New York University Tandon School of Engineering
Computer Science and Engineering

CS-GY 6923: Project

Shuhan Cai sc12227@nyu.edu
Github: CSGY-6923-Project

Introduction

Understanding how neural language models scale with model size has become a central topic in modern machine learning research. While scaling laws for natural language models are now well established, far less is known about how these laws transfer to non-linguistic symbolic domains such as music. Symbolic music presents a particularly interesting testbed: it exhibits strict syntactic rules, hierarchical temporal structure, and long-range dependencies that differ fundamentally from natural language.

In this project, I investigate scaling laws for neural sequence models trained on symbolic music represented as ABC notation. Using large-scale data derived from the Lakh MIDI Dataset, I study how model size affects validation loss, learning efficiency, and musical structure emergence. I conduct a systematic comparison between decoder-only Transformer models and recurrent neural networks (LSTMs), and analyze how architectural choices influence both quantitative metrics and qualitative musical outputs.

First, I build a complete preprocessing pipeline converting raw MIDI files into a large-scale textual music corpus as ABC notation. Second, I empirically derive scaling trends for Transformers and RNNs under controlled training conditions. Third, I analyze the musical characteristics of generated samples, highlighting both the strengths and limitations of autoregressive models in learning musical form.

Data

Dataset selection. I use the Lakh MIDI Dataset (LMD), which contains over 170,000 MIDI files spanning a wide range of genres and compositional styles. LMD is well suited for scaling studies due to its size, diversity, and permissive licensing. To enable language-model-style training, all MIDI files were converted into ABC notation, a human-readable textual representation of music.

Conversion pipeline. MIDI files were converted to ABC notation using the `midi2abc` library. During conversion, corrupted or malformed MIDI files were automatically discarded. Extremely short and long sequences were filtered out to fit within the model context window.

Tokenization strategy. I adopt a character-level tokenization scheme over raw ABC text. Each character (including musical symbols, bar markers, whitespace, and newlines) is treated as a token. This choice results in a compact vocabulary of 99 symbols and avoids the need for domain-specific token engineering.

Alternative tokenization approaches considered include note-level tokens and music-aware composite tokens. While these approaches may encode higher-level musical semantics, they significantly complicate preprocessing and introduce ambiguity across voices and time signatures. Character-level tokenization provides maximal flexibility and ensures syntactic validity is learned directly from data.

Example:

```
X:1
T:Example Tune
```

```
M:4/4
L:1/8
K:C
C D E F | G A B c |
```

Under character-level tokenization, this sequence is represented as:

```
['X', ':', '1', '\n',
 'T', ':', 'E', 'x', 'a', 'm', 'p', 'l', 'e', ' ', 'T', 'u', 'n', 'e', '\n',
 'M', ':', '4', '/', '4', '\n',
 'L', ':', '1', '/', '8', '\n',
 'K', ':', 'C', '\n',
 'C', ' ', 'D', ' ', 'E', ' ', 'F', ' ', '|', ' ',
 'G', ' ', 'A', ' ', 'B', ' ', 'c', ' ', '|']
```

Newlines and whitespace are preserved as explicit tokens, allowing the model to learn structural boundaries such as headers, measures, and phrase breaks directly from data.

The following shows the distribution of file lengths in the dataset:

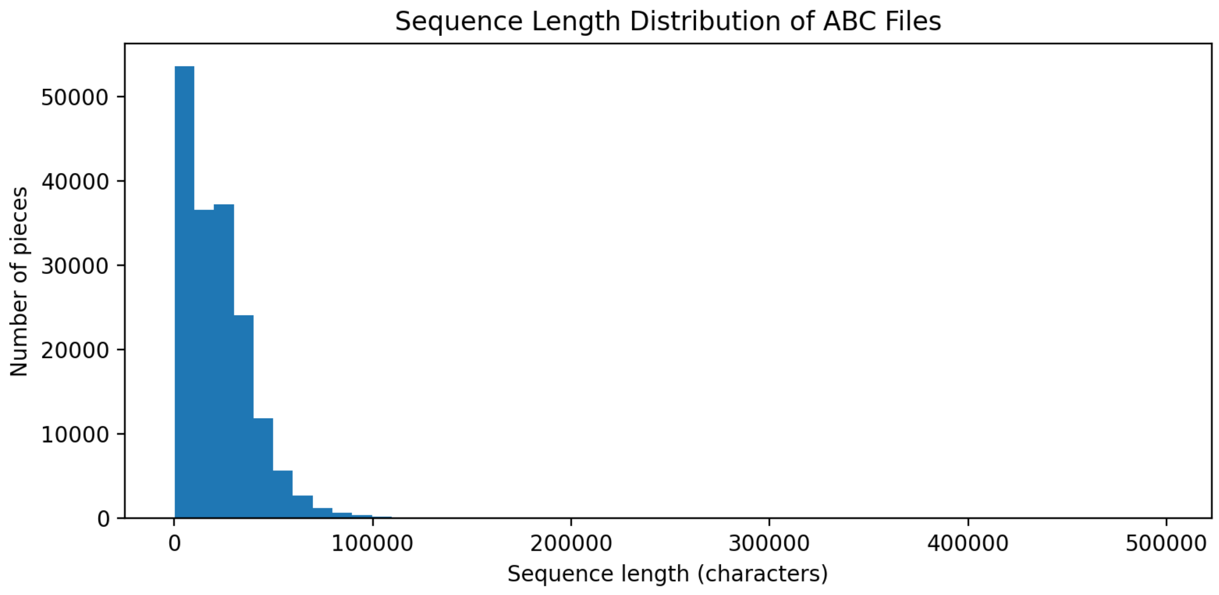

Figure 1: Sequence length distribution.

The majority of ABC files exceed the model's context window, indicating that training samples typically capture only local musical structure rather than complete pieces. This partially explains the model's difficulty in learning long-range musical form such as A–B–A structure.

Dataset statistics. After preprocessing, the corpus contains over one billion characters. We split the data into training, validation, and test sets using a 98% / 1% / 1% ratio. There are over 100M tokens in training set, ensuring that scaling experiments are not data-limited.

Methods

Model architectures. We train a family of decoder-only Transformer models with increasing parameter counts. Model sizes range from approximately 1M parameters to 59M parameters, achieved by varying the number of layers, attention heads, and hidden dimensions while keeping the tokenization scheme fixed.

Table 1: Transformer model configurations.

Model	Layers	Heads	Hidden Dim	Parameters
Tiny	2	2	128	0.41M
Small	4	4	256	3.17M
Medium	6	6	384	10.66M
Large	8	8	512	25.23M
XL	12	10	640	59.06M

Context Window Length I train all models using a fixed-length context window, where each training sample consists of a contiguous subsequence of characters from an ABC file. Due to the length distribution of the dataset, most musical pieces substantially exceed the model's context window.

This design choice reflects a practical trade-off between computational feasibility and representational capacity. While larger context windows could, in principle, capture longer-range musical structure, they would significantly increase memory usage and training cost.

As a result, the model primarily learns local musical dependencies, such as rhythmic consistency, short melodic fragments, and syntactic validity of ABC notation. This limitation directly explains the model's difficulty in generating long-range musical form, such as A–B–A structure, observed in later experiments.

Attention Mechanism All Transformer models use standard causal self-attention, identical to the mechanism employed in decoder-only language models such as GPT. Each token attends to all previous tokens within the context window, ensuring autoregressive generation.

I intentionally avoid introducing music-specific inductive biases, such as bar-aware attention or hierarchical structure, in order to isolate the effect of model scale. This design choice allows us to attribute observed improvements in performance primarily to increased model capacity rather than architectural specialization.

While more sophisticated attention mechanisms may improve long-range musical structure, they would confound the interpretation of scaling behavior, which is the primary focus of this study.

Training setup. All Transformer models are trained using the same training configuration: AdamW optimizer, cosine learning rate schedule with warmup, fixed batch size measured in tokens, and identical training data. Each model is trained for exactly one epoch for the scaling comparison, following established scaling law methodology.

Sequence Length Training is performed using fixed-length character sequences sampled from the ABC corpus via a sliding window. Each sequence represents a contiguous segment of a musical piece rather than an entire composition.

This design choice reflects both computational constraints and properties of the data distribution. Most ABC files substantially exceed feasible context lengths, and increasing sequence length would lead to quadratic growth in attention cost without guaranteed gains in long-range musical modeling.

Empirically, this setup encourages the model to learn local musical regularities, such as rhythmic consistency and short melodic fragments, while avoiding overfitting to absolute positional patterns. The absence of explicit long-range structure in generated samples is therefore a natural consequence of both the training objective and the sequence length choice.

Regularization To promote training stability and fair comparison across model sizes, I apply a minimal and consistent regularization strategy. All models use dropout with probability 0.1 in attention and feedforward layers, along with AdamW optimization and weight decay of 0.1.

I deliberately avoid aggressive regularization techniques such as label smoothing or data augmentation. Given the large scale of the training corpus, the dominant risk is underfitting rather than overfitting, particularly for larger models.

This restrained regularization setup ensures that observed performance differences primarily reflect model capacity and architectural scaling rather than regularization artifacts.

Results

Scaling results. Validation loss consistently decreases with increasing model size. When plotted on a log-log scale against parameter count, the results closely follow a power-law relationship of the form

$$L(N) = a \cdot N^{-\alpha} + c, \alpha = 0.6299$$

where N denotes the number of parameters.

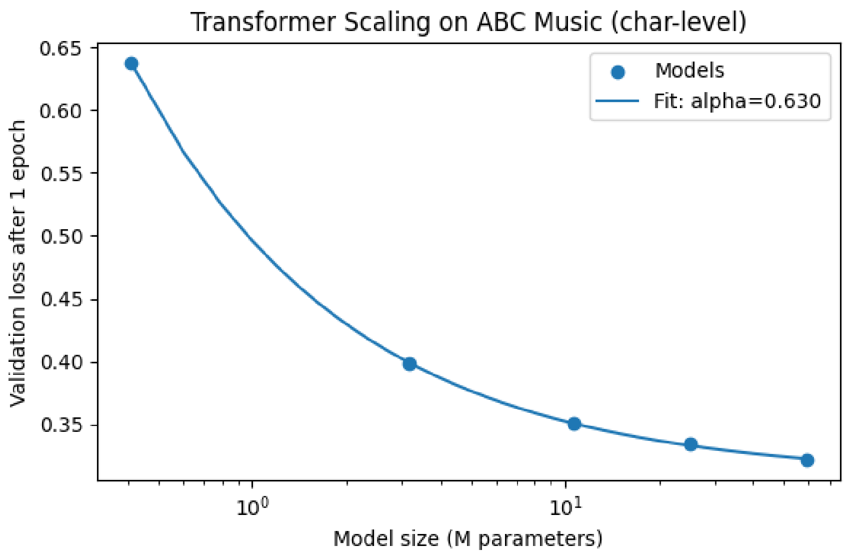

Figure 2: Transformer scaling curve with power-law fit.

The fitted exponent α indicates diminishing but consistent returns as model size increases.

These findings partially align with known scaling laws for language models. Similar to the trends reported by Kaplan et al. and the compute-optimal analysis of Chinchilla, larger models learn the underlying data distribution more efficiently. However, unlike natural language, the music domain exhibits weaker long-range dependencies and lower semantic density per token.

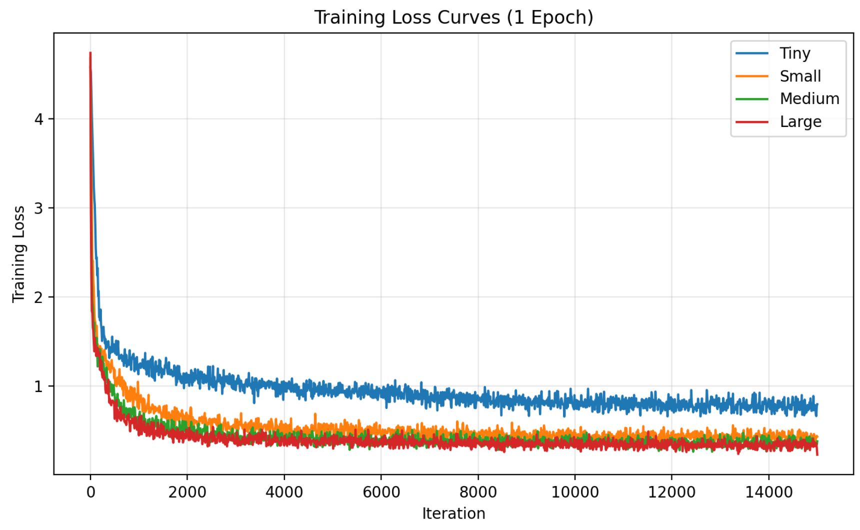

Figure 3: Training curves.

Figure 3 shows the training loss curves for models of increasing scale. As model size increases, training loss decreases more rapidly during early optimization and converges to a lower final value. This indicates that larger models are more effective at fitting the underlying distribution of ABC music tokens.

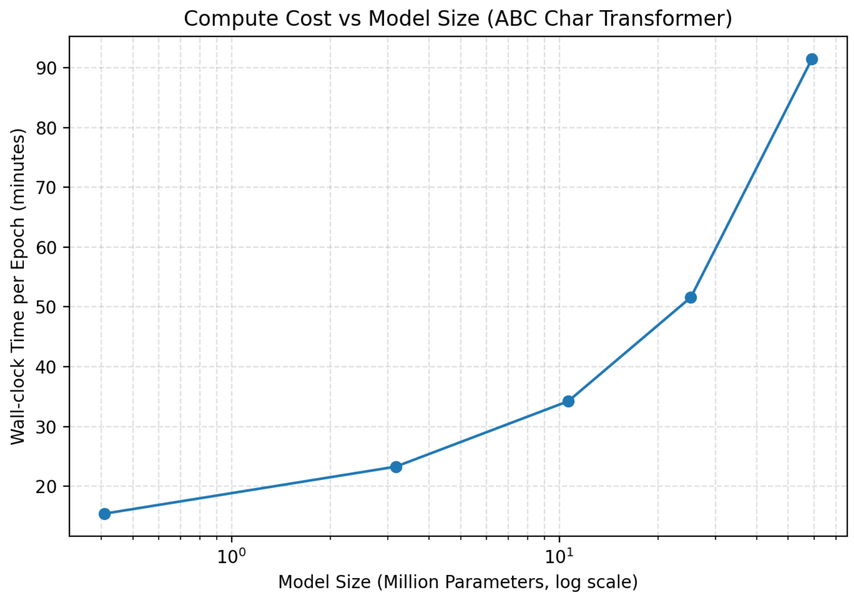

Figure 4: Compute Cost.

Figure 4 illustrates the wall-clock training time as a function of model size. Training time increases superlinearly with the number of parameters, reflecting the quadratic cost of self-attention and increased memory traffic during forward and backward passes.

Despite the higher computational cost, larger models achieve lower loss in fewer training steps, indicating improved sample efficiency. This trade-off suggests that, within our computational budget, allocating resources to moderately larger models yields better performance than training smaller models for extended durations.

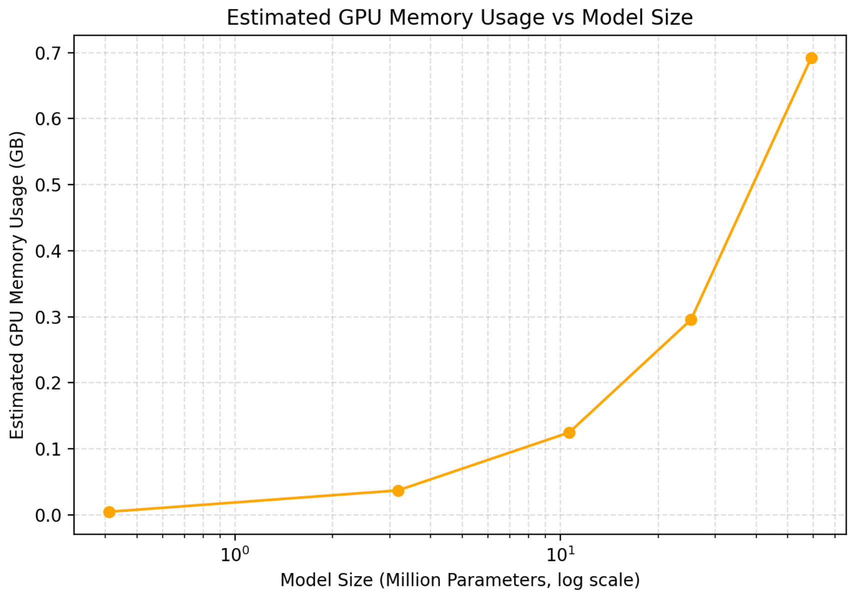

Figure 5: Gpu memory estimate.

Figure 5 reports peak GPU memory usage for different model configurations. Memory consumption scales approximately linearly with model size, driven primarily by parameter storage, activation tensors, and optimizer states.

Compute-Optimal Training My experiments suggest that compute-optimal training in the music domain differs in important ways from standard language modeling. While larger Transformer models consistently improve validation loss and sample quality, the marginal benefit of continued training rapidly diminishes once the dataset has been sufficiently covered.

Empirically, we observe that increasing model capacity yields greater gains than extending training duration. Validation loss reaches very small number within a epoch, after which further optimization leads to overfitting or stagnation. This behavior indicates that the effective entropy of character-level ABC data is relatively low compared to natural language.

RNN models. To compare architectures, I train a family of LSTM-based language models with parameter counts matched as closely as possible to the Transformer models. All RNNs use the same character-level tokenization and are trained on the same data for one epoch.

Scaling behavior. While RNN validation loss also decreases with increasing model size, the rate of improvement is substantially slower than that of Transformers. The RNN scaling curve exhibits a flatter slope, indicating poorer scaling efficiency.

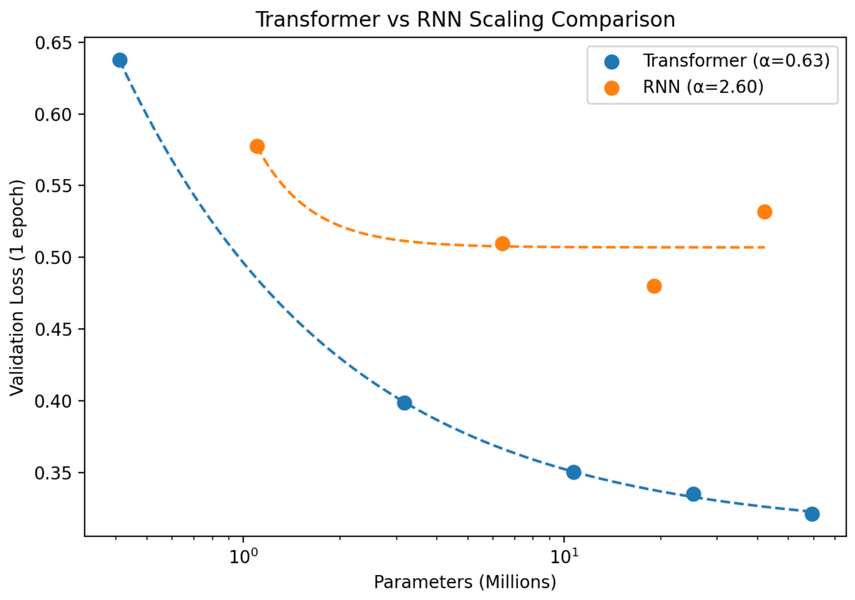

Figure 6: Comparison of Transformer and RNN scaling behavior.

Comparison. Transformers outperform RNNs at comparable parameter counts, both in final validation loss and in training efficiency. This gap widens as model size increases, suggesting that self-attention mechanisms are significantly better suited for capturing long-range musical dependencies than recurrent architectures.

Best Model Training. I selected the largest feasible Transformer model supported by our computational budget, consisting of approximately 59.06M parameters. The model follows a standard decoder-only autoregressive architecture trained on character-level ABC notation

The model was trained on a large-scale ABC corpus converted from the Lakh MIDI Dataset. Training proceeded for as many tokens as permitted within the project timeline.

After training, I decided to use the weights generated from the third epoch as the best model weights. After testing, I found that:

$$\text{Val Loss} = 0.375162 \quad | \quad \text{Val PPL} = 1.455$$
$$\text{Test Loss} = 0.381023 \quad | \quad \text{Test PPL} = 1.464$$

Sample Generation. In total, we generated: 10 conditional samples and 10 unconditional samples.

Table 2: Conditional samples.

Metric	Result
Total samples	10
Syntactically valid (parseable ABC)	10(100%)
Successfully converted to MIDI	10(100%)
Nontrivial music (contains pitched notes)	10(100%)

Table 3: Unconditional samples.

Metric	Result
Total samples	10
Syntactically valid (parseable ABC)	10(100%)
Successfully converted to MIDI	10(100%)
Nontrivial music (contains pitched notes)	9(90%)

All samples were converted to MIDI using Music21.

Please see here: Mid out

The model demonstrates strong mastery of ABC syntax and produces MIDI-playable outputs with high reliability, especially under conditional prompting.

Musical coherence

Conditional samples generally exhibit:Stable rhythmic structure,consistent meter,plausible note durations and bar alignment.Meaningful interaction between melodic and percussion voices.Maintains a coherent low-register bass pattern across multiple bars, with percussion voices reinforcing meter consistency.

Unconditional samples, by contrast:Frequently overuse rests and exhibit long spans of silence

This behavior reflects the model's uncertainty when no structural context is provided.

Tonality, harmony, and rhythm

The model reliably respects declared key signatures, maintains internal rhythmic consistency and produces harmonically plausible chord clusters.

However, I also observe that frequent key resets in multi-voice outputs, weak long-range harmonic plannin and local coherence without global direction

The model learn

The model clearly learned,ABC syntax rules,meter-consistent rhythmic continuation,local melodic fragments,and style continuation under conditional prompts...

Despite attempts to induce A–B–A structure through prompt engineering, generated samples consistently failed to exhibit true melodic recall.

Discussion

Key Insights from Scaling Experiments

My scaling experiments reveal several consistent trends across both Transformer and RNN architectures. First, increasing model size leads to systematic reductions in training and validation loss, confirming that larger models are better able to capture the statistical structure of symbolic music sequences. However, the magnitude of improvement diminishes as model size grows, indicating diminishing returns beyond a certain scale.

Second, Transformers consistently outperform RNN-based models at comparable parameter counts. This advantage becomes more pronounced at larger scales, suggesting that attention-based architectures are more effective at modeling long-range dependencies in music sequences.

Finally, computational cost grows rapidly with model size, particularly for Transformers due to the quadratic complexity of self-attention. These observations highlight a fundamental trade-off between model scale, performance, and computational feasibility.

Interpretation in the Context of Music Modeling

Music modeling presents unique challenges compared to natural language modeling. Symbolic music contains strong local structure (e.g., rhythm, meter, bar alignment) but also exhibits long-range dependencies such as thematic repetition and harmonic progression. Our results suggest that Transformers are well-suited for capturing local and medium-range musical coherence, especially when conditioned on a musical prefix.

However, even large Transformer models struggle with high-level musical form, such as long-range melodic recall or A–B–A structure. This limitation reflects both architectural constraints and properties of the training data. The ABC representations derived from MIDI files rarely encode explicit section-level structure, making it difficult for models to learn long-term musical planning purely from token-level supervision.

These findings align with prior work in music generation, which suggests that hierarchical or explicitly structured representations are often necessary to model global musical form effectively.

Design Decisions and Their Impact Several design choices significantly influenced model performance. Character-level tokenization was chosen for its simplicity and robustness to syntax variation in ABC notation. This approach enabled high syntactic validity and reliable MIDI conversion but limited the model's ability to reason about higher-level musical units such as notes, measures, or phrases.

The decision to prioritize moderate context window lengths balanced computational efficiency and musical coherence. While longer context windows may improve long-range dependency modeling, memory constraints and diminishing empirical gains motivated a more conservative choice.

Conditional generation proved to be substantially more effective than unconditional sampling. Providing structured prompts anchored the model in a valid musical context, reducing syntactic errors and improving musical coherence. This highlights the importance of conditioning and context in autoregressive music generation.

Limitations and Future Work This study has several limitations. First, training was constrained by computational resources, limiting both model scale and total training tokens. Larger models trained for longer durations may yield stronger results. Second, the dataset lacks explicit annotations for musical form, which likely contributed to the model's inability to learn long-range structure.

Future work could explore hierarchical architectures, such as combining Transformers with latent structure models or multi-scale representations. Alternative tokenization schemes, including note-level or event-based representations, may also improve musical abstraction. Additionally, incorporating structural annotations or reinforcement-based objectives could encourage models to generate music with clearer global organization.

Conclusion

In this project, we investigated scaling laws for neural sequence models in the symbolic music domain using ABC notation. Through systematic experiments across model sizes and architectures, we demonstrated that larger models consistently achieve lower loss and improved sample quality, with Transformers outperforming RNNs at comparable scales.

Our results show that scaling benefits in music modeling resemble those observed in natural language processing, but with important domain-specific differences. While larger models improve local musical coherence and syntactic validity, they do not automatically acquire high-level musical form. This highlights the limits of purely autoregressive training on unstructured symbolic data.

Overall, this study deepened our understanding of compute–performance trade-offs in music modeling and underscored the importance of representation and architectural design when applying scaling principles beyond text. These insights provide a foundation for future work on more structured and musically informed generative models.

Appendices

Generated ABC Music Sample

Listing 1: 1

```
X: 1
T: from /root/autodl-tmp/data/midi_raw/lmd_full/4/44978f394ffed2f23247187e41c72f13.mid
M: 4/4
L: 1/8
Q:1/4=140
K:C % 0 sharps
V:1
%%MIDI program 0
[bf][bf]/2[bf]/2 [bf][bf]/2[bf]/2 [bf][^a^f] [bg][g-e-]| \
[ge]8| \
z8| \
ee ee ee dc|
B2 ee ee Bf-| \
f2 e2<e2 ec| \
ee ee e2 dc| \
BB BB BB AG|
AB cA2G3| \
ee ee ee dc| \
g2 ag2e3| \
ee ee ee dc|
B2 ee ee Bf-| \
f2 e2<e2 ec| \
ee ee e2 dc| \
BB BB BB AG|
AB cA2G3| \
bb bb aa ga-| \
ag ^fg bb ba-| \
a2 z2 c'c' c'a|
bb bb aa ga-| \
a2 z2 c'c' c'a| \
bb bb aa ga-| \
a2 z2 c'c' c'a|
bb bb aa ga-| \
a2 z2 c'c' c'a| \
bb bb aa ga| \
z8|
B2 ee ee Bf-| \
f2 e2<e2 ec| \
ee ee e2 dc| \
BB BB BB AG|
AB cA2G3| \
ee ee ee dc| \
g2 ag2e3| \
ee ee ee dc|
B2 ee ee Bf-| \
f2 e2<e2 ec| \
ee ee e2 dc| \
BB BB BB AG|
AB cA2G3| \
z8| \
z8| \
z8|
z8| \
z8| \
z8| \
z8|
z8| \
z8| \
z8| \
z8|
z8| \
z8| \
z8| \
z8|
z8| \
z6 ee| \
ee ee e2 dc| \
BB BB BB AG|
AB cA2G3| \
z6 ee| \
ee ee e2 dc| \
BB BB BB AG|
AB cA2G3| \
z8| \
ee ee ee dc| \
BB BB BB AG|
AB cA2G3| \
z8| \
ee ee ee dc| \
BB BB BB AG|
AB cA2G3| \
ee ee ee dc| \
g2 ag2e3| \
ee ee ee dc|
B2 ee ee Bf-| \
f2 e2<e2 ec| \
ee ee e2 dc| \
BB BB BB AG|
AB cA2G3| \
z8| \
z8| \
z8|
z8| \
z8| \
z8| \
z8|
z8| \
z8| \
z8| \
z6 ee|
ee ee e2 dc| \
BB BB BB AG| \
AB cA2G3|
V:2
%%MIDI program 0
z8| \
z8| \
z8| \
z8|
z8| \
z8| \
z8| \
z8|
z8| \
z8| \
z8| \
z8|
z8| \
z8| \
z8| \
z8|
z8| \
z6 ee| \
ee ee e2 dc| \
BB BB BB AG|
AB cA2G3| \
z8| \
ee ee ee dc| \
BB BB BB AG|
AB cA2G3| \
ee ee ee dc| \
g2 ag2e3| \
ee ee ee dc|
g2 ag2e3| \
ee ee ee dc| \
BB BB BB AG| \
AB cA2G3|
z8| \
z8| \
z8| \
z8|
z8| \
ee ee ee Bf-| \
f2 e2<e2 ec| \
ee ee ea2g|
z8| \
ee ee ee Bf-| \
f2 e2<e2 ec| \
ee ee ea2g|
z8| \
ee ee ee dc| \
g2 ag2e3| \
ee ee ee dc|
g2 ag2e
---------------
```

Listing 2: 2

```
X: 1
T: from /root/autodl-tmp/data/midi_raw/lmd_full/4/447291346e90312d01cf44a27de07ccf.mid
M: 4/4
L: 1/8
Q:1/4=160
K:Bb % 2 flats
```

```
V:1
z8| \
z8| \
z8| \
z8|
z6
%%MIDI program 1
%%MIDI program 1
B/2zB/2-| \
B3/2z/2 cz/2c3z/2f-| \
fc2-c/2-[dc]/2 z3B| \
A/2BB/2 z/2B/2z/2B2A/2 B/2zB/2-|
B2 z3/2B/2 z3c/2z/2| \
d/2d/2z/2d/2 z/2d/2z/2d2cz/2d/2z/2| \
e/2e/2z/2e/2 z/2fz/2 g3/2z/2 fz/2d/2-| \
dz6z/2d/2|
z/2d/2z/2d/2 z/2d/2z/2d2c>d[cB-]/2| \
B/2A/2z6z/2B/2| \
z/2B/2z/2B/2 z/2B/2z/2B3/2z/2A/2 Bc/2B/2-| \
B3/2z6B/2|
z/2B/2z/2B/2 z/2B/2z/2B3/2z/2A/2- [B-A]/2B/2c/2B/2-| \
B3/2z6B/2| \
z/2B/2z/2B/2 z/2B/2z/2B2d2c/2| \
B2 z4 z/2B/2z/2B/2|
z/2B/2z/2B2c/2 [BA-]/2A/2B/2z/2 c/2zd/2-| \
dz6z| \
z2 z/2B3/2 z/2c2dz/2| \
c4- c3/2F2-F/2-| \
F4- Fz/2C2-C/2-| \
C4- C3/2z2z/2| \
z6 z/2B,-[C-B,]/2|
CB,/2<A,/2 [B,A,-]A,/2B,/2<A,/2B,zA,3/2-| \
A,6 z2| \
D3/2z2C/2 z/2B,3/2 z/2zA,/2| \
B,A,/2F,4z/2 A,/2zE,/2|
F,>E, F,>E, F,2<A,2| \
Cz Cz C/2z3/2 B,/2Cz/2| \
C3/2C/2 C3z/2B,/2 z/2A,3/2-| \
A,z4z/2E/2 zz|
F/2z/2G/2z/2 A/2B3/2 z/2B3/2 Bz| \
D/2z3/2 C/2z/2B,3/2z/2[B,A,]/2A,/2 F,3/2z/2| \
z8| \
z8|
z8| \
z8| \
z8| \
z8|
z8| \
z8| \
z8| \
z8|
z8| \
z8| \
z8| \
z8|
z8| \
z8| \
z8| \
z6 A2|
z/2Bz/2 B2 c3z/2z/2| \
c3/2z/2 cB/2>A/2 B/2z/2A/2-[AF-]/2 F2-| \
F4 E/2z3z/2| \
z3/2D/2<E/2E/2z F/2z/2G/2A/2 z/2B3/2-|
B/2z/2B3/2z/2B zc2c/2-[cB]/2| \
A/2<B/2[AE]/2z/2 F2- F/2z/2A/2F2-F/2-| \
F2- F/2zA2-A/2 EF/2z/2| \
G6- [A-G]/2A3/2-|
A/2z/2B z/2B2z/2B zd/2z/2| \
z/2c/2z/2BA/2z/2[FC]3z3/2| \
z/2E3/2 zD3/2C/2B,3/2z/2A,-| \
A,/2z/2G,6-G,|
z6 B2| \
z/2Bz/2 B3/2z/2 c3-c/2z/2| \
c3/2z/2 cB/2>A/2 B/2z/2A/2-[AF-]/2 F2-| \
F3/2z/2 E/2z4z3/2|
zD/2E/2 zE/2z/2 F/2G/2A/2z/2 B2-| \

---------------
```

Listing 3: 3

```
X: 1
T: from /root/autodl-tmp/data/midi_raw/lmd_full/4/448a20cb249e1e2036559cddf2ae358c.mid
M: 4/4
L: 1/8
Q:1/4=120
K:G % 1 sharps
V:1
K:C % 0 sharps
% Last note suggests minor mode tune
z8| \
%%MIDI program 35
E,,2- E,,/2z/2E,,2-E,,/2z/2 E,,3/2z/2| \
 (3E,,4E,,4E,,4| \
C,,2- C,,/2z/2C,, G,,,3/2z/2 G,,,D,,|
C,,2- C,,/2z/2C,,2<G,,,2G,,,| \
 (3A,,,4A,,,4A,,,4| \
D,,2- D,,/2z/2D,, D,,3/2z/2 G,,,3/2z/2| \
E,,2- E,,/2z/2E,,2-E,,/2z/2 D,,^C,,|
C,,2- C,,/2z/2C,,2<D,,2D,,| \
G,,,3z G,,,2- G,,,/2z/2D,,| \
G,,3G,, D,,3/2z/2 G,,3/2z/2| \
E,,2- E,,/2z/2E,, B,,,2 A,,,3/2z/2|
E,,2- E,,/2z/2E,, B,,,2 E,,D,,| \
-[D,,G,,,-]G,,,/2^A,,,B,,,3/2 D,,3z| \
D,,2- D,,/2z/2D,,2<D,,2D,,| \
G,,,3z G,,,2 zD,,|
G,,3D,, G,,2 D,,C,,| \
B,,,3^F,, B,,2 B,,,2| \
E,,,2 zB,,, E,,2 E,,,2| \
A,,,2- A,,,/2z/2A,,, E,,3A,,,|
D,,3A,,, F,,2 E,,3/2z/2| \
D,,2- D,,/2z/2A,, A,,2 D,,2| \
G,,2- G,,/2z/2G,, G,,2 G,,,2| \
C,,2- [D,,-C,,]/2D,,3/2 E,,2 C,,2|
G,,3
V:2
%%MIDI channel 10
%Percussion
z8| \
z8| \
z8| \
z8|
z8| \
z8| \
z8| \
z8|
z8| \
z8| \
z2 D,,z2D,,/2z/2 D,,/2z3/2| \
z2 D,,/2z2z/2D,,/2z/2 D,,/2z3/2|
z2 D,,/2z2z/2D,,/2z/2 D,,/2z3/2| \
z2 D,,/2z2z/2D,,/2z/2 D,,/2D,,/2z| \
z4 zD,,/2z/2 D,,/2z3/2| \
z2 D,,/2z2z/2D,,/2z/2 D,,/2z3/2|
z2 D,,/2z2z/2D,,/2z/2 D,,/2z3/2| \
z2 D,,/2z2z/2D,,/2z/2 D,,/2z3/2| \
z2 D,,z2D,,/2z/2 D,,/2z3/2| \
z2 D,,/2z2z/2D,,/2z/2 D,,/2z3/2|
z2 D,,/2z2z/2D,,/2z/2 D,,/2z3/2| \
z2 D,,/2z2z/2D,,/2z/2 D,,/2D,,/2z| \
z4 zD,,/2z/2 D,,/2z3/2| \
z2 D,,/2z2z/2D,,/2z/2 D,,/2D,,/2z|
z4 zD,,/2z/2 D,,/2z3/2| \
z2 D,,z2D,,/2z/2 D,,/2z3/2| \
z2 D,,/2z/2D,,/2D,,/2 z2 D,,2-| \
D,,2 D,,2 z2 D,,D,,|
z4 zD,,/2z/2 D,,/2z/2D,,/2z/2| \
z2 D,,z3 D,,D,,/2D,,/2| \
z/2D,,/2z D,,/2zD,,/2 z2 D,,2-| \
D,,2 D,,2 z2 D,,D,,/2D,,/2|
z/2 (3D,,D,,D,,D,,/2z4z| \
D,,z D,,z3 D,,D,,/2D,,/2|
---------------
```

Listing 4: 4

```
X: 1
T: from /root/autodl-tmp/data/midi_raw/lmd_full/4/44970c09b327eb73ac0d7f2077df7b79.mid
M: 4/4
L: 1/8
Q:1/4=120
K:Eb % 3 flats
V:1
%***Missing time signature meta command in MIDI file
```

```
% Last note suggests minor mode tune
%%MIDI program 22
z8| \
z8| \
z8| \
z8|
z8| \
z8| \
z8| \
z8|
 (3BcB (3ABc B/2c/2z/2A/2 z2| \
c/2zB/2 zA/2B/2 z/2c/2z2e/2z/2| \
B/2c/2z/2B/2 zA/2B/2 z/2c/2z2e/2z/2| \
B/2c/2z/2B/2 z2 c/2z3/2 g3/2z/2|
f3/2zc/2z/2e3/2z/2f3/2z| \
z/2f/2z/2f/2 z/2f/2z/2e/2 z/2f/2z/2g3/2z| \
B/2c/2B/2c/2 z/2G/2z/2B3/2z/2c/2 z/2e3/2| \
z/2f3/2 zf/2z/2 f3/2zc/2z/2e/2-|
ez/2f3/2z f/2z/2f/2z/2 f/2z/2f/2z/2| \
e/2z/2f/2z/2 g3/2zg/2z/2g/2 z/2a3/2| \
z/2a3/2 z2 [BA]/2B/2B/2c/2 z/2B/2z/2A/2-| \
Az/2B/2 z/2B/2z/2B/2 z/2B/2z/2B/2 z/2[BB]/2z/2[cB]/2|
z/2B/2z/2B/2 z2 [BA]/2B/2B/2c/2 z/2B/2z/2A/2-| \
Az/2B/2 z/2B/2z/2B/2 z/2B/2z/2B/2 z/2[BB]/2z/2[cB]/2| \
z/2B/2z/2B/2 z2 [BA]/2B/2B/2c/2 z/2B/2z/2A/2-| \
Az/2B/2 z/2B/2z/2B/2 z/2B/2z/2B/2 z/2[BB]/2z/2[cB]/2|
z/2B/2z/2B/2 z2 [BA]/2B/2B/2c/2 z/2B/2z/2A/2-| \
Az/2B/2 z/2B/2z/2B/2 z/2B/2z/2B/2 z/2[BB]/2z/2[cB]/2| \
z/2B/2z/2B/2 z2 [BA]/2B/2B/2c/2 z/2B/2z/2A/2-| \
Az/2B/2 z/2B/2z/2B/2 z/2B/2z/2B/2 z/2[BB]/2z/2[cB]/2|
z/2B/2z/2B/2 z2 [BA]/2B/2B/2c/2 z/2B/2z/2A/2-| \
Az/2B/2 z/2B/2z/2B/2 z/2B/2z/2B/2 z/2[BB]/2z/2[cB]/2| \
z/2B/2z/2B/2 z2 [BA]/2B/2B/2c/2 z/2B/2z/2A/2-| \
Az/2B/2 z/2B/2z/2B/2 z/2B/2z/2B/2 z/2[BB]/2z/2[cB]/2|
z/2B/2z/2B/2 z2 [BA]/2B/2B/2c/2 z/2B/2z/2A/2-| \
Az/2B/2 z/2B/2z/2B/2 z/2B/2z/2B/2 z/2[BB]/2z/2[cB]/2| \
z/2B/2z/2B/2 z2 [BA]/2B/2B/2c/2 z/2B/2z/2A/2-| \
Az/2B/2 z/2B/2z/2B/2 z/2B/2z/2B/2 z/2[BB]/2z/2[cB]/2|
z/2B/2z/2B/2 z2 [BA]/2B/2B/2c/2 z/2B/2z/2A/2-| \
Az/2B/2 z/2B/2z/2B/2 z/2B/2z/2B/2 z/2[BB]/2z/2[cB]/2| \
z/2B/2z/2B/2 z2 [cA]/2[cB]/2[cB]/2[cc]/2 z/2B/2z/2A/2-| \
Az/2B/2 z
---------------
```

Listing 5: 5

```
X: 1
T: from /root/autodl-tmp/data/midi_raw/lmd_full/4/44dc10ac8547537616e7c1659e38321d.mid
M: 1/4
L: 1/16
Q:1/4=176
K:C % 0 sharps
V:1
%%MIDI channel 10
z4| \
z4| \
z4| \
z4|
z4| \
z4| \
z4| \
z4|
z4| \
[^F,,B,,,]/2zF,,/2 [F,,^C,,]/2z^A,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2z^A,,/2|
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 ^A,,/2z3/2|
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2z^A,,/2|
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 ^A,,/2z3/2|
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 ^A,,/2z3/2|
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 ^A,,/2z3/2|
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 ^A,,/2z3/2|
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 ^A,,/2z3/2|
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 ^A,,/2z3/2|
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 ^A,,/2z3/2|
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 F,,/2zF,,/2| \
[^F,,B,,,]/2z3/2 ^A,,/2z3/2| \
[^F,,B,,,]/2z3/2 ^A,
---------------
```