Symbolic music generation using Mamba architecture

Jan Retkowski Miłosz Łopatto Jakub Stępniak

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 TODO**

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

1. 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) [9], Variational Autoencoders (VAEs) [11] and Generative Adversarial Networks (GANs) [3] were used to successfully generate music in symbolic formats. However, after popularisation of Transformers [13] and Diffusion Models [8] 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 the more efficient. However, they still suffer from quadratic GPU memory, as well as inference time complexity. To mitigate those problems State Space Models (SSMs) [6] were developed. Recently they have been experiencing gaining popularity and even beating transformers on some of their most signature tasks in the field of nature 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 to further test 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.

2. Data

We decided to use MIDI [14] 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 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) [7] [12] . It contains over 200 hours of virtuosic piano performances. The data comes from ten years of International Piano-e-Competition. It is one of the most popular datasets, which allows 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. The tokenization was conducted using miditok [2] library. It contains methods for tokenising MIDI data into most of the most popular formats. We decided to opt for REMI [10] format with learned Byte-Pair Encodings.

3. 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 (ideal for matrix calculations), and HBM, which is slower but has a larger capacity. The main bottleneck of

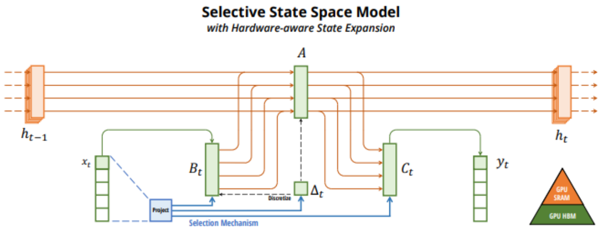

Figure 1. Selective SSM [4]

this approach is managing the data transfer between these memory types.

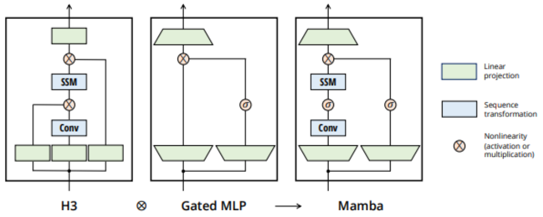

Figure 2. Mamba block [4]

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. Whole architecture is build from many Mamba block which are computed layer by layer.

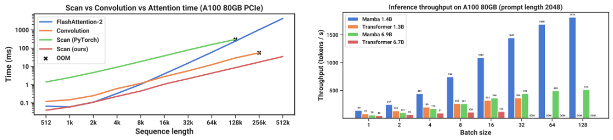

Figure 3. Benchmarks [4]

4. Training Setup

TODO

5. Experiments

5.1. 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 $0, 1, ..., 1999$ and trained the tried to make model generate it after receiving 0 token as imput. During the training loss quickly approached zero. When testing the model it managed to generate the entire sequence correctly, abeit it requred tunung the generation function's parameters.

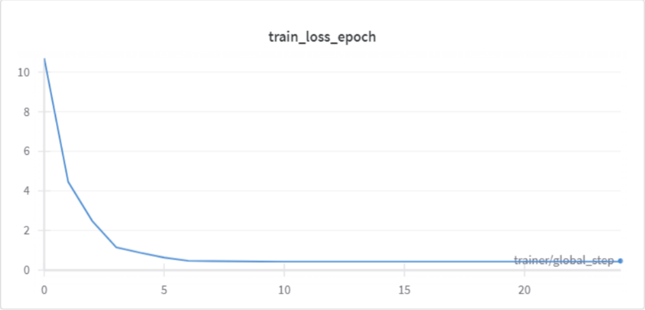

Figure 4. Loss during training on synthetic data

6. Conclusions

TODO

References

[1] Hao-Wen Dong. *MusPy Documentation*. https://salu133445.github.io/muspy/. 2020.

[2] Nathan Fradet et al. "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. URL: https://archives.ismir.net/ismir2021/latebreaking/000005.pdf.

[3] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[4] Albert Gu and Tri Dao. "Mamba: Linear-Time Sequence Modeling with Selective State Spaces". In: *arXiv preprint arXiv:2312.00752* (2023).

[5] Albert Gu, Karan Goel, and Christopher Ré. *Efficiently Modeling Long Sequences with Structured State Spaces*. 2022. arXiv: 2111.00396 [cs.LG].

[6] Albert Gu et al. *Combining Recurrent, Convolutional, and Continuous-time Models with Linear State-Space Layers*. 2021. arXiv: 2110.13985 [cs.LG].

[7] Curtis Hawthorne et al. *MAESTRO Dataset*. https://research.google/resources/datasets/maestro/. 2018.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG].

[9] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[10] 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. Seattle, WA, USA: Association for Computing Machinery, 2020, pp. 1180–1188. ISBN: 9781450379885. DOI: 10.1145/3394171.3413671. URL: https://doi.org/10.1145/3394171.3413671.

[11] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: 1312.6114 [stat.ML].

[12] Google Magenta Project. *MAESTRO Dataset*. https://magenta.tensorflow.org/datasets/maestro. 2018.

[13] Ashish Vaswani et al. "Attention is All You Need". In: 2017. URL: https://arxiv.org/pdf/1706.03762.pdf.

[14] Wikipedia contributors. *MIDI — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=MIDI&oldid=1153160350. [Online; accessed 5-May-2023]. 2023.