Scaling Laws for Transformer Models in Symbolic Music Generation

CS-GY 6923: Machine Learning

Name: Eswanth Sriram Chengalasetty

Abstract

This project studies scaling laws for language models trained on symbolic music represented in ABC notation. We train families of decoder-only Transformer (GPT) models and recurrent (LSTM) baselines while varying model parameter count N and holding data and training duration constant. Validation loss is observed to follow a power-law relationship of the form $L = aN^{-\alpha} + c$. Results demonstrate that Transformer architectures exhibit more favorable scaling behavior than RNNs at comparable parameter counts. The largest feasible Transformer model was trained to convergence and evaluated both quantitatively and qualitatively through unconditional and conditional music generation.

1 Introduction

Scaling laws offer a powerful way to understand how model performance changes as capacity, data, and compute increase. In language modeling, these laws have shown that larger models tend to improve in a predictable manner, shaping how modern architectures are designed. However, most existing work focuses on natural language, and it remains unclear whether the same scaling behavior applies to other structured sequence domains.

Symbolic music generation provides a compelling test case. Unlike natural language, symbolic music representations must follow strict syntactic rules in order to be valid. A single misplaced character can break an entire piece. At the same time, music contains long-range structure, including repetition, rhythm, and motifs that span hundreds of tokens. This combination of fragile syntax and long-context dependencies makes symbolic music both challenging and well suited for studying scaling behavior.

In this project, I investigate scaling laws for symbolic music generation using ABC notation, a compact text-based format that explicitly encodes musical structure. Starting from the Lakh MIDI Dataset, I built a preprocessing pipeline to convert MIDI files into ABC notation and clean the resulting data. The final corpus contains approximately 13 billion tokens. To preserve ABC's strict syntax, I use character-level tokenization, resulting in a vocabulary of 97 unique characters.

To study scaling behavior, I train a series of decoder-only Transformer (GPT-style) models with increasing parameter counts, along with four LSTM-based RNN models as a baseline. All models in the scaling study are trained on the same dataset using the same tokenization scheme and for exactly one epoch (About 100M), allowing for a controlled comparison focused on model capacity and architecture. After identifying the best-performing configuration, I further train the largest Transformer model to convergence by resuming training from its initial checkpoint.

Model performance is evaluated using validation loss and test-set perplexity, along with qualitative analysis of generated music. Generated ABC samples are converted back into MIDI or audio to verify syntactic validity and are manually listened to to assess musical coherence. I also evaluate conditional generation by prompting the model with structured ABC headers specifying musical attributes such as time signature and key.

Overall, this project demonstrates that Transformers exhibit clear power-law scaling behavior in the symbolic music domain and scale more efficiently than recurrent models under identical training constraints. The results suggest that scaling principles developed for language models extend naturally to symbolic music generation.

2 Data and Preprocessing

2.1 Dataset

The dataset used in this project is derived from the full Lakh MIDI Dataset, which contains approximately 179,000 MIDI files spanning a wide range of musical styles, genres, and levels of compositional complexity. This dataset was chosen primarily due to its scale and diversity, making it well suited for studying how model performance changes as a function of model capacity. Rather than relying on an existing symbolic music corpus, the entire data preprocessing pipeline was constructed as part of this project, beginning from raw MIDI files.

All MIDI files were converted into symbolic music representations using the command-line tool `midi2abc`. This conversion process was fully automated and applied uniformly across the dataset, with no manual intervention or hand-editing of individual files. The conversion preserved important musical information such as key signatures, time signatures, and multi-voice structure when present, allowing both monophonic and polyphonic pieces to be represented in the final corpus. ABC notation was selected as the target representation due to its compact, text-based format and its explicit encoding of musical structure, including rhythm, meter, and key, which makes it particularly suitable for sequence modeling.

During the conversion process, approximately 5,000 MIDI files were found to be corrupted or failed to convert successfully and were therefore discarded. These failures were handled conservatively, and no additional filtering was applied based on musical quality, genre, or structure. As a result, the final dataset remains broadly representative of the original Lakh MIDI collection. All preprocessing steps were fully automated, and no manual curation of individual files was performed at any stage.

2.2 Tokenization

Character-level tokenization was employed throughout all experiments. This decision was motivated by the strict and fragile syntax of ABC notation, where single characters often carry significant musical meaning. Symbols such as bar lines, repeats, accidentals, and duration markers play a critical role in defining musical structure, and merging them into larger subword units risks producing syntactically invalid or musically uninterpretable output. For this reason, subword tokenization approaches such as Byte Pair Encoding were intentionally avoided.

By operating at the character level, the model is forced to learn the exact syntactic structure required to produce valid ABC notation. This approach also aligns naturally with the symbolic nature of the data and avoids introducing unnecessary complexity into the preprocessing pipeline. The final vocabulary consists of 97 unique characters, constructed from the entire cleaned dataset. No special tokens such as padding or end-of-sequence markers were introduced, and the same fixed vocabulary was shared across all models to ensure consistency in training and evaluation.

2.3 Dataset Statistics

After conversion and cleaning, the final corpus contained approximately 13 billion characters, substantially exceeding the minimum requirement of 100 million tokens for the scaling study.

This large corpus allowed all models in the scaling experiments to be trained on the same dataset while holding the amount of data constant across different model sizes.

The dataset was partitioned into fixed splits of 98% training, 1% validation, and 1% test. These splits were applied consistently across all models and experiments, ensuring that differences in performance could be attributed to model capacity and architecture rather than differences in data exposure. Aside from removing files that failed conversion or lacked required ABC headers, no additional filtering based on sequence length or musical content was performed. This decision was made to preserve the natural distribution of musical structures present in the original dataset and to avoid introducing subjective preprocessing heuristics.

3 Methods

3.1 Model Architectures

This project evaluates two families of sequence models commonly used for language modeling tasks: Transformer-based models and recurrent neural networks. Both architectures were adapted to operate on symbolic music represented in ABC notation using character-level prediction.

The Transformer models follow a decoder-only architecture based on the standard `nanoGPT` implementation. These models use causal self-attention, ensuring that predictions at each timestep depend only on previously observed tokens. Each Transformer block consists of multi-head self-attention followed by a position-wise feedforward network, with pre-layer normalization applied before each sub-layer. This configuration is widely used in modern GPT-style models due to its training stability and strong empirical performance. The context window was fixed at 256 tokens for all Transformer models to maintain consistency across scaling experiments. Token embedding and output projection layers were implemented separately, without weight tying.

As a recurrent baseline, Long Short-Term Memory (LSTM) networks were trained for comparison. The LSTM models were single-directional, as bidirectional recurrence is incompatible with causal sequence modeling. Each model consists of an embedding layer followed by a stack of LSTM layers and a linear output projection. Hidden states were reset between batches, treating each batch as an independent sliding window over the data. Gemini has aided me in this script's generation to train. This setup mirrors the training paradigm used for the Transformer models and allows for a controlled comparison between architectures. Dropout was applied within the LSTM layers to mitigate overfitting.

3.2 Scaling Configurations

To study scaling behavior, multiple model sizes were trained within each architecture family while holding the dataset and training protocol fixed. For the Transformer models, five configurations were evaluated, ranging from approximately 1.4 million parameters to approximately 107 million parameters. These configurations varied the number of layers, attention heads, and embedding dimensions while keeping the context length and tokenization scheme constant.

For the recurrent baseline, four LSTM configurations were trained with increasing depth and hidden dimension, resulting in parameter counts spanning a comparable range. Although Transformers and RNNs differ fundamentally in their computational mechanisms, these configurations were chosen to enable comparison at similar parameter scales. All model configurations used in the scaling study are summarized in Table 1.

Table 1: Model configurations used

Model	Type	Layers	Heads	Parameters
Tiny	GPT	3	3	∼1.4M
Small	GPT	4	4	∼5.0M
Medium	GPT	6	8	∼20M
Large	GPT	8	12	∼50M
XL	GPT	12	12	∼107M
RNN-Tiny	LSTM	2	–	∼1.1M
RNN-Small	LSTM	3	–	∼3.6M
RNN-Medium	LSTM	4	–	∼10.7M
RNN-Large	LSTM	5	–	∼42.2M

3.3 Training Protocol

All models in the scaling study were trained under identical conditions to isolate the effect of model capacity on performance. Each model was trained for exactly one epoch (about 100M tokens). This ensured that all models were exposed to the same number of training tokens, satisfying the requirements for empirical scaling-law analysis.

The same character-level tokenization scheme, batch size measured in tokens, and cosine learning rate decay schedule were used across all scaling runs. Optimization was performed using the AdamW optimizer, and the training objective was standard cross-entropy loss over next-character prediction. Transformer models were trained using mixed-precision on CUDA-enabled GPUs for computational efficiency, while LSTM models were trained in full precision. Whenever possible, both architectures were trained on comparable hardware to avoid confounding architectural comparisons with hardware effects.

Following the scaling experiments, the largest Transformer model (GPT-XL) was selected as the best-performing configuration. This model was resumed from its scaling-study checkpoint and trained for additional iterations (12,000 steps) to further reduce validation loss. No additional hyperparameter tuning was performed during this stage; the extended training represents a continuation of the scaling study rather than a separate optimization process.

3.4 Evaluation Metrics

Model performance was primarily evaluated using validation cross-entropy loss, which serves as the standard metric in scaling-law analyses. Scaling behavior was analyzed by fitting the functional form $L = aN^{-\alpha} + c$ to validation losses as a function of model parameter count.

For the final GPT-XL model, additional evaluation was performed on the test set. Gemini has aided me in creating the script to generate samples from the best model. Test-set perplexity was computed to provide a more interpretable measure of sequence modeling performance. Qualitative evaluation was also conducted by generating symbolic music samples in ABC notation and listening on a ABC notation player website. Generated outputs were assessed for syntactic validity, defined as successful playback, as well as for overall musical coherence through manual listening.

4 Results

This section presents the empirical results of the scaling experiments and qualitative evaluation of symbolic music generation. We analyze scaling behavior for Transformer models, compare them to recurrent baselines, and evaluate generated samples from the best-performing model.

4.1 Transformer Scaling

Figure 1 shows validation cross-entropy loss as a function of model parameter count for the Transformer (GPT) models trained in the scaling study. The five models span over two orders of magnitude in size, from approximately 1.4×10^6 to 1.1×10^8 parameters. All models were trained for exactly one epoch on the same training split, ensuring a controlled comparison.

Validation loss decreases monotonically as model size increases. When plotted on log-log axes, the relationship between loss and parameter count is approximately linear, consistent with a power-law relationship of the form

$$L = aN^{-\alpha} + c.$$

A nonlinear fit yields a scaling exponent of $\alpha \approx 0.593$, which is comparable to values reported in natural language modeling. This suggests that symbolic music generation exhibits similar scaling behavior despite the stricter syntactic constraints imposed by ABC notation.

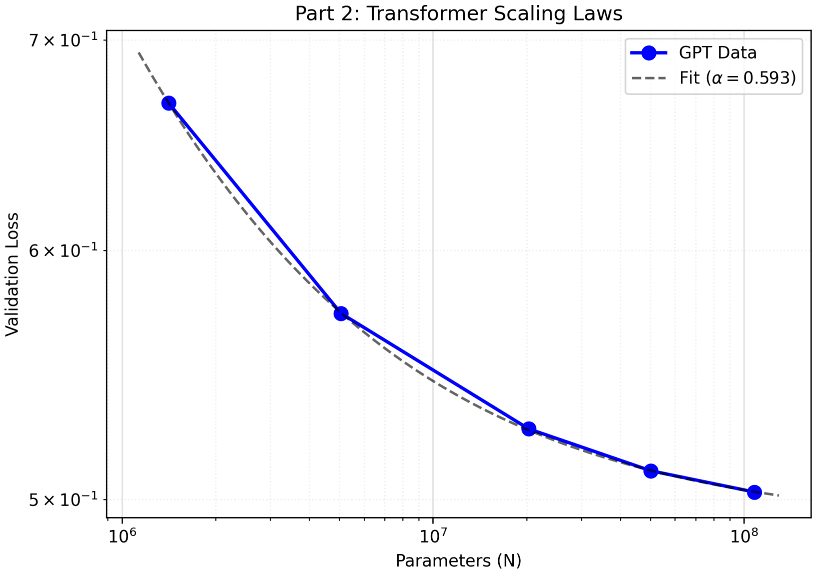

Figure 1: Transformer (GPT) scaling results. Validation cross-entropy loss versus parameter count on log-log axes. The dashed line shows the fitted power-law curve with exponent $\alpha = 0.593$, indicating predictable scaling behavior as model size increases.

4.2 RNN Scaling and Architectural Comparison

Figure 2 presents the corresponding scaling results for the LSTM-based recurrent neural networks. Four RNN configurations were evaluated under the same training conditions as the Transformer models.

While validation loss initially decreases as RNN model size increases, the overall improvement is modest. Notably, the largest RNN configuration exhibits a slight increase in validation loss, suggesting diminishing returns and potential optimization or capacity limitations at larger scales. This behavior contrasts with the smooth and monotonic scaling observed for the Transformer models.

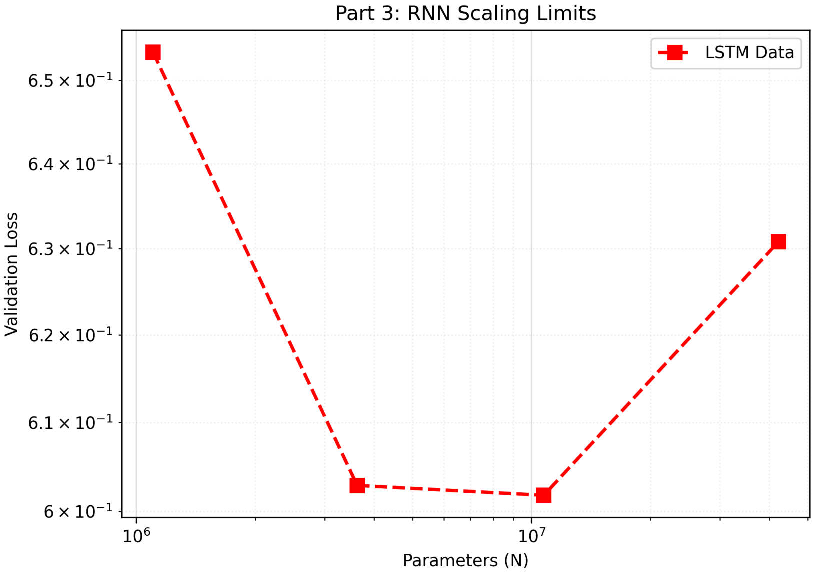

Figure 2: RNN (LSTM) scaling results. Validation cross-entropy loss versus parameter count. Loss decreases weakly with model size and shows non-monotonic behavior at larger scales, indicating limited scaling efficiency compared to Transformers.

Figure 3 directly compares Transformer and RNN scaling behavior on the same axes. Across the comparable parameter counts, Transformer models consistently achieve lower validation loss than RNN models. The performance gap is already visible at smaller scales and widens substantially as model size increases, highlighting the superior scaling efficiency of self-attention-based architectures.

4.3 Sample Generation and Qualitative Evaluation

The largest Transformer model (GPT-XL) was trained for additional iterations beyond the scaling study and used to generate symbolic music samples in ABC notation. A decent of generated samples were syntactically valid and could be successfully played audio but some only few notes and some did not play at all, with an estimated validity rate of approximately 75%. Please find the playable audio files in the google drive provided.

Qualitatively, many samples exhibit coherent musical structure, including consistent key and meter, repetition, and simple phrase-like organization. Conditional generation experiments show that the model generally follows structured prompts such as specified time signatures and key

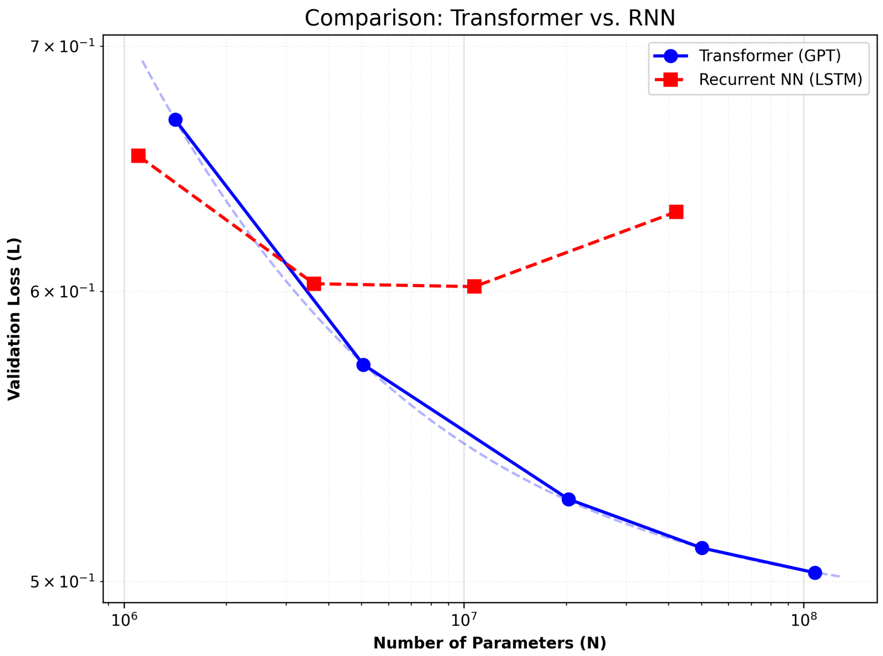

Figure 3: Direct comparison of Transformer and RNN scaling behavior. Transformer models achieve lower validation loss at all tested parameter scales, and the gap increases as model size grows.

signatures, with rhythmic structure particularly influenced by meter prompts. While long-range global structure remains imperfect, these results indicate that increased model capacity leads to both quantitative and qualitative improvements.

5 Discussion

This project provides clear evidence that architectural choice plays a central role in how models scale for symbolic music generation. While both Transformers and recurrent models benefit from increased capacity, the results show that Transformers scale more predictably and efficiently under identical training conditions.

5.1 Key Insights from Scaling

The Transformer models exhibit a smooth and monotonic reduction in validation loss as model size increases, closely following a power-law relationship. The fitted scaling exponent $\alpha \approx 0.593$ is comparable to values observed in natural language modeling, suggesting that symbolic music generation obeys similar scaling dynamics despite being a non-linguistic and highly structured domain. This result was somewhat surprising given the strict syntax of ABC notation, but it reinforces the idea that scaling laws are robust across sequence modeling tasks.

In contrast, the LSTM-based models showed much weaker scaling behavior. Although validation loss initially decreased with increased model size, improvements were modest and became inconsistent at larger scales. This indicates that simply adding parameters to a recurrent architecture does not yield the same performance gains as scaling a Transformer. The difference highlights how much architectural inductive bias matters when modeling long and structured sequences like music.

5.2 Implications for Music Modeling

Music contains long-range dependencies such as repeated motifs, rhythmic patterns, and harmonic structure that often span dozens or hundreds of tokens. Transformers are particularly well suited to this setting because self-attention allows distant parts of a sequence to interact directly. The results suggest that this global context modeling is critical for symbolic music generation and is a major reason Transformers outperform recurrent baselines.

Recurrent models, even when scaled up, must compress long-term information into a fixed-size hidden state. The observed scaling limits suggest that this mechanism struggles to capture broader musical structure, especially under fixed training budgets. As a result, Transformers not only achieve lower loss but also make better use of each token seen during training.

5.3 Design Choices and Practical Constraints

Character-level tokenization was a deliberate design choice in this project. Although it significantly increases sequence length, it preserves the fragile syntax of ABC notation, where individual characters encode meaningful musical information. The qualitative results suggest that this trade-off was worthwhile, as many generated samples are audibly coherent and respect basic musical constraints.

From a practical standpoint, scaling to larger models introduced real computational challenges. While smaller models could be trained locally, the largest Transformer model exceeded the capabilities of the available hardware. Training the GPT-XL model required access to an NVIDIA A100 GPU via Google Colab, highlighting a common limitation in modern deep learning research: meaningful scaling experiments increasingly depend on access to specialized compute resources. This constraint ultimately shaped which models were feasible to train and evaluate.

5.4 Limitations and Future Directions

Despite strong scaling results, several limitations remain. Character-level modeling places a heavy burden on the model to learn musical structure from very long sequences, which likely contributes to imperfect long-range coherence and occasional syntactic errors. Additionally, while many samples are audibly playable, strict syntactic validity is not guaranteed, particularly for unconditional generation.

Future work could explore hierarchical or music-aware tokenization schemes to improve efficiency and global structure. Scaling the dataset further and training for longer durations may also improve musical coherence. Finally, incorporating explicit musical constraints or structure-aware objectives could help guide generation toward more consistent and expressive compositions.

6 Conclusion

This project showed me that scaling laws are not just a property of natural language models, but extend cleanly into symbolic music generation as well. By training both Transformer and recurrent models under the same constraints, I was able to see firsthand how model architecture shapes scaling behavior. Transformer models scaled smoothly and predictably with size, while RNNs showed much weaker and less reliable improvements as parameters increased. This made it clear that simply adding capacity is not enough; how a model processes information matters just as much as how large it is.

Working with symbolic music also gave me a new appreciation for how challenging structured sequence modeling can be. Even with strict syntax and long-range dependencies, larger Transformer models were able to generate music that actually sounds musical, especially when given simple conditioning prompts. It was surprising to see recognizable rhythmic patterns, repetition, and basic structure emerge from character-level modeling alone. At the same time, the imperfections in the output made it clear that there is still a large gap between local coherence and true long-form musical structure.

One of the biggest takeaways from this project was how real computational constraints shape research. Training the largest model required moving to an A100 GPU on Google Colab, since my local machine simply could not handle models at that scale. This experience made the trade-offs behind scaling very tangible and emphasized why access to compute is such a limiting factor in modern deep learning work.

Overall, this project was a strong learning experience in both theory and practice. It reinforced why scaling laws are such a powerful tool for understanding model behavior and showed that music is a rich and promising domain for further exploration. There is still plenty of room to improve tokenization, structure modeling, and training efficiency, and the results here make me excited to keep experimenting with larger models, better representations, and longer context windows in future work.

References

[1] Eswanth Sriram Chengalasetty. *Scaling Laws for Symbolic Music Generation (Code Repository)*. GitHub, 2025. `https://github.com/Eswanth0131/ml_music_project`

[2] Eswanth Sriram Chengalasetty. *Scaling Laws for Symbolic Music Generation (Datasets and Model Artifacts)*. Google Drive, 2025. `https://drive.google.com/drive/folders/1m3_EeTz2uSYvga_28-SJ3BGKcA8a0pUh`