Not Quite My Tempo: Symbolic Music Generation With Mamba Architecture

Jan Retkowski
TODO@gmail.com

Milosz Lopatto
TODO@gmail.com

Jakub Stepniak
TODO@gmail.com

Faculty of Electronics and Information Technology
Warsaw University of Technology
Warsaw, Poland

***Abstract*—Over the past few years, a lot of effort has been put into making Transformers more and more efficient. However, they still suffer from quadratic memory and inference time complexity. Recently they have been experiencing increasing competition from state space models (SSMs), that avoid some of transformer's drawbacks. In this paper we test one of state-of-the-art SSM models called Mamba in symbolic music generation. To achieve this we conducted several experiments on MAESTRO MIDI dataset. Our results show that while mamba can be used to generate novel musical scores, their quality leaves much to be desired.**

***Index Terms*—deep learning, transformer, music, symbolic music generation, neural network, generative, unsupervised learning, state space model, ssm, mamba**

I. Introduction

Symbolic music generation has been an area of active research over the past decades. In the beginning it was dominated by classical methods like Hidden Markov Models. During the advent of deep learning in the last decade, neural networks managed to become a go-to method for this task. During those years a lot of architectures, including Long Short-term Memory networks (LSTMs) [8], Variational Autoencoders (VAEs) [10] and Generative Adversarial Networks (GANs) [3] were used to successfully generate music in symbolic formats. However, after popularisation of Transformers [14] and Diffusion Models [7] in the field broad of deep learning, they seem to be gaining more and more dominance in the area of symbolic music generation. In the case of transformers, which are the go-to architecture in the area of natural language processing, a lot of effort has been put into small iterative improvements, to make them more efficient. However, they still suffer from quadratic GPU memory and inference time complexity. To mitigate those problems State Space Models (SSMs) [5] were developed. Recently, they have been gaining popularity and even beating transformers on some of their most signature tasks in the field of natural language processing, while being significantly smaller [4]. They seem to be especially suited for long time dependencies. Musical data, which often contains such long dependencies, seem to be suited for further testing the performance of SSMs. Therefore we decided to use current state-of-the-art SSM model called Mamba [4] to generate music in symbolic format.

II. Data

We decided to use MIDI [15] music representation. It is by far the most popular format in the field of symbolic music and allows us to freely use the largest symbolic music datasets available. Because of the model's size, a lot of data is required to reduce the chance of overfitting. The dataset we chose is MAESTRO (MIDI and Audio Edited for Synchronous TRacks and Organization) [6] [13]. It contains over 200 hours of virtuosic piano performances. The data comes from ten years of the International Piano-e-Competition. It is one of the most popular datasets, allowing us to easily compare our results to other models. The data was downloaded using Muspy [1] library, which allows for convenient handling of symbolic musical data. Tokenization was conducted using the miditok [2] library. It contains methods for tokenising MIDI data into most of the most popular formats. We decided to opt for REMI [9] format with learned Byte-Pair Encodings.

III. Architecture

Mamba utilises a novel approach called selective state space models (SSMs), which enhance traditional state space models by allowing parameters to be functions of the input. This design enables the model to selectively propagate or forget information along the sequence length, dependent on the current input token. Mamba integrates this selective SSM approach into a simplified end-to-end architecture, foregoing traditional components like attention or MLP blocks found in architectures like Transformers.

The main difference between Selective SSM and traditional SSM is the input-dependence, which mimics what the attention mechanism in Transformers does—essentially assessing whether a specific token is important or not. However, this feature sacrifices the ability of traditional SSMs to precompute all learnable matrices (Δ, A, B, C) and their configurations based on sequence length in a single inference pass. To address this, we introduce a mechanism of Parallel Associative Scan (similar to Prefix Sum) that requires storing precomputed calculations, leading to higher memory usage but still maintaining linear computation.

To enhance efficiency further, the authors proposed using a hardware-aware algorithm that leverages two main types of GPU memory: SRAM, which is fast but has limited capacity

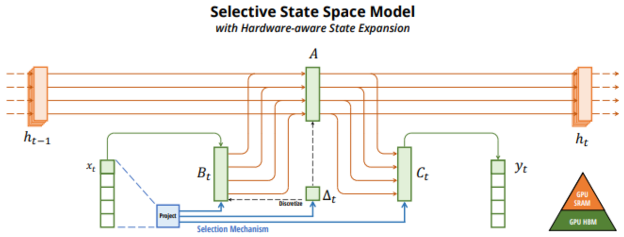

Fig. 1. Selective SSM [4]

TABLE I
MODEL COMPARISON [4]

Architecture	Complexity	Memory	Performance
Transformer	$O(N^2)$	$O(N^2)$	great
RNN	$O(N)$	$O(N)$	poor
SSM	$O(N)$	$O(N)$	poor
Selective SSM (Mamba)	$O(N)$	$O(N)$	great.

(ideal for matrix calculations), and HBM, which is slower but has a larger capacity. The main bottleneck of this approach is managing the data transfer between these memory types.

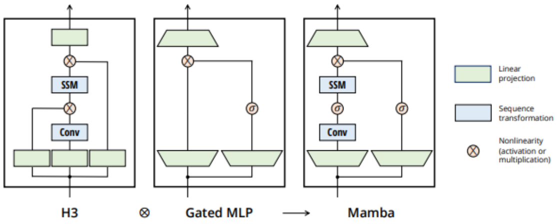

Fig. 2. Mamba block [4]

Selective SSM is a crucial component of the Mamba block, but the system also includes linear layers that increase dimensionality, nonlinear layers, and gating connections. The whole architecture is built from many Mamba blocks, which are computed layer by layer.

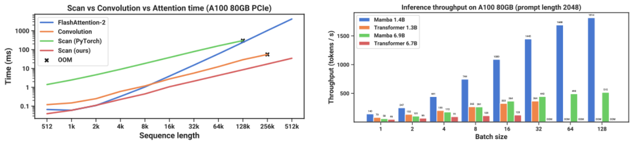

Fig. 3. Benchmarks [4]

IV. Experiments

A. Synthetic Data

To first check that our training setup is correct, we decided to overfit the model to synthetic data. For this purpose, we generated sequence $0, 1, ..., 1999$ then tried to make the model recreate it after receiving the 0 token as input. During the training, the loss quickly approached zero. When testing the model, it managed to generate the entire sequence correctly, albeit it required tuning the generation function's parameters.

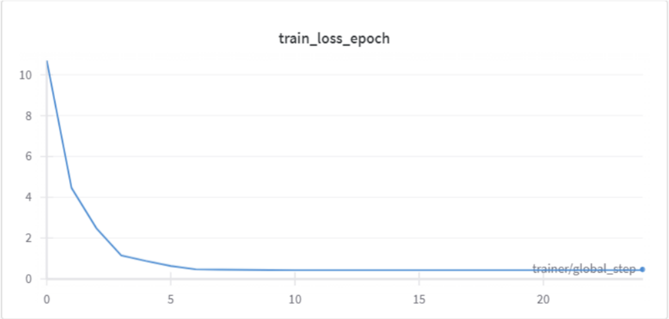

Fig. 4. Loss during training on synthetic data

We conducted a series of experiments to evaluate the performance of the Mamba architecture in symbolic music generation. The experiments are outlined as follows:

1) **Simple Sequence Training**: We began by training the model on a simple numerical sequence (e.g., 0, 1, 2, 3, 4, ...). This experiment was designed to verify the model's ability to learn and reproduce basic patterns. The main goal of this part was to catch any implementation bugs as soon as possible.
2) **Single Musical Piece Training**: Next, we trained the model on a single musical piece. This experiment aimed to assess the model's capacity to understand and generate music from a limited dataset. Even though it may seem to be a trivial task, playing with configuration was necessary – both training and inference.
3) **Comprehensive Training on Multiple Pieces**: Finally, we conducted full-scale training using a diverse set of musical pieces from the MAESTRO dataset. This experiment was intended to evaluate the model's performance in generating complex and varied musical compositions.

Each experiment was carefully monitored, and the generated outputs were analyzed to determine the model's effectiveness in capturing musical structures and patterns.

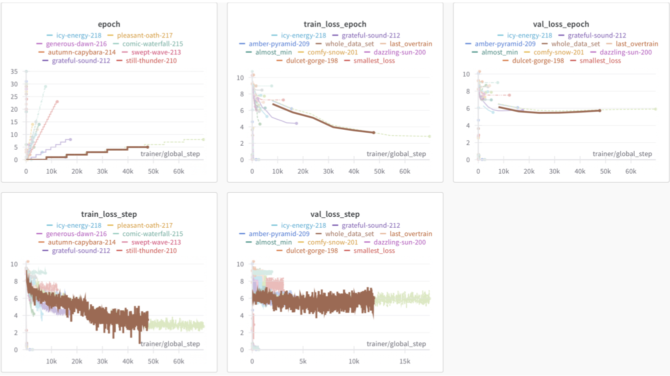

Fig. 5. Training and validation losses in various experiments

V. Conclusions

Although Mamba managed to generate novel samples, they weren't of high quality. This may be due to our models being unable to handle the complexities of virtuosic music. The model might be more suited for simpler datasets that do not contain such intricate musical scores.

A. Future work

While Mamba offers better performance than transformers, it comes at the cost of output quality. A promising direction for future research is to experiment with hybrid architectures that combine the strengths of both models. Examples of such architectures include **MambaFormer** [12] and **Jamba** [11]. It is also worth trying different datasets to see how Mamba handles them.

References

[1] Hao-Wen Dong. Muspy documentation. https://salu133445.github.io/muspy/, 2020.

[2] Nathan Fradet, Jean-Pierre Briot, Fabien Chhel, Amal El Fallah Seghrouchni, and Nicolas Gutowski. MidiTok: A python package for MIDI file tokenization. In *Extended Abstracts for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference*, 2021.

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[4] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[5] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state-space layers, 2021.

[6] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Maestro dataset. https://research.google/resources/datasets/maestro/, 2018.

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[9] Yu-Siang Huang and Yi-Hsuan Yang. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, page 1180–1188, New York, NY, USA, 2020. Association for Computing Machinery.

[10] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[11] Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

[12] Jongho Park, Jaeseung Park, Zheyang Xiong, Nayoung Lee, Jaewoong Cho, Samet Oymak, Kangwook Lee, and Dimitris Papailiopoulos. Can mamba learn how to learn? a comparative study on in-context learning tasks. *arXiv preprint arXiv:2402.04248*, 2024.

[13] Google Magenta Project. Maestro dataset. https://magenta.tensorflow.org/datasets/maestro, 2018.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017.

[15] Wikipedia contributors. Midi — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=MIDI&oldid=1153160350, 2023. [Online; accessed 5-May-2023].