Structured State-space Models

EPFL- EECI PhD School 2025

Leonardo Massai*

l.massai@epfl.ch

Ecole Polytechnique Fédérale de Lausanne*

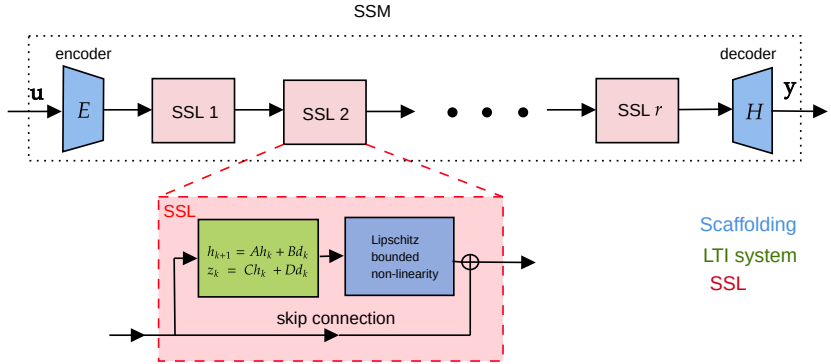
State Space Models

What is an SSM?

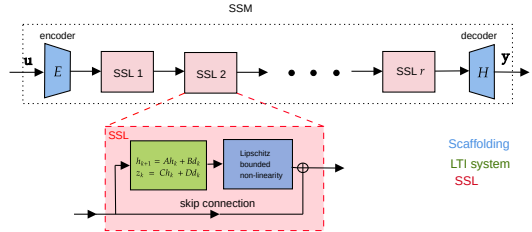
An SSM is an input/output operator defined through a dynamical system consisting of multiple layers, each composed of

- A **discrete-time LTI system**. It can be discretized from a CT or directly modeled in DT.
- A pre- and post-processing of its input and output, typically involving **static nonlinear function** (Wiener models). All the processing done around the LTI system is also known as the SSM **scaffolding**.

Example of SSM structure (LRU)



State Space Models



SSM structure

Input/output operator $\mathcal{M} : \mathbf{u} \in \mathcal{L}^{n_u} \mapsto \mathbf{y} \in \mathcal{L}^{n_y}$

- **LTI system**:

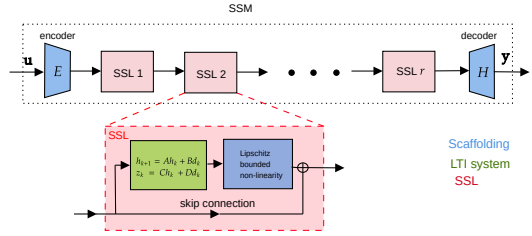
$$g_{(A,B,C,D)} : \begin{cases} x_{k+1} & = Ax_k + Bd_k, \;\; x_0 = 0 \\ z_k & = Cx_k + Dd_k \;, \end{cases}$$

  with $h \in \mathbb{R}^{n_h}$, $d, z \in \mathbb{R}^n$. A common requirement is that A must be Schur, ensuring system stability (ISS).

- The output of g is fed to a **static nonlinearity** that belongs to a family of parametrized functions $\sigma_\xi : \mathbb{R}^n \mapsto \mathbb{R}^n$ depending on the parameter $\xi \in \mathbb{R}^m$.

- The **scaffolding** commonly includes a linear encoder/decoder defined by the matrices $E \in \mathbb{R}^{n \times n_u}, H \in \mathbb{R}^{n_y \times n}$. These are just linear transformations of the form $\mathbf{y} = E\mathbf{u}$.

The LRU



LRU

- For a fixed number of layers $r \in \mathbb{N}$, the LRU is a parametrized operator $\mathcal{M}_\theta : \mathbf{u} \in \mathcal{L}^{n_u} \mapsto \mathbf{y} \in \mathcal{L}^{n_y}$ with:

$$\theta \in \mathcal{P} = \{\{A_i, B_i, C_i, D_i, \xi_i\}_{1 \leq i \leq r}, E, H\}$$

- An LRU [a] with $r \in \mathbb{N}$ layers processes its input $\mathbf{u}$ in the following manner:

$$\begin{aligned} &\text{(encoder)} && \mathbf{y}_0 = E\mathbf{u} \\ &\text{(SSL)} && \mathbf{y}_i = \sigma_i\left(g_i(\mathbf{y}_{i-1})\right) + \mathbf{y}_{i-1}, \quad 1 \leq i \leq r \\ &\text{(decoder)} && \mathbf{y} = H\mathbf{y}_r \end{aligned}$$

[a] *Resurrecting Linear Recurrences*, Orvieto et al., 2023

LTI Systems

Why LTI systems?

1. Output can be computed both recursively (during inference) and via convolution: $z_k = \sum_{\tau=0}^{k-1} CA^{k-\tau-1}Bd_\tau + Dd_k$ allowing parallelization and **fast computing** during training (parallel scan).
2. Properties such as **stability** are easy to enforce.

Enforcing stability: complex diagonal parametrization

- We want to parametrize Schur matrices $A \in \mathbb{R}^{n_h \times n_h}$. A convenient way to do it is to use diagonal complex-valued matrices.
- Almost all matrices $A \in \mathbb{R}^{n_h \times n_h}$ are diagonalizable over $\mathbb{C}$.
- Given a diagonalizable A we can write $A = P\Lambda P^{-1}$ and construct the similar system with state $\tilde{x}_k := P^{-1}x_k, \tilde{B} := P^{-1}B$ and output $\tilde{z}_k = \tilde{C}\tilde{x}_k + Dd_k = Cx_k + Dd_k = z_k$ where $\tilde{C} := CP$.
- This implies that instead of learning (A, B, C, D), one can learn $(\Lambda, \tilde{B}, \tilde{C}, D)$, where $\Lambda, \tilde{B}, \tilde{C}$ are complex-valued, and Λ is diagonal.

LTI systems

A formulation with Λ with conjugate pairs

- When A is of even dimension with complex conjugate pairs of eigenvalues, one can write:

$$\Lambda = \begin{bmatrix} \bar{\Lambda} & \\ & \text{conj}(\bar{\Lambda}) \end{bmatrix}, \quad \tilde{B} = \begin{bmatrix} \bar{B} \\ \text{conj}(\bar{B}) \end{bmatrix}, \quad \tilde{C} = [\bar{C} \;\; \text{conj}(\bar{C})]$$

  where $\bar{\Lambda} \in \mathbb{C}^{n_\lambda \times n_\lambda}, \bar{B} \in \mathbb{C}^{n_\lambda \times n}, \bar{C} \in \mathbb{C}^{n_z \times n_\lambda}$, while $D \in \mathbb{R}^{n_z \times n}$, with $n_\lambda = \frac{n_h}{2}$.

- Substituting you get

$$\begin{bmatrix} \bar{x}_{k+1} \\ \text{conj}(\bar{x}_{k+1}) \end{bmatrix} = \begin{bmatrix} \Lambda \bar{x}_k \\ \text{conj}(\Lambda \bar{x}_k) \end{bmatrix} + \begin{bmatrix} \bar{B} u_k \\ \text{conj}(\bar{B}) u_k \end{bmatrix} \tag{1}$$

$$z_k = \bar{C}\bar{x}_k + \text{conj}(\bar{C}\tilde{x}_k) + Du_k = 2\mathcal{R}e(\bar{C}\bar{x}_k) + Du_k \tag{2}$$

- We can train the smaller subsystem $(\bar{\Lambda}, \bar{B}, \bar{C})$ and compute the output as $z_k = 2\mathcal{R}e(\bar{z}_k) + Du_k$ for further reduction of the computational burden without any approximations.
- Parameterize $\bar{\Lambda} = \text{diag}\,(\lambda_1, \ldots, \lambda_{n_\lambda})$ with $\lambda_j = e^{-e^{\mu_j}} e^{\theta_j i}$ ensuring $|\lambda_j| = e^{-e^{\mu_j}} < 1 \,\forall\, \mu_j \in \mathbb{R}$.

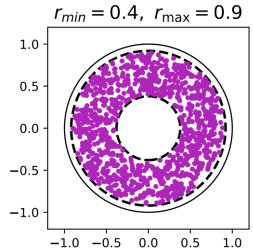
Initialization

Initialization Strategies

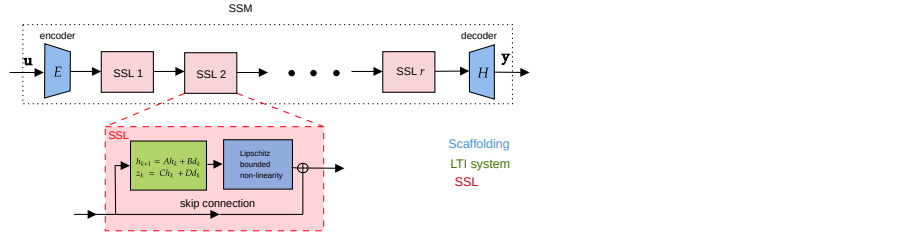
- All parameterizations aim to ensure that $|\lambda_j| < 1$ and $|\lambda_j| \approx 1$ to push the system towards marginal stability. Intuitively, this allows the system to have long-range memory and prevents the signal from past inputs from quickly dying out.

Towards marginal stability for long dependencies

- The idea is to constrain the eigenvalues to lie on a disc inside the unitary circle and close to the stability boundary.
- Let u_1, u_2 be independent uniform random variables on the interval $[0, 1]$. Let $0 \leq r_{\min} \leq r_{\max} \leq 1$. Compute $\nu = -\frac{1}{2} \log \left(u_1 \left(r_{\max}^2 - r_{\min}^2 \right) + r_{\min}^2 \right)$ and $\alpha = 2\pi u_2$. Then $\exp(-\nu + i\alpha)$ is uniformly distributed on the ring in $\mathbb{C}$ between circles of radii $r_{\min}$ and $r_{\max}$.



Recap SSM (LRU)



- LTI system in the SSL layer:

$$\text{From } \begin{cases} x_{k+1} &= Ah_k + Bd_k \\ z_k &= Ch_k + Dd_k, \end{cases} \quad \text{to} \quad \begin{cases} \tilde{x}_{k+1} &= \bar{\Lambda}\tilde{x}_k + \bar{B}d_k \\ \tilde{z}_k &= \bar{C}\tilde{x}_k \\ z_k &= 2\mathcal{R}e(\tilde{z}_k) + Dd_k \end{cases} \quad \text{with } \bar{\Lambda} = \begin{bmatrix} e^{-e^{\mu_1}} e^{\theta_1 i} & & \\ & \ddots & \\ & & e^{-e^{\mu_{n_\lambda}}} e^{\theta_{n_\lambda} i} \end{bmatrix}$$

where $\bar{\Lambda} \in \mathbb{C}^{n_\lambda \times n_\lambda}, \bar{B} \in \mathbb{C}^{n_\lambda \times n_u}, \bar{C} \in \mathbb{C}^{n_y \times n_\lambda}$, while $D \in \mathbb{R}^{n_\eta \times n_u}$, with $n_\lambda = \frac{n_x}{2}$

- Training parameters: $\Theta = \left\{ \left\{ \bar{\Lambda}_j, \bar{B}_j, \bar{C}_j, D_j, \xi_i \right\}_{j \in \{1,\dots,r\}}, E, H \right\}$
- Hyperparameters: $n, n_x, r, \text{hyper}(\sigma)$

Computational efficiency

$$\begin{cases} x_{k+1} &= Ax_k + Bd_k, x_0 = 0 \\ z_k &= Cx_k + Dd_k, \end{cases} \qquad x_{k+1} = \sum_{j=0}^{k} A^j B d_{k-j} = (\mathcal{K} \star d)_k, \quad k = 0, \dots, T-1$$

where $\mathcal{K} = \begin{bmatrix} B, AB, \dots, A^{T-2}B, A^{T-1}B \end{bmatrix}$ is called kernel and the convolution operator for causal sequences f, g of length T is defined as

$$(f \star g)_k = \sum_{j=0}^{T-1} f_j g_{k-j} = \sum_{j=0}^{k} f_j g_{k-j}, \quad (g_i = 0 \text{ for } i < 0)$$

Convolutions can be parallelized:

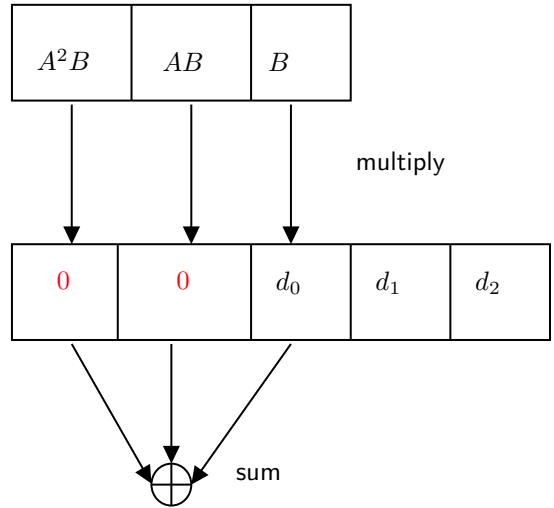
- $x_1 = Bd_0$
- $x_2 = ABd_0 + Bd_1$ (does not depend on x_1)
- $x_3 = A^2Bd_0 + ABd_1 + Bd_2$ (does not depend on x_1, x_2)
- $\dots$

Given the input sequence $d_0, d_1, \dots, d_{T-1}$, the whole trajectory $x_1, x_2 \dots, x_T$ can be computed in one shot once the kernel $\mathcal{K}$ is materialized.

Computational efficiency

$$(\mathcal{K} \star d)_0 = \sum_{j=0}^{2} \mathcal{K}_j d_{-j} = \mathcal{K}_0 d_0 + \mathcal{K}_1 d_{-1} + \mathcal{K}_2 d_{-2}$$

$k = 0$

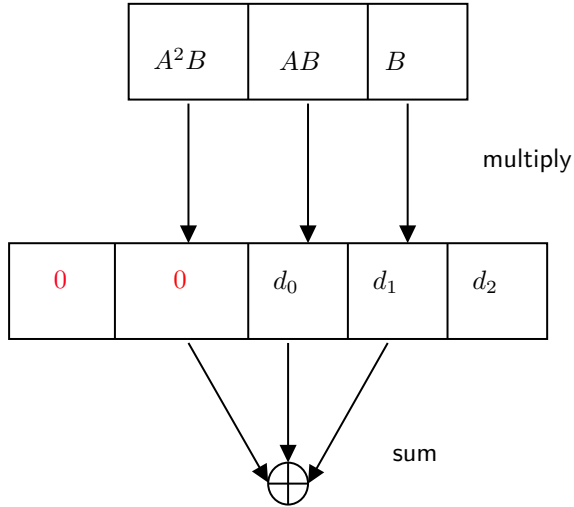


$$x_1 = Bd_0$$

Computational efficiency

$$(\mathcal{K} \star d)_1 = \sum_{j=0}^{2} \mathcal{K}_j d_{1-j} = \mathcal{K}_0 d_1 + \mathcal{K}_1 d_0 + \mathcal{K}_2 d_{-1}$$

$k = 1$

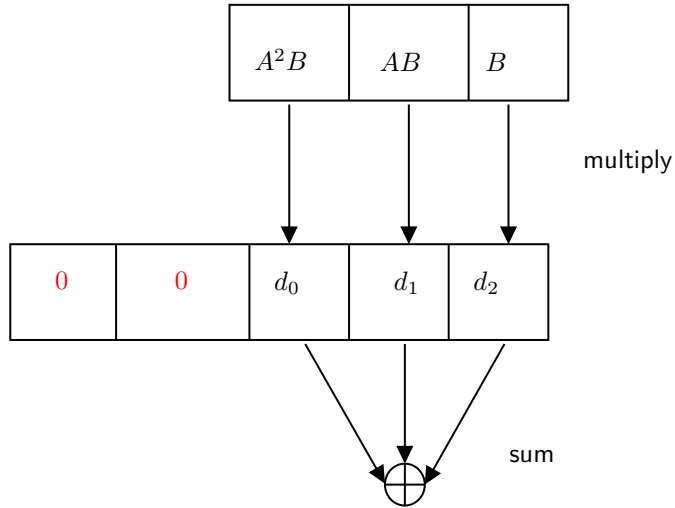


$$x_2 = \; ABd_0 \; + Bd_1$$

Computational efficiency

$$(\mathcal{K} \star d)_2 = \sum_{j=0}^{2} \mathcal{K}_j d_{2-j} = \mathcal{K}_0 d_2 + \mathcal{K}_1 d_1 + \mathcal{K}_2 d_0$$

$k = 2$



$$x_3 = A^2Bd_0 + ABd_1 + Bd_2$$

Computational efficiency

Convolutional form and parallel scan

- The elements of $\mathcal{K} = \left[B, AB, \ldots, A^{T-2}B, A^{T-1}B\right]$ can be computed in parallel.
- This can be done via parallel scan: given a binary associative operator $\bullet$ (i.e. $(a \bullet b) \bullet c = a \bullet (b \bullet c)$) and a sequence $[a_1, a_2, \ldots, a_L]$ a scan operation (sometimes referred to as all-prefix-sum) returns the sequence

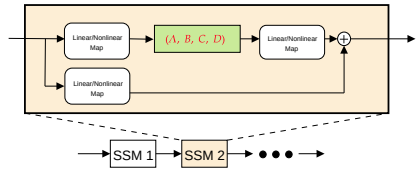
$$[a_1, (a_1 \bullet a_2), \ldots, (a_1 \bullet a_2 \bullet \ldots \bullet a_L)]$$

- The complexity of simulating a DT LTI system for a sequence of inputs of length T is $\mathcal{O}(T \log_2 T)$ times the complexity of the product $\Lambda \cdot \Lambda$. (for Λ diagonal it scales as $\mathcal{O}(n_\lambda)$, for non-diagonal matrices its complexity is $\mathcal{O}\left(n_\lambda^{2.3737}\right)$).

Scaffoldings

Is linear enough? Scaffolding to the rescue

- Expressiveness of linear systems is enhanced by pre- and post-processing the input/output via static linear and non-linear transformations. This process is called **scaffolding**.



- All maps are static and applied element-wise. Non-linearities should be Lipschitz continuous to preserve stability properties.

Scaffoldings

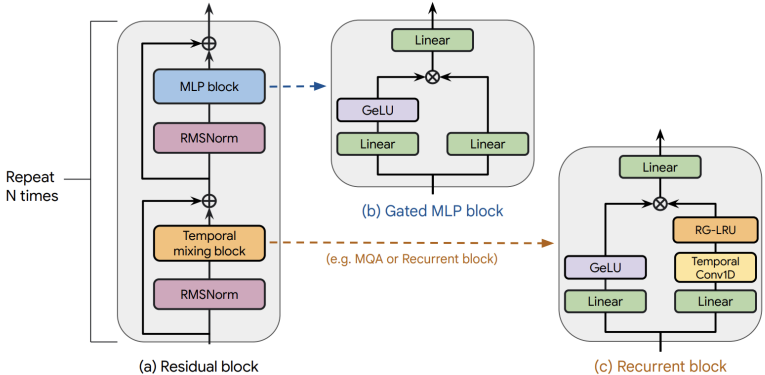


Figure: Griffin Scaffolding.

SSMs architecture

A taxonomy of SSMs

Model	Features				
	Parametrization	Discretization	Structure	Implementation	Scaffolding
S4 [12]	LTI	Bilinear	SISO	Convolution and Recurrence	MLP / H3
S4D [18]	LTI	Exact	SISO	Convolution and Recurrence	MLP / H3
S5 [19]	LTI	Exact / Bilinear	MIMO	Parallel Scan	MLP / H3
LRU [20]	LTI	None	MIMO	Parallel Scan	MLP / H3
S6 [10]	LTV	Exact	MIMO	Custom Parallel Scan	Mamba
RG-LRU [21]	LTV	None	MIMO	Custom Parallel Scan	Mamba / Hawk / Griffin

Performance

Model	LRA Task [%]						
	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	avg.
Random	10.00	50.00	50.00	10.00	50.00	50.00	36.67
Transformer [2] (paper results)	36.37	64.27	57.46	42.44	71.40	FAIL	53.66
S4 [12] (paper results)	59.60	86.82	90.90	88.65	94.20	96.35	86.09
S4D [18] (paper results)	60.52	87.34	91.09	88.19	93.96	92.80	85.65
S5 [19] (paper results)	**62.15**	89.31	**91.40**	88.00	**95.33**	**98.58**	**87.46**
LRU [20] (paper results)	60.20	**89.40**	89.90	**89.00**	95.10	94.20	86.30
S6 [10]	38.02	82.98	72.14	69.82	69.26	67.32	66.59
RG-LRU [21]	32.34	71.75	66.58	61.15	73.38	69.53	62.45

Figure: SSMs performance.

SSMs with prescribed $\mathcal{L}_2$-bound γ

- What if we want to impose something "more" than stability and we want a prescribed $\mathcal{L}_2$-bound?

L2RU: SSMs with prescribed $\mathcal{L}_2$-bound γ [1]

1. We can parametrize DT LTI systems with prescribed $\mathcal{L}_2$-bound γ using the Real Bounded Lemma:

$$g(A,B,C,D) \text{ has } \mathcal{L}_2\text{-gain } \gamma \iff P \succ 0, \quad \begin{bmatrix} A^\top PA - P + C^\top C & A^\top PB + C^\top D \\ B^\top PA + D^\top C & B^\top PB + D^\top D - \gamma^2 I \end{bmatrix} \prec 0$$

2. Then, we can use the knowledge of the Lipschitz constant of each nonlinearity and encoder/decoder matrices to find a free parametrization such that the SSM operator $\mathcal{M}$ has prescribed bound $\hat{\gamma}$

$$\text{L2RU parametrization: } \psi : \theta \in \mathbb{R}^m \mapsto \{\{\underbrace{A_i, B_i, C_i, D_i}_{g_i}, \underbrace{\xi_i}_{\sigma_i}\}_{1 \le i \le r}, E, H\} \text{ s.t. } \mathcal{M}_{\psi(\theta)} \text{ has } \mathcal{L}_2\text{-bound } \hat{\gamma}.$$

- Find out more at `https://arxiv.org/abs/2503.23818`

[1] *Free Parametrization of $\mathcal{L}_2$-bounded State Space Models*, Leonardo Massai, Giancarlo Ferrari-Trecate, 2025