Exploring the Effectiveness of RNN-LSTM and RNN-GRU in Monophonic Music Generation Using ABC Notation

Milind Uttam Nemade[1], Satheesh Babu[2], Shakir Khan[3],

[1] Professor, Department of AI-DS, K. J. Somaiya Institute of Technology, Mumbai,

[2] Professor, Faculty of Pharmacy, Lincoln University College, Malaysia,

[3] University Centre for Research and Development, Chandigarh University, Mohali 140413, India and College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, Saudi Arabia,

mnemade@somaiya.edu:https://orchid.org/0000-0002-3051-3056, satheeshbabu@lincoln.edu.my, sgkhancs@gmail.com:https://orchid.org/0000-0002-7925-9191

Abstract: Recent advances in artificial intelligence have significantly influenced the field of music generation, particularly in monophonic melody generation. Researchers have explored AI architectures, including Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU). Here we conducted RNN-LSTM and GRU experiment for evaluation of objective and subjective parameters of monophonic music generation. For objective evaluation we used are percentage of valid songs, presence of musical structure (key, meter, notes), repetition score and length similarity score and for subjective evaluation we used musical coherence and creativity parameters. LSTM is slower to train than GRUs, potentially leading to a high loss. If we could swap the GRU for an LSTM it lowers the loss. High score of key musical scale for GRU indicates songs are structurally sound than LSTM, even if the loss is high. Repetition score for LSTM is lower than GRU, indicating more variety, generation of creative, diverse music, which is crucial for quality. A high score for length similarity suggests the GRU is producing songs of appropriate length, supporting quality even with high loss. Poor musical coherence and low creativity contributing to high loss and model needs more data or training to improve.

Keywords: Music generation; RNN; LSTM; GRU; ABC Notation.

Introduction

Artificial Intelligence (AI) has revolutionized the field of music creation by enabling systems to independently compose, evaluate, and replicate musical content with limited human involvement. Among the various methodologies, AI-based music composition has shown its potential across diverse domains such as education, music therapy, entertainment, and collaborative artistic production [1]. An important differentiation within this domain lies between monophonic and polyphonic music generation. Polyphonic systems focus on generating complex arrangements with multiple notes played simultaneously, whereas monophonic models are confined to creating melodies composed of single-note sequences. Due to its structural simplicity, monophonic generation serves as an ideal entry point for studying musical structure and modelling temporal dependencies [2][16].

Recurrent Neural Networks (RNN) is simple and suitable for learning time-dependent data but fight with vanishing gradient problem which affect long-term dependencies. RNNs [3, 15, 17], along with their advanced versions LSTM [14] and GRUs have shown exceptional capability in handling sequential data, which is very important for music generation. It is a streamlined version of LSTMs, offer comparable performance with fewer parameters, faster training times, and easier optimization, making them attractive for smaller datasets and limited-resource environments. While RNNs are proficient at identifying temporal patterns [4], LSTM and GRU models are designed to address challenges like long-term dependency learning and the vanishing gradient problem. This enables them to better retain information from prior musical notes, which is crucial in producing coherent musical sequences. These sequence-learning capabilities form the core of this research.

The main objective of this work is to conduct a detailed performance assessment of RNN, LSTM, and GRU models in the context of monophonic music generation. The evaluation includes both objective indicators such as repetition score, length similarity, percentage of song validity, and musical structural presence and subjective criteria like musical coherence and creativity. Additionally, the study examines model training efficiency, behavioural patterns, and the aesthetic quality of the generated compositions, thereby offering a thorough understanding of each model's appropriateness for real-world applications in AI-driven music creation.

Related work

Monophonic melody generation is typically framed as a sequence modeling task, where at each time-step t, the goal is to predict the next note event, including both pitch and often duration. Recurrent Neural Networks (RNNs) were among the earliest deep learning methods used for this task. In a basic RNN, the hidden state h_t is updated as follows:

$$h_t = \sigma(Wh_{t-1} + Ux_t + b), \qquad \hat{y}_t = \text{softmax}(V h_t + c),$$

Where x_t is the one-hot encoded input (often the previous note), W, U, and V are learnable weight matrices, and σ is the activation function. Table 1 shows comparison of RNN based models for monophonic music generation based on strengths and limitations.

Table 1: Comparison of RNN-Based Models for Monophonic Music Generation

Model	Strengths	Limitations
RNN	Simple architecture, suitable for short sequences	Poor long-term memory, vanishing gradients
LSTM	Good at learning long-term dependencies, expressive	Slow training, high resource usage
GRU	Faster convergence, fewer parameters, easy tuning	Slightly less expressive than LSTM in complex tasks

LSTM cell helped mitigate this issue by adding gate structure to switch the movement of information [5]. LSTM variants, including the input, forget, and output gates, enable the model to preserve memory over long sequences, facilitating the generation of coherent motifs across many bars of music [6]. Additionally, another approach to LSTM is GRU, using two gates (update and reset) while often providing similar performance with less computational overhead [7].

In practice, modern melody generation models often use stacked LSTMs or GRUs, with two or three layers, each containing 128 to 512 units. For instance, one study used two-layer LSTM architecture with

300 and 200 units to model Bach chorale melodies [8]. RNN-based models like Melody RNN [9], developed by Google Magenta, introduced domain-specific tricks such as lookback inputs (feeding in notes from two bars ago) and attention layers to progress the melody generation class.
The ABC notation dataset [10] is widely used in monophonic music research. It provides symbolic representations of folk melodies and serves as an effective benchmark for model training and evaluation. Other datasets [11-13] include Nottingham, Irish Folk Songs, and JSB Chorales, though the latter contains polyphonic elements and is less suitable for purely monophonic studies.
Table 2: Summary of Common Datasets in Monophonic Music Generation

Dataset Name	Description	Format	Suitability
ABC Notation	Folk melodies in symbolic format	Text-based	High
Nottingham	English folk tunes	MIDI/ABC	High
Irish Folk Songs	Traditional Irish tunes	ABC	High
JSB Chorales	Harmonized chorales by J.S. Bach	MIDI	Medium (contains polyphony)

This review highlights the need for a unified benchmarking framework and dataset pre-processing standard to enable better comparison and reproducibility across different studies in monophonic music generation.

Experimental Methodologies:

Monophonic music generation experiment conduction flow for different AI models is shown in fig.1.

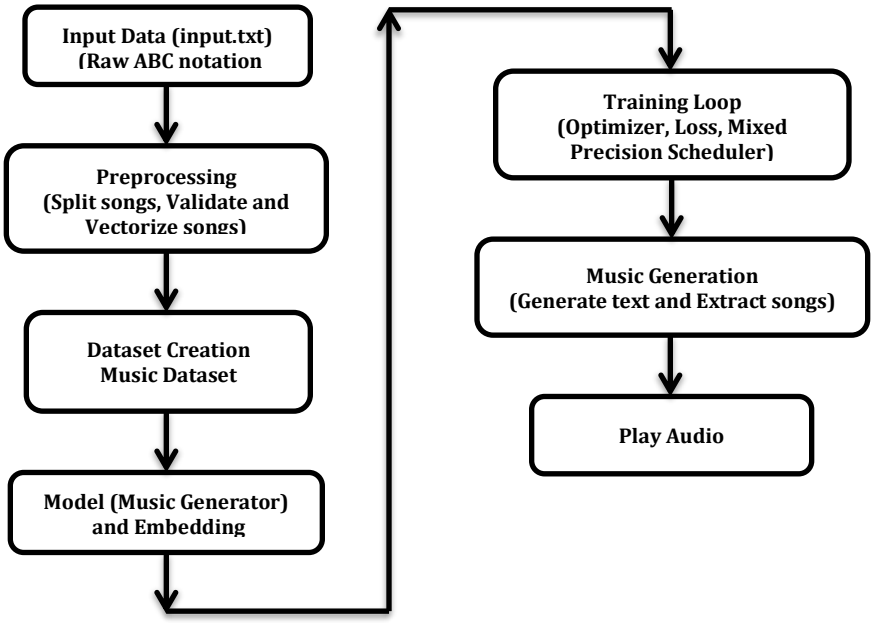

Fig.1 Monophonic Music Generation Experiment Conduction Flow

At start collection of music stored as text, written in a special format called ABC notation. Here each song is like a recipe for music, with instructions for notes, rhythms, and structure. The next step cleans up the music data to make sure it's usable. It checks each song to confirm it has essential parts, a rhythm pattern, and actual notes. It also creates a list of all the unique symbols used in the songs and turns the text into numbers so the computer can work with it. In the next data organization it takes the numbered music data and breaks it into smaller chunks, like short snippets of songs. Each chunk is paired with the next part of the song, so the program can learn to predict what comes next. These

chunks are grouped into batches to make learning efficient. The music model is designed to recognize patterns in the music snippets and guess what should come next in a sequence. It's built with layers that process the data step-by-step, remembering past notes to make better predictions, kind of like a musician recalling a melody. In training process the model studies the music snippets to get better at predicting the next note. It looks at each snippet, makes a guess, checks how wrong it was, and adjusts itself to improve. In song creation phase the trained model generates new music. It starts with a small piece of text (like the beginning of a song) and keeps adding one symbol at a time, using what it learned to decide what fits best. It then splits the long string of symbols into separate songs and makes sure they're formatted correctly. The next step tries to turn the generated songs into sounds you can listen to. In final song evaluation stage, it checks songs, whether they have proper musical parts (like keys and rhythms), how similar they are to the original songs, and whether they're creative or just repetitive.

Objective Evaluation Parameters:

Objective evaluation is data driven metrics which assess the generated songs. These parameters are calculated automatically by analyzing the songs structure, content and similarity to training data [18].

- **Percentage of valid songs:** It checks how many generated songs can be successfully converted into playable music. A song is valid if it can be processed by music conversion tools without errors. If this parameter value is >80% it indicates that model is generating usable music, even if the loss is high.
- **Presence of Musical Structure (Key, Meter, Notes):** It examines whether generated songs contain essential musical component. Key indicates musical scale (e.g. C major). Meter specifies the rhythm pattern (e.g. 4/4 time). Notes are actual musical notes (e.g. C, D, E). High scores here (e.g. >80% for key and meter) mean the songs are structurally sound, even if the loss is high.
- **Repetition Score:** It assesses how repetitive a song is by comparing the number of unique symbols (like notes or rhythms) to the total symbols. A lower score means less repetition, indicating more variety. Repetition score <0.3 is ideal.
- **Length Similarity:** It compares the average length of generated songs (in symbols) to the average length of the original songs in the training data. It ensures that the generated songs match the training data's style. A score >0.8 suggests the model is producing songs of appropriate length, supporting quality even with high loss.

Subjective Evaluation Parameters:

Subjective evaluation is a qualitative assessment to gauge the musical quality and appeal of the generated songs. These parameters are less about numbers and more about how the songs "feel" or sound, often requiring listening or interpretation. Fig. 2 shows subjective analysis work flow for Monophonic Music Generation.

- **Musical Coherence:** It evaluates whether the generated songs sound like real music, with a logical flow of notes, rhythms, and structure. Coherence is what makes music enjoyable. Even with a high loss, coherent songs mean the model is capturing meaningful patterns. A "Good" qualititive rating indicates the songs are musically structured, which is promising.
- **Creativity:** It assesses how original and varied the generated songs are, rather than sounding like copies of the training songs or repeating the same patterns. The repetition score (objective

metric) rate creativity as “High,” “Moderate,” or “Low” (e.g., repetition <0.3 is “High”). A “High” rating suggests the model is producing diverse songs, which is a success even with high loss.

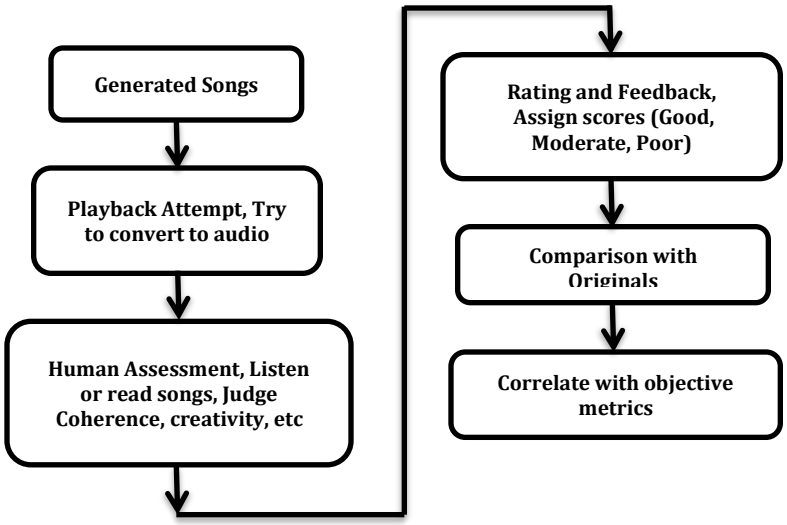

Fig.2 Subjective analysis work flow for Monophonic Music Generation

Results:

We conducted RNN-LSTM and GRU experiment for evaluation of objective and subjective parameters of monophonic music generation. We followed the experiment conduction flow as given in Fig.1. ABC Nottingham Dataset (1200 number of songs) is used to train RNN using group of songs, which we have already generated. The model we created will be based on LSTM and GRU architecture, where to preserve information about the time-based relationships, we used state vector between consecutive characters. The models final output is then fed into a entirely linked dense layer. Here we output a softmax over each character in the vocabulary, and then sample from this distribution to predict the next character.

The RNN, LSTM, and GRU models were trained using the prepared dataset with a batch size of 32. To optimize model performance, categorical cross-entropy was selected as the loss function, while the Adam optimizer was applied with a learning rate of 0.001. This setup proved effective in guiding the learning process. To assess the excellence of the generated music, the RNN-LSTM and RNN-GRU outputs were compared. Some common parameters settings for RNN-LSTM and RNN-GRU models are,

Testing Model:

- Batch Size: 32,
- Sequence Length: 100,
- Vocabulary Size: 95.

Hyperparameter Setting and Optimization:

- Number of training iterations=3000
- Batch Size=10
- Sequence length= 100
- Learning rate= 0.001
- Optimizer: Adam

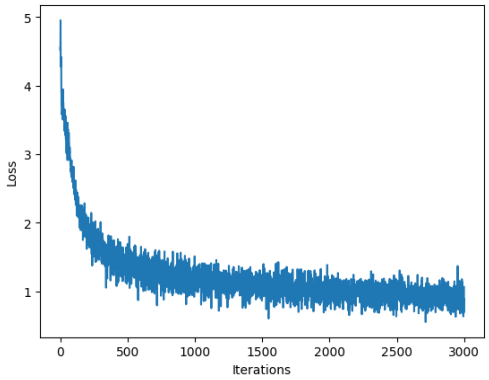

Fig.3. Loss against number of iterations for RNN-LSTM

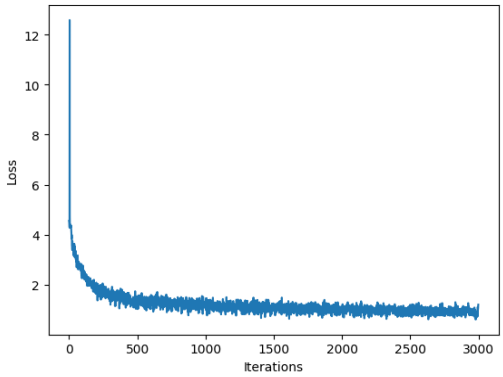

Fig.4. Loss against number of iterations for RNN-GRU

Model Parameters: Sequential Model

- Embedding Dimension: 256
- Number of RNN units: 1024

Table 3: Model Summary for RNN-LSTM

Type of layer	Shape output	Parameters
Embedding	(1, None, 256)	24,320
LSTM	(1, None, 1024)	5,246,976
Dense	(1, None, 95)	97,375

Table 4: Model Summary for RNN-GRU

Type of layer	Shape output	Parameters
Embedding	(1, None, 256)	24,320
GRU	(1, None, 1024)	3,938,304
Dense	(1, None, 95)	97,375

Table 5: Comparison of Model parameters with RNN-LSTM and RNN-GRU Models

Parameters	RNN-LSTM Model	RNN-GRU Model
Scalar loss of untrained model computed	4.5543604	4.5530705
Total	5,368,671 (20.48 MB)	4,059,999 (15.49 MB)
Trainable	5,368,671 (20.48 MB)	4,059,999 (15.49 MB)
Analyse Music Quality by Number of generated songs	3	2

Analyse Music Quality by Average song length	320.33 characters	433 characters
Structural Analysis-Song with key signature	33.33%	100%
Structural Analysis- Songs with meter signature	0%	0%
Structural Analysis- Songs with musical notes	100%	100%
Objective Quality Metrics- Average repetition score (0-1, lower is better)	0.843	0.866
Objective Quality Metrics- Length Similarity to original song	0.785	0.943
Objective Quality Metrics- Percentage of valid songs	100%	100%
Subjective Quality Metrics- Musical Coherence	Poor-Lacking consistent musical structure	Poor-Lacking consistent musical structure
Subjective Quality Metrics- Creativity	Low-Highly repetitive output	Low-Highly repetitive output

Conclusions

In this work we used RNN-LSTM with GRU models for music generation by comparing RNN-LSTM and RNN-GRU models. Due to complex structure of LSTM it takes longer to learn patterns. With 50 epochs of training, an LSTM might not have learned as much as a simpler model. LSTM is slower to train than GRUs, potentially leading to a high loss. If we could swap the GRU for an LSTM it lowers the loss. LSTM require more memory and computing power since they store and update more information at each step than GRU.

Key indicates the musical scale. High score >80% for key for GRU indicates songs are structurally sound, even if the loss is high. Repetition score for LSTM is lower than GRU, indicating more variety, generation of creative, diverse music, which is crucial for quality. A length similarity score >0.8 suggests the GRU is producing songs of appropriate length, supporting quality even with high loss. Subjective quality metrics shows poor musical coherence which suggest sound will be chaotic and model needs more data or training to improve. While low creativity mean the model is stuck on simple patterns, contributing to high loss.

References

1. E. R. Miranda, *Handbook of Artificial Intelligence for Music*. Springer, 2021.
2. J. P. Briot, G. Hadjeres, and F.-D. Pachet, "*Deep Learning Techniques for Music Generation",* Springer, 2019.
3. Alexander Agung Santoso Gunawan, Ananda Phan Iman, Derwin Suhartono, "*Automatic Music Generator Using Recurrent Neural Network*", International Journal of Computational Intelligence Systems, Vol. 13(1), 2020, pp. 645–654.
4. Nishal Silva, Luca Turchet, "*Real-Time Pattern Recognition of Symbolic Monophonic Music*", ACM ISBN 979-8-4007-0968-5/24/09, AM '24, Milan, Italy, pp. 308-317, September 18–20, 2024.
5. S. Hochreiter and J. Schmidhuber, "*Long Short-Term Memory*," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
6. M. Lech and T. Kostek, "*Music Modeling with LSTM Recurrent Neural Networks*," Available: http://aaltodoc.aalto.fi, 2016.

7. Chung, J., Gulcehre, C., Cho, K., & Bengio, Y., "*Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*", *arXiv preprint arXiv:1412.3555*, 2014
8. Dawen, H., & Chia-Yi, W., "*Modeling Bach Chorales with LSTM Networks*", *arXiv preprint arXiv:1609.07846*, 2016
9. Magenta, Google Brain, "*Melody RNN: A Model for Generating Music with Recurrent Neural Networks*", Magenta Project, Retrieved from https://magenta.tensorflow.org/, 2017
10. ABC Notation Dataset. [Online]. Available: https://abcnotation.com.
11. MAESTRO Dataset. [Online]. Available: https://magenta.tensorflow.org/datasets/maestro.
12. Irish Folk Music Dataset. [Online]. Available: https://thesession.org.
13. Nottingham Dataset. [Online]. Available: http://abc.sourceforge.net/NMD/.
14. Raghu Vamsi Uppuluri, **"***Music Generation Using Recurrent Neural Networks: An LSTM Approach to Melodic and Harmonic Composition",* International Journal of Research Publication and Reviews (IJRPR), Vol (5), Issue (10), pp. 1623-1628, October (2024).
15. Hanbing Zhao, Siran Min, Jianwei Fang, Shanshan Bian, "*AI-driven music composition: Melody generation using Recurrent Neural Networks and Variational Autoencoders*", Alexandria Engineering Journal, Springer, pp. 258-270, 2025.
16. Dorien Herremans, Ching-Hua Chuan, and Elaine Chew, "*A Functional Taxonomy of Music Generation Systems*", ACM Comput. Surv. 50, 5, Article 69 (September 2017).
17. JianWu, Changran Hu, Yulong Wang, Xiaolin Hu, and Jun Zhu, "*A Hierarchical Recurrent Neural Network for Symbolic Melody Generation*", arXiv:1712.05274v2 [cs.SD], 5th Sep. 2018.
18. Hanbing Zhao, Siran Min, Jianwei Fang, Shanshan Bian, "*AI-driven music composition: Melody generation using Recurrent Neural Networks and Variational Autoencoders", Elsevier* Alexandria Engineering Journal (AEJ), pp.258–270, 17 February 2025.