Scaling Law in MIDI-Based Symbolic Music Sequences

Shashwat Saket
shashwat.saket@nyu.edu

Instructor: Prof. Pavel Izmailov
Github Code: https://github.com/shashwatsaket46/Scaling-Law-Musical-MIDI-Data–ML-Project

CS-GY 6923: Optional Project.

1 Introduction

As the world is moving towards the multimodal data processing, music plays an important role in sequence modeling. Listening music was always passionate about. And as symbolic music have a very distinct domain which fosters the scaling behavior of models. Particularly speaking, the authors in this paper worked on Neural scaling laws which delineates the performance boost and changes when model size, computation and dataset size is increased. Previous works displays a relationship between LLM models and natural language which henceforth procures a deterministic power law functions between scale of the model and its validation loss inversely [1]. This enlightens the architectural developments of modern gigantic models. There are many empirical studies which have focused on text based data and have obtained amazing results. However, it leaves an open ended opportunities on nontextual data, and musical data having octaves and symphonies are prima facie for this paper.

Dissimilar to natural language, in case of symbolic music representation of data requires a hierarchical or temporal structure which can be denoted using encoded structure having notations like ABC. We need to focus on modelling the long-term dependencies and consistent structures of musical data around the temporal features. Because of this, the symbolic music is a challenging domain to model the representative architectural evaluation as several factors such as inductive biases are needed to be mitigated.

Historically, the long short term memories were the backbone of recurrent neural networks and have provided the significant architecture in training symbolic music data and the generation of music using them. It has been observed that for short term dependencies, the LSTM based models were procuring good results in generating the music, but for long term dependencies, as the authors in this papers have used, it does provide a very coherent musical outputs, as scaling disasters such as vanishing gradients and limitations such as context propagation were the key challenges in this domain. Opposite to this, the transformers based models such as GPT equipped with self attention have potential to interact directly in a

sequence tokens which amplifies the generative capacities in long range modeling domains, an essential component in researching the symbolic music representations and its scaling behavior [2].

In this paper, the authors have examined the scaling laws associated with the training of data consisting of symbolic music using neural sequences. The transformers and LSTM based gigantic models were trained using several model hyperparameters tuning focused on optimization of loss and validation loss. Further, the authors in this paper have developed a demonstration for autoregressive music using the best-trained transformer model deduced in this paper and then changed the ABC format to MIDI format music. The results lucidly points that the transformers performed better than LSTM when comparing using the scaling law metric, enlightening the significance of attention models in the field of structured sequences.

2 Data Collection and Processing

2.1 Dataset Description

We are using the MIDI-Caps dataset from Huggingface, which is a large-scale collection of data consisting of symbolic music in the MIDI files format [3] amalgamated with the text-based description, which was initially introduced to study the multimodal behavior of the data having music and text both. This experiment focusses specifically on the symbolic music and henceforth would be analyzing the modal behavior of music.

There are several factors which resulted in selecting the MIDI-Caps dataset [3]. Primarily, it contains a very diverse music corpus and have varied style of music in MIDI files, providing an apt choice for our large scale representations using the transformers. Other than that, it holds the details of pitch, duration and timing, and hence foster us to contrast the LSTM based models with transformers on a metric such as neural scaling laws.

2.2 Preprocessing

The MIDI-Caps files after extracting gave me thousands of MIDI files, each files consisting some kind of musical data performed by some artists. The first step in preprocessing mandates to change the MIDI files into ABC style notations [5], which stores the significant details such as pitch, octaves, musical structures and pauses. Further the processing step would consists of following pipeline:

1. The raw midi files are taken and converted into a music symbols, particularly having the text based notations such as ABC.

2. All the musical signatures like the metadata having meter can be accessed whenever required.

3. Filtering out the MIDI files which had short music or data, and henceforth not capable of training has been avoided.

4. Resulting data has been partitioned into fixed sequences of token available for training.

Because of this conversion, the essence of music is retained from the raw original MIDI data while providing an approach for autoregressive language modeling.

2.3 Tokenization

In this section, the tokenization approaches used in this paper would be discussed. Tokenization is used to create a vocabulary for the training data. With this, a custom vocabulary based tokenization scheme has been created, as given in the ABC notation corpus. There are other options like character level tokenization, sentence level tokenization, etc. But we created a regex based tokenization having:

- Octave markers and the pitch symbols are preserved in token. (e.g., `Ĉ`)
- The duration modifiers such as `z`, `z2` are preserved.
- The fractional values such as the token of duration having integer values like 2, 2/3.
- Structural markers and repeated symbols like `|`, `|:`
- Headers having the encoded musical context such as `K:C`.

Using this strategy, the efficiency and the musical expressiveness can be retained. The sequence length can be minimized with this strategy than the character level tokenization, and musical complexity can be better be managed with our tokenization, making it flexible enough for the patterns of musical data.

2.4 Dataset Visualization and its statistics

- The size of the vocabulary is 8004 tokens.
- The training validation and test split is 98

We got the distribution of data having the having several musical symbols, and several rare tokens having less frequencies. This type of distribution plays a significant role in a symbolic musical dataset as it stores the regular musical octaves as well as the non common events. There could be several musical random noise which needs a more precise data cleaning.

2.5 Discussion on dataset

Our dataset consisting of MIDI data, which we then convert to ABC to have all the notations of musical sequences properly. Next we tokenized the data using our regex-based tokenizer, which fosters an efficient mode of the musical pattern representation. Although it tokenizes the key aspects and musical notations like octaves, the grammatical correctness is not being

explicitly maintained, and hence a post-processing must be performed on this data. The representation we have is prepared for understanding the scaling laws. In this part, we would be discussing the model architecture, the setup that we used in our training, evaluation metrics, the hyperparameters used, and the model size for evaluating the symbolic music.

3 Experimentation Model

3.1 Model Architectures

In this section, we will be discussing different variants of natural language sequence models: transformer-based architectures with several parameters and model sizes, long short-term memory-based models, and a small nanoGPT-based model in initial training [4]. Transformers are self-attention-based models and is based on decoder-only architecture, enabling it to handle autoregressive modeling of data, which follows some principles of nanoGPT. The LSTM models primarily have recurrent neural networks and are having constraints such as vanishing gradients.

Transformers have positional encoding, token encoding, multi-stacked self-attention, and a fully connected layer towards the vocabulary while training the data. It also have a multi headed causal self-attention, layer normalization, and an in-decoder-only setting.

LSTM models have several RNN units connected with each other finally going to a linear output. This helps them to represent data having sequences, but the vanishing and exploding gradient problem needs to be precisely handled here. We have also trained a model on nanoGPT which was a fast model for any medium sized data. This is easy to train and have the capability to adapt to data efficiently.

Model	Layers	Heads	Embedding Dim	Context Length
nanoGPT-Tiny	4	4	128	256
Transformer Tiny	4	4	128	256
Transformer Small	4	6	192	256
Transformer Medium	6	8	256	256
Transformer Large	8	8	384	256

Table 1: Transformer architecture configurations used in scaling experiments used in this paper.

Table 1 denotes the model architecture we used in the nanoGPT, Transformer along with its layer, heads, embedding dimensions, context length details. As the model size is increasing, the layers, heads, dimensions are increasing.

Table 2 denotes the number of parameters used for each model used in this paper. Starting from 29 million parameters as a tiny model, we have trained large model until 184 million parameters.

Model Scale	Parameters	Parameters (Millions)
NanoGPT Tiny	29,244,672	29.24M
Transformer Tiny	57,674,752	57.675M
Transformer Small	87,101,952	87.102M
Transformer Medium	118,501,888	118.502M
Transformer Large	184,840,704	184.841M

Table 2: Transformer-based model scales and number of parameters used.

Model	Embedding Dim	Hidden Size	Layers
LSTM-57M	256	1536	3
LSTM-87M	1536	1536	2
LSTM-118M	3584	256	2
LSTM-184M	3584	1536	3

Table 3: LSTM architectural configurations used in scaling experiments.

Table 3 denotes the LSTM models used in this paper and then embeddinng dimensions, hidden size, number of layers for model.

3.2 Hyperparameters and Training setup

Every model used in this paper is optimizing using Adam Optimizer and have a crossentropy loss for the next token prediction. We considered the stability across each model and generalized used of Adam optimizer on sequence modeling tasks. We implement a decaying learning rate and amplified scaling in our models. Batch size is dependent on the compute unit and constrained for achieving a stable training loss convergence.

3.3 Scaling law studies and its design

The focus of this paper is to study and analyze the scaling laws of neural sequences having musical data and for this we have followed a constant tokenization scheme and objective for the training. After this, the validation loss can be analyzed to understand the scaling law effects. Because of this, we have periodically stored the validation loss and henceforth used it as our primary metric. Also a very tiny nanoGPT based model has been trained in our experiment to understand the training behavior and also analyze minimum capacity required to represent the model complexity.

3.4 Evaluation metric

The primary evaluation metric here is the validation loss for each model using the cross-entropy loss function which provides the mean negative log-likelihood. The predictive uncertainty is one of the significant measures that we have used here, and for this the

validation loss was an apt choice. We have also generated the music samples using the autoregressive sampling and for this once the generated music is stored in ABC notations based format, it is converted to MIDI format.

4 Results

4.1 Training a small NanoGPT model

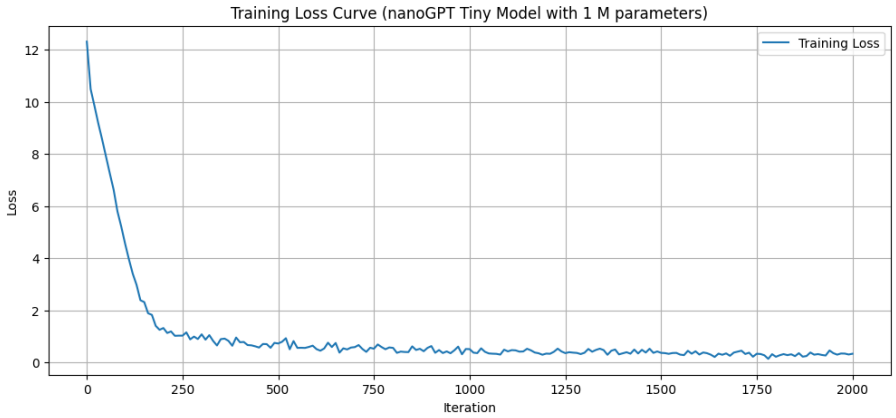


Figure 1: Figure represents the training loss over iteration for the nanoGPT based model used for understanding the pattern

Figure 1 displays the training loss vs iteration graph for nanoGPT based tiny model having 29 million parameters. This curve displays a rapid decrease in loss in early iterations and then the loss values are decreasing the a gradual manner. Even with less parameters and capacity, the model seems to train well on the given set of data. We also see some plateaus which refers to the fact that because of small model capacity, the representation is limited and gets exhausted.

4.2 Scaling behavior of Transformers

Figure 2 displays a validation loss for each given model architecture which has been trained on our data consisting of symbolic music. Our model parameters ranges from 57 million to 186 million parameters and a constant token having 100 million tokens are being used.

As a consequence, we can say that the bigger transformer model is able to represent the data better than the smaller models, therefore getting lower validation loss. The overall trend follows a monotonic decreasing behavior on validation loss. A short training horizon and smaller scales have the limitations on our training computational units.

This indicates that as we are increasing the model size, the performance is getting better, which follows the scaling law on neural sequences.

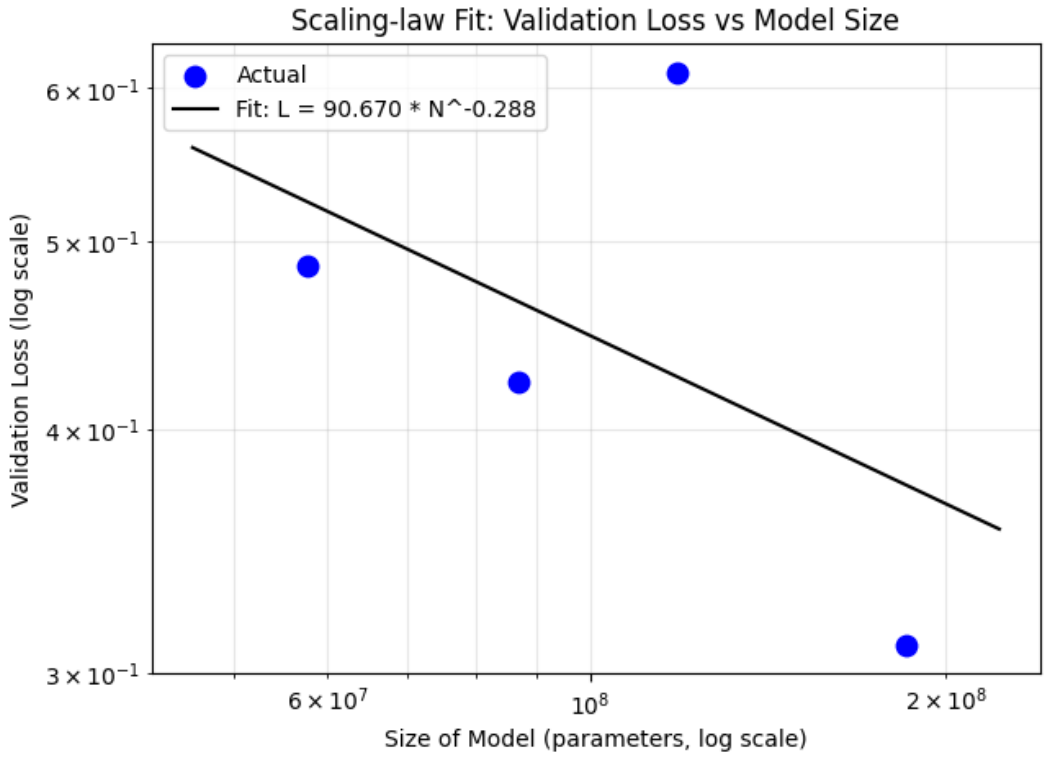


Figure 2: Representation of scaling for transformmers on several parameters over 100 M token

4.3 Scaling behavior on LSTM models

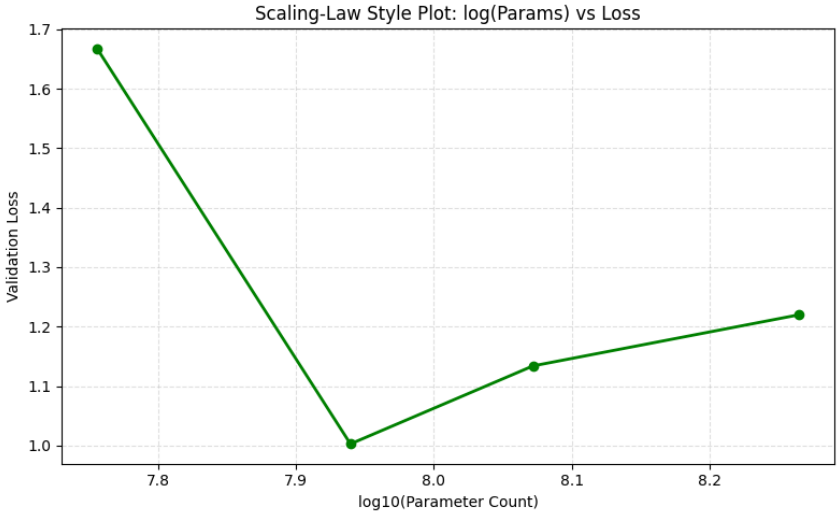


Figure 3: Representation of validation loss over the LSTM models with different parameters on 100 M tokens.

Figure 3 represents the validation loss on Long Short term Memory based models having four different model sizes. It is lucid that the model performance increases from 57M to 87M parameters model but for the bigger models, the performance is monotonically decreasinng.

This proves that the in contrast to transformers, the LSTM models does not shows a monotonic decrease in validation loss. As recurrent models suffers from vanishing and exploding gradient problem, thus we can say that after a certain model size, the LSTM based model exhausts its capacity to represent the data properly.

4.4 Comparison of scaling laws

Figure 4 directly compares the validation loss for our transformer based model and LSTM based model on a similar scale.

At all the time, it is observed that the transformer based models are achieving a lower validation loss than the LSTM based models. Also, it is lucid that transformers based model follows a proper monotonically decreasing validation loss as we increase the model parameter size which is not true for LSTM based models.

4.5 Computation Cost and dynamics of training

Our training curves displays a constant convergence on the transformer models. We also observed that the the higher dimension models having large number of parameters are converging slower than the smaller models but are converging to a lower validation loss. In contrast to transformers, the LSTM based models displays a faster initial convergence but creates a plateau for larger models.

We were given a fixed computation constraints on time and GPU constraints. The initial part for loading the data and performing initial tests on nanoGPT has been performed on L4

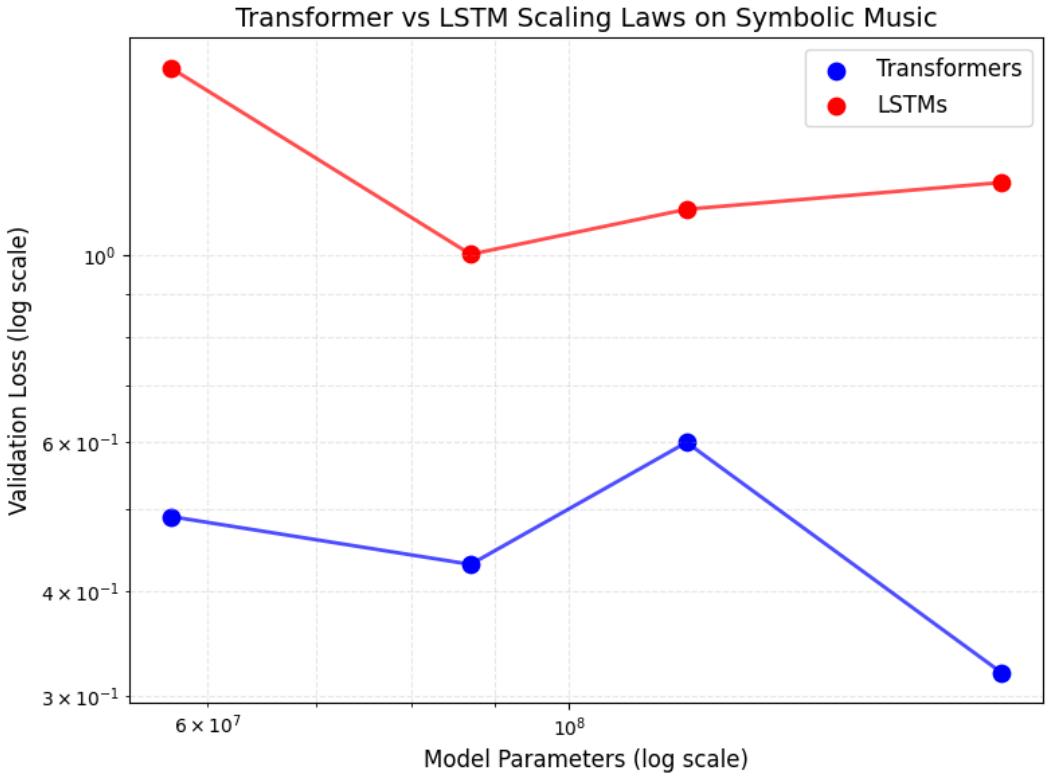


Figure 4: Comparing both transformer based models and LSTM based models on a similar plot having different parameter with validation loss on 100 M tokens

GPU, and the training for larger parameters based transformers and LSTM based models are performed on A100 GPU with 80GB RAM.

4.6 Generated Music evaluation on qualitative factors

We used the best transformer model, i.e. Large-Transformer for training the final data with 200 million tokens. Figure 4 displays the training loss convergence for our final model. Using this, we generated a sequence containing the symbolic music using the auto regressive sampling methodology. The sequences that are generated are processed to convert to ABC format and then to MIDI format consequently for analysis. The music contains rhythmic resonance and pitch properly, however as there is scope of improvement on grammatical correctness. Altogether, our model is able to understand the musical patterns and is able to generate notes, octaves and rhythms properly.

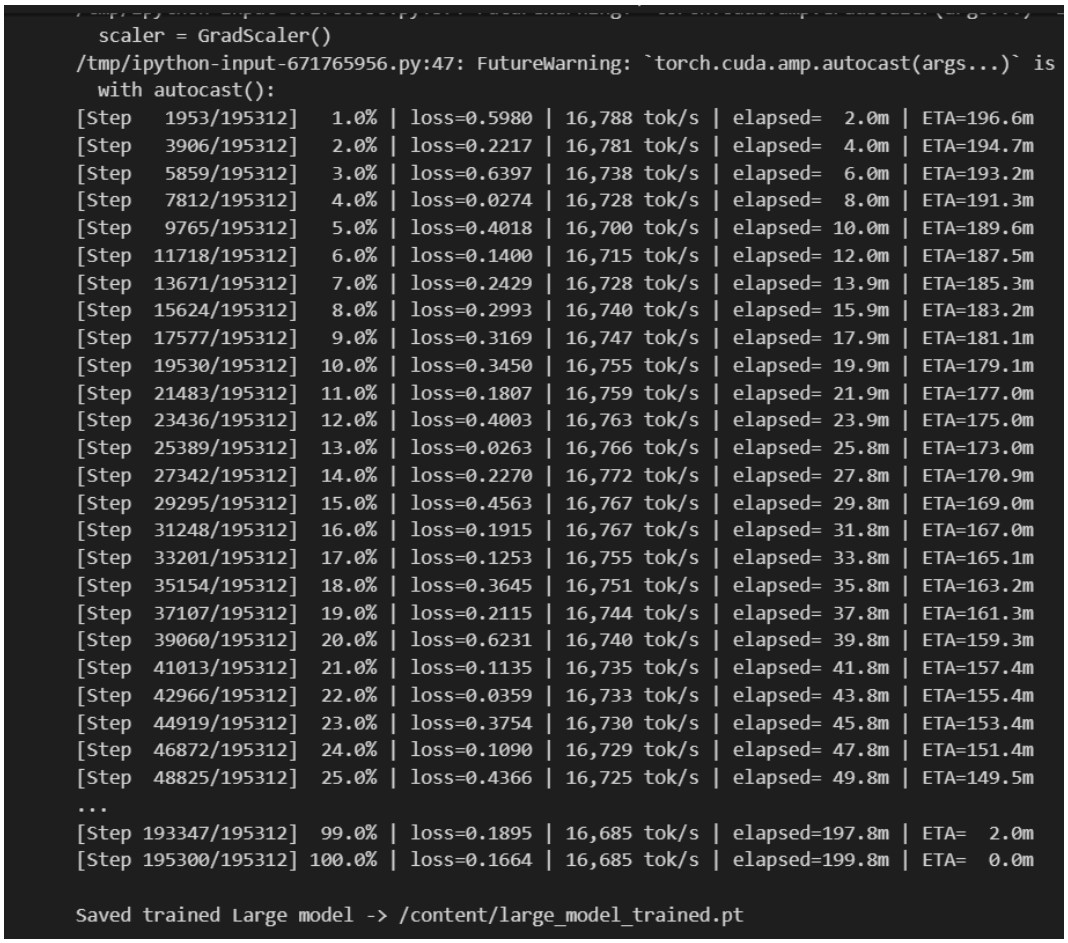


Figure 5: Training loss for our best model used for generating the music

5 Discussion

5.1 Scaling Behavior on different families of Models

The neural scaling tasks lucidly displays a contrast on attention based models using transformers and recurrent neural networks based models using LSTM when we model it on symbolic music data. We see a monotonic decrease in validation loss when we see the transformer based models as we increase the parameter size which maintains a power law consistency on musical data. This proves that it is possible to extend the neural scaling laws to non textual data.

Conversely, we see that LSTM based models displays a non monotonic behavior on validation loss when we increase the model size and hence it can be stated the model size increase cannot directly influence the performance of LSTM based models in a limited iterations.

5.2 Implications for Symbolic Music Generation

We observed a hierarchical structure on symbolic music which includes octaves, tunes, and rhythmic patterns of music having correct pitch which makes our generated music pleasant to hear. As the scaling laws proves, we observed that the transformer models are able represent the musical data properly because of their attention based training, which enhances the performance on the contextual window.

The sequence based models used in LSTM architecture does not properly fit the decreasing validation loss because of which it is clear that the sequence length has a specific constraints over representation of models. This becomes largely visible as we increaase the model size and henceforth we can say that it highly affect from the vanishing and exploding gradients on symbolic music.

5.3 Impact of Architectural and Training Choices

There are several decisions that have impacted the final music generation output. As we trained all the models keeping other factors such as data split, data preprocessing constant have resulted in a stable decrease in validation loss. Because of this, the increased model size primarily entails to the performance improvements. Our choice of regex based tokenization also have high impact on storing the meaningful context in music while we are able to manage the sequence lengths. This highly influences data preprocessing and henceforth result in a constant improvement in training qualities. Introduction of nanoGPT based model also represents the convergence, but also displays the constraints of representation on such a small model.

5.4 Limitations and Future Direction

There are some limitations in this experiments. Given more time, more efforts can be invested preprocessing of data, the regex pattern is able retrieve the musical notations, but more cleaning on pauses and grammatical correctness can improve the performance. Other than that, the compute unit limitation was a factor which restricted a use of further parameter improvement. Increasing the model size in transformer was getting towards a better output as 256 million parameter based model were training better for 100 million token based training. Single iteration training also was a factor which if increased have capacity to increase model perfomance. I would have also worked on selecting a more precise metric for musical data which handles the grammatical correctness as well.

Any future work towards this have a potential to optimize context windows, iterations on training models, preprocessing training data, and selection of mixed metrics focused on musical tunes. Human based reinforced metrics along with music aware evaluation can improve the harmonics of musical quality. The grammatical correctness improvement can highly improve the generation of resonating structures of music.

6 Conclusion

This work primarily focus on scaling laws when there is a non textual musical symphonies based sequence models and its contrast of training using transformer and LSTMs, and a tiny nanoGPT model as a initial model for understanding the pattern within a limited constraints on computational GPU. The results of this experiment proves the neural scaling laws property, hereby stating that predictability of performance increases as we increase the model size using larger parameters, decreasing the validation loss. Conversely, the other family of model based on recurrent neural networks based on LSTM does not scale properly while we increase the model size even with enhanced parameters, it does not monotonically decrease the validation loss. The generated music with different models also confirms our statement that increased model parameters provides a better music generation.

Using this, it can be stated that the attention based models provides a better modeling in a musical setting with long sequences and is able to capture the hierarchical notes given in a musical symphonies. Given our limited compute budget, Transformer based architectural models displayed amazing results on scaling law power fit patterns from neural modeling. This gives a good example on how the choice of models affect the training and representation of computationally scale the transfer in a musical data.

Code for this can be found at Github

References

[1] Kaplan, J., McCandlish, S., Henighan, T., et al. (2020). *Scaling Laws for Neural Language Models*. arXiv preprint arXiv:2001.08361.

[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). *Attention Is All You Need*. Advances in Neural Information Processing Systems (NeurIPS), 30.

[3] Melechovský, J., Roy, A., and Herremans, D. (2024). *MidiCaps: A Large-Scale MIDI Dataset with Text Captions*. arXiv preprint arXiv:2406.02255.

[4] Karpathy, A. (2023). *nanoGPT: A Minimal, Efficient Implementation of GPT*. GitHub repository.

[5] ABC Notation. *ABC Notation Home Page*. Available at: `https://abcnotation.com`.