Symphony by Sequence: Learning to Compose with Neural Networks

BAX 423 PROJECT REPORT (GROUP 7)

Angel Mary Oviya, Emily Wong, Mikaela McLean, Sharang Saxena, Via Lin

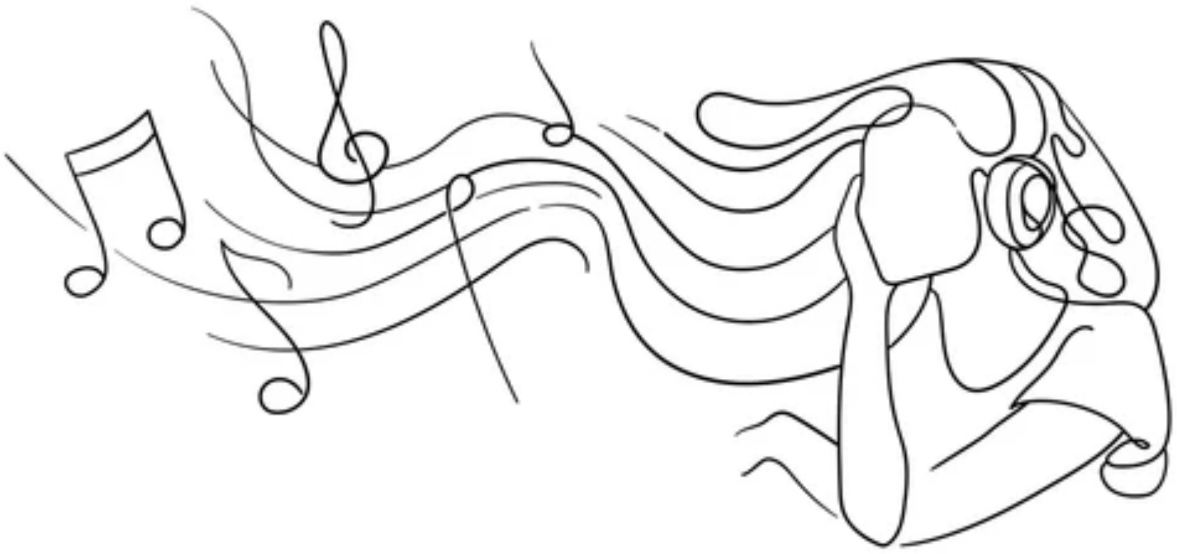


TABLE OF CONTENTS

Abstract

This project evaluates the effectiveness of RNN, LSTM, and Transformer architectures for symbolic music generation. Trained on tokenized classical MIDI data, each model was assessed on loss convergence, musical coherence, and readiness for deployment. The Transformer demonstrated superior performance due to its ability to model long-range dependencies via attention mechanisms. It now powers a Music-as-a-Service prototype that enables dynamic, parameterized music generation.

1 | BUSINESS OBJECTIVE

In a world increasingly shaped by digital experiences, music remains one of the most powerful and most under-automated tools for emotional impact. Whether it's the rising tension of a game, the calm flow of a meditation app, or the pacing of a short-form video, background music defines how content is felt. But behind every well-timed melody is a problem: original, mood-specific music is expensive, slow to produce, and notoriously difficult to scale. This project began with a provocative question: *What if machines could compose on cue?*

We set out to build a generative system that could create expressive, adaptive music in real time using deep learning. Our objective was to evaluate and compare Recurrent Neural Networks (RNN), Long Short-Term Memory networks (LSTM), and Transformer models on their ability to generate symbolic music, with the ultimate goal of powering a Music-as-a-Service (MaaS) platform. This system would allow developers to generate royalty-free music dynamically, parameterized by tempo, mood, and style. No human composers, no licensing bottlenecks, just scalable creativity on demand.

By turning raw data into music and predictive sequences into composition, we reimagined how businesses can access and control one of the most emotionally resonant elements in their digital products: sound.

2 | IMPLEMENTATION

To construct a robust music generation system, we implemented three autoregressive sequence models—RNN, LSTM, and Transformer, each of which was trained from scratch using symbolic musical data encoded in MIDI format. MIDI was chosen for its compact representation of musical events (pitch, duration, velocity, instrument channel), making it ideal for tokenization and sequence modeling. Unlike raw audio, which requires convolutional feature extraction or spectrogram analysis, MIDI simplifies preprocessing and focuses the model's learning on compositional structure rather than acoustics.

Sequences were tokenized into pitch-duration pairs and embedded into a dense vector space, where each token's representation was drawn from an embedding matrix $E \in R^{V \times d}$, with V representing vocabulary size and d the embedding dimension. These embeddings served as inputs to each model architecture.

The RNN was implemented using a standard tanh activation and learned hidden state transitions:

$$h_t = tanh(W_x x_t + W_h h_{t-1} + b)$$

where x_t is the input embedding at time t, W_x and W_h are leanable weight matrices, and b a bias vector.

The **LSTM** introduces a memory cell $c_t$and gating mechanisms:

Forget gate: $f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$ Updated memory: $c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t$

Input gate: $i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$ Output gate: $o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$

Candidate state: $f_t = tanh(W_c[h_{t-1}, x_t] + b_c)$ Updated hidden state: $h_t = o_t \odot tanh(c_t)$

Here, σ is the sigmoid function and $\odot$ denotes element-wise multiplication. In contrast, the Transformer model eliminates recurrence entirely and relies on self-attention mechanisms:

$$\text{Attention}(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

where $Q, K, V \in R^{L \times d_k}$ are the query, key, and value matrices derived from input embeddings, d_k is the dimensionality of the keys, and L is the sequence length. Sinusoidal positional encodings were added to retain order information in the absence of recurrence. All models were trained to minimize sparse categorical cross entropy. Output logits z were passed through a temperature-scaled softmax:

$$p_i = \frac{e^{z_i/T}}{\Sigma_j e^{z_i/T}}$$

with lower temperatures yielding more deterministic generations and higher temperatures increasing creativity.

We encountered several implementation challenges. First, due to limited computational resources, we could not leverage pre-trained models or large-scale datasets such as MAESTRO or Lakh MIDI. Instead, we trained all models from scratch on a curated classical MIDI dataset. All models were trained for 10 epochs to ensure consistency under limited compute resources. While the Transformer is capable of longer training, we observed stable convergence within this range.

Another challenge was balancing model complexity with generation quality. Transformers, though powerful, are memory-intensive. Colab environments constrained batch size and sequence length, necessitating gradient clipping and learning rate scheduling to maintain training stability. We also observed that attempts to generate separate tracks for multiple instruments introduced sparsity in the token space, degrading model performance. Thus, we opted to generate a single token stream simulating dual-instrument polyphony through pitch range and rhythmic density, rather than interleaving instrument-switch tokens that were underrepresented in the training data. This sparsity reflects a core challenge in symbolic music modeling: as complexity increases (e.g., multi-track or multi-instrument output), so does the token vocabulary and data imbalance, making it harder for models to learn musically coherent patterns.

Despite these limitations, each model successfully learned symbolic musical structures. By combining lightweight preprocessing with principled architecture design, we laid the foundation for real-time music generation that is both computationally efficient and musically expressive.

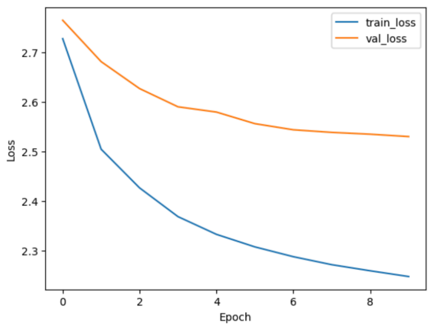
3 | METRICS OF SUCCESS

To assess model performance, we trained each architecture from scratch on a curated classical MIDI dataset using next-token prediction. All models minimized sparse categorical cross-entropy and were trained with early stopping and learning rate scheduling. All models, including the Transformer, were trained for 10 epochs to enable fair comparison under the same training budget. Despite its increased depth and capacity, the Transformer achieved reliable convergence within this window. The following table summarizes key training metrics and output characteristics for each model.

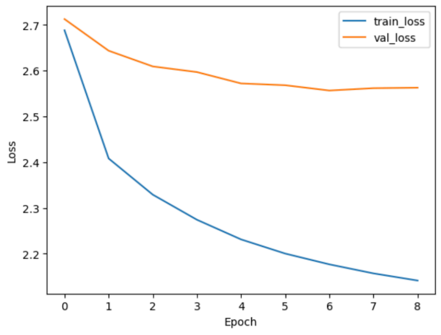
Model	Architecture	Total Parameters	Training Epochs	Best Validation Loss	Final Accuracy	Generation Quality (T=0.8)
RNN	2-layer GRU with embedding and dropout	669,975	10	~2.53	~31.8%	Basic melodic structure
LSTM	2-layer LSTM with embedding and dropout	881,943	10	~2.56	~33.6%	Improved rhythm and phrasing
Transformer	2-layer Transformer with attention, positional encoding, and FFN	726,999	10	~2.56	~29%	Coherent, long range melodic structure

In addition, the loss curves below illustrate how training and validation loss evolved over the 10 training epochs for all three models. The Transformer exhibited steady and stable convergence, with a smooth reduction in loss and minimal overfitting. The LSTM reached stability within 8–10 epochs, while the RNN showed greater volatility and a less consistent learning trajectory throughout training.

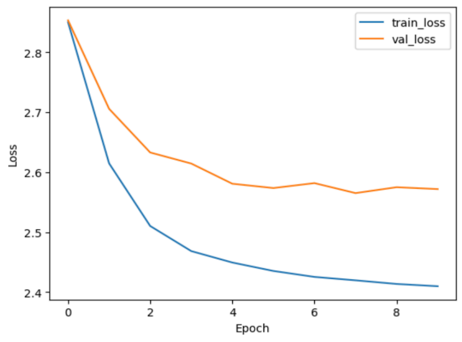
RNN



LSTM



Transformer



For subjective evaluation, each model generated 200-token sequences at T = 0.8. Five human evaluators rated the samples on a 1–5 rubric assessing melodic continuity, harmonic structure, rhythmic coherence, and phrase resolution. The Transformer received the highest average rating (4.3), followed by LSTM (3.5) and RNN (2.7), confirming its superior ability to model long-range musical dependencies.

4 | ROLE OF ANALYTICS

Analytics guided every phase of the project, from preprocessing and feature engineering to architectural tuning and evaluation. Exploratory analysis of token frequency distributions led to pruning underrepresented pitch-duration pairs and informed decisions about fixed sequence lengths. Entropy scores and n-gram diversity metrics were used to quantify output creativity, while attention heatmaps in the Transformer model provided interpretability by revealing how prior musical context influenced token prediction.

The core modeling task, next-token prediction in symbolic sequences, is inherently predictive. Each model estimates the conditional probability, generating the next token based on historical context. This probabilistic framing enabled sequence continuation and real-time music composition.

The project also incorporated prescriptive elements. By evaluating multiple model architectures and sampling temperatures, we prescribed optimal configurations for specific use cases: RNNs for short motifs, LSTMs for stable rhythm, and Transformers for long-form melodic development. These insights, combined with parameter tuning and human-in-the-loop review, guided design choices that enhanced both musicality and operational performance.

5 | EXECUTION AND RESULTING OUTPUT

All models were implemented in TensorFlow and trained using Colab notebooks with GPU acceleration. During generation, the system used a sliding-window autoregressive loop: given the last 32 tokens, it predicted the next token using argmax or temperature sampling, then appended it to the sequence and repeated. Outputs were decoded to MIDI format and post-processed to ensure rhythmic validity.

While initial experiments explored splitting musical content across three separate tracks to simulate multi-instrument composition, this approach was not adopted in the final model. Instead, the Transformer's output stream combines two musical lines within a single track. This structure was chosen to simplify token conditioning and preserve sequential coherence. Though the codebase contained hooks for additional instrument layering, early tests indicated a loss of generation quality when conditioning on instrument-switching tokens due to data sparsity and vocabulary inflation. As a result, the final implementation uses one combined symbolic track that simulates polyphony through pitch range and rhythm density, effectively mimicking two-instrument output while maintaining modeling efficiency.

6 | IMPLEMENTATION AND SCALE

The successful development of the Transformer model provides a strong foundation for a scalable Music-as-a-Service (MaaS) platform. While the current implementation runs effectively within a Google Colab notebook, the roadmap to transition this prototype into a production-ready service would require several key steps in deployment, optimization, and user interface design.

Future work would prioritize deploying the Transformer model via a Flask API, containerizing it with Docker, and serving it through a high-performance system such as TensorFlow Serving. This API would accept inputs such as a seed sequence, generation length, and a temperature parameter to control creativity, returning the final composition in MIDI format. To support real-time applications in gaming and content creation, critical optimizations including caching for frequent requests, model quantization, and batch inference would be explored to reduce latency. A robust post-processing pipeline would remain essential to refine the output by removing duplicate notes and correcting timing errors.

A user-facing interface would be developed to allow deeper customization, including parameters for mood, key, and length. To broaden the system's musical range, the training dataset would be expanded beyond solo piano to include multiple instruments and styles. These improvements would be supported by more powerful infrastructure such as cloud-based TPUs or multi-GPU systems, along with a more sophisticated evaluation framework incorporating structured metrics like n-gram overlap, tonal centroid distance, and entropy diversity. Together, these upgrades will enhance output variety, training efficiency, and musical fidelity, moving the system from prototype to production-ready MaaS infrastructure.

7 | CONCLUSION

This project demonstrates the technical and commercial viability of AI-generated symbolic music. Through comparative modeling and rigorous evaluation, we identified the Transformer as the optimal architecture for generating structured, expressive, and stylistically consistent musical sequences. Its ability to simulate multi-instrument compositions and deliver real-time performance makes it suitable for integration into scalable music-generation systems. This work lays the foundation for a future in which generative models support creative workflows not by replacing human composers, but by enabling dynamic, data-driven musical experiences at scale.