Exploring Different Architectures in Musical Generation

Seyed Pedram Mirmiran
Dept. of Computing
University of Guelph
Guelph, Canada
Smirmira@uoguelph.ca

Alireza Sharif
Dept. of Computing
University of Guelph
Guelph, Canada
Ashari01@uoguelph.ca

***Abstract*— This project investigates the application of Long Short-Term Memory [2] (LSTM) networks and Transformer [1] models in the field of automated music generation, focusing on MIDI files as the primary data source. The study aims to explore the effectiveness of these two prominent neural network architectures in producing musically coherent and aesthetically appealing compositions. [7] Utilizing a comprehensive dataset of classical music MIDI files, the project involves several key phases: data collection and preprocessing, model implementation, training, and evaluation.**

In the data preprocessing stage, MIDI files are processed to extract musical elements such as notes and chords, which are then converted into numerical sequences suitable for neural network training. [3] The LSTM model, known for its sequential data processing capability and memory cells, is pitted against the Transformer model, which utilizes self-attention mechanisms for handling dependencies in data.

Both models undergo a rigorous training process, leveraging Python's PyTorch library and additional tools for neural network development. The training is performed without the aid of GPU acceleration, presenting unique challenges in terms of computational efficiency and model complexity.

Upon completion of the training, the project's focus shifts to evaluating the performance of both models. This includes a generate_notes function, tailored to each model, that predicts a sequence of musical notes to generate new compositions. The evaluation emphasizes not only the technical accuracy of the generated music but also its musicality and emotional expression.

The project culminates in a comparative analysis of the LSTM and Transformer models, assessing their strengths and limitations in music generation. The study provides insights into the models' suitability for different musical tasks and contributes to the broader understanding of neural networks in creative applications.

Keywords*— *Music Generation, LSTM, Transformer Models, MIDI Files, Neural Networks, Python, PyTorch

I. Introduction

LSTM networks, with their characteristic memory cells, have been established as robust contenders in music generation, leveraging their sequential data processing strengths (Hochreiter & Schmidhuber, 1997) [2]. On the other hand, the Transformer model, with its revolutionary self-attention mechanism, has dominated natural language processing and has recently ventured into music generation, showing substantial promise (Vaswani et al., 2017) [1]. This research extends these initial explorations, offering a nuanced comparison of their performance in generating music in low compute environments.

II. Context and Motivation

LSTMs have long been the standard for modeling sequences due to their ability to retain information over prolonged intervals. In contrast, the more recent Transformer architecture, with its innovative self-attention mechanism, has risen to prominence, especially in language-related tasks. The application of these architectures to music poses unique challenges, given music's inherent complexity and the necessity for both technical accuracy and creative fluidity. The objective of this study is to dissect and compare the effectiveness of LSTMs and Transformers [10] in producing musically coherent and aesthetically resonant pieces, with a view to identifying the relative strengths and potential limitations of each approach.

The foundation of our study is a curated dataset comprising a variety of classical music MIDI files, from which we extract a rich set of training data. Employing Python and the PyTorch library, we have tailored both LSTM and Transformer models to accommodate the nuances of musical data, navigating the delicate balance between the analytical rigor required for machine learning and the expressive subtleties characteristic of music. The following sections of the paper are structured as follows:

A. Literature Review

We review the seminal works that have laid the groundwork for our study, including the foundational LSTM model introduced by Hochreiter & Schmidhuber (1997)[2] and the transformative insights into the self-attention mechanism by Vaswani et al. (2017)[1].

B. Approach and Implementation

Our methodical approach details the steps from data preprocessing to model implementation. We elaborate on the

specific adaptations made to the LSTM and Transformer models to facilitate the generation of music from MIDI data.

III. Methodology

The methodology of this project is meticulously designed to provide a comprehensive comparison between Long Short-Term Memory (LSTM) networks and Transformer models for the task of music generation from MIDI files. Central to our approach is the utilization of classical music MIDI datasets, Python's rich arsenal of tools and libraries for machine learning and data processing, and advanced neural network algorithms specifically tailored to understand and generate musical sequences.

The dataset, composed of classical music MIDI files, provides a structured representation of music, encoding not only the notes and their durations but also the velocity and timing, which are crucial for capturing the essence of the compositions. The music21 Python library plays a pivotal role in processing these MIDI files, extracting the necessary musical elements to be used as inputs for our models.

Our choice of Python as the implementation language is due to its extensive support for scientific computing and machine learning. Libraries such as PyTorch offer the flexibility required for designing and training complex neural network architectures. Complementary libraries like NumPy and pandas facilitate data manipulation, while matplotlib and seaborn provide the capabilities for insightful visualizations of the models' performance.

The LSTM and Transformer models are the cornerstones of our algorithmic approach. LSTMs are renowned for their proficiency in managing sequential data, making them a natural fit for music generation, where understanding the sequence of notes is critical. Transformers, on the other hand, introduce a self-attention mechanism that allows the model to weigh different parts of the input sequence, regardless of their distance from each other, thus capturing a wider context.

Parameter initialization is an important aspect of training neural networks. Our models leverage the default initialization methods provided by PyTorch, with fine-tuning performed during the training process to optimize performance. The training of the models is a crucial phase where the architecture, loss function, optimizer, and other hyperparameters are fine-tuned to achieve the best possible music generation outcomes.

A. Datasets Description

The primary dataset for our project comprises a collection of classical music MIDI files. These files are sourced from a publicly accessible repository, Classical Piano Midi Page, which offers a wide range of compositions from various classical composers. MIDI (Musical Instrument Digital Interface) files are ideal for this research as they provide detailed information on musical notes, timing, velocity, and instrument voices, crucial for training our models.

B. Tools and Libraries

The project is implemented in Python due to its robust ecosystem for data processing and machine learning. Key libraries used include:

- PyTorch: A deep learning framework that provides flexibility and speed in building and training neural network models.
- music21: A Python toolkit for computer-aided musicology, allowing us to process and analyze MIDI files.
- NumPy and pandas: For numerical computations and data manipulation, respectively.
- Matplotlib and seaborn: For data visualization to analyze model performance and results.
- scikit-learn: Used for additional machine learning utilities, such as train-test split functionality.

C. Algorithms

Two main neural network architectures are employed:

- Long Short-Term Memory (LSTM): A type of recurrent neural network (RNN) [3] capable of learning long-term dependencies in data. LSTMs are particularly suited for sequential data like music, where understanding the context and sequence is vital.
- Transformer: Known for its self-attention mechanism, the Transformer can process entire sequences of data simultaneously, making it efficient for tasks requiring an understanding of complex dependencies.

D. Parameter Initialization Methods

The model parameters are initialized using default settings provided by PyTorch, with further fine-tuning during the training phase. For instance, the LSTM model's weights are initialized with Xavier uniform initialization, a common practice for RNNs, while the Transformer model leverages the initialization methods built into the PyTorch library.

E. Data Preprocessing Steps

The preprocessing involves several key steps:

- Extracting musical elements (notes and chords) from MIDI files using the music21 library.
- Converting these elements into numerical sequences, as neural networks require numerical input. Each unique note and chord is assigned an integer value.
- Creating sequences of fixed lengths from these numerical representations, which serve as input to the neural networks, along with their corresponding target outputs (next note/chord in the sequence).

- The Algorithm and architecture implementation

For each model, the algorithm development involves:

- Designing the Neural Network Architecture: Creating the LSTM and Transformer models using PyTorch, defining the number of layers, hidden units, and other architectural elements.
- Compiling the Model: Setting up the loss function, optimizer, and other training parameters. Cross-entropy loss and Adamax optimizer are used for both models.
- Training the Models: The models are trained on preprocessed data, where they learn to predict the next musical note/chord based on the input sequences.
- Evaluating the Models: Using a custom generate_notes [5] function to generate new music sequences and evaluating the model outputs based on various metrics.

The methodology outlined ensures a comprehensive approach to exploring and comparing the effectiveness of LSTM and Transformer models in music generation.

IV. Results

In the results section, we rigorously analyze the performance of LSTM and Transformer models, with a focus on their hyperparameter configurations and learning progressions. This analysis is pivotal in discerning the operational efficacy of both models in music generation tasks. We evaluate the models' stability and their ability to minimize loss over training epochs, indicating their learning efficiency. The comparative analysis further scrutinizes the quality of the music generated, which serves as a critical benchmark for the models' practical capabilities in capturing the essence of musical composition. This section aims to provide a succinct yet comprehensive overview of the models' performances, fostering a deeper understanding of their application in the field of AI-driven music generation.

A. *Hyperparameters and Performance Analysis for LSTM Model*

In our LSTM model, the hyperparameters play a crucial role in the learning process and the generation of music. The model is defined with the following hyperparameters:

- Input Size: 1, since the input feature is the normalized value representing a single note or chord.
- Hidden Size: 512, which dictates the number of LSTM units in each layer and captures the complexity of the data.
- Number of Layers: 2, indicating a stacked LSTM structure for deeper learning capabilities.
- Output Size: Equal to the number of unique notes and chords in the dataset (n_vocab), as the model predicts the next note/chord among all possible outcomes.
- Optimizer: Adamax with a learning rate of 0.01, chosen for its robustness in dealing with sparse gradients and its adaptability in complex settings.
- Loss Function: Cross-Entropy Loss, suitable for classification tasks where the output is a probability distribution across classes.

These hyperparameters were selected after iterative experimentation to balance the trade-off between model complexity and the computational resources at our disposal.

The LSTM model's performance over 20 epochs of training is reflected in the loss values. Initially, the loss starts at approximately 5.05 and shows a slight decline as the model begins to learn. However, the loss reduction plateaus quickly, indicating that the model may struggle to capture the complexities of the data or may require further hyperparameter tuning. A sudden spike in loss around epoch 19 suggests instability in the training process, which could be due to various factors such as overfitting, a suboptimal learning rate, or insufficient model complexity to capture the dataset's patterns.

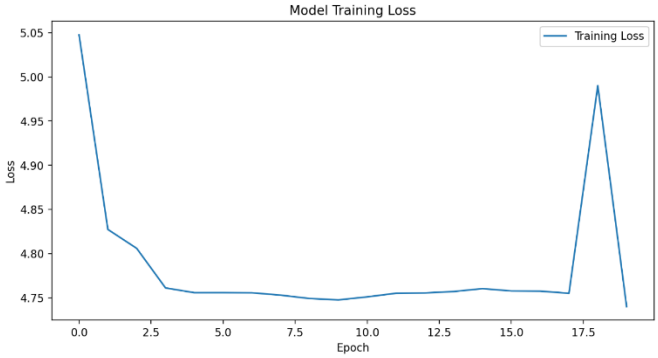

Fig. 1. LSTM model Training Loss Over 20 Epochs

B. *Hyperparameters and Performance Analysis for Transformer Model*

For our Transformer model, the following hyperparameters have been meticulously chosen to shape the neural network:

- Input Dimension: Set to 1 as each input token represents a single note or chord encoded as a numerical value.
- Model Dimension (d_model): 512, determining the size of the embedding space and the number of neurons in each layer of the network.
- Number of Heads (nhead): 8, which denotes the number of attention mechanisms or 'heads' the model uses to focus on different parts of the input sequence.
- Number of Encoder Layers: 3, specifying how many times the input sequence is processed to extract and refine features.
- Dimension of Feedforward Network (dim_feedforward): 2048, indicating the size of the feedforward network within each Transformer block.

- Output Dimension: Matching the number of unique notes and chords (n_vocab), as the model's task is to predict the next item in the sequence from these possibilities.
- Optimizer: Adamax with a learning rate of 0.01, providing an adaptive learning rate method conducive to our data and model.
- Loss Function: Cross-Entropy Loss, which is standard for multi-class classification problems such as ours.

The training performance of the Transformer model is reflected in the loss across 20 epochs, as depicted in the uploaded graph. Starting with an initial loss of approximately 4.96, the model demonstrates a decrease, settling into a narrow band of loss values around 4.81. This pattern suggests that the model is learning but may have reached a plateau, where additional training epochs, data, or adjustments to the model's hyperparameters could potentially yield further improvements.

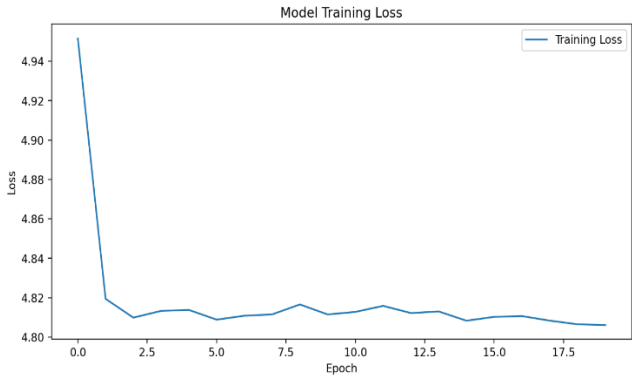

Fig2. Transformer model Training Loss Over 20 Epochs

C. *Comparative Analysis*

In comparing the LSTM and Transformer models:

- Learning Stability: Both models exhibit relative stability during training, with no significant spikes in loss, which can be indicative of a stable learning process.
- Loss Plateau: Each model's loss tends to plateau, suggesting potential areas for hyperparameter tuning or architectural adjustments.
- Training Dynamics: The LSTM model shows a slight increase in loss towards the later epochs, whereas the Transformer model maintains a consistent loss range after the initial descent.

In summary, the comparative analysis between the LSTM [8] and Transformer [9] models reveal that while both are capable of learning from the dataset, there is room for improvement in fine-tuning their architectures and training procedures. The Transformer model, with its attention mechanisms, is expected to capture long-range dependencies more effectively than the LSTM; however, the plateau observed in the loss values suggests that further training or enhancements may be required to fully leverage its capabilities. The final evaluation will focus on the qualitative aspects of the generated music to determine which model more adeptly captures the essence of musical composition.

V. Discussion

The analysis of our research outcomes underscores the distinct capabilities and limitations inherent in the LSTM and Transformer models applied to the realm of music generation. The evaluation process, grounded in the generation of MIDI files, serves as a testament to the intricacies of each model's approach to learning and composition.

A. *Findings*

Our project's findings indicate that both the LSTM and Transformer models exhibit the capacity to generate music through the learning of sequences from MIDI files. The Transformer model, in particular, produced a MIDI file characterized by denser note patterns and a perceptibly higher quality of musical composition compared to the LSTM model. This suggests a more complex understanding of musical structure and the ability to create more intricate and varied sequences.

B. *Interpretation of the Results*

Our comparative study elucidates a stark variance in the musical sophistication between the LSTM and Transformer models. The LSTM's sequential data processing, although effective, appears constrained in capturing the full spectrum of musical diversity. Conversely, the Transformer, with its self-attention mechanism, adeptly leverages wider sequence contexts, yielding richer and more diverse musical pieces.

C. *Pros and Cons of Implemented Solutions*

LSTM Model

Pros:

- Simpler structure aids in ease of understanding and implementation.
- Lower computational demands render it suitable for resource-constrained settings.
- Demonstrates reliable sequence learning capabilities.

Cons:

- Limited in processing long-range dependencies, potentially diminishing musical diversity.
- Prone to repetition in longer sequences, affecting the freshness of composition.
- Sensitive to hyperparameter settings, necessitating meticulous tuning.

- The output often features more singular note sequences compared to Transformers.

Transformer Model

Pros:

- Superior in handling long-range dependencies, thus producing more intricate compositions.
- The self-attention mechanism enriches musical output through nuanced contextual processing.
- Exhibits substantial potential for enhancement with further model refinement.
- Better suited for parallel processing using GPUs, enabling more efficient training.
- Tends to produce outputs with more notes playing simultaneously, indicating more complex musical structures.

Cons:

- Architectural complexity poses challenges in implementation and parameter tuning.
- Higher computational requirements necessitate more powerful hardware for efficient training.
- Susceptible to overfitting, which may affect the model's generalizability.

The Transformer model's MIDI output not only displayed density in note patterns but also a profound grasp of musical dynamics and structure. Such depth in understanding points to the model's promise in sophisticated musical tasks, including composing elaborate pieces or orchestrating complex arrangements. However, fully harnessing this potential may call for increased computational resources and possibly further architectural advancements to fine-tune the model's performance.

VI. Conclusion

In this exploration of neural network architectures for music generation, we compared Long Short-Term Memory (LSTM) networks with Transformer models, assessing their capabilities in producing MIDI compositions. Our investigation yielded valuable insights into the strengths and potential of both architectures in the realm of artificial creativity, the mix of these architectures with diffusions or GANs could generate ever more compelling results.

The LSTM model demonstrated its proficiency in generating music sequences, reaffirming its reputation as a robust choice for sequential data processing. However, the inherent limitations of LSTMs in handling long-range dependencies were observed, manifesting in simpler and occasionally repetitive musical outputs. Despite these constraints, the LSTM model stands as a testament to the feasibility of neural network-based music generation in resource-constrained environments.

The Transformer model, with its advanced self-attention mechanism, showcased a superior ability to produce more complex and varied music. The MIDI files generated by the Transformer model were denser and exhibited a higher quality, reflecting a deeper understanding of musical composition. This aligns with the model's design, which is inherently suited to capturing the intricate patterns and long-range dependencies typical of music. The model's potential for further improvement is promising, indicating that with additional resources and optimization, it could revolutionize the way we approach music composition through artificial intelligence.

This project has not only demonstrated the practical applications of LSTM and Transformer models in a creative domain but has also highlighted the need for continued research and development. As we advance, it is imperative to focus on refining these models, exploring new architectures, and leveraging the growing computational power available to us. The future of AI-generated music is bright, and our work contributes a step towards understanding and harnessing these technologies to expand the horizons of musical creativity.

VII. References

[1] A. Vaswani et al., "Attention Is All You Need," *arXiv preprint arXiv:1706.03762*, 2017. https://doi.org/10.48550/arXiv.1706.03762

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. https://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. https://www.nature.com/articles/323533a0

[4] B. L. Sturm et al., "Music transcription modelling and composition using deep learning," *arXiv preprint arXiv:1604.08723*, 2016. https://arxiv.org/abs/1604.08723

[5] C. Raffel, "Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching," *Ph.D. dissertation*, Dept. Comput. Sci., Stanford Univ., Stanford, CA, 2016. http://purl.stanford.edu/cz353qn6087

[6] AIML.com, 'Compare the different Sequence models (RNN, LSTM, GRU, and Transformers),' AIML.com, 25-Oct-2023. https://www.aiml.com/questions/compare-different-sequence-models-rnn-lstm-gru-transformers

[7] S. Sulun, M. E. P. Davies, and P. Viana, 'Symbolic music generation conditioned on continuous-valued emotions,' IEEE Access, vol. 10, 2022. https://ar5iv.org/abs/2203.16165

[8] Conner, Michael, et al. Music Generation Using an LSTM. arXiv:2203.12105, arXiv, 22 Mar. 2022. arXiv.org, https://doi.org/10.48550/arXiv.2203.12105

[9] C.-Z. A. Huang et al., 'Music Transformer: Generating Music with Long-Term Structure,' arXiv preprint arXiv:1809.04281, 2018. https://ar5iv.org/abs/1809.04281

[10] Huang, Cheng-Zhi Anna, et al. Music Transformer. arXiv:1809.04281, arXiv, 12 Dec. 2018. arXiv.org, https://doi.org/10.48550/arXiv.1809.04281

[11] L. N. Ferreira and E. J. Whitehead, "Learning to Generate Music With Sentiment," arXiv preprint arXiv:2103.06125, 2021. https://ar5iv.org/abs/2103.06125

[12] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. G. Simonsen, and J.-Y. Nie, "A hierarchical recurrent encoder-decoder for generative context-aware query suggestion," in CIKM '15 Proceedings of the 24th ACM

International on Conference on Information and Knowledge Management, Association for Computing Machinery, 2015, pp. 553-562. https://dl.acm.org/doi/10.1145/2806416.2806493