 430 unique compositions by 13 composers, providing a rich and diverse source of classical piano music.

We focus exclusively on the MIDI files for our tokenization process. The tokenization approach is based on the method described by Oore et al. (2018) [5], which has proven effective for music generation tasks. This particular tokenization scheme was chosen for its comprehensive representation of musical events and its ability to capture both temporal and pitch information efficiently.

Our vocabulary consists of several types of events:

- Note-on events: 128 events (indices 1-128) representing the start of a note.
- Note-off events: 128 events (indices 129-256) representing the end of a note.
- Time-shift events: 125 events (indices 257-381) representing time between events, with each unit equal to 8ms, allowing for up to 1 second of time shift.
- Velocity events: 32 events (indices 382-413) representing how hard a key is pressed.
- Special tokens: Padding (index 0), start (index 414), and end (index 415) tokens.

This vocabulary structure results in a total size of 416 tokens, providing a balance between expressiveness and model complexity.

The tokenization process involves several key steps:

- MIDI Parsing: We parse each MIDI file, converting musical events into their corresponding token indices. This process handles note events, time shifts, and velocity changes, as well as special cases like pedal events and tempo changes.
- Sequence Sampling: To create manageable input sequences for training, we employ two sampling methods:
  - Random sampling of sequences with a specified length from the full MIDI sequences.
  - Specific sampling from the end of each input sequence to ensure the model learns how to properly conclude musical pieces.

  The chosen fixed length for our sequences is 512 tokens, which provides a good balance between capturing musical phrases and maintaining computational efficiency.
- Data Augmentation: To increase the diversity of our training data, we apply two augmentation techniques:
  - Pitch Transposition: We shift the pitch of the sequences up and down by -5, -1, 0, 3, and 6 semitones. This range was chosen to maintain musical plausibility while significantly expanding the dataset.
  - Time Stretching: We apply time stretches of 0.95, 1, and 1.05 to each sequence, subtly altering the tempo without drastically changing the musical character.
- Final Processing: We add start and end tokens to each sequence and pad them to a uniform length using the padding token. The augmented and sampled data is then shuffled to ensure random order during training.

The tokenization process results in a dataset of fixed-length sequences of integers, where each integer corresponds to a musical event in the vocabulary. This representation preserves the temporal and pitch information of the original MIDI files, transforming them into a format suitable for training sequence models such as RNNs or Transformers for music generation tasks.

After tokenization and augmentation, our dataset expanded to approximately 64,000 sequences, each with a length of 600 tokens (including start and end tokens). We split this data into training and validation sets using an 85-15 ratio, providing a substantial training set while retaining a significant portion for validation to ensure model generalization.

B. Model

Our music generation model employs a decoder-only transformer architecture, also known as an auto-regressive transformer, as it predicts the next token based on the previous tokens in the sequence. This architecture is widely used for text generation and is increasingly being applied to music generation as well.

The model begins with an input embedding layer that converts token indices into dense vector representations. Following this, the model includes a sinusoidal positional encoding layer that adds crucial information about the position of tokens in the sequence, enhancing the model's ability to capture order-related dependencies.

To ensure robustness during training, an input dropout layer is applied after the embedding and positional encoding, helping to mitigate overfitting. The core of our model consists of multiple decoder layers, each configured with parameters such as model dimension, number of attention heads, feed-forward dimension, maximum relative distance for positional embeddings, and dropout rates, allowing for a flexible and powerful architecture.

In the forward pass, the input tensor undergoes scaling after embedding and positional encoding are added. This process is

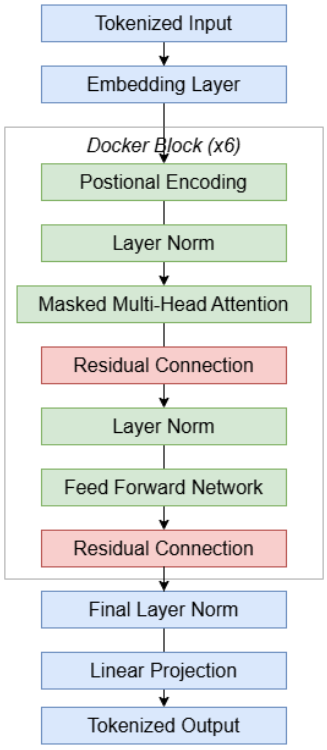

Fig. 1. Transformer Decoder Block Workflow

followed by passing the data through the configured decoder layers. Finally, a layer normalization step is applied to match the behavior of a traditional transformer decoder. The output from this layer is projected to the vocabulary size using a linear transformation.

The multi-head relative self-attention mechanism within each decoder layer computes attention scores based on both the content and relative positions of tokens:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T + R}{\sqrt{d_k}}\right)V \quad (1)$$

where Q, K, and V are the query, key, and value matrices, and R represents the relative position embeddings. The relative position embeddings are retrieved based on the sequence length. This dual approach, which combines relative positional attention with sinusoidal absolute positional encoding, enhances the model's understanding of musical structure by capturing local patterns and long-range dependencies effectively.

The sinusoidal absolute positional encoding is defined as follows:

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}\right) \quad (2)$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}\right) \quad (3)$$

Furthermore, masking is applied to ensure that predictions for a given position depend only on known outputs at previous positions, preventing any influence from future tokens.

After the attention mechanism, each decoder layer incorporates a feed-forward network consisting of two linear transformations with a GELU activation function in between. This activation function has been shown to perform well in transformer architectures, offering smoother gradients compared to other commonly used functions. The network employs kaiming weight initialization technique to help maintain consistent variance across layers, which promotes stable gradient flow during training. Dropout is applied after the attention and feed-forward processes to prevent overfitting and improve generalization.

C. Training

The model was trained using the Adam optimizer. The training employed a custom learning rate scheduler designed to increase the learning rate linearly during the initial warmup steps and decrease it proportionally to the inverse square root of the step number thereafter. This approach helps stabilize training by allowing the model to adapt to the data gradually, especially in the early stages, which can lead to better convergence compared to standard schedulers.

To evaluate the model's performance, a masked loss function was implemented, utilizing cross-entropy loss as the base metric. This function computes the average loss while excluding padded elements in the target sequences, ensuring that the model focuses on learning from valid data points and does not get penalized for padding. The formula for the masked loss function is given by:

$$\text{Masked Loss} = \frac{\sum_{i=1}^{N} \text{Loss}(y_i, \hat{y}_i) \cdot \text{Mask}(y_i)}{\sum_{i=1}^{N} \text{Mask}(y_i)} \quad (4)$$

where y_i is the ground truth, $\hat{y}_i$ is the predicted output, and $\text{Mask}(y_i)$ is 1 for valid positions and 0 for padding.

The hyperparameters that gave the best results are:

- Model Dimension : 256, allowing for a balance between capacity and computational efficiency.
- Number of Layers: 6, providing sufficient depth to capture complex patterns in the data.
- Number of Heads: 8, enhancing the model's ability to focus on different parts of the input simultaneously.
- Feedforward Dimension: 1024, enabling effective transformation of input representations.

The model was trained for 50 epochs, with performance metrics being recorded for both training and validation sets to monitor overfitting and ensure robust learning throughout the training process.

D. Evaluation Metrics

To evaluate the performance of the music generation model, the following metrics were utilized: accuracy, F1 score, and perplexity.

- Accuracy:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

- F1 Score:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

where

$$Precision = \frac{TP}{TP + FP} \quad \text{and} \quad Recall = \frac{TP}{TP + FN} \tag{7}$$

- Perplexity:

$$\text{Perplexity} = e^{-\frac{1}{N}\sum_{i=1}^{N}\log(P(w_i))} \tag{8}$$

where N is the total number of valid tokens and $P(w_i)$ is the predicted probability of the i-th token.

The combination of accuracy, F1 score, and perplexity offers a well-rounded evaluation for the music generation model. Accuracy measures the overall correctness of predictions, while the F1 score balances precision and recall, ensuring that relevant musical elements are generated while minimizing irrelevant ones. Perplexity assesses the model's ability to predict sequences of musical tokens, with lower values indicating better performance in generating coherent music. Together, these metrics effectively capture both the technical and creative aspects of music generation.

E. Music Generation

The trained transformer model can now be used for music generation using the music_generation file. It can either start with a single token or use an existing MIDI file as a prompt for predicting the sequences. The generated sequence is then converted to MIDI and subsequently to a WAV audio file using FluidSynth, a software synthesizer that uses a soundfont to add instrument sounds. The final WAV file is ready for playback in the notebook itself or for further processing, turning digital music data into listenable audio.

IV. Results

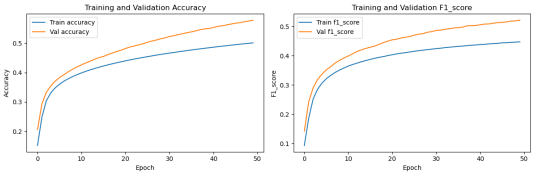

Fig. 2. Accuracy and F1 score over epochs for transformer

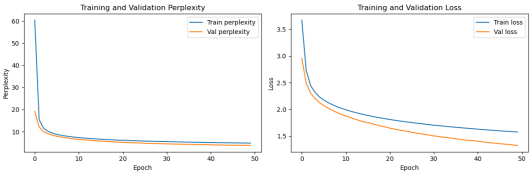

Fig. 3. Perplexity and loss over epochs for transformer

The training process shows steady improvements in model performance metrics, such as accuracy, F1-score, perplexity, and loss, over the epochs. Starting from a low initial accuracy of around 0.15 in epoch 1, the model reaches a training accuracy of approximately 0.47 and a validation accuracy of around 0.55 by epoch 50. F1-scores also improve, aligning well with the accuracy trend and indicating balanced precision and recall. Perplexity decreases over time, suggesting that the model's predictions are becoming more certain, while the gradual reduction in loss indicates convergence.

Therefore, we can conclude that the consistent learning rate decay is effective in aiding stability as the model continues to learn, and with more training, even better results could be achieved.

V. Comparison with LSTM-RNN model:

LSTM-RNN is another popular architecture for music generation, using recurrent connections and gated memory cells to process sequential information. However, it has some drawbacks compared to transformers:

- Limited parallel processing: LSTMs process sequences step-by-step, which slows down training.
- Struggles with long-term dependencies: Although memory cells help, they can still forget important information.
- Higher inference cost: Sequential processing leads to increased computational demands.

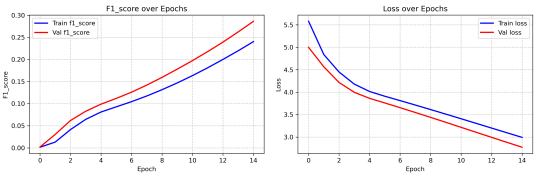

Fig. 4. F1 score and loss over epochs for the LSTM-RNN model

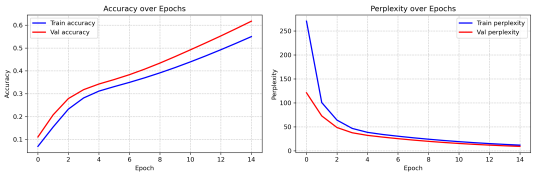

Fig. 5. Accuracy and perplexity over epochs for the LSTM-RNN model

In a test with an LSTM model (3 bidirectional layers, 256 hidden units, AdamW optimizer with 0.01 weight decay, learning rate schedule), the model achieved decent results after 15 epochs (validation accuracy $\sim$ 61, perplexity $\sim$ 9.1, F1 score $\sim$ 0.28). However, these results could not be trusted as the music generated was of lower quality than transformers. So we could say that the transformers did better.

VI. Conclusions

In this study, we demonstrated a transformer model's effectiveness in symbolic music generation through comprehensive training and also by comparing it with LSTM-RNN model. The generated audio files produced decent music, and with

more computational resources and extended training, even better results could be achieved. Future work could focus on leveraging architectural advancements such as:

- Transformer-XL) [6], which better handles long-range dependencies
- Compound Word Transformer) [7], which uses a Linear Transformer backbone for improved symbolic music encoding and faster training

For evaluation metrics, instead of using both accuracy and F1 scores, we could:

- Choose one of these metrics alongside a more suitable generative task metric like modified BLEU and perplexity
- Incorporate music-specific metrics based on pitch, rhythm, and structure
- Use audio comparison metrics like PEAQ (Perceptual Evaluation of Audio Quality) or MUSHRA (MUltiple Stimuli with Hidden Reference and Anchor), eliminating the need for expert human evaluation

Additionally, expanding the dataset to include various musical instruments for multi-track MIDI could enhance the model's capability to generate richer, multi-instrument compositions.

VII. References

References

[1] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-Attention with Relative Position Representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, 2018, pp. 464–468.

[2] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, et al., "Music Transformer: Generating music with long-term structure," in *International Conference on Learning Representations*, 2019.

[3] R. Xiong, Y. Yang, D. He, K. Zheng, and T. Zheng, "On Layer Normalization in the Transformer Architecture," in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020, pp. 10524–10533.

[4] D. Hendrycks and K. Gimpel, "Gaussian Error Linear Units (GELUs)," *arXiv preprint arXiv:1606.08415*, 2016.

[5] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This Time with Feeling: Learning Expressive Musical Performance," *arXiv preprint arXiv:1808.03715*, 2018.

[6] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context," *arXiv preprint arXiv:1901.02860*, 2019.

[7] Y. Hsiao, S. Dai, and B. Kong, "Compound Word Transformer: Learning to Compose Full-Song Music over Dynamic Directed Hypergraphs," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, pp. 7966–7974, 2021.