From system identification to sequence models: a primer on Structured State-Space Models

Fabio Bonassi *, **Per Mattsson** *, **Thomas B. Schön** *

* *Department of Information Technology, Uppsala University, 75105 Uppsala, Sweden. E-mail:* name.surname@it.uu.se

In recent years, the machine learning community has developed increasingly flexible and accurate models for sequence data. Consider, for example, Transformers and their widespread applications in large language models. Despite their state-of-the-art performance, many of these architectures are hindered by poor scalability with respect to sequence length T. For example, in basic Transformers both the computational cost and the number of parameters scale as $\mathcal{O}(T^2)$, making these models expensive and largely overparametrized when long sequences need to be processed. The Structured State-space Model (SSM) [1] was proposed to overcome these limitations. A simplified SSM architecture—depicted in Figure 1—consists in a sequence of layers (SSL) made from a linear dynamical component, a static nonlinearity, and a skip connection. Between layers, linear projections or layer normalization blocks might also be present, but they are omitted here for simplicity.

There are several reasons behind the interest in SSMs. First, since the dynamics are linear-in-the-states, they can be very efficiently unrolled over extremely long sequences. By parallelizing computations (sub)linear time complexities can be achieved. Moreover, since they are recurrent architectures, the number of learnable parameters is independent of T, making them less prone to overparametrization. Lastly, they excel in extremely-long memory [2].

Structured State-Space Models (SSMs) build on two well-established pillars of system identification: block-oriented models and state-space models. From this perspective, SSMs can be seen as *deep state-space Wiener models trained via gradient descent-based simulation error minimization* [3]. Their effectiveness hinges on a structured state-space formulation that enables computational efficiency during both training and inference, which is paramount in machine learning applications. To make this possible, SSMs address two well-known issues in state-space identification [4]: (*i*) the structure and parametrization of the dynamical component, and (*ii*) the initialization of the dynamics' learnable parameters. In what follows, we summarize the strategies employed by the three main SSM architectures to address these issues.

Discrete-time Linear Recurrent Units

Liner Recurrent Units (LRU) are discrete-time SSMs where each layer $\ell \in \{1, ..., L\}$ is described by

$$\text{SSL}_\ell : \begin{cases} x_{k+1} = \Lambda x_k + \Gamma B u_k, & \text{(1a)} \\ \eta_k = \mathfrak{Re}(C x_k) + D u_k, & \text{(1b)} \\ y_k = \sigma(\eta_k) + F u_k, & \text{(1c)} \end{cases}$$

with $x_k \in \mathbb{C}^{n_x}$ being the state, $u_k \in \mathbb{R}^{n_u}$ the input, $\eta_k \in \mathbb{R}^{n_h}$ the hidden output, and $y_k \in \mathbb{R}^{n_y}$ the output.

Parametrization Λ is diagonal, complex-valued, and structurally Schur, i.e., $\Lambda = \text{diag}(\lambda_1, ..., \lambda_{n_x})$ with

$$\lambda_j := e^{-\exp(\mu_j) + i \exp(\theta_j)}. \quad (2)$$

The SSL can hence learn stable, complex dynamics ($|\lambda_j| < 1$). Because the state vector is complex, the real part is extracted in the hidden output transformation (1b), which is equivalent to enforcing the complex-conjugateness of the state vector [3]. As shown in [2], it is paramount to normalize the gain of state components, to ensure they have comparable orders of magnitude. In [2] states are normalized with respect to the ℓ_2 gain, $\Gamma = \text{diag}(\gamma_1, ..., \gamma_{n_x})$,

$$\gamma_j = \sqrt{1 - |\lambda_j|^2} = \sqrt{1 - e^{-2\exp(\mu_j)}}, \quad (3)$$

albeit the static gain $\gamma_j = 1 - |\lambda_j| = 1 - e^{-\exp(\mu_j)}$ can alternatively be adopted. The other matrices are dense matrices of proper dimensions: B and C are generally complex-valued, while D and F are real-valued.

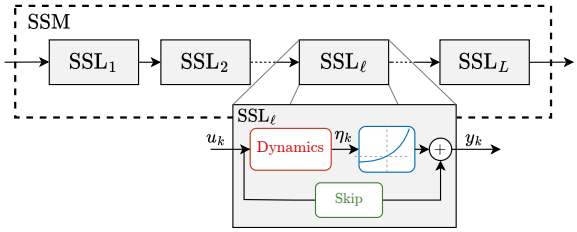


Fig. 1. Schematic of a general SSM.

Initialization A crucial ingredient for learning accurate SSMs, and state-space models in general [4], is the initialization strategy. Orvieto et al. [2] proposed to sample the initial eigenvalues $\lambda_1, ..., \lambda_n$ from a suitable circular crown sector (see Figure 2a), from which the initial μ_j and θ_j are computed. Ideally, such a sector should be densely sampled (c.f. Figure 3 of [2]) by increasing the state dimension, thus avoiding bad local minima and limiting the frequency bias [5]. The other matrices are initialized randomly.

Efficient training The computational efficiency of LRUs builds on two pillars. First, owing to the diagonality of Λ, computing x_{k+1} given x_k and u_k scales linearly with n_x, which is especially important for large models. Second, the linear dynamics of LRUs enable the computation of the state trajectory $x_{0:T}$ to be parallelized across the time dimension, [1] leading to (sub)linear scaling. This is possible because the state update function is a binary associative operation that allows for Parallel Scan [7].

Continuous-time diagonal reparametrization

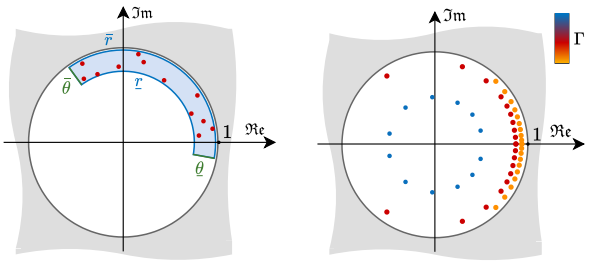
Despite the discrete-time formulation (1) being the most intuitive, original SSMs have often been reparametrized in the continuous-time domain,

$$\text{SSL}_\ell : \begin{cases} \dot{x}(t) = \Gamma A x(t) + \Gamma B u(t), & \text{(4a)} \\ \eta(t) = \mathfrak{Re}(C x(t)), & \text{(4b)} \\ y(t) = \sigma(\eta(t)) + F u(t). & \text{(4c)} \end{cases}$$

Parametrization Γ is a learnable diagonal matrix with strictly positive entries that acts as a time-scaling operator, dilating or contracting time [3; 8]. In the popular S5 architecture [9], A is defined as a diagonal, complex-valued, Hurwitz matrix. That is, $A = \Lambda = \text{diag}(\lambda_1, ..., \lambda_{n_x})$, where

⋆ This research was financially supported by *Kjell och Märta Beijer Foundation.*

[1] Traditional RNNs are nonlinear state-space models [6] and must be unrolled recursively over time, i.e., x_k is required to compute x_{k+1}. Therefore, they do not allow for parallelization across the time axis.



(a) (b)

Fig. 2. Eigenvalues initialization (a) by random sampling in a circular crown sector versus (b) by discretizing the HiPPO-LegS' diagonal component with scalar Γ.

$$\lambda_j = -e^{\mu_j} + ie^{\theta_j}. \tag{5}$$

Because these models process (regularly sampled) discrete-time signals, the continuous-time system (4) is discretized, typically via zero-order hold or bilinear transformation, yielding a system similar to (1). A continuous-time parametrization is useful for two main reasons. First, it enables handling sequences with varying sampling rates during both training and inference. Second, the discretization step has been shown [2] to induce a gain regularization, thereby enhancing the model's numerical conditioning.

Initialization As with LRUs, initialization is paramount. Smith et al. [9] proposed to initialize $A = \Lambda$ with the diagonal component of the HiPPO-LegS matrix [1]. This matrix—projecting time-series onto a low-dimensional space of Legendre polynomials—can be decomposed into a complex diagonal component plus a (discarded) rank-1 component. The ZOH discretization of the resulting $\Gamma\Lambda$ has eigenvalues illustrated in Figure 2b. The time-scale Γ is initialized sampling from a uniform distribution. [2]

Efficient training The differentiability of the discretization step allows these models to be trained with the same Parallel Scan algorithm as LRUs.

Remark 1. The architecture that kickstarted the interest in SSMs was the S4 architecture [1]. These models follow the form of (4), with a HiPPO-LegS-like diagonal plus low-rank A matrix which allows for an efficient simulation in the frequency domain (via FFT) [3].

Remark 2. S4 and S5 revolved around the specific HiPPO-LegS initialization due to its alleged long-term memory property. While the HiPPO matrix provides a reasonable set of initial eigenvalues—at least if Γ is properly initialized—it is has later been shown to be not fundamental [2]. Any randomized initialization can lead to similar (or even superior) levels of accuracy, provided that the initial eigenvalues are neither too slow nor too fast.

Selective Structured State-space Models

A recent development are the selective SSMs (S6) which have been proposed as part of the Mamba architecture [10]. Its dynamics can be considered *Linear Input Varying* (LIV). That is, letting the subscript $_{[u(t)]}$ denote the instantaneous dependency on $u(t)$, each S6 layer reads

$$\text{Selective SSL}_\ell\colon \begin{cases} \dot{x}(t) = \Gamma_{[u(t)]}\Lambda x(t) + \Gamma_{[u(t)]}B_{[u(t)]}\,u(t), & \text{(6a)} \\ \eta(t) = \mathfrak{Re}\big[C_{[u(t)]}\,x(t)\big], & \text{(6b)} \\ y(t) = \sigma(\eta(t)) + Fu(t). & \text{(6c)} \end{cases}$$

Parametrization Let $\mathrm{lin}(u, \Phi)$ be a vector-to-matrix affine transformation of u parametrized by Φ. Then, $\Gamma_{[u(t)]} = \mathrm{diag}\big(\mathrm{softmax}(\mathrm{lin}(u_t; \Phi_\Gamma))\big)$, $B_{[u(t)]} = \mathrm{lin}(u_t; \Phi_B)$, and $C_{[u(t)]} = \mathrm{lin}(u_t; \Phi_C)$ represent the input-varying system matrices. Λ is parametrized and initialized as in (4) and (5). This parametrization enables the system to *adaptively gate* inputs (amplifying or suppressing them), thus allowing for state reset and memory retention. These operations are crucial in many machine learning tasks, but are not achievable with traditional SSMs [11].

Efficient training Despite being more involved than both the LRU (1) and S5 (4) formulations, this architecture can still be trained with linear efficiency via the Parallel Scan algorithm. Achieving this required the hardware-aware CUDA implementation in [10], which was crucial in mitigating GPU bandwidth bottlenecks.

Discussion

While SSMs are closely related to classical state-space models, novel factors have contributed to their recent success. First, their *depth* enables more complex dynamics to be learned. Second, the use of *carefully chosen initialization strategies*, which select eigenvalues that are neither too slow, nor too fast. Combined with the large number of states typically employed, this results in rich dynamics. Lastly, and most importantly, the *model structures* and *training strategies* are tailored for extreme computational scalability, supported by open, GPU-optimized implementations, which are essential in modern machine learning.

Despite these advantages, several open problems remain, where we believe that the control community could offer valuable insights. Among them is the SSM's frequency bias [5], which leads to underfitting of components at higher frequencies than the fastest initial eigenvalue. Moreover, the discussed parametrization are non-minimal, and thus not invariant to coordinate transformations. Conversely, minimal canonical forms are computationally inefficient due to the increased cost of dense matrix product.

[2] Usually with support $[0.001/\tau_s, 0.1/\tau_s]$, where τ_s is the (average) sampling time of the training data. This ensures that the initial eigenvalues are close enough to the unit circle, cf. Figure 2b.

REFERENCES

[1] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," in *The Tenth International Conference on Learning Representations*, 2022.

[2] A. Orvieto, S. L. Smith, A. Gu, A. Fernando, C. Gulcehre, R. Pascanu, and S. De, "Resurrecting recurrent neural networks for long sequences," *International Conference on Machine Learning*, vol. abs/2303.06349, pp. 26670–26698, 2023.

[3] F. Bonassi, C. Andersson, P. Mattsson, and T. B. Schön, "Structured state-space models are deep wiener models," in *20th IFAC Symposium on System Identification (SYSID)*, 2024.

[4] L. Ljung, *System identification: Theory for the user*. Prentice Hall information and system sciences series, Prentice Hall, 1998.

[5] A. Yu, D. Lyu, S. H. Lim, M. W. Mahoney, and N. B. Erichson, "Tuning frequency bias of state space models," in *The Thirteenth International Conference on Learning Representations*, 2025.

[6] F. Bonassi, M. Farina, J. Xie, and R. Scattolini, "On recurrent neural networks for learning-based control: recent results and ideas for future developments," *Journal of Process Control*, vol. 114, pp. 92–104, 2022.

[7] G. E. Blelloch, "Prefix sums and their applications," 1990.

[8] J. Weigand, G. I. Beintema, J. Ulmen, D. Görges, R. Tóth, M. Schoukens, and M. Ruskowski, "State derivative normalization for continuous-time deep neural networks," *IFAC-PapersOnLine*, vol. 58, no. 15, pp. 253–258, 2024.

[9] J. T. H. Smith, A. Warrington, and S. W. Linderman, "Simplified state space layers for sequence modeling," in *The Eleventh International Conference on Learning Representations*, 2023.

[10] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," in *First Conference on Language Modeling*, 2024.

[11] W. Merrill, J. Petty, and A. Sabharwal, "The illusion of state in state-space models," in *41st International Conference on Machine Learning*, 2024.