SSM Review

Alexander Shekhovtsov*

1 Sequence Modeling Backbones

Deep causal backbones (Fig. 1) process sequential data in a streaming, autoregressive manner by interleaving causal mixing layers (causal self-attention, RNN, or SSM) with per-token MLPs. The causal layer is the sole component that communicates across time; the MLP provides nonlinear expressivity per position.

We work primarily in discrete time. At each step t the model receives inputs $x_1, \dots, x_t$, produces an output η_t, and a probabilistic head yields

$$p(y_t \mid x_1, \dots, x_t) = p_{\text{head}}\big(y_t;\ \eta(x_1, \dots, x_t)\big). \tag{1}$$

In autoregressive language modeling, $y_t = x_{t+1}$.

Backbone structure. All modern deep sequence models follow a repeating block:

$$x_t^{(\ell+1)} = \text{MLP}^{(\ell)}\big(\text{CausalLayer}^{(\ell)}(x_1^{(\ell)}, \dots, x_t^{(\ell)})\big), \tag{2}$$

alternating (1) a **time-mixing layer** that can communicate between positions causally, and (2) a **per-token MLP** applied independently at each position. The three principal causal layer choices are:

Causal layer	**Example architectures**
Causal Attention	Transformer, GPT
Recurrent cell	Deep RNN / LSTM / GRU
Linear filter	SSM (S4, Mamba, etc.)

Motivation. Applications span a wide range of rates and context lengths—from discrete text tokens (10^3–10^5 context) to raw audio at 44,100 Hz ($\sim 10^5$ steps per second). The central question across all settings is: *how should a model represent its past?* The answer drives the design of SSMs.

*Heavily leveraging Claude Sonnet 4.6

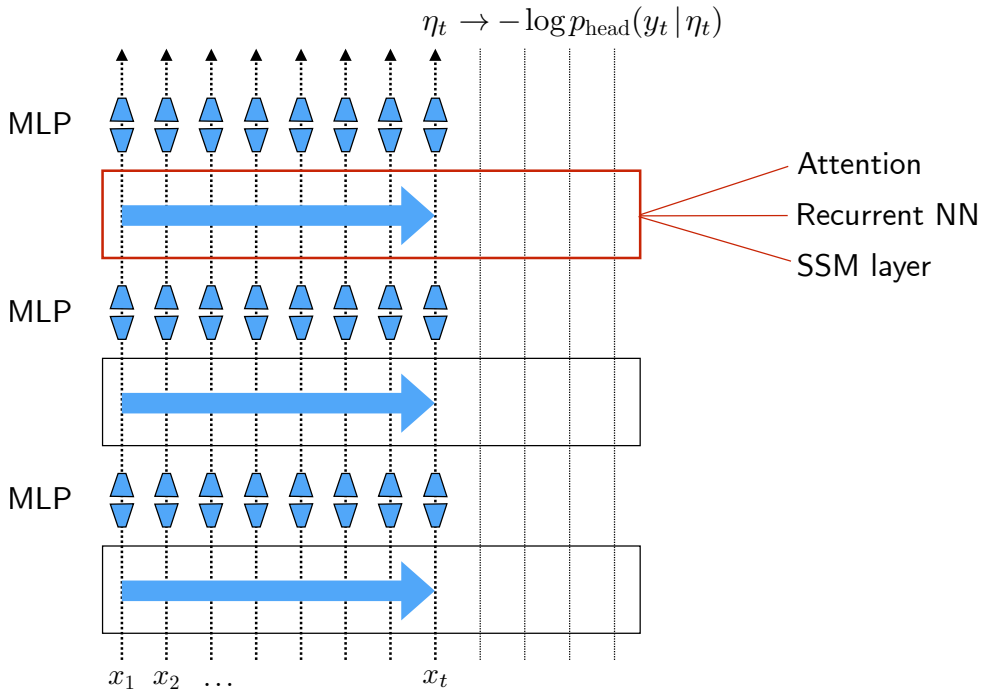

Figure 1: General architecture of a deep causal backbone. The causal layer mixes information across time; the MLP is a per-token nonlinearity.

2 Limitations of Transformers and RNNs

2.1 Transformers: quadratic scaling

Causal self-attention computes

$$y_t = \frac{\sum_{s \le t} e^{\langle q_t, k_s \rangle} v_s}{\sum_{s \le t} e^{\langle q_t, k_s \rangle}}, \qquad q_t = W^Q u_t,\ k_s = W^K u_s,\ v_s = W^V u_s.$$

Training requires $\mathcal{O}(N^2 d)$ operations and $\mathcal{O}(N^2)$ memory (or $\mathcal{O}(Nd)$ with FlashAttention (Dao et al., 2022)), with $\mathcal{O}(\log N)$ sequential depth. **Inference** requires an $\mathcal{O}(Nd)$ KV cache that grows without bound with sequence length, and $\mathcal{O}(Nd)$ compute per token.

At audio rates ($N \approx 10^5$ steps/second), $N^2 = 10^{10}$ makes training intractable.

2.2 RNNs: serial training

RNNs maintain an $\mathcal{O}(m)$ state updated via $h_t = f(h_{t-1}, u_t; \theta)$, giving $\mathcal{O}(m)$ inference memory and $\mathcal{O}(m^2)$ per-step compute—ideal for streaming. However, the recurrence is strictly serial: the sequential depth is $\mathcal{O}(N)$ for nonlinear f, making training on long sequences slow. Moreover, the N-step Jacobian product $\prod_t \nabla f$ leads to vanishing or exploding gradients, requiring careful architecture engineering (LSTM, GRU) whose stability guarantees degrade as N grows with data length.

2.3 Design requirements

Model	$\mathcal{O}(1)$ inference memory	Parallel training
Attention	No (KV cache grows)	Yes
RNN	Yes	No (serial)
SSM	**Yes**	**Yes**

The key insight: *restricting to linear recurrences enables parallel computation and explicit stability control, while retaining constant-memory recurrent inference.* This train/infer duality is the engine of all modern SSM architectures.

3 Expressivity of Linear Causal Layers

A single linear layer has limited expressivity, but a linear SSM layer embedded in a deep network with MLP blocks is powerful. The same principle underlies CNNs (linear convolutions + pointwise nonlinearities $\to$ translation-equivariant universal approximation) and Deep Sets (linear aggregation + nonlinearities $\to$ permutation-equivariant universal approximation). The SSM layer provides linear causal structure; MLP blocks provide nonlinear expressivity.

3.1 The cumulative sum

The simplest linear causal layer is the cumulative sum:

$$h_t = h_{t-1} + u_t \implies h_t = \sum_{s \le t} u_s. \tag{3}$$

This is order-blind ((a, b) and (b, a) yield the same h_2), yet with constant input $u_t = 1$ it tracks position $h_t = t$. A deep model with cumulative sum layers and nonlinearities can implement position-dependent aggregation. Section 8.2 shows that all SSMs reduce to this form via a change of variables.

4 Linear State Space Models

Definition 4.1 (Linear Time-Invariant SSM Layer)**.** Given matrices $A \in \mathbb{R}^{m\times m}$, $B \in \mathbb{R}^{m\times d}$, $C \in \mathbb{R}^{p\times m}$, the SSM maps $(u_t)_{t\ge1}$, $u_t \in \mathbb{R}^d$, to $(y_t)_{t\ge1}$, $y_t \in \mathbb{R}^p$, via

$$h_t = A\,h_{t-1} + B\,u_t, \quad h_0 = 0, \tag{4}$$
$$y_t = C\,h_t. \tag{5}$$

A governs autonomous state evolution; B injects the input; C projects the state to output. Time-invariance means the same transformation applies at every step.

4.1 Universality in deep networks

State update	Readout	Universal for fading-memory causal maps?
$h_t = A\,h_{t-1} + B\,u_t$	linear $C\,h_t$	No
$h_t = A\,h_{t-1} + B\,u_t$	polynomial $\phi(h_t)$	Yes
$h_t = A(u_t)\,h_{t-1} + B\,u_t$	linear $C\,h_t$	Yes

(Grigoryeva and Ortega, 2019)

A single linear SSM with linear readout is not universal; with a nonlinear readout (or equivalently, embedded in a deep network with MLP blocks) it is. Making A input-dependent (as in Mamba, Section 7.4) achieves universality with a linear readout.

4.2 Convolution form and the train/infer duality

Unrolling the recurrence from $h_0 = 0$:

$$y_t = \sum_{s=0}^{t-1} \underbrace{C\,A^s\,B}_{K_s}\; u_{t-s} = (K * u)_t. \tag{6}$$

The sequence $K_s = CA^sB$ is the *impulse-response kernel*. For scalar input-output, the SSM is equivalent to a causal convolution with infinite-support kernel parameterized through A.

Mode	Compute	Memory
Recurrence (inference)	$\mathcal{O}(m^2)$ per step	$\mathcal{O}(m)$
Convolution via FFT (training)	$\mathcal{O}(Nm \log N)$ for full sequence	$\mathcal{O}(Nm)$

Training computes the kernel $(K_0, \ldots, K_{N-1})$ from (A, B, C) and applies the convolution theorem $y = \mathcal{F}^{-1}(\mathcal{F}(K) \odot \mathcal{F}(u))$ in parallel with $\mathcal{O}(\log N)$ depth. Inference uses the recurrence with $\mathcal{O}(m)$ state. This is still suboptimal in computation cost and poorly extends to the general vector case $d > 1$. Furthermore, for input-dependent A_t (Section 7.4), the FFT approach is unavailable; the prefix scan of Section 8 handles the general case.

5 Eigenvalue Structure and the Prony Decomposition

Next, we give two basic examples of linear filters and later we will see that in fact all linear filters reduce to a linear combination of those.

5.1 First-order filter: exponential moving average

For $m = 1$, $r \in (-1, 1)$:

$$h_t = r\,h_{t-1} + u_t \quad \Longrightarrow \quad K_s = r^s. \tag{7}$$

The kernel is a geometric sequence (EMA). $r \to 1$: long memory (timescale $\tau \approx 1/(1-r)$); $r \to 0$: short memory; $r < 0$: alternating-sign (high-pass) filter.

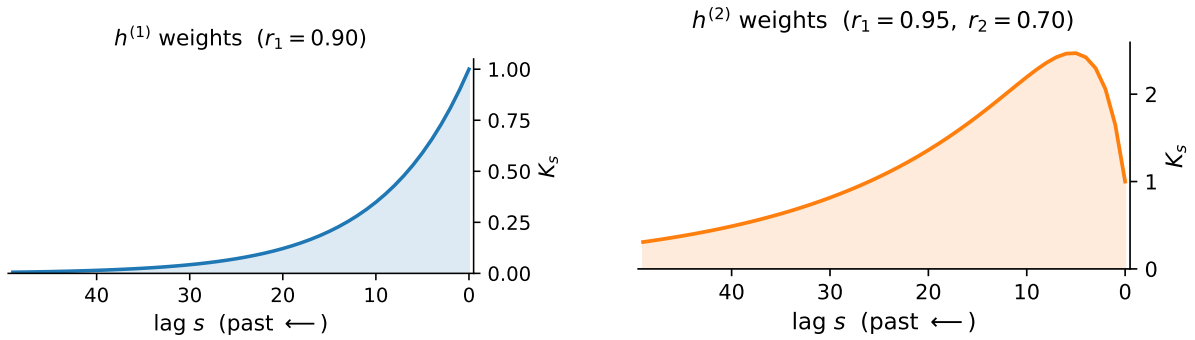

Figure 2: Impulse-response kernels of a first-order EWA (left) and a second-order EWA (right).

5.2 Second-order filter: chaining two EWAs

Applying a second EWA on top of the first:

$$h_t^{(2)} = r_2\, h_{t-1}^{(2)} + h_t^{(1)} = r_2\, h_{t-1}^{(2)} + r_1\, h_{t-1}^{(1)} + u_t.$$

As a single matrix recurrence:

$$\underbrace{\begin{pmatrix} h_t^{(1)} \\ h_t^{(2)} \end{pmatrix}}_{h_t} = \underbrace{\begin{pmatrix} r_1 & 0 \\ r_1 & r_2 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} h_{t-1}^{(1)} \\ h_{t-1}^{(2)} \end{pmatrix}}_{h_{t-1}} + \underbrace{\begin{pmatrix} 1 \\ 1 \end{pmatrix}}_{B} u_t.$$

A is lower triangular with eigenvalues r_1, r_2. After diagonalization, each eigenvector coordinate evolves as a scalar EWA; the combined kernel (for any linear readout C) is a sum of two geometric sequences, $K_s = \alpha_1 r_1^s + \alpha_2 r_2^s$—a strictly richer shape than any single EWA (Figure 2, right).

5.3 Second-order filter: harmonic oscillator

A 2×2 rotation block,

$$h_t = r \begin{pmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{pmatrix} h_{t-1} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u_t, \tag{8}$$

has kernel $K_s = r^s \cos(s\omega)$ (a damped sinusoid). At $r = 1$: sustained oscillation with no decay. The frequency ω selects which temporal frequency of the input history is tracked, making this block particularly valuable for audio.

5.4 General case: diagonalization

If $A = V \Lambda V^{-1}$ with $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_m)$, the transformed state $\tilde{h}_t = V^{-1} h_t$ evolves as m decoupled scalar filters:

$$\tilde{h}_t^{(j)} = \lambda_j\, \tilde{h}_{t-1}^{(j)} + \tilde{b}_j\, u_t, \quad j = 1, \ldots, m.$$

Since A has real entries, complex eigenvalues appear as conjugate pairs $\lambda_j = r_j e^{i\omega_j}$. To keep all arithmetic real we use the *real Jordan form*

$$A = V_{\mathbb{R}}\, J_{\mathbb{R}}\, V_{\mathbb{R}}^{-1}, \tag{9}$$

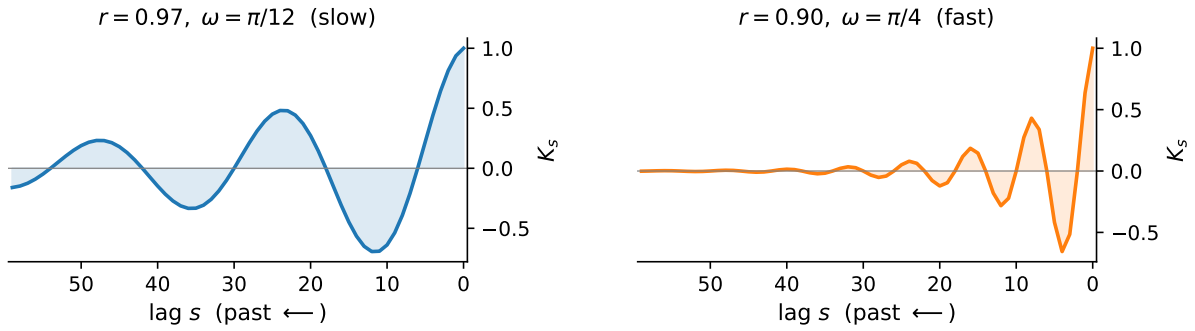

Figure 3: Harmonic oscillator kernels: damped ($r < 1$, left) and near-unit-circle ($r \approx 1$, right).

where, writing p real eigenvalues $\lambda_1, \ldots, \lambda_p \in \mathbb{R}$ and q conjugate pairs with parameters (r_j, ω_j), $j = 1, \ldots, q$ (so $p + 2q = m$):

$$J_{\mathbb{R}} = \begin{pmatrix} \lambda_1 & & & & & & \\ & \ddots & & & & & \\ & & \lambda_p & & & & \\ & & & r_1 \begin{pmatrix} \cos\omega_1 & -\sin\omega_1 \\ \sin\omega_1 & \cos\omega_1 \end{pmatrix} & & & \\ & & & & \ddots & & \\ & & & & & r_q \begin{pmatrix} \cos\omega_q & -\sin\omega_q \\ \sin\omega_q & \cos\omega_q \end{pmatrix} \end{pmatrix}.$$

Real eigenvalues give 1×1 blocks (EWA filters); conjugate pairs give 2×2 rotation-scaling blocks (harmonic oscillators). Parameterizing via $(\lambda_1, \ldots, \lambda_p, r_1, \omega_1, \ldots, r_q, \omega_q)$ uses m real scalars in place of a general $m \times m$ matrix, and all arithmetic remains real.

5.5 The Prony decomposition

In the complex diagonal basis, the kernel decomposes as

$$\boxed{K_s = \sum_{j=1}^{p} \alpha_j \, r_j^s + \sum_{j=1}^{q} 2\rho_j \, r_j^s \cos(\omega_j s + \phi_j),} \quad s = 0, 1, 2, \ldots \tag{10}$$

where $p + 2q = m$, with real amplitudes α_j, ρ_j and phases ϕ_j determined by (A, B, C). Every kernel of an m-th order real LTI system is *exactly* of this form.

Eigenvalue placement determines kernel character:

- $r \in (0, 1)$ (real positive): monotonically decaying EMA, timescale $\tau \approx 1/(1 - r)$.
- $\lambda = 1$: flat equal-weight average (requires normalization).
- $r \in (-1, 0)$ (real negative): alternating-sign (high-pass) filter.
- Complex pair, $|\lambda| < 1$: exponentially damped sinusoid.
- Complex pair, $|\lambda| = 1$: sustained sinusoid—the *only* mechanism for stable non-decaying memory.

- $|\lambda| > 1$: diverging kernel; not used.

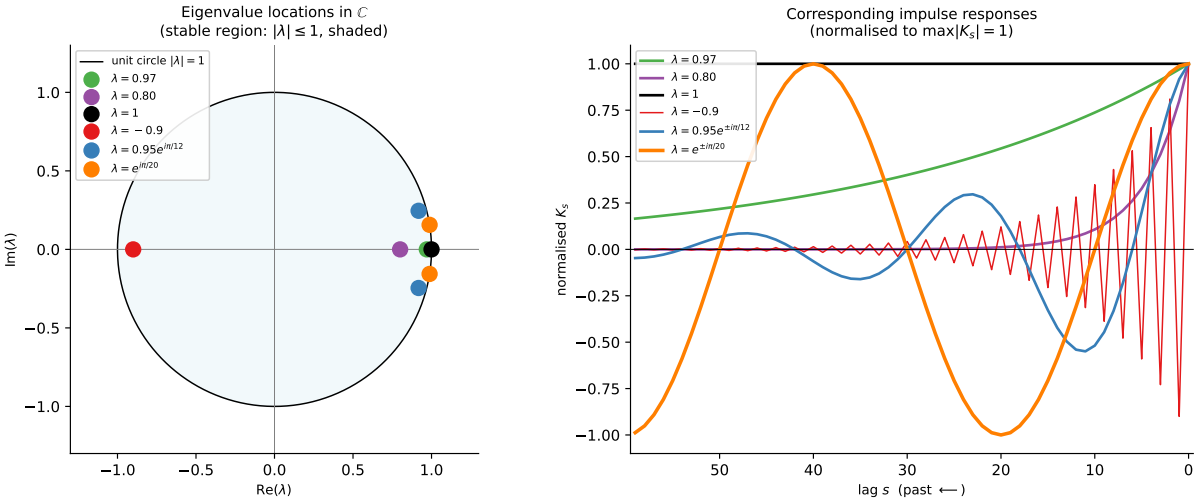

Figure 4: Eigenvalue location (left) and corresponding impulse-response kernels (right).

Example 5.1 ($m = 4$, two conjugate pairs). Two conjugate pairs $(0.98\, e^{\pm i\pi/12}, 0.80\, e^{\pm i\pi/3})$ yield a kernel with structure at two timescales simultaneously (Figure 5).

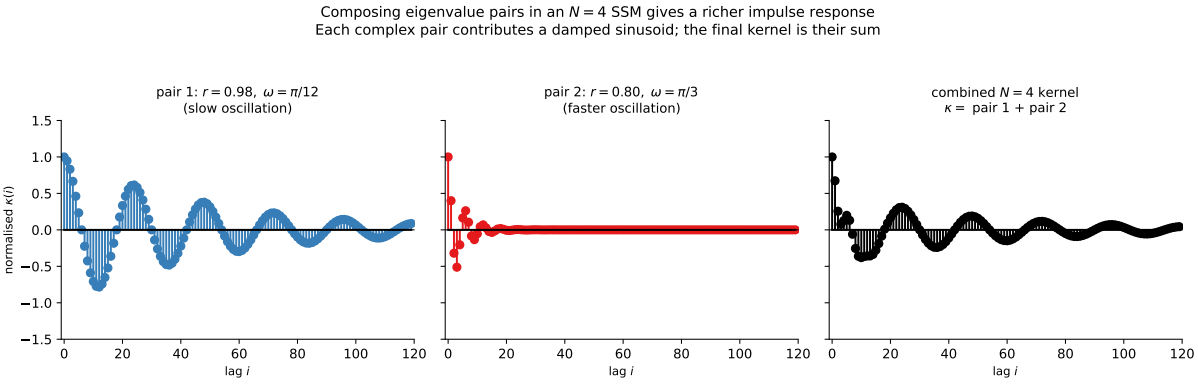

Figure 5: m=4 SSM with two conjugate pairs; individual damped sinusoids (left/center) and combined kernel (right).

Key Insight. *Eigenvalue design is kernel design: the Prony decomposition specifies exactly which kernels an m-th order LTI system can express—all finite sums of EMA and damped-sinusoidal terms with at most m summands.*

6 Memory and Eigenvalue Placement

The Prony decomposition makes the relationship between eigenvalue location and memory capacity explicit. Each term decays as r_j^s, so the kernel vanishes exponentially whenever all $|\lambda_j| < 1$, with timescale $\tau_j \approx 1/(1 - r_j)$. Designing eigenvalues close to (but inside) the unit circle is therefore the practical recipe for long-range memory while retaining stability.

Eigenvalues on the unit circle ($|\lambda_j| = 1$) are the only way to avoid this exponential decay, but the useful cases are limited. $\lambda = 1$ gives a flat running average that diverges on

sustained non-zero input and requires explicit normalization. $\lambda = -1$ gives an alternating-sign kernel with infinite memory but oscillating weights—not useful for smooth recall of distant history. The only genuinely useful unit-circle mechanism is a conjugate pair $\lambda = e^{\pm i\omega}$, which retains the frequency-ω projection of the input history as a sustained sinusoid. In practice, even this is rarely used directly: most architectures keep $r_j < 1$ and simply push eigenvalues close to the unit circle when the task requires long memory.

Remark 6.1 (Fading memory and vanishing gradients). The gradient $\partial\mathcal{L}/\partial h_0 \propto A^\top$ vanishes when $\rho(A) < 1$ for exactly the same spectral reason the kernel vanishes—both are controlled by powers of A. The SSM framework resolves the associated training difficulty by making A a learnable parameter with eigenvalues parameterized so they remain inside the unit circle by construction (e.g. $r_j = \sigma(-\nu_j)$ with unconstrained ν_j), while allowing the model to push $r_j \to 1$ when the task demands long memory.

7 Continuous Time, Discretization, and Variable Steps

7.1 Continuous-time SSM and ZOH discretization

The natural starting point is the continuous-time linear ODE

$$\dot{h}(\tau) = \lambda h(\tau) + b\,u(\tau), \quad \lambda < 0. \tag{11}$$

Under the *zero-order hold* (ZOH) assumption—u held constant for a step of duration Δ—the exact solution is

$$h(\tau + \Delta) = e^{\lambda\Delta}\,h(\tau) + \frac{e^{\lambda\Delta} - 1}{\lambda}\,b\,u. \tag{12}$$

This is precisely the discrete recurrence $h_t = \bar{A}(\Delta)\,h_{t-1} + \bar{B}(\Delta)\,u_t$ with

$$\bar{A}(\Delta) = A^\Delta, \qquad \bar{B}(\Delta) = (I - A)^{-1}(I - A^\Delta)\,B, \tag{13}$$

where $r = e^{\lambda\Delta_0}$ reparameterizes the decay rate ($r^\Delta = e^{\lambda\Delta}$). The Prony-basis scalar formula is the exact discretization; no matrix exponentiation is needed. Many SSM papers (S4, Liquid S4) start from ODEs and discretize; the discrete Prony formulas are equivalent.

7.2 Variable step sizes

For inputs arriving at irregular intervals Δ_k, replace (A, B) with $(\bar{A}(\Delta_k), \bar{B}(\Delta_k))$ at each step. In the Prony basis this is cheap: EWA blocks require r^{Δ_k} and one division; rotation blocks reduce to scaling $r \to r^{\Delta_k}$ and angle addition $\omega \to \Delta_k\omega$, with a closed-form 2×2 inverse. The full m-dimensional update costs $\mathcal{O}(m)$ scalar operations.

This mechanism handles diverse modalities without architectural change: text ($\Delta = 1$), uniformly downsampled audio ($\Delta = D$), or irregular sensor logs (variable Δ_k).

7.3 Reduction to $A{=}I$ for variable step sizes

Define cumulative time $T_t = \sum_{s=1}^{t} \Delta_s$ (a prefix sum). For variable-step EWA $h_t = r^{\Delta_t} h_{t-1} + \bar{b}_t u_t$, setting $z_t = r^{-T_t} h_t$ gives

$$z_t = z_{t-1} + r^{-T_t}\,\bar{b}_t\,u_t, \qquad y_t = r^{T_t} c\,z_t.$$

The $A{=}I$ reduction (Section 8.2) applies uniformly across all variants: fixed integer steps ($T_t = t$), fixed real steps ($T_t = t\Delta$), variable discrete steps ($T_t = \sum \Delta_s$), and input-dependent steps (Section 7.4).

Key Insight. *In all cases: first compute the prefix sum of step sizes, then compute the prefix sum of position-encoded inputs. Two prefix sums suffice.*

7.4 Mamba: input-dependent parameters

Mamba (Gu and Dao, 2023) uses a linear SSM recurrence with Δ, B, C all input-dependent:

$$\Delta_t = \text{softplus}(b_\Delta + W_\Delta u_t) \in \mathbb{R}^m, \quad A_t = \exp(\Delta_t \odot \lambda), \quad B_t = W_B u_t, \quad C_t = W_C u_t, \tag{14}$$

where $\lambda \in \mathbb{R}^m_{<0}$ is a fixed learned parameter. Δ_t controls the per-channel forgetting rate; B_t and C_t control what is written and read. Making A_t input-dependent places Mamba in the third universality row (Section 4.1).

Since A_t is input-dependent, no fixed kernel exists. Mamba trains using the affine-map prefix scan of Section 8.1 with per-channel scalar composition.

Under the $A{=}I$ view, Mamba is a prefix sum over data-dependent positional encodings:

1. Compute $\Delta_t = \text{softplus}(b_\Delta + W_\Delta u_t)$.
2. Prefix-sum step sizes: $T_t = \sum_{s \le t} \Delta_s$.
3. Encode inputs: $\tilde{u}_t = r^{-T_t} \odot B_t u_t$.
4. Prefix-sum encoded inputs: $z_t = \sum_{s \le t} \tilde{u}_s$.
5. Decode: $y_t = r^{T_t} \odot C_t z_t$.

8 Parallel Training via Prefix Scan

For time-varying parameters (e.g., input-dependent A_t), the convolution approach is unavailable. The *parallel prefix scan* provides an $\mathcal{O}(\log N)$-depth alternative.

Definition 8.1 (Prefix scan)**.** Given $\phi_1, \ldots, \phi_N$ and an associative operator $\oplus$, compute all prefix products ϕ_1, $\phi_1 \oplus \phi_2$, … A binary reduction tree achieves this in $\lceil \log_2 N \rceil$ parallel rounds with $\mathcal{O}(N)$ total work.

8.1 SSM step as an affine map

The SSM recurrence $h_t = A_t h_{t-1} + b_t$ is an affine map. Representing each step as $\Phi_t = (A_t, b_t)$ with composition

$$\Phi_t \circ \Phi_{t-1} = (A_t A_{t-1},\ A_t b_{t-1} + b_t),$$

the prefix scan over $(\Phi_1, \ldots, \Phi_N)$ computes all N states in $\mathcal{O}(\log N)$ depth. In the Prony basis this decomposes into per-block scalar operations:

EWA block. $(d_2, b_2) \circ (d_1, b_1) = (d_2 d_1, d_2 b_1 + b_2)$. Cost: 2 multiplies, 1 add.

Rotation block. Using $R(\theta_2)R(\theta_1) = R(\theta_2 + \theta_1)$:

$$(d_2, \theta_2, b_2) \circ (d_1, \theta_1, b_1) = \big(d_2 d_1, \;\; \theta_2 + \theta_1, \;\; d_2 R(\theta_2)\, b_1 + b_2\big). \tag{15}$$

The A-part costs 1 multiply + 1 add; the b-part is a complex multiply (one hardware FMA).

Total: all N states in $\mathcal{O}(\log N)$ depth, $\mathcal{O}(Nm)$ total work, with 3 flops/EWA channel and ≈ 8 flops/harmonic block—versus $\mathcal{O}(m^2)$ for a dense recurrence.

8.2 Reduction to $A{=}I$: connection to RoPE

Any SSM with Prony-basis blocks reduces to a plain prefix sum via a change of variables.

EWA channel. For $h_t = r h_{t-1} + b u_t$, $y_t = c h_t$, define $z_t = r^{-t} h_t$:

$$z_t = z_{t-1} + r^{-t} b\, u_t, \qquad y_t = c\, r^t\, z_t.$$

A prefix sum over position-encoded inputs $r^{-t} b\, u_t$ with position-encoded readout $c r^t$.

Rotation block. For $h_t = r R(\omega) h_{t-1} + b u_t$, define $z_t = r^{-t} R(-t\omega) h_t$:

$$z_t = z_{t-1} + r^{-t} R(-t\omega)\, b\, u_t, \qquad y_t = r^t\, z_t^{\mathsf{T}} R(-t\omega)\, c. \tag{16}$$

A prefix sum with *rotational* position encoding. The input and output use the same rotation $-t\omega$—this is precisely **RoPE** (Su et al., 2021).

Key Insight. *RoPE is not an independent design choice: it is the natural encoding that emerges when reducing an oscillatory SSM to a prefix sum. Any model using RoPE-like encodings is implicitly an SSM with oscillatory eigenvalue structure.*

8.3 Cost summary

Model	Training compute	Training depth	Infer/step	Infer memory
Transformer	$\mathcal{O}(N^2 d)$	$\mathcal{O}(\log N)$	$\mathcal{O}(Nd)$	$\mathcal{O}(Nd)$
Deep RNN	$\mathcal{O}(Nm^2)$	$\mathcal{O}(N)$	$\mathcal{O}(m^2)$	$\mathcal{O}(m)$
SSM (scan)	$\mathcal{O}(Nm)$	$\mathcal{O}(\log N)$	$\mathcal{O}(m^2)$	$\mathcal{O}(m)$

9 Unifying Linear Attention, RoPE, and SSMs

Linear attention, the Performer, RetNet, and SSMs are instances of a single recurrence

$$H_t = A\, H_{t-1} + k_t\, v_t^{\mathsf{T}}, \qquad y_t = q_t^{\mathsf{T}} H_t, \tag{17}$$

but they arrive at it from opposite directions. **Linear attention and the Performer** start from softmax attention and relax it toward a linear-in-N recurrence, preserving a normalization denominator along the way. **RetNet** starts from the SSM side and makes the state matrices input-dependent so as to mimic the key–query–value structure of attention, obtaining a masked linear attention without a normalization term. Both paths converge on essentially the same recurrence; the practical difference is whether normalization is carried along.

9.1 Linear attention as $A = I$

Replacing the softmax in attention with a kernel feature map ϕ (Katharopoulos et al., 2020) gives

$$y_t \approx \frac{\phi(q_t)^\top \sum_{s \le t} \phi(k_s)\, v_s^\top}{\phi(q_t)^\top \sum_{s \le t} \phi(k_s)},$$

where the matrix state accumulates as

$$H_t = H_{t-1} + \phi(k_t)\, v_t^\top, \qquad y_t = \phi(q_t)^\top H_t. \tag{18}$$

This is an SSM with $A = I$ (no forgetting), matrix-valued state $H_t \in \mathbb{R}^{d \times d}$, and input-dependent projection maps playing the roles of keys, queries, and values—but the normalization denominator is kept because the model is designed to approximate the original probabilistic softmax attention.

9.2 Performer

The Performer (Choromanski et al., 2021) constructs ϕ to be an unbiased estimator of $\exp(q^\top k)$ using random Fourier features. With $\omega_1, \dots, \omega_r \sim \mathcal{N}(0, I)$:

$$\phi(x) = \frac{e^{-\|x\|^2/2}}{\sqrt{r}} \left(e^{\omega_1^\top x}, \dots, e^{\omega_r^\top x}\right)^\top, \qquad \phi(q)^\top \phi(k) \xrightarrow{r \to \infty} e^{q^\top k}.$$

Same recurrent structure as (18) and the same normalization denominator, now with a provable approximation guarantee to softmax attention.

9.3 RetNet: from SSM toward attention

The Retentive Network (Sun et al., 2023) takes the opposite starting point: it begins with the SSM

$$h_t = A\, h_{t-1} + B\, u_t, \qquad y_t = C\, h_t,$$

and asks how to make it as expressive as a transformer. The bottleneck in a standard (time-invariant) SSM is that B and C are fixed matrices: every input u_t is projected the same way regardless of content. Transformers, by contrast, compute content-dependent keys $k_t = W_K u_t$ and queries $q_t = W_Q u_t$.

Making B and C input-dependent. RetNet replaces the fixed matrices with input-dependent projections combined with a positional rotation (which arises naturally from the SSM eigenvalue structure, as shown in Section 8.2):

$$k_t = R(t\omega)\, W_K u_t \in \mathbb{R}^m, \qquad q_t = R(t\omega)\, W_Q u_t \in \mathbb{R}^m, \qquad v_t = W_V u_t \in \mathbb{R}^d.$$

Setting $B_t = k_t$, $C_t = q_t^\top$, and $A = \gamma I$ (scalar exponential forgetting with $\gamma \in (0, 1)$), the state becomes matrix-valued $H_t \in \mathbb{R}^{m \times d}$ and the recurrence reads

$$H_t = \gamma\, H_{t-1} + k_t\, v_t^\top, \qquad y_t = H_t^\top q_t. \tag{19}$$

Unrolling to a masked linear attention. Unrolling (19) from $H_0 = 0$ gives

$$y_t = \sum_{s=1}^{t} \gamma^{t-s} \left(q_t^\mathsf{T} k_s\right) v_s.$$

This is a score-weighted sum over past values, just like attention, but with two key differences: the score is a plain dot product $q_t^\mathsf{T} k_s$ (no softmax), and instead of a learned or normalized weighting the contribution of position s is attenuated by γ^{t-s}—an exponential forgetting mask.

In matrix form, with $Q, K \in \mathbb{R}^{m\times N}$ (columns are q_t, k_t) and $V \in \mathbb{R}^{N\times d}$:

$$Y = \left(Q^\mathsf{T} K \odot D\right) V, \qquad D_{ts} = \begin{cases} \gamma^{t-s} & s \le t \\ 0 & s > t, \end{cases} \tag{20}$$

where D is a causal exponential-decay mask. There is no normalization row in (20)—the decay factor γ^{t-s} provides implicit soft forgetting without requiring a partition-function denominator.

Complexity.

- **Parallel (training):** form $Q^\mathsf{T} K \in \mathbb{R}^{N\times N}$, apply mask D, multiply by V; cost $O(N^2(m+d))$. (FlashAttention cannot be directly applied because there is no softmax, but the quadratic form allows highly parallel matrix-multiply implementations.)
- **Recurrent (inference):** maintain $H_t \in \mathbb{R}^{m\times d}$; update $H_t = \gamma H_{t-1} + k_t v_t^\mathsf{T}$ and read $y_t = H_t^\mathsf{T} q_t$; cost $O(md)$ memory and $O(md)$ arithmetic per token, constant in N.

Key Insight. *RetNet derives the key–query–value decomposition from the SSM side: making B and C input-dependent yields transformers' expressiveness, while the SSM structure replaces the softmax with an exponential decay mask and eliminates the need for explicit normalization.*

9.4 The unifying family

All three models are instances of the single recurrence (17), differing only in the choice of A and whether a normalization term is carried:

$$\boxed{H_t = A\, H_{t-1} + k_t\, v_t^\mathsf{T}, \qquad y_t = q_t^\mathsf{T}\, H_t.} \tag{21}$$

A	Model	Normalization
$A = I$, $k_t = \phi(W_K u_t)$	Linear attention	yes
$A = I$, $k_t = \phi(W_K u_t)$	Performer	yes
$A = \gamma I$, $k_t = R(t\omega) W_K u_t$	RetNet	no
$A = R(\theta)$	Linear attention + RoPE	yes
$A = rR(\theta)$, $r < 1$	Damped oscillatory SSM	—

When $A = R(\theta)$, the score between query t and key s is $q_t^\mathsf{T} R((t-s)\theta)k_s$—depending only on the lag $t-s$, which is the relative-position structure of RoPE. The oscillatory eigenvalue structure and RoPE are two faces of the same 2×2 rotation block in A.

The normalization column reflects the different starting points: linear attention and the Performer are designed to approximate softmax attention (a normalized distribution), so they preserve the denominator. RetNet is designed to augment an SSM, where no such normalization is natural; the exponential decay mask γ^{t-s} provides implicit forgetting without requiring a partition function.

Key Insight. *Linear attention, RoPE, RetNet, and SSMs form a single family parameterized by $H_t = AH_{t-1} + k_t v_t^\mathsf{T}$. The choice of A determines memory structure; k_t and q_t determine what is written and read. The two research directions—linearizing attention vs. enriching SSMs—converge on the same recurrence from opposite ends.*

10 Fast Weights and Associative Memory

Note: Further connections to check.

Substituting $k_t = R(t\omega)W_K u_t$ and $v_t = W_V u_t$ directly into the RetNet state recurrence (19) gives

$$H_t = \sum_{s=1}^{t} \gamma^{t-s}\, R(s\omega)\, W_K\, (u_s u_s^\mathsf{T})\, W_V^\mathsf{T},$$

exposing the state as an **exponentially-weighted moving average of rotated, projected outer products** of the input with itself. This connects RetNet—and, more broadly, the entire linear SSM family—to several older research threads.

Associative (Hopfield) memory. The Hopfield network (Hopfield, 1982) stores memories as outer-product weight updates $W \mathrel{+}= xx^\mathsf{T}$ and recalls them via $y = Wx$. The RetNet state does exactly this on a sequence: each step writes a new (key, value) association $k_t v_t^\mathsf{T}$ into H_t, weighted by how recently it occurred. Readout $y_t = H_t^\mathsf{T} q_t$ is then content-addressed retrieval from the accumulated associative store.

Fast weights. Schmidhuber (1992) proposed *fast weights*: a slow network learns to write a rapidly-changing weight matrix, and a fast network reads from it. The outer-product write $H_t \mathrel{+}= k_t v_t^\mathsf{T}$ is precisely a fast-weight update, with the query-based readout $y_t = H_t^\mathsf{T} q_t$ as the fast-network pass. In Ba et al. (2016) this mechanism was studied in the context of modern RNNs.

Fast weights as linear transformers. Schlag et al. (2021) proved that linear (non-softmax) attention and fast-weight memories are equivalent: both maintain a matrix state via rank-1 outer-product updates and retrieve via a linear projection of the query through that state. RetNet is a fast-weight memory with *exponential forgetting* ($\gamma < 1$) and *rotary position encoding* baked into the keys and queries.

Rank and capacity. Because each update $k_t v_t^\mathsf{T}$ is rank-1, the state H_t has rank at most $\min(t, m, d)$. The SSM state dimension m is therefore a hard capacity ceiling: at most

m independent (key, value) associations can be stored simultaneously. This is the same capacity bound as a Hopfield network with m-dimensional patterns.

Second-order statistics and RoPE. The score $(q_t^\intercal R((t-s)\omega)\, k_s)$ is a bilinear form in the current and past input. The rotation $R((t-s)\omega)$ introduces a lag-dependent phase, allowing the query to selectively attend to inputs at specific lags—which is RoPE. The full retrieval $y_t = \sum_s \gamma^{t-s}(q_t^\intercal k_s)v_s$ computes lag-dependent second-order cross-statistics between the current query and the stored history. This connects to frequency-domain covariance estimation and adaptive filtering in signal processing.

11 Attention Details

11.1 Flash Attention

Let $k_s, q_t, v_s \in \mathbb{R}^d$ be the key, query, and value vectors, $s, t \in [1, N]$. The attention output for position t is

$$o_t = \frac{\sum_{s<t} \exp(q_t^\top k_s) v_s}{\sum_{s<t} \exp(q_t^\top k_s)}. \tag{22}$$

Both sums can be accumulated online over s:

$$h_t = \sum_{s<t} \exp(q_t^\top k_s) v_s, \tag{23}$$

$$Z_t = \sum_{s<t} \exp(q_t^\top k_s), \tag{24}$$

giving $o_t = h_t / Z_t$.

References

Jimmy Ba, Geoffrey E. Hinton, Volodymyr Mnih, Joel Z. Leike, and Audrinas Plaquet. Using fast weights to attend to the recent past. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations (ICLR)*, 2021. arXiv:2009.14794.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2205.14135.

Lyudmila Grigoryeva and Juan-Pablo Ortega. Universal discrete-time reservoir computers with stochastic inputs and linear readouts using non-homogeneous state-affine systems. *Journal of Machine Learning Research*, 20(1):2604–2637, 2019. arXiv:1712.00754.

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

John J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020. arXiv:2006.16236.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021. arXiv:2102.11174.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jian Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.