Structured State Space Models

Nicolas Stellwag [1]

Abstract

Structured *State Space Models (SSMs)* are an upcoming family of sequence models that show promising performance on a wide range of tasks. For instance, experiments with small models indicate that structured SSMs begin to outperform transformers in language modeling. Additionally, they exhibit desirable properties such as efficient sampling strategies for training and inference. In this seminar paper, I provide a gentle introduction to structured SSMs and give an overview over their advantages, disadvantages, applications, and my predictions of future developments.

1. Introduction

Large-scale sequence modeling becomes more and more relevant. The most prominent example is language modeling, which is largely dependent on scaling up the context windows models can take into account. Existing architectures are inherently flawed with respect to that problem. Transformers scale quadratically in the sequence length, resulting in slow inference and high resource requirements. RNNs suffer from poor performance, mainly caused by their inability to retain long-term context in the hidden state. Structured state space models are a promising alternative because they offer two alternative formulations: A recurrent formulation that allows for efficient inference, and a parallel formulation based on convolutions that allows for parallel training and circumvents gradient problems (Gu et al., 2022b). Additionally, structured SSMs are a theoretically stronger model family for continuous input signals. Transformers and classical RNNs are designed for discrete sequences, having no notion of continuity. In contrast, structured SSMs are based on mathematical foundations that explicitly model the input sequence as a continuous signal. That allows them to handle varying sampling intervals, even within single sequences, without any modifications.

My main contribution is a gentle introduction to structured state space models and their advantages and disadvantages. Additionally, I provide an overview over applications and corresponding state-of-the-art architectures, and give my prediction for further developments in the field.

[1]Technical University of Munich, Munich, Germany. Correspondence to: Nicolas Stellwag <nicolas.stellwag@tum.de>.

2. Background

2.1. Online Function Approximation

Main Idea In machine learning, the main idea of online function approximation is to provide models with mathematically principled, compressed representations of sequence histories. The first architecture that followed this approach was *Legendre Memory Units (LMUs)* (Voelker et al., 2019), but in the following I will introduce a more general solution.

HiPPO Framework The HiPPO framework (Gu et al., 2020) generalizes LMUs to a more abstract framework for online function approximation. It takes in a continuous input signal $u(t)$ and aims at approximating it with respect to a probability measure $\mu(t)$. The measure can be conceptualized as a weight function for the approximation error. Different instantiations of the HiPPO framework refer to different choices of measure. For instance, picking $\mu(t)$ as a window with limited context and uniform value yields LMUs. Since the approximation is to be computed online, meaning only $u_{\leq \tau}$ is known at time τ, the measure's support $(-\infty, \tau]$ varies. As a consequence, the measure $\mu^{(\tau)}(t)$ is a function of time t and parameterized by the current point in time τ.

The compression happens by projecting $u(t)$ onto a polynomial function basis $\{g_n\}_{n<N}$. The basis functions are chosen such that they are orthogonal with respect to the measure, meaning $\forall n, m < N : \langle g_n, g_m \rangle_{\mu^{(t)}} = \delta_{n,m}$ where the inner product with respect to any measure is defined as $\langle f, g \rangle_\mu = \int_0^\infty f(t)g(t)d\mu(t)$. The compressed representation of the input function history up to τ is the basis functions' coefficients, which are computed by projecting the input function onto the respective basis function via inner product: $c_n^{(\tau)} = \langle u_{\leq \tau}, g_n \rangle_{\mu^{(\tau)}}$. Recalling that the goal is to compute the compressed representation $c(t) \in \mathbb{R}^N$ online, the idea now is to differentiate the coefficients with respect to time. It turns out that $\frac{d}{dt}c(t)$ approximately evolves as a system of linear ordinary differential equations of the form

$$\frac{d}{dt}c(t) = A(t)c(t) + B(t)u(t) \tag{1}$$

where $A(t) \in \mathbb{R}^{N\times N}$ and $B(t) \in \mathbb{R}^{N\times 1}$ can be derived in closed-form for the respective measure. While the operators A, B depend on time in the general case, Gu et al. (2020) show that they are time-invariant for all HiPPO instantiations they work with. For the actual computation, the dynamics have to be discretized, which means they take the general form $c_{k+1} = A_k c_k + B_k u_k$. This introduces a discretization step size parameter Δt, which is crucial for performance.

Theoretically, the approximation of $u(t)$ can now be restored from the coefficients as $\hat{u}(t) = \sum_{n<N} c_n(t) g_n(t)$. Figure 1 visualizes the reconstructed signal for two different instantiations of the HiPPO framework. That being said, in practice the approximation is not actually realized. Instead, the models work directly on the compressed representation of the approximated signal history $c(t)$.

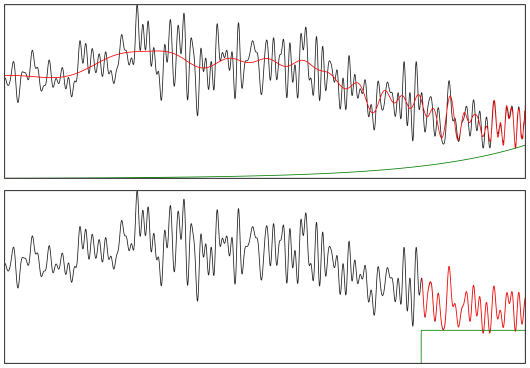

Figure 1. Plots of input signal $u(t)$ (black), HiPPO reconstruction $\hat{u}(t)$ (red), and measure function $\mu(t)$ (green) for the HiPPO instantiation *LegS* (exponentially decaying measure), and *LegT* (uniform, windowed measure) (Gu et al., 2020).[2]

2.2. State Space Models

Gu et al. (2021) introduce the *Linear State Space Layer (LSSL)*, the first (non-structured) SSM that incorporates ideas from the HiPPO framework. It can be written as

$$\frac{d}{dt}x(t) = Ax(t) + Bu(t) \tag{2}$$

$$y(t) = Cx(t) + Du(t) \tag{3}$$

where $u(t) \in \mathbb{R}$ is the input signal, $x(t) \in \mathbb{R}^N$ is the *state*, $y(t) \in \mathbb{R}^M$ is the output, and $A \in \mathbb{R}^{N\times N}, B \in \mathbb{R}^N, C \in \mathbb{R}^{M\times N}, D \in \mathbb{R}^M$ are learnable parameter matrices. $Du(t)$ acts as a projected residual connection and is irrelevant for any further derivations, which is why I omit it from now on.

A big difference to other sequence models is that the LSSL layer's continuous dynamics are manually discretized using a timescale parameter Δt. That makes it a good choice for settings such as dealing with irregularly spaced time series. Note that for sequences that are not based on a continuous signal, Δt can also be learned as a model parameter.

With random initialization, the LSSL does not achieve any noteworthy performance. But when provided with the compressed input function history using ideas from the HiPPO framework, it becomes a strong model. Equation (2) of the LSSL takes the same form as Equation (1). That allows to integrate the HiPPO framework by initializing A and B with the derived HiPPO matrix and bias for the chosen instantiation. It should be noted that the state dynamic does not stay fixed after the initialization. A and B are further optimized during training of the layer.

Finally, I address how to model multichannel sequences with LSSLs. The definition above describes a map $\mathbb{R} \to \mathbb{R}^M$ for each sequence element, but what is needed for sequences with H channels is a map $\mathbb{R}^H \to \mathbb{R}^H$. Gu et al. (2021) achieve this by training one LSSL per input-channel, and combining the parallel LSSL layers with a position-wise MLP $\mathbb{R}^{H\cdot M} \to \mathbb{R}^H$.

3. Structured State Space Models

3.1. Efficient Computation as Convolution

Problem Primitive state space models such as LSSL-based architectures have advantages such as dealing well with irregularly spaced sequences, and allowing for dynamic selection of discretization interval during training and inference. But fundamentally they are still recurrent architectures, which means they suffer from the typical problems such as slow training and gradient issues during training (Gu et al., 2022b). That also impacts their performance on long sequences, even though the HiPPO framework integration provides a theoretically strong fundament for good long-range performance.

Convolutional Formulation Gu et al. (2022b) introduce an alternative way to compute SSM outputs for all sequence positions at the same time. This enables fast, parallel training and eliminates gradient problems during training caused by unrolling the recurrent computational graph. Note that this formulation is not mutually exclusive with the recurrent formulation. You can train in parallel using the following convolutional formulation, but still do efficient, recurrent inference.

Plugging the LSSL's recurrent formulation (see Equations (2) and (3)) into itself a few times yields a closed-form solution for the output after several recurrent steps

$$y_k = \overline{CA^kB}u_0 + \cdots + \overline{CAB}u_{k-1} + \overline{CB}u_k \tag{4}$$

where for instance $\overline{A}$ is the discretized version of A. Therefore, it can be computed as a single *long convo-*

[2] Visualization code from: https://github.com/state-spaces/s4

lution $y = \overline{K} * u$, meaning a convolution with a kernel $\overline{K} = \left(\overline{C}\overline{A}^i\overline{B}\right)_{i\in[L]}$ of the same length L as the input signal.

The convolution itself can be computed efficiently in the Fourier domain, but the kernel is non-trivial to compute due to the powers of A (Gu et al., 2022b). The following approaches aim at finding an efficient way to compute the kernel by enforcing certain structures of A, thereby introducing the model family of *Structured SSMs*.

Normal Plus Low-Rank (S4) Performing the conjugation $(A, B, C) \sim (V^{-1}AV, V^{-1}B, CV)$ does not change the output an SSM computes, it only performs a base change of the state. That means one can conjugate A without changing the SSMs behavior. In principle, that allows to diagonalize the HiPPO matrix via conjugation $A = V^{-1}\Lambda V$, which would make computing the convolution kernel trivial and computationally inexpensive. Unfortunately, this primitive diagonalization is numerically unstable for larger states, because V would have entries exponential in the state size.

Instead, Gu et al. (2022b) exploit that HiPPO matrices can be decomposed as the sum of a normal matrix and a low-rank correction $A = N - PQ^T = V\Lambda V^* - PQ^T$ where $V \in \mathbb{C}^{N\times N}$ is unitary, Λ is diagonal, and $P, Q \in \mathbb{R}^{N\times r}$ form a low-rank factorization. Furthermore, the normal plus low-rank form can be turned into a diagonal plus low-rank form by the conjugation $V\Lambda V^* - PQ^T = V(\Lambda - (V^*P)(V^*Q)^*)V^*$.

Using the diagonal plus low-rank decomposition, the convolution kernel $\overline{K}$ can be efficiently evaluated using a few tricks. I briefly sketch Gu et al. (2022b)'s approach: *1)* Instead of computing $\overline{K}$ directly, they compute its spectrum by evaluating its *truncated generating function* at the roots of unity. From this representation, $\overline{K}$ can be efficiently recovered using the inverse Fourier transform. *2)* Evaluating the generating function does not involve a power of $\overline{A}$ anymore, but an inverse. There is an identity called *Woodbury matrix identity*, which reduces the problem of computing the inverse of any matrix plus low-rank correction $(M + LR^*)^{-1}$ to computing the inverse of the matrix M^{-1}. That can be used to reduce computing $\overline{A^{-1}}$ to the final, truly diagonal case. *3)* Computing the diagonal case is equivalent to the computation of so-called *Cauchy kernels*, which is a common problem with stable, near-linear time algorithms.

The resulting structured SSM is called *S4*. As already mentioned, it can be evaluated with two different algorithms. *S4 Recurrence* is the recurrent formulation and one step can be evaluated in $\mathcal{O}(N)$ operations. *S4 Convolution* can be used to evaluate the convolution kernel in $\mathcal{O}(N + L)$ operations and space, which can then be applied in $\mathcal{O}(L \cdot \log L)$. Note that this is a significant advantage over transformers, which are quadratic in the sequence length L.

Diagonal Further work such as *S4D* (Gu et al., 2022a), *DSS* (Gupta et al., 2022), and *GSS* (Mehta et al., 2023) shows that dropping the low-rank correction term does not heavily influence performance and allows for simpler and more efficient computation. The resulting *diagonal structured SSMs* form the basis of most advanced structured SSM architectures, including the ones discussed in further sections.

3.2. Addressing Input-Selectivity

Problem While structured SSMs have desirable properties with respect to computation and efficiency, their performance on discrete sequence modeling tasks is not on par with transformers yet. Their core problem is often referred to as missing *(input-)selectivity*, which means the parameters A, B, C, D, Δ, and therefore the computed operator, are independent of the input. Consequently, SSMs fail even on simple synthetic tasks such *Induction Head* (retrieving the token after special token), *Associative Recall* (retrieving a specific value from key-value input) (Fu et al., 2023), and *Selective Copying* (copying a sequence to unevenly spaced locations marked by special tokens) (Arjovsky et al., 2016). In the following, I present three architectures that address this issue. *H3* (Fu et al., 2023) and *Hyena* (Poli et al., 2023) try to achieve some input-selectivity by designing an architecture around "classical" structured SSMs. *Mamba* (Gu & Dao, 2023) goes one step further and explicitly turns some parameters into functions of the input.

H3 Fu et al. (2023) introduce the H3 architecture, which is specifically designed to solve Induction Head and Associative Recall and achieves a large jump in performance for both of them. Both of these tasks require the ability to perform an operation on a token based on the previous token's value. H3 starts off by computing 3 projections Q, K, V of the current input token u, which are loosely related to the projections of the attention mechanism. Then, the layer output is calculated as

$$\mathrm{H3}(Q, K, V) = Q \odot \mathrm{SSM}_{\mathrm{diag}}\big(\mathrm{SSM}_{\mathrm{shift}}(K) \odot V\big) \quad (5)$$

which can be conceptualized as two steps: *1)* K is fed through a so-called *Shift-SSM*, an SSM with A fixed to shift the state one position "down" ($x_0^{(t)}$ becomes 0, $x_1^{(t)}$ becomes $x_0^{(t)}$ and so on). Even though B is learned in practice, from assuming it to be fixed to the first basis vector e_1 and recalling Equation (2), one can see that this keeps a local history of the scalar SSM inputs in the state vector. The Shift-SSM's output interacts with V via Hadamard product, thereby relating the previous token's K to the current token's V. *2)* The output of this operation is fed through a "normal" diagonal SSM, and the final layer output is computed as Hadamard product with Q.

Additionally, the authors introduce a hardware-optimized CUDA kernel called *FlashConv*, that speeds up the convolution computation by fusing operations in the SRAM and

only then writing back to HBM, similar to the well-known *FlashAttention* (Dao et al., 2022).

Hyena By introducing Hyena, Poli et al. (2023) generalize the concept of the H3 layer, thereby drastically improving the performance on recall and reasoning tasks such as the ones mentioned above. To demonstrate the analogy to H3, I first provide a different view on SSMs and H3: SSMs effectively compute an *implicitly parameterized long convolution*, which means the convolution kernel's elements are functions of the actual model parameters. H3's convolutions and Hadamard products can be rewritten as matrix multiplications $\text{H3}(q, k, v) = D_q S_\psi D_k S_\varphi \cdot v$ where $D_x = \text{diag}(x)$ and S_ψ, S_φ are Töplitz matrices for the convolution kernels realized by the diagonal SSM and Shift-SSM respectively. The Hyena layer generalizes that to $y = D_x^{N_H} S_h^{N_H} \dots D_x^1 S_h^1 \cdot v$. That means H3 is an instantiation of Hyena with $N_H = 2$ and a constrained first kernel. Of course, in practice all these matrices are not materialized, but computed by convolutions in the Fourier domain and Hadamard products.

Strictly speaking, the Hyena framework is not just an SSM architecture, but a more general framework of interleaved implicitly parameterized long convolutions and data-controlled gating by Hadamard products. Poli et al. (2023) compute their kernels as $h_t = \text{Window}(t) \cdot \text{MLP}\big(\text{PositionalEncoding}(t)\big)$ where Window is a weighting function for the sequence positions, and most commonly chosen as an exponential decay. It is still appropriate to view Hyena as an architectural framework for SSMs. Since SSMs also realize a convolution with an implicit kernel, the Hyena convolutions with the kernels specified above can simply be replaced with SSM layers.

Mamba H3 and Hyena try to achieve some input selectivity by designing an architecture around classical SSMs. With Mamba, Gu & Dao (2023) go one step further and actually compute the SSM parameters Δ, B, C as learnable functions of the input. This results in a massive boost in performance, with Mamba being the first SSM-based model to solve the Selective Copying task and being the first to match the performance of transformers on language modeling.

This performance increase is no free lunch. Since the parameters are now not only functions of the time but also of the input, the convolutional formulation is not possible anymore. To mitigate this, Gu & Dao (2023) introduce a hardware-optimized scan algorithm that fuses the discretization and a limited number of recurrence steps to a single CUDA kernel, preventing most IO operations between SRAM and HBM.

4. Applications & Performance

Long-Range Sequence Modeling Since SSMs scale subquadratically with the sequence length, they allow for deep models, even when large context windows are required. Based on the architectures presented in this seminar paper, there is follow-up work that specifically tackles long range tasks. Hasani et al. (2023) combine SSMs with ideas from *liquid time-constant networks* (Hasani et al., 2021). Gu et al. (2023) derive an alternative formulation of the HiPPO framework and introduce novel instantiations. Smith et al. (2023) fuse multiple S4 layers to a multi-input multi-output *S5* layer. Table 1 provides an overview over *Long Range Arena (LRA)* (Tay et al., 2021) benchmark scores. I also include *Big Bird* (Zaheer et al., 2020), a strong transformer baseline explicitly designed to handle long-range dependencies. The comparison shows that SSM-based architectures perform significantly better than transformer-based architectures.

Language Modeling Table 2 shows results of SSMs and a transformer baseline on the WikiText-103 (Merity et al., 2017) language modeling task. For these smaller models, SSM-based models designed for input-selectivity start to exceed transformer-baselines. Experiments with large-scale models are not publicly available yet.

Other Applications Potential applications of SSMs are manifold. Qiao et al. (2024) introduce a Mamba-based multimodal LLM. Zhu et al. (2024) present a drop-in replacement for vision transformers. Liu et al. (2024); Ruan & Xiang (2024) perform 2D image segmentation, Xing et al. (2024) 3D volume segmentation, and Yang et al. (2024) 4D volume-video segmentation. Fei et al. (2024) train a diffusion model based on SSMs. Liang et al. (2024) introduce a 3D point cloud processing backbone. SSMs are also applied to other domains, such as speech (Goel et al., 2022; Zhang et al., 2024), time series (Zhang et al., 2023; Patro & Agneeswaran, 2024), and graphs (Wang et al., 2024; Behrouz & Hashemi, 2024).

5. Conclusion

Structured SSMs allow for fast parallel training and efficient recurrent inference, which makes them a promising model family for sequence modeling. The fact that most state-of-the-art models rely on Mamba as a backbone indicates that input-selectivity is crucial for a lot of tasks, though. Consequently, I suspect "pure" linear time invariant SSMs will only be used for niche applications such as irregularly spaced sequences in the future. That does not make SSMs useless for tasks that require input-selectivity, such as language modeling. The trend in language modeling seems to go further into the direction of scaling up models as much as possible, which is why efficiency becomes more and more relevant. Replacing transformer layers by SSM layers allows researchers to make fine-grained trade-offs between expressivity and efficiency for individual layers, potentially resulting in higher performance of the full models.

References

Arjovsky, M., Shah, A., and Bengio, Y. Unitary evolution recurrent neural networks. In Balcan, M. and Weinberger, K. Q. (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1120–1128. JMLR.org, 2016.

Behrouz, A. and Hashemi, F. Graph mamba: Towards learning on graphs with state space models. *CoRR*, abs/2402.08678, 2024. doi: 10.48550/ARXIV.2402.08678.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Fei, Z., Fan, M., Yu, C., and Huang, J. Scalable diffusion models with state space backbone. *CoRR*, abs/2402.05608, 2024. doi: 10.48550/ARXIV.2402.05608.

Fu, D. Y., Dao, T., Saab, K. K., Thomas, A. W., Rudra, A., and Ré, C. Hungry hungry hippos: Towards language modeling with state space models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Goel, K., Gu, A., Donahue, C., and Ré, C. It's raw! audio generation with state-space models. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S. (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 7616–7633. PMLR, 2022.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. *CoRR*, abs/2312.00752, 2023. doi: 10.48550/ARXIV.2312.00752.

Gu, A., Dao, T., Ermon, S., Rudra, A., and Ré, C. Hippo: Recurrent memory with optimal polynomial projections. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Gu, A., Johnson, I., Goel, K., Saab, K., Dao, T., Rudra, A., and Ré, C. Combining recurrent, convolutional, and continuous-time models with linear state space layers. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 572–585, 2021.

Gu, A., Goel, K., Gupta, A., and Ré, C. On the parameterization and initialization of diagonal state space models. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022a.

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022b.

Gu, A., Johnson, I., Timalsina, A., Rudra, A., and Ré, C. How to train your HIPPO: state space models with generalized orthogonal basis projections. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Gupta, A., Gu, A., and Berant, J. Diagonal state spaces are as effective as structured state spaces. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Hasani, R. M., Lechner, M., Amini, A., Rus, D., and Grosu, R. Liquid time-constant networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 7657–7666. AAAI Press, 2021. doi: 10.1609/AAAI.V35I9.16936.

Hasani, R. M., Lechner, M., Wang, T., Chahine, M., Amini, A., and Rus, D. Liquid structural state-space models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Liang, D., Zhou, X., Wang, X., Zhu, X., Xu, W., Zou, Z., Ye, X., and Bai, X. Pointmamba: A simple state space model for point cloud analysis. *CoRR*, abs/2402.10739, 2024. doi: 10.48550/ARXIV.2402.10739.

Liu, J., Yang, H., Zhou, H., Xi, Y., Yu, L., Yu, Y., Liang, Y., Shi, G., Zhang, S., Zheng, H., and Wang, S. Swin-umamba: Mamba-based unet with imagenet-based pretraining. *CoRR*, abs/2402.03302, 2024. doi: 10.48550/ARXIV.2402.03302.

Mehta, H., Gupta, A., Cutkosky, A., and Neyshabur, B. Long range language modeling via gated state spaces. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Patro, B. N. and Agneeswaran, V. S. Simba: Simplified mamba-based architecture for vision and multivariate time series. *CoRR*, abs/2403.15360, 2024. doi: 10.48550/ARXIV.2403.15360.

Poli, M., Massaroli, S., Nguyen, E., Fu, D. Y., Dao, T., Baccus, S., Bengio, Y., Ermon, S., and Ré, C. Hyena hierarchy: Towards larger convolutional language models. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 28043–28078. PMLR, 2023.

Qiao, Y., Yu, Z., Guo, L., Chen, S., Zhao, Z., Sun, M., Wu, Q., and Liu, J. Vl-mamba: Exploring state space models for multimodal learning. *CoRR*, abs/2403.13600, 2024. doi: 10.48550/ARXIV.2403.13600.

Ruan, J. and Xiang, S. Vm-unet: Vision mamba unet for medical image segmentation. *CoRR*, abs/2402.02491, 2024. doi: 10.48550/ARXIV.2402.02491.

Smith, J. T. H., Warrington, A., and Linderman, S. W. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena : A benchmark for efficient transformers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Voelker, A., Kajic, I., and Eliasmith, C. Legendre memory units: Continuous-time representation in recurrent neural networks. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 15544–15553, 2019.

Wang, C., Tsepa, O., Ma, J., and Wang, B. Graph-mamba: Towards long-range graph sequence modeling with selective state spaces. *CoRR*, abs/2402.00789, 2024. doi: 10.48550/ARXIV.2402.00789.

Xing, Z., Ye, T., Yang, Y., Liu, G., and Zhu, L. Segmamba: Long-range sequential modeling mamba for 3d medical image segmentation. *CoRR*, abs/2401.13560, 2024. doi: 10.48550/ARXIV.2401.13560.

Yang, Y., Xing, Z., and Zhu, L. Vivim: a video vision mamba for medical video object segmentation. *CoRR*, abs/2401.14168, 2024. doi: 10.48550/ARXIV.2401.14168.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontañón, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. Big bird: Transformers for longer sequences. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Zhang, M., Saab, K. K., Poli, M., Dao, T., Goel, K., and Ré, C. Effectively modeling time series with simple discrete state spaces. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Zhang, X., Zhang, Q., Liu, H., Xiao, T., Qian, X., Ahmed, B., Ambikairajah, E., Li, H., and Epps, J. Mamba in speech: Towards an alternative to self-attention. *CoRR*, abs/2405.12609, 2024. doi: 10.48550/ARXIV.2405.12609.

Zhu, L., Liao, B., Zhang, Q., Wang, X., Liu, W., and Wang, X. Vision mamba: Efficient visual representation learning with bidirectional state space model. *CoRR*, abs/2401.09417, 2024. doi: 10.48550/ARXIV.2401.09417.

A. Applications & Performance

Table 1. Long Range Arena (Tay et al., 2021) scores for long range models. Higher is better. All results taken from (Hasani et al., 2023).

	MODEL / TASK (INPUT LENGTH)	AAN 4000	PATH-X 16384	AVG.
(GU ET AL., 2022B)	S4	87.09	88.10	80.48
(GU ET AL., 2022A)	S4D-INV	91.09	92.80	85.50
(GUPTA ET AL., 2022)	DSS	87.6	85.0	79.45
(HASANI ET AL., 2023)	LIQUID-S4	91.20	96.66	87.32
(GU ET AL., 2023)	S4-LEGS/FOUT	90.30	×	78.50
(SMITH ET AL., 2023)	S5	88.26	85.25	82.46
(ZAHEER ET AL., 2020)	BIG BIRD	59.29	×	55.01

Table 2. WikiText-103 (Merity et al., 2017) test perplexity for language models. Lower is better. If not specified otherwise, results are taken from the model papers. *Includes two transformer layers. **WikiText-103 results only published in response to reviews: https://openreview.net/forum?id=AL1fq05o7H (visited: 12 Jun 2024)

	MODEL	PERPLEXITY	#PARAM
(GU ET AL., 2022B)	S4	21.0	249M
(FU ET AL., 2023)	HYBRID-H3*	18.5	125M
(POLI ET AL., 2023)	HYENA-3	18.5	125M
(GU & DAO, 2023)	MAMBA**	16.3	125M
(POLI ET AL., 2023)	TRANSFORMER BASELINE	18.6	125M