Evaluating LSTM, GRU, and Transformer Architectures for Music Note Pattern Generation with Small MIDI Datasets

Cristoval Neo Sasono
Computer Science Department
School of Computing and Creative Arts
Bina Nusantara University,
Jakarta, Indonesia 11480
cristoval.sasono@binus.ac.id

Brandon Salim
Computer Science Department
School of Computing and Creative Arts
Bina Nusantara University,
Jakarta, Indonesia 11480
brandon.salim@binus.ac.id

Kenneth Jayadi Yu
Computer Science Department
School of Computing and Creative Arts
Bina Nusantara University,
Jakarta, Indonesia 11480
kenneth.jayadi@binus.ac.id

Feri Setiawan
Computer Science Department
School of Computing and Creative Arts
Bina Nusantara University
Jakarta, Indonesia 11480
feri.setiawan@binus.edu

Raymond Bahana
Computer Science Department
School of Computing and Creative Arts
Bina Nusantara University
Jakarta, Indonesia 11480
raymond.bahana@binus.edu

***Abstract*— The application of deep learning models in music generation has gained quite the attention, with several architectures like Long Short-Term Memory (LSTM), Gated Recurrent Units(GRU), and Transformers, demonstrating their potential in modeling sequential data. This research in particular will aim to evaluate the effectiveness of these different architectures in generating musical note patterns using a small data set of just 130 MIDI files. The models were assessed both quanitatively through metrics such as loss, accuracy, and F1 score; as well as quantitatively through visualized pattern generation through MuseScore.**

Results show that LSTM outperformed GRU and Transformer by gaining the highest F1 score of 0.2176 in 10 epochs, 0.3726 in 15 epochs, and 0.6737 in 30 epochs. GRU showed some success, with a faster training speed and generation capabilities akin to what LSTM generates, while Transformers performed surprisingly worse, stagnating in an accuracy of (22%) with its loss decreasing quite slow. These findings highlight the limitations of Transformers in small-data scenarios and emphasize the suitability of LSTM and GRU for such contexts. Future work could explore hybrid architectures and data augmentation to enhance model performance in resource-constrained environments.

***Index Terms*—Music Generation, Limited Dataset, LSTM, GRU, Transformer**

I. Introduction

In recent years, we have seen quite a significant growth in the development of artificial intelligence, which in many aspects, has taken center stage in our lives and technology. One aspect of artificial intelligence that has seen the substantial advancement is in its generative capabilities, which we can see in AI such as ChatGPT, and its ability to generate text at images on prompt. Pushing beyond that, the generation capabilities of artificial intelligence is being intersected with the objective to generate music. This has captured a lot of attention, especially with the advancements in deep learning techniques.

AI-driven music generation takes advantage of neural networks in which these techniques can learn musical structures, styles, and patterns to generate new, yet coherent compositions. Deep learning models such as Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM) networks, Gated Recurrent Unit (GRU), the more recent Transformer architecture, as well as many more models have shown quite promising results in generating the sequential and layered nature of music.

For example, a study published in IEEE Xplore discusses the use of Generative Adversarial Networks (GANs) in music generation, highlighting how these models can produce mellifluous music by learning from complex sound patterns [1]. Another example is a study presented in the International Journal of Machine Learning and Computing, which discusses the implementation of RNN-LSTMs for generating Carnatic music. Their approach involves a modular pipeline that predicts various musical components, such as motifs, tunes, endnotes, and ornamentations called gamakas. The study demonstrates the ability of deep learning models to capture the intricate structure of Indian classical music, achieving training accuracies up to 98% for certain modules [2].

Building upon this foundation, our research focuses on developing an AI model to generate music using MIDI files of classical music. However, unlike many existing studies that emphasize large datasets, our study pivots towards exploring the capabilities of different models in generating music with a small, limited dataset. This direction reflects the practical

979-8-3503-3954-3/23/$31.00

constraints of resource availability and aims to address the challenges faced in such scenarios.

To achieve this, we selected three prominent deep learning architectures for evaluation: Long-Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and the more recent Transformer model. Our study aims to evaluate how these models perform in generating coherent and stylistically relevant music under the constraints of a small data set. By analyzing the strengths and limitations of each model in this context, we aim to contribute to the understanding of their applicability in resource-limited scenarios and provide insights into optimizing music generation tasks with minimal data.

II. RELATED WORKS

Long Short-Term Memory networks (LSTMs) have seen more use in the field of AI music generation in recent times. Dua et al. (2020) [3] proposed an LSTM-based model that emphasises on precise chord and rhythm prediction, allowing better capture of harmonic progressions in the structure, while also showing that LSTMs has an advantage over RNNs in complex harmonic modelling. Kumar et al. (2020) [4], on the other hand, utilized LSTMs to generate Indian Carnatic music by incorporating motifs and gamakas (note transitions) to preserve the unique structure. The LSTM allowed the model to retain and manage those motifs over long sequences, allowing for better adherence to the genre. For classical music, Kumar and Ravindran (2019) [5] demonstrated an LSTM architecture combined with reinforcement learning, aiming to improve the composition of polyphonic music, showing how LSTMs allowed the model to understand the structure of polyphonic music, albeit missing some characteristics such as repetition and tension. Colombo and Gerstner (2018) [6] used LSTMs to build a model that is able to compose in multiple styles within classical music, and despite lowered performance for more complex styles, achieved generally satisfactory results.

Other than LSTMs, Gated Recurrent Units (GRUs) have also been explored in the realm of AI music generation. Nayebi and Vitelli (2015) [7] compared the performance of LSTM and GRU networks for algorithmic music generation, concluding that LSTMs produced more musically plausible outputs than GRUs. However, recent studies have investigated hybrid architectures combining LSTM and GRU cells to leverage the strengths of both models. For instance, a 2023 study proposed a music generation system utilizing a character-level RNN with a hybrid LSTM-GRU architecture, trained on a dataset of 340 tunes in ABC notation. This approach aimed to capture both long-term dependencies and short-term patterns in music, enhancing the quality and diversity of generated compositions [8].

Other than LSTMs, Transformer models have emerged as an alternative due to their self-attention mechanism, which allows the capture long-term dependencies more effectively than LSTM, a previously major limitation. Huang et al. (2018) [9] refined this further by introducing relative attention mechanism, which improves memory effectiveness and allowed for structural coherence to be maintained for a relatively long time. Wu and Yang (2020) [10] used a Transformer model to generate jazz music, and despite getting decent results for a complex genre, showed how erratic usage of pitches tend to be problem.

Our work here aim to further investigate the performance of LSTM, GRU, and Transformer architectures in music generation, particularly focusing on their effectiveness in resource-constrained environments with limited datasets.

III. METHODOLOGY

A. Research Flow

We established a structured research methodology, which can be seen in Figure 1.

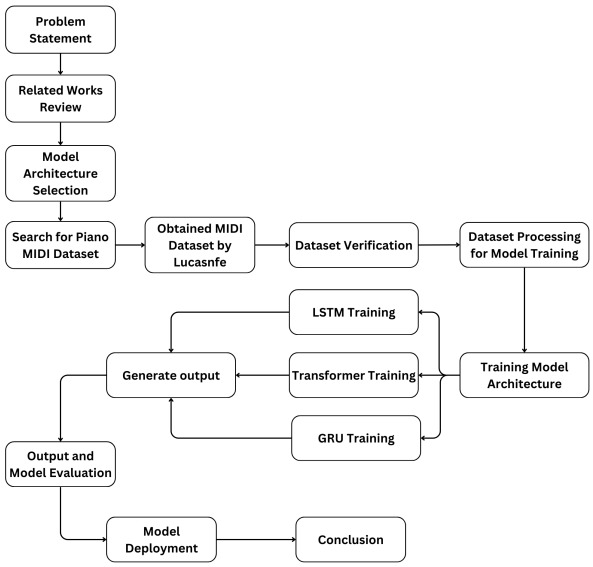

Fig. 1. Research Flow Diagram

a) Problem Statement: This research starts with defining the problem of generating coherant musical note patterns using limited resources and limited data set. Main objective being to evaluate and compare the performance of three deep learning models, namely LSTM, GRU, and Transformer.

b) Related Works Review: We did research on existing literature to see and understand how these models were utilized previously. It helps to establish relevance and novely of our study.

c) Model Architecture Selection: Based on insights from literature review, three models were chosen for comparison: Long Short-Term Memory (LSTM) networks, Gated Recurrent Units (GRU), and Transformer models. Each model represents a unique approach to handling sequential data and offers distinct advantages.

d) Data Set Selection, Verification, Processing: We scoured the internet for a piano MIDI data set. We eventually found one that we could use for our goals and processed it for use. Further elaboration will be in the Data Set subsection.

e) Model Training: The three chosen architectures—LSTM, GRU, and Transformer—are trained on the processed dataset.

f) Generate Output and Model Evaluation: Once trained, the models are used to generate musical note patterns. The generated outputs are evaluated for coherence, relevance, and quality. Performance metrics and qualitative assessments are used to compare the models' effectiveness.

g) Conclusion and Deployment: The study concludes with a discussion of the findings, highlighting the strengths and limitations of each model under the constraints of the small dataset. Insights from the evaluation are used to identify potential areas for improvement and future research. The trained models are then deployed in a distributed system to be tested by users.

TABLE I
MAIN PYTHON LIBRARIES USED

Library Name	Description
tensorflow	A deep learning framework widely used for building and training neural networks. Provides essential tools for machine learning workflows.
music21	A toolkit for computer-aided musicology, used for parsing, analyzing, and processing MIDI files.

B. Data Set

The dataset used for this research was sourced from GitHub[1], provided by Lucas N. Ferreira [11]. The ADL Piano MIDI dataset contains 11,086 piano pieces spanning various genres.

Information on this dataset, it is derived from the Lakh MIDI dataset, a collection of 45,129 unique MIDI files matched to entries in the Million Song Dataset. Since most pieces in the Lakh MIDI dataset include multiple instruments, only tracks featuring instruments from the "Piano Family" (MIDI program numbers 1-8) were extracted, resulting in 9,021 unique piano MIDI files. These files were then augmented with approximately 2,065 additional files scraped from publicly available online sources.

C. Data Set Processing

To narrow the dataset's scope, we chose to use the Classical MIDI subset, which contains approximately 1,300 files. After selecting a specific genre from the dataset, we start processing the training data that will be used by the models by randomly selecting 10% of the files, resulting in the training set of 130 MIDI files. This reduced scope allows us to work with the given resources and time that we have, aligining with the specific objective of analyzing which models work best under these limited constraints.

Following the selected MIDI files, it was further parsed using the music21 library, which allows us to extract the essential music features, including notes, chords, durations, tempos, time signatures, and key signatures. Notes and chords were represented in machine-readable formats, with individual pitches extracted for chords. Rests were labeled to maintain the integrity of the musical sequences. Additional features like durations, tempos (defaulting to 120 BPM when unspecified), time signatures (defaulting to "4/4"), and key signatures (defaulting to C major/A minor) were also extracted and mapped to integer indices.

However, we decided to just focus on its note pattern generation for this study. This decision reflects the primary goal of evaluating the models' ability to learn and generate coherent melodic sequences based on note patterns. This simplifies the complexity of the task, while ensuring that the models could generate something cohesive. All the other aspect will be controlled post production.

The input sequence constructed from the notes are using a sliding window approach to create sequences of 50 elements. Each sequence's subsequent note was designated as the target output. After that, the notes were normalized to facilitate model training, and the processed sequences were serialized for efficient storage and reuse.

D. Deep Learning Architectures

In this project, we employ three distinct deep learning architectures to explore their capabilities in music generation: Long-Short-Term Memory (LSTM) networks, Gated Recurrent Units (GRU), and Transformer models.

a) Long-Term-Short Memory: LSTM networks are a type of recurrent neural network (RNN) designed to capture long-term dependencies in sequential data. They utilize memory cells and gating mechanisms to effectively manage information flow, making them particularly suitable for tasks like music generation, where understanding temporal relationships is crucial. Studies have demonstrated the efficacy of LSTM in generating coherent musical compositions by learning existing musical sequences [12].

b) Gated Recurrent Unit: GRUs are a simplified variant of LSTMs, offering a more streamlined architecture with fewer parameters. They employ gating mechanisms to control information flow without maintaining a separate memory cell, which can lead to faster training and reduced computational requirements. GRUs have been applied in music generation tasks, effectively modeling sequential data and capturing temporal dependencies in musical compositions [13].

c) Transformer: Transformer models leverage self-attention mechanisms to process sequential data, allowing for the capture of long-range dependencies without relying on recurrent connections. This architecture enables parallel processing of input sequences, leading to improved scalability. In the realm of music generation, Transformer-based models have shown promise in generating compositions with coherent structure and style [14].

E. Training Device Specifications

All training and evaluations were conducted on a single machine with the following specifications:

- **Device:** MacBook Pro
- **Model:** 13-inch, 2020, Four Thunderbolt 3 ports

[1]See full dataset at https://github.com/lucasnfe/adl-piano-midi

- **Processor:** 2 GHz Quad-Core Intel Core i5
- **Graphics:** Intel Iris Plus Graphics 1536 MB
- **Memory:** 16 GB 3733 MHz LPDDR4X
- **Operating System:** macOS Sonoma 14.5

The limited computational power of this setup influenced the choice of dataset size and number of epochs, as well as the training time for each model.

F. Evaluation Techniques

To evaluate our trained architecture models, we employed a total of 4 quantitative and 1 qualitative measures to provide insights into their performance in a resource constrained environment paired with a small data set.

a) Training Loss - Categorical Cross-Entropy Loss: Categorical Cross-Entropy Loss was selected as a measure to see the disparity between the predicted probability distributions with the actual labels. It is quite effective for multi-class classification tasks, such as predicted the next note in a sequence. The lower the value means the more the model's predictions closely align with the true distribution [15]. This measure was chosen due to its reliable measure of how well the model learns to predict the target sequence over successive epochs, which allowed us to comparatively evaluate the architecture's learning capabilities.

b) Training Accuracy: Training accuracy was used to measure the proportion of correct predictions made by the model out of all predictions during the training process. While it provides a clear evaluation of overall correctness, it is quite valuable for tracking the model's general improvement and ability to predict the note sequences.

c) F1 Score: The F1 Score is the harmonic mean of precision and recall. This is quite important as it offers a nuanced view of performance, notably in sequence generation tasks with class imbalances or perhaps repetitive outputs [16]. This was chosen due to being able to give further insight other than accuracy for evaluating diverse and relevant musical sequences.

d) Training Time: Training time was recorded to measure the computational efficiency of each model. This metric captures the duration required for training each architecture over a fixed number of epochs, which is particularly important in resource-constrained scenarios where computational efficiency is as critical as model performance.

e) Qualitative Assessment - Pattern Generation: In addition to quantitative metrics, a qualitative assessment was conducted by analyzing the structure of the generated music through the midi editor *MuseScore*. We inspect the output manually to determine each model's abilities to generate music, as the quality and preference of music is quite subjective and cannot be captured by quantitative metrics alone.

IV. Result and Discussion

Based on our research, the performance of each deep learning architectures revealed a distinct difference in their ability to learn and generate music through a data set of just 130 MIDI files and with the given environment. The information of their training loss and training accuracy can be seen in Table II and III with their graphs in Figure 2 and 3 respectively, while the F1 Scores can be seen in Table IV. Notably, LSTM emerged as the most effective, followed by GRU. Surprisingly, Transformers, which are often celebrated for their cutting-edge capabilities, performed significantly worse than expected in this resource-constrained scenario.

TABLE II
Training Loss for LSTM, GRU, and Transformer Models

Epoch	LSTM Loss	GRU Loss	Transformer Loss
1	4.2907	4.4379	4.4313
2	4.1601	4.2652	4.2766
3	4.1092	4.2141	4.2350
4	4.0553	4.1943	4.2068
5	3.9902	4.1797	4.1909
6	3.9103	4.1643	4.1686
7	3.8192	4.1375	4.1573
8	3.7008	4.1094	4.1417
9	3.5667	4.0852	4.1329
10	3.4322	4.0651	4.1157
11	3.2981	4.0466	Not Trained Further
12	3.1748	4.0295	Not Trained Further
13	3.0520	4.0129	Not Trained Further
14	2.9349	3.9939	Not Trained Further
15	2.8237	3.9767	Not Trained Further

TABLE III
Training Accuracy for LSTM, GRU, and Transformer Models

Epoch	LSTM Acc.	GRU Acc.	Transformer Acc.
1	0.2215	0.2250	0.2208
2	0.2215	0.2255	0.2220
3	0.2213	0.2256	0.2220
4	0.2221	0.2255	0.2219
5	0.2243	0.2255	0.2218
6	0.2270	0.2255	0.2217
7	0.2330	0.2255	0.2215
8	0.2427	0.2255	0.2215
9	0.2531	0.2255	0.2218
10	0.2680	0.2262	0.2215
11	0.2837	0.2267	Not Trained Further
12	0.2988	0.2264	Not Trained Further
13	0.3149	0.2267	Not Trained Further
14	0.3319	0.2271	Not Trained Further
15	0.3470	0.2276	Not Trained Further

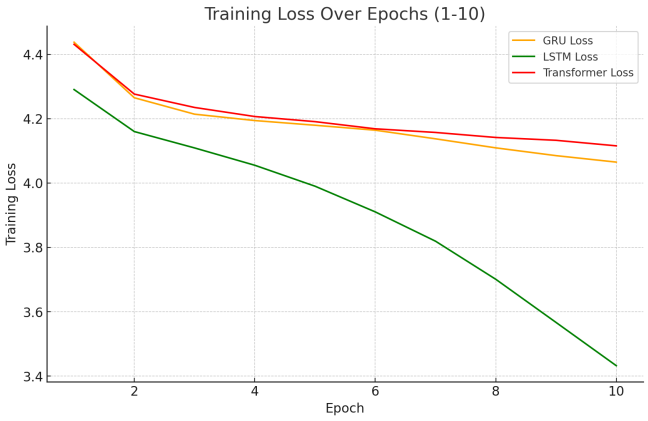

Fig. 2. Training Loss Graph

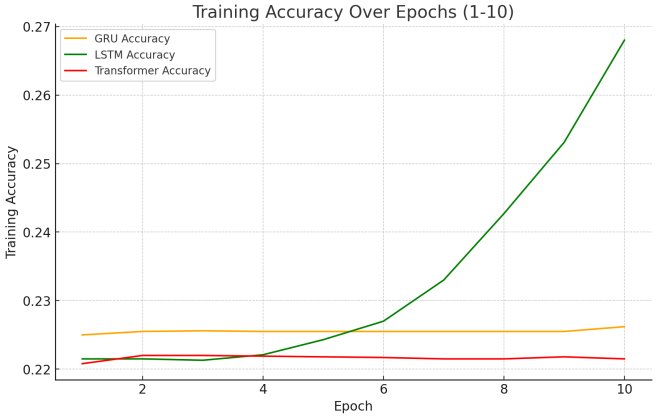

Fig. 3. Training Accuracy Graph

TABLE IV
F1 SCORE RESULTS FOR LSTM, GRU, AND TRANSFORMER MODELS

Epoch	LSTM	GRU	Transformer
10	0.2176	0.0878	0.0835
15	0.3726	0.0939	Not Trained Further
30	0.6737	Not Trained Further	Not Trained Further

A. Transformer's Poor Performance Explained

Transformers are known for their ability to model complex relationships in data using self-attention mechanisms. However, they lack inherent inductive biases, such as recurrence and locality, which are embedded in models like LSTMs and GRUs. While this design allows Transformers to excel with large and diverse datasets, it makes them data-hungry, relying heavily on extensive examples to learn patterns and generalize effectively.

In this study, the dataset comprised only 130 MIDI files (10% of the original dataset), representing a resource-constrained environment. With such limited data:

- **Lack of Inductive Biases:** Transformers struggled to generalize effectively, leading to its poor performance.
- **Complex Architecture:** The sophisticated nature of Transformers, which is advantageous with larger datasets, proved to be a disadvantage here, resulting in stagnant performance during training.

Other studies have observed similar trends in Transformer performance on small datasets. For example:

- In the study "Experimental Study of LSTM and Transformers for Fall Detection on Smartwatches" [17], CNN-LSTM models consistently outperformed Transformers in terms of F1-Score across all dataset sizes. The authors noted, "Transformers, due to the lack of inductive biases, need a lot of data to gain optimal performance."
- Another study, "A Comparison of LSTM and BERT for Small Corpus" [18], found that LSTMs statistically outperformed BERT (a Transformer-based model) on a small, task-specific dataset. The researchers concluded, "For smaller datasets, BERT overfits more than simple LSTM architectures."

Our findings align with these observations. Transformers in this study showed minimal improvement in loss and accuracy during training, stagnating at a loss of 4.1 and an accuracy of 22% by the 10th epoch. Further training did not yield noticeable improvements. In contrast, LSTMs and GRUs demonstrated steady progress, with LSTMs achieving a loss of 1.78 and an accuracy of 53% by the 30th epoch. This difference underscores the Transformers' inability to adapt to the constraints of small limited data sets.

TABLE V
TRAINING DURATION FOR LSTM, GRU, AND TRANSFORMER MODELS

Model	Training Time per Epoch
LSTM	35 minutes
GRU	15 minutes
Transformer	25 minutes

The training durations presented in Table V further illustrate the computational efficiency of the models. GRU trained significantly faster (15 minutes per epoch) than both Transformer (25 minutes per epoch) and LSTM (35 minutes per epoch). However, this speed came at the cost of performance, as GRU's accuracy and F1 Score remained lower than those of LSTM. The Transformer's intermediate training time did not translate into meaningful performance gains, further supporting the decision to cease training after early stagnation.

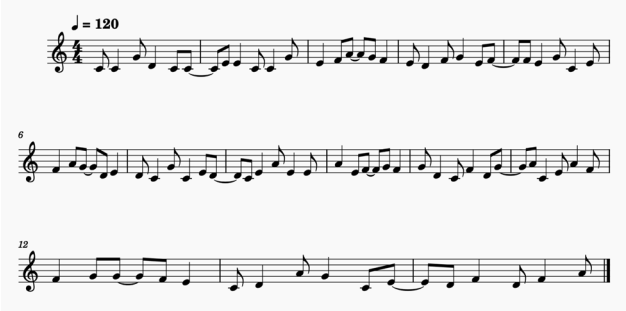

Fig. 4. LSTM Generation visualized in MuseScore

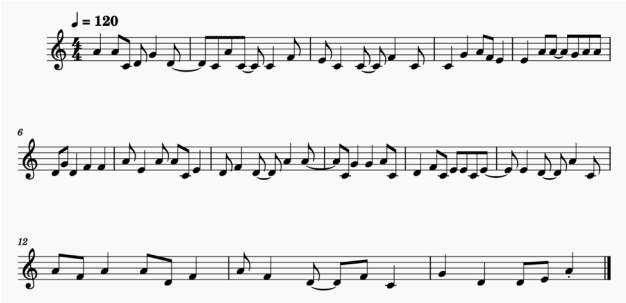

Fig. 5. GRU Generation visualized in MuseScore

B. Qualitative Assessment

We generated examples from each models by following a set of parameters. They are all in treble clef C major, 4/4 time

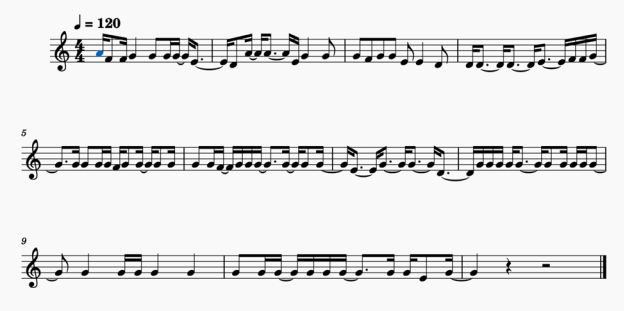

Fig. 6. Transformer Generation visualized in MuseScore

signature, 75 total notes with durations of 1.0 and 0.5. In terms of note pattern generation, visualized in Figures 4, 5, and 6, we can see a stark difference in how they generate.

Both LSTM and GRU are able to generate some interesting patterns. However, in the case for Transformer, we can see that it is able to generate something at the start, but it will slowly devolve into repeating in this case, the note G4 with small derivations. When investigating its probability distribution during debugging, the note G4 was seen with a probability of 94% generating. Attempts to counteract this issue, such as excluding G4 from the probability distribution, resulted in the model transferring this repetitive behavior to another note. This behavior underscores the Transformer's reliance on larger datasets to effectively model diverse patterns.

V. Conclusion and Future Work

To conclude, this research evaluated the performance of three different deep learning architectures, which are LSTM, GRU, and Transformer, for music note pattern generation by using a small data set of 130 MIDI files.

From those models, LSTM shows as the most effective, both showing good results for its quantitative and qualitative metrics. It achieved the highest F1 score with the best accuracy. GRU was able to train faster with its qualitative metrics being able to keep up with LSTM, its quantitative metrics falls behind compared to LSTM.

Transformers, despite their supposed better performance, was seen performed significantly worse than expected in this study. Their reliance on large datasets and lack of inductive biases, such as recurrence and locality, lead it to issues in its generation ability, which is also reflected in its quantitative metrics. This underscores the limitations of Transformers in small dataset scenarios and highlights the importance of matching model architecture to dataset size and characteristics.

Looking ahead, future research could explore hybrid architectures or fine-tuning techniques to optimize Transformers for small datasets. Additionally, investigating the impact of data augmentation and incorporating more musical features (e.g., dynamics and polyphony) could perhaps enhance the models' performance

Supplementary Codes

All the codes used in this paper can be accessed through the following link: https://github.com/CristovalNS/AI-FinalProject-AllModels

CRediT authorship contribution statement

Cristoval Neo Sasono: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data Curation, Writing - Original Draft, Visualization, Project administration. **Brandon Salim:** Software, Validation, Investigation, Resources **Kenneth Jayadi Yu:** Investigation, Funding acquisition, Writing - Review & Editing. **Feri Setiawan:** Supervision. **Raymond Bahana:** Supervision.

References

[1] H. Zhang, L. Xie, and K. Qi, "Implement music generation with gan: A systematic review," in *2021 International Conference on Computer Engineering and Application (ICCEA)*, pp. 352–355, 2021.

[2] N. Kumar, P. Ashwin, and H. Ananthakrishnan, "Mellisai - an ai generated music composer using rnn-lstms," *International Journal of Machine Learning and Computing*, vol. 10, pp. 247–252, 02 2020.

[3] M. Dua, R. Yadav, D. Mamgai, and S. Brodiya, "An improved rnn-lstm based novel approach for sheet music generation," *Procedia Computer Science*, vol. 171, pp. 465–474, 2020. Third International Conference on Computing and Network Communications (CoCoNet'19).

[4] N. H. Kumar, P. S. Ashwin, and H. Ananthakrishnan, "MellisAI - an AI generated music composer using RNN-LSTMs," *International Journal of Machine Learning and Computing*, vol. 10, no. 2, pp. 247–252, 2020.

[5] H. Kumar and B. Ravindran, "Polyphonic music composition with lstm neural networks and reinforcement learning," 2019.

[6] F. Colombo and W. Gerstner, "Bachprop: Learning to compose music in multiple styles," 2018.

[7] A. Nayebi and L. Vitelli, "Algorithmic music generation with recurrent neural networks," *CS224D Course Reports, Stanford University*, 2015.

[8] V. Thanammal, M. Kishanthini, and M. Gokuldhev, "Generating musical compositions with gru and lstm neural networks," *International Journal of Advanced Trends in Engineering and Management (IJATEM)*, vol. 2, no. 4, pp. 195–202, 2023.

[9] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," 2018.

[10] S.-L. Wu and Y.-H. Yang, "The jazz transformer on the front line: Exploring the shortcomings of ai-composed music through quantitative measures," 2020.

[11] L. N. Ferreira, L. H. Lelis, and J. Whitehead, "Computer-generated music for tabletop role-playing games," 2020.

[12] M. Conner, L. Gral, K. Adams, D. Hunger, R. Strelow, and A. Neuwirth, "Music generation using an lstm," 2022.

[13] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014.

[14] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, C. Hawthorne, A. M. Dai, M. D. Hoffman, and D. Eck, "Music transformer: Generating music with long-term structure," *arXiv preprint arXiv:1809.04281*, 2018.

[15] J. Terven, D. M. Cordova-Esparza, A. Ramirez-Pedraza, E. A. Chavez-Urbiola, and J. A. Romero-Gonzalez, "Loss functions and metrics in deep learning," 2024.

[16] Z. C. Lipton, C. Elkan, and B. Narayanaswamy, "Thresholding classifiers to maximize f1 score," 2014.

[17] S. T. Haque, M. Debnath, A. Yasmin, T. Mahmud, and A. H. H. Ngu, "Experimental study of long short-term memory and transformer models for fall detection on smartwatches," *Sensors*, vol. 24, no. 19, 2024.

[18] A. Ezen-Can, "A comparison of lstm and bert for small corpus," 2020.