CARACAL: NONCOMMUTING ROTATIONS IN STATE-SPACE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

State-space models (SSMs) combine parallel training with fixed-memory decoding, but diagonal transitions limit their state-tracking capacity: these models cannot solve NC^1-complete problems.[1] Non-diagonal transitions expand this capacity in models such as block-diagonal SLiCE and DeltaProduct, but dense block composition and repeated per-token updates increase training cost. The central design problem is to combine non-diagonal state transitions with the efficient parallel training of state-space duality (SSD). We resolve this tension with CARACAL, a non-diagonal SSM whose noncommuting $\mathrm{SO}(3)$ rotations compose through quaternions. This structure yields SO3SSD, combining hard state tracking with SSD's efficiency. CARACAL provably exceeds the state-tracking capacity of diagonal SSMs and solves the A_5 benchmark with a single layer. On finite-state automata, it achieves the highest mean accuracy among evaluated parallel models. We also obtain the lowest measured test perplexities in our language-model comparison, competitive zero-shot accuracy, and leading MAD compression and selective-copying results. Including control preparation, SO3SSD trains faster than every tested baseline at 512K tokens and batch one, outperforming the fastest by $1.44\times$ even when recurrent baselines use smaller states.

1 INTRODUCTION

Structured state-space models (SSMs) and linear recurrent attention combine parallel training with autoregressive generation whose memory does not grow with sequence length. Input-dependent gating has made these models competitive for language modeling by learning how to retain information and update stored associations (Gu & Dao, 2024; Dao & Gu, 2024; Lahoti et al., 2026). In state-tracking tasks, each token specifies an operation on the result of previous operations, as in following changes to entities in a narrative or executing program instructions (Merrill et al., 2024). Supporting these computations efficiently depends on the structure of the state transition matrix.

Diagonal transitions commute, including Mamba-3's fixed-plane rotations (Lahoti et al., 2026). At fixed depth and logarithmic precision, diagonal input-dependent SSMs admit TC^0 circuit simulations, whereas group state tracking includes NC^1-complete problems (Merrill et al., 2024; Barrington, 1989). Non-diagonal transitions expand this computational reach: DeltaProduct and RWKV-7 combine stronger state tracking with language-modeling results (Siems et al., 2025; Peng et al., 2025). DeltaProduct's training cost grows with the number of Householder factors per token (Siems et al., 2025). PD-SSM uses discrete routing trained with surrogate gradients (Terzic et al., 2025). Block-diagonal SLiCE composes dense blocks, with cost increasing with block size (Walker et al., 2025). Mamba-2's state-space duality (SSD) exploits scalar transitions for efficient parallel training through matrix multiplications (Dao & Gu, 2024). Hard state tracking calls for richer transitions, yet SSD derives its efficiency from scalar transitions. Motivated by this tension, we ask:

Can a general-purpose token mixer cross the TC^0 ceiling of diagonal SSMs to solve hard state-tracking problems, yet retain SSD's training efficiency and competitive performance in language modeling and associative recall?

[1] At fixed depth and logarithmic precision, assuming $\mathrm{TC}^0 \neq \mathrm{NC}^1$ (Merrill et al., 2024).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We answer this question affirmatively with CARACAL,[2] a non-diagonal SSM that crosses the diagonal expressivity barrier with SSD's training efficiency. CARACAL solves the tested A_5 benchmark with one layer, achieves the lowest measured language-model perplexity in our comparison, and delivers strong associative recall. SO3SSD leads long-sequence training at batch one even with control preparation included and smaller states for the baselines. Our contributions are:

- **Noncommuting $\mathrm{SO}(3)$ transitions with SSD's efficiency.** We prove that a single head with noncommuting $\mathrm{SO}(3)$ transitions tracks A_5 using nine state coordinates, positive damping, a linear readout, and logarithmic precision (Theorem 4).
- **Exact integration at arbitrary rotation angles.** We derive closed-form first-order-hold updates for damped rotations (Hochbruck & Ostermann, 2010). They exactly integrate the state input at arbitrary rotation angles when the dynamics are constant over each interval (Proposition 2).
- **SO3SSD: leading long-sequence training throughput.** We reduce the full first-order-hold recurrence to scalar SSD contractions through quaternion prefixes (Proposition 3). At 512K tokens and batch one, SO3SSD is $1.44\times$ faster than the fastest baseline with a smaller state, including control preparation (Section 5.4).
- **Single-layer A_5 learning, strong recall, and lowest measured perplexity.** A single layer exceeds the A_5 benchmark's 90% accuracy threshold at every tested length (3–20), with median selected depth one over five seeds at each length (Section 5.1). On MAD (Poli et al., 2024), CARACAL-Δ leads on compression and selective copying, with gains persisting at similar parameter counts to GDN2 (Section 5.2). Our 45M and 177M language models achieve the lowest measured test perplexity in our comparison and competitive zero-shot accuracy (Section 5.3).

2 RELATED WORK

We briefly situate our work within the literature on state-space models and linear attention. Appendix A provides an extended discussion.

Rotations and noncommuting transitions. Mamba-3 combines input-dependent planar rotations with two-endpoint discretization, retaining state-space duality (SSD) through cumulative rotation angles (Lahoti et al., 2026). CARACAL makes the axes input-dependent, requiring ordered rotation products, which we implement using quaternions. DeltaProduct composes generalized Householder transformations, with per-token computation increasing with the number of factors (Siems et al., 2025). Block-diagonal SLiCE preserves noncommuting products in independent dense blocks (Walker et al., 2025). Each CARACAL head shares one scaled 3×3 rotation across its state blocks. Adding blocks increases memory capacity while the ordered transition product still needs only one rotation and one decay scalar. This structure also gives explicit integration coefficients and a reduction to scalar SSD.

Associative memory. Gated DeltaNet combines global forgetting with the delta rule (Yang et al., 2025); Gated DeltaNet-2 separates channel-wise erasure and writing (Hatamizadeh et al., 2026). KLA derives its write gain from propagated uncertainty (Shaj et al., 2026). Our rotations and write gains depend directly on tokens. CARACAL-Δ applies prediction-error editing after rotation, using a key determined by the integration coefficients (Appendix C.4).

State tracking. Circuit bounds and group classifications expose limitations of diagonal SSMs (Merrill et al., 2024; Shakerinava et al., 2026). RWKV-7 and Gated DeltaProduct establish recognition of every regular language through multilayer constructions (Peng et al., 2025; Siems et al., 2025). DeltaProduct also tracks finite subgroups of $\mathrm{SO}(3)$ with two reflection factors (Siems et al., 2025, Theorem 4). Our analysis realizes this group-tracking capability within our layer using one head, positive damping, and a linear readout (Section 4.4).

[2]Named after the Caracal, a wild cat; we are inspired by the midair rotations of falling cats (Montgomery, 1993).

3 BACKGROUND

State-space models. A linear SSM evolves a hidden state $\mathbf{s}(t) \in \mathbb{R}^{d_s}$ under an input $u(t)$ and reads out $y(t)$:

$$\dot{\mathbf{s}}(t) = \mathbf{A}(t)\mathbf{s}(t) + \mathbf{B}(t)u(t), \qquad y(t) = \mathbf{C}(t)^{\mathsf{T}}\mathbf{s}(t) + Du(t), \tag{1}$$

where $\mathbf{A}(t) \in \mathbb{R}^{d_s \times d_s}$, $\mathbf{B}(t), \mathbf{C}(t) \in \mathbb{R}^{d_s}$, and D is a scalar skip coefficient. Input-dependent SSMs condition these coefficients and the discretization on token embeddings $\mathbf{e}_1, \ldots, \mathbf{e}_T \in \mathbb{R}^d$ (Gu & Dao, 2024). For p value channels sharing dynamics, $\mathbf{S}_n \in \mathbb{R}^{p \times d_s}$ collects their states after token n, giving the discrete recurrence

$$\mathbf{S}_{n+1} = \mathbf{S}_n\mathbf{M}_n^{\mathsf{T}} + \mathbf{V}_n, \qquad \mathbf{y}_n = \mathbf{S}_n\mathbf{C}_n + D\mathbf{u}_n, \tag{2}$$

where $\mathbf{u}_n \in \mathbb{R}^p$ contains the values; the arriving embedding $\mathbf{e}_{n+1}$ determines $\mathbf{M}_n$ and the state input $\mathbf{V}_n \in \mathbb{R}^{p \times d_s}$. Holding $\mathbf{A}_n$ constant over a step of size h_n gives $\mathbf{M}_n = e^{h_n \mathbf{A}_n}$; the input hold determines $\mathbf{V}_n$ through variation of constants (Appendix B.2). These recurrences admit standard parallel scans (Smith et al., 2023; Gu & Dao, 2024).

State-space duality. Mamba-2's SSD layer uses scalar transitions $\mathbf{M}_n = \alpha_n \mathbf{I}$, $\alpha_n \in (0, 1)$, with $\mathbf{V}_n = \mathbf{u}_{n+1}\mathbf{B}_{n+1}^{\mathsf{T}}$, absorbing the step size into $\mathbf{B}_{n+1}$ (Dao & Gu, 2024). In the dual attention form, $\mathbf{B}_n, \mathbf{C}_n$ are keys and queries, $\mathbf{u}_n$ are values, and $\mathbf{S}_n$ is the *fast-weight matrix* (Yang et al., 2024). With zero initial state, stacking the vectors as rows gives

$$\mathbf{Y} = (\mathbf{L} \odot (\mathbf{C}\mathbf{B}^{\mathsf{T}}))\mathbf{U} + D\mathbf{U}, \qquad L_{tj} = \mathbf{1}_{j \le t} \prod_{k=j}^{t-1} \alpha_k. \tag{3}$$

Here $\mathbf{Y}$ stacks the outputs, $\odot$ is elementwise multiplication, and empty products equal one. For a fixed chunk size, SSD's work grows linearly with sequence length, while most arithmetic runs on matrix multiplication units (Dao & Gu, 2024, Section 6).

Rotations and unit quaternions. Three-dimensional rotations form the group $\mathrm{SO}(3) = \{R \in \mathbb{R}^{3\times 3} : R^{\mathsf{T}}R = I,\ \det R = 1\}$. A *quaternion* $q = (q_0, \mathbf{q})$ consists of a scalar part $q_0 \in \mathbb{R}$ and a vector part $\mathbf{q} \in \mathbb{R}^3$. SO3SSD represents each rotation prefix by a unit quaternion with four real coordinates. Quaternion multiplication composes the rotations in order, while quaternion conjugation applies the resulting rotation to the keys and queries. The conjugate $\bar{q} = (q_0, -\mathbf{q})$ negates the vector part. Quaternion multiplication is the *Hamilton product*

$$a \otimes b = \big(a_0 b_0 - \mathbf{a}^{\mathsf{T}}\mathbf{b},\ a_0\mathbf{b} + b_0\mathbf{a} + \mathbf{a} \times \mathbf{b}\big). \tag{4}$$

To rotate $\mathbf{v} \in \mathbb{R}^3$, first embed it as the *pure quaternion* $(0, \mathbf{v})$. For a unit quaternion $q \in \mathbb{H}_1 = \{q : q_0^2 + \|\mathbf{q}\|_2^2 = 1\}$, the rule $(0, \mathbf{R}(q)\mathbf{v}) = q \otimes (0, \mathbf{v}) \otimes \bar{q}$ defines the rotation $\mathbf{R}(q)$.

Proposition 1 (Quaternion double-cover of $\mathrm{SO}(3)$)**.**

(i) $(\mathbb{H}_1, \otimes)$ *is a group with identity* $\mathbf{1}_{\mathbb{H}} = (1, \mathbf{0})$ *and inverse* $q^{-1} = \bar{q}$ *for every* $q \in \mathbb{H}_1$.

(ii) $\mathbf{R} : \mathbb{H}_1 \to \mathrm{SO}(3)$ *is a surjective group homomorphism with kernel* $\{\pm\mathbf{1}_{\mathbb{H}}\}$. *For all* $q, q_1, q_2 \in \mathbb{H}_1$,

$$\mathbf{R}(q_2 \otimes q_1) = \mathbf{R}(q_2)\mathbf{R}(q_1), \qquad \mathbf{R}(q) = \mathbf{R}(-q). \tag{5}$$

(iii) For every $\boldsymbol{\omega} \in \mathbb{R}^3$, *with* $\Omega = \|\boldsymbol{\omega}\|_2$ *and* $[\boldsymbol{\omega}]_\times \mathbf{v} = \boldsymbol{\omega} \times \mathbf{v}$,

$$q(\boldsymbol{\omega}) = \big(\cos(\Omega/2),\ (\sin(\Omega/2)/\Omega)\boldsymbol{\omega}\big) \in \mathbb{H}_1, \qquad \mathbf{R}(q(\boldsymbol{\omega})) = e^{[\boldsymbol{\omega}]_\times}, \tag{6}$$

where $\sin(\Omega/2)/\Omega = 1/2$ *at* $\Omega = 0$ *by continuity.*

This is a standard result in rotation geometry (Gallier, 2001, Chapter 9); for completeness, we give a proof in Appendix B.1. Section 4.3 uses (iii) to convert axis-angle controls to quaternions and (ii) to compose their rotations.

State tracking. State tracking asks a sequence model to determine the state reached after a sequence of updates (Merrill et al., 2024). A finite-state automaton provides the standard setting: each input symbol selects a transition, and the model predicts the resulting state at every step (Sarrof et al., 2024; Terzic et al., 2025). Invertible transitions generate a finite group G; tracking a word $g_1 \cdots g_T$ then means returning the products $g_t \cdots g_1$ at all prefixes. Even permutations of five objects form the 60-element group A_5. A *faithful representation* is an injective, product-preserving map from G to a matrix group, as used in Theorem 4. The *word problem* asks whether the final product is the identity. For A_5, it is NC^1-complete (Barrington, 1989), making permutation composition a standard test of state-tracking expressivity (Merrill et al., 2024; Shaj et al., 2026). Fixed-depth diagonal SSMs at logarithmic precision admit TC^0 simulations and therefore cannot solve this problem at arbitrary lengths if $\mathrm{TC}^0 \neq \mathrm{NC}^1$ (Merrill et al., 2024). Appendix B.4 gives the circuit definitions and the arithmetic assumptions used in our comparison.

4 CARACAL

CARACAL gives structured SSMs noncommuting transitions whose ordered products remain compact as memory width grows (Section 4.1). We integrate inputs exactly under a first-order hold (Section 4.2), then use quaternion prefixes to reduce the recurrence to scalar SSD in *SO3SSD* (Section 4.3). A single head supports NC^1-complete group tracking (Section 4.4). Section 4.5 specifies the complete layer.

4.1 NONCOMMUTING $\mathrm{SO}(3)$ STATE TRANSITIONS

Each input selects a rotation axis and angle for every head. Rotations about different axes need not commute. Each head stores $\mathbf{S}_n \in \mathbb{R}^{p\times 3m}$: every row is a value channel partitioned into m three-dimensional blocks, called *state triples*. All triples in the head share the rotation $\mathbf{R}_n = e^{[\boldsymbol{\omega}_n]_\times}$. The direction and length of $\boldsymbol{\omega}_n$ set its axis and angle. The transition matrix is $\mathbf{M}_n = \alpha_n \operatorname{diag}(\mathbf{R}_n, \ldots, \mathbf{R}_n)$, where $\alpha_n = e^{-\gamma_n}$ and $\gamma_n \geq 0$ controls decay. Using the Background's keys $\mathbf{B}_n$, queries $\mathbf{C}_n$, and values $\mathbf{u}_n$, the head computes

$$\boxed{\begin{aligned}\mathbf{S}_{n+1} &= \mathbf{S}_n\mathbf{M}_n^{\mathsf{T}} + \mathbf{u}_n(\mathbf{F}_n^-\mathbf{B}_n)^{\mathsf{T}} + \mathbf{u}_{n+1}(\mathbf{F}_n^+\mathbf{B}_{n+1})^{\mathsf{T}},\\ \mathbf{y}_n &= \mathbf{S}_n\mathbf{C}_n + D\mathbf{u}_n.\end{aligned}} \tag{7}$$

The operators $\mathbf{F}_n^{\pm}$ integrate the previous and current inputs through the rotation, as derived below. A layer contains H heads and $N = 3Hpm$ recurrent coordinates: m sets memory width and H sets the number of independent rotations. Every transition product remains a scaled rotation on each triple, with $\|\mathbf{M}_{t-1}\cdots\mathbf{M}_s\|_2 = \exp(-\sum_{j=s}^{t-1}\gamma_j)$. Decay controls contraction independently of the axes.

Comparison of state transition matrices. Mamba-3 rotates within fixed coordinate planes, so transition products reduce to sums of angles in those planes (Lahoti et al., 2026). CARACAL changes the plane with the input, giving rotations whose order must be preserved. DeltaProduct obtains rotations through products of Householder reflections (Siems et al., 2025); two reflections already represent every element of $\mathrm{SO}(3)$. A nonidentity rotation shared across m triples changes a $2m$-dimensional subspace, requiring at least $2m$ Householder reflections to represent the same $3m \times 3m$ operator (Appendix B.1). We store this rotation product in one quaternion, independent of m, with one additional scalar for decay.

4.2 EXACT EXPONENTIAL INTEGRATION

We derive $\mathbf{F}_n^{\pm}$ by integrating the state input through the damped rotation. Mamba-1/2 add the sampled state input at the interval's end; Mamba-3 weights both endpoints (Lahoti et al., 2026). In our notation, its exponential-trapezoidal rule uses $\mathbf{F}_n^- = (1-\lambda_n)\rho_n\mathbf{M}_n$ and $\mathbf{F}_n^+ = \lambda_n\rho_n\mathbf{I}$, with learned $\lambda_n \in [0,1]$. We retain the two input samples and specialize first-order-hold exponential integration (Hochbruck & Ostermann, 2010) to the damped rotation, integrating the exponential against the interpolated input exactly.

Set $\mathbf{K}_n = [\boldsymbol{\omega}_n]_\times$. On each unit interval, hold $\mathbf{A}_n = \mathrm{diag}(-\gamma_n \mathbf{I}_3 + \mathbf{K}_n, \ldots)$ and the gain ρ_n constant, and interpolate the state input $\mathbf{u}\mathbf{B}^\mathsf{T}$ linearly between its samples. This first-order hold (FOH) requires integrating decay along the rotation axis and damped oscillation in its perpendicular plane. The two scalar weights are

$$f_-(z) = \frac{(z-1)e^z + 1}{z^2}, \qquad f_+(z) = \frac{e^z - 1 - z}{z^2}, \qquad f_-(0) = f_+(0) = \tfrac{1}{2}. \tag{8}$$

Proposition 2. *For the interval model above, set $\Omega_n = \|\boldsymbol{\omega}_n\|_2$, $z_n = -\gamma_n + i\Omega_n$, and $a_n^\pm = f_\pm(-\gamma_n)$. Equation (7) is exact with block-diagonal $\mathbf{F}_n^\pm$ whose kth 3×3 blocks, $1 \le k \le m$, are*

$$(\mathbf{F}_n^\pm)_{kk} = \rho_n \left[a_n^\pm \mathbf{I}_3 + \frac{\mathrm{Im}\, f_\pm(z_n)}{\Omega_n} \mathbf{K}_n + \frac{a_n^\pm - \mathrm{Re}\, f_\pm(z_n)}{\Omega_n^2} \mathbf{K}_n^2 \right], \tag{9}$$

with $(\mathbf{F}_n^\pm)_{kk} = \rho_n a_n^\pm \mathbf{I}_3$ when $\Omega_n = 0$.

Equation (9) applies each coefficient with two cross products and a vector combination, giving constant work per triple. At an undamped half-turn about a unit axis $\mathbf{a}$, symmetric trapezoidal integration assigns zero to a constant perpendicular input $\mathbf{v}$. Exact integration instead gives $(2\rho_n/\pi)\mathbf{a} \times \mathbf{v}$. Appendix B.2 proves the formula and gives its continuous limits for stable evaluation near zero.

4.3 SO3SSD: Accelerating Caracal using Quaternions

SO3SSD evaluates Caracal's changing-axis recurrence through scalar SSD contractions (Dao & Gu, 2024). Mamba-3 implements fixed-plane rotations by accumulating angles and rotating the keys and queries (Lahoti et al., 2026, Propositions 3–4). Changing the axis requires ordered rotation products, which we compute through a quaternion prefix scan shared across the state triples within each head.

Let $\mathbf{Q}_0 = \mathbf{I}$ and let $\mathbf{Q}_n$ apply $\mathbf{R}_{n-1} \cdots \mathbf{R}_0$ to every state triple. Then $\mathbf{M}_{t-1} \cdots \mathbf{M}_j = (\prod_{k=j}^{t-1} \alpha_k)\mathbf{Q}_t \mathbf{Q}_j^\mathsf{T}$. For a write entering memory at time j, applying $\mathbf{Q}_j^\mathsf{T}$ to its key and $\mathbf{Q}_t^\mathsf{T}$ to the query at t absorbs the intervening rotation, leaving a scalar decay mask. The orthogonal coordinate change avoids division by accumulated decay.

This factorization also admits a scan over 3×3 rotation matrices. SO3SSD uses quaternions for two standard properties (Brown, 1989, Sections 3–4):

(1) **Compact prefixes.** Each rotation prefix occupies four real coordinates instead of nine matrix entries, independently of the number of state triples sharing it.
(2) **Normalization.** Accumulating angles in fixed planes avoids products of stored rotation matrices. Changing-axis matrix prefixes can drift from orthogonality under rounding. Quaternion prefixes replace matrix re-orthogonalization with a scalar norm rescaling, which we apply in FP32. This controls norm drift; angular rounding error remains.

Set $q_n = q(\boldsymbol{\omega}_n)$ using Proposition 1(iii), and compute $\pi_0 = \mathbf{1}_\mathbb{H}$, $\pi_{n+1} = q_n \otimes \pi_n$ by an associative Hamilton-product scan. The homomorphism in Proposition 1(ii) gives $\mathbf{R}(\pi_{n+1}) = \mathbf{R}(q_n)\mathbf{R}(\pi_n)$. Thus $\mathbf{R}(\pi_n) = \mathbf{R}_{n-1} \cdots \mathbf{R}_0$, so the quaternion scan computes exactly the rotation prefixes defining $\mathbf{Q}_n$. The next proposition gives the SSD operands, including both first-order-hold inputs.

Proposition 3. *For Eq. (7), set $\mathbf{h}_0 = \mathbf{0}$, $\mathbf{h}_n = \mathbf{F}_{n-1}^+ \mathbf{B}_n$ for $n \ge 1$, and define $\widehat{\mathbf{B}}_n = \mathbf{Q}_{n+1}^\mathsf{T}(\mathbf{F}_n^- \mathbf{B}_n + \mathbf{M}_n \mathbf{h}_n)$, $\widehat{\mathbf{C}}_n = \mathbf{Q}_n^\mathsf{T} \mathbf{C}_n$, and $\widehat{D}_n = D + \mathbf{C}_n^\mathsf{T} \mathbf{h}_n$.*

(i) The state $\widehat{\mathbf{S}}_n = (\mathbf{S}_n - \mathbf{u}_n \mathbf{h}_n^\mathsf{T})\mathbf{Q}_n$ satisfies

$$\widehat{\mathbf{S}}_{n+1} = \alpha_n \widehat{\mathbf{S}}_n + \mathbf{u}_n \widehat{\mathbf{B}}_n^\mathsf{T}, \qquad \mathbf{y}_n = \widehat{\mathbf{S}}_n \widehat{\mathbf{C}}_n + \widehat{D}_n \mathbf{u}_n. \tag{10}$$

(ii) For $0 \le n < T$, stack $\mathbf{y}_{n+1}, \widehat{\mathbf{C}}_{n+1}, \widehat{\mathbf{B}}_n, \mathbf{u}_n, \mathbf{u}_{n+1}$ as rows of $\mathbf{Y}, \widehat{\mathbf{C}}, \widehat{\mathbf{B}}, \mathbf{U}^-, \mathbf{U}^+$. If $\mathbf{S}_0 = \mathbf{0}$, then

$$\mathbf{Y} = (\mathbf{L} \odot (\widehat{\mathbf{C}}\widehat{\mathbf{B}}^\mathsf{T}))\mathbf{U}^- + \mathrm{diag}(\widehat{D}_1, \ldots, \widehat{D}_T)\mathbf{U}^+, \quad L_{nj} = \mathbf{1}_{j \le n} \prod_{k=j+1}^{n} \alpha_k. \tag{11}$$

Equation (11) is scalar SSD (Eq. (3)) with a token-local correction. Algorithm 1 forms its operands through chunk-local quaternion prefixes, sharing rotation preparation across value channels. A work-efficient tree scan takes $O(TN)$ work and storage with $O(\log T)$ depth; decoding retains $O(N)$ state. Our fused implementation uses ordered chunk-boundary transfers overlapped with local work (Appendix C.5). Appendix B.3 proves the reduction and derives a four-coordinate suffix sum for the quaternion backward pass.

4.4 State-tracking expressivity

The following theorem specializes the finite rotation-group tracking constructions of Grazzi et al. (2025); Siems et al. (2025) to our recurrence, with positive damping and a linear readout.

Theorem 4. *Let G be a finite group with a faithful representation in $\mathrm{SO}(3)$. There exist a* CARACAL *head with $p = 3$, $m = 1$, $N = 9$, $\gamma = \log(4/3)$, and a fixed linear classifier $\mathbf{W}$ such that*

$$\arg\max_{g \in G} [\mathbf{W}\mathbf{y}_t]_g = g_t \cdots g_1 \qquad \textit{for every word } g_1 \cdots g_T \in G^* \textit{ and } 1 \le t \le T.$$

The parameters are independent of T. Under relative rounding, $O(\log(T+1))$ mantissa and exponent bits suffice through length T.

For a faithful representation $\varrho : G \hookrightarrow \mathrm{SO}(3)$, set $\mathbf{S}_0 = \mathbf{I}_3$ and suppress input writes. Token g_t selects $\mathbf{R}_{t-1} = \varrho(g_t)$, giving $\mathbf{S}_t = (3/4)^t \varrho(g_t \cdots g_1)^\mathsf{T}$. Choose a unit query $\mathbf{C}$ that is fixed by none of the nonidentity rotations in $\varrho(G)$. The distinct unit vectors $\mathbf{v}_g = \varrho(g)^\mathsf{T}\mathbf{C}$ define classifier rows $\mathbf{v}_g^\mathsf{T}$. The correct score is $(3/4)^t$; every other score is strictly smaller. Damping therefore preserves the class ordering. Appendix B.4 gives the parameter choices and error bounds.

Because the A_5 word problem is NC^1-complete (Barrington, 1989), this capability exceeds that of fixed-depth diagonal SSMs under Merrill et al. (2024)'s logarithmic-precision assumptions and $\mathrm{TC}^0 \neq \mathrm{NC}^1$. SO3SSD performs the noncommuting computation in its quaternion prefix scan. The transformed keys and queries encode these ordered products before scalar SSD evaluates the remaining recurrence.

The proof controls accumulated rotation error with $O(\log(T+1))$ mantissa bits. Exponents of magnitude $O(T)$ represent the decaying state and scores using another $O(\log(T+1))$ bits. The parameters are fixed; the numerical guarantee allows precision to grow with T. Training and length generalization at fixed precision are evaluated separately in Section 5.1 and Appendix D.1.

4.5 Architecture

Each layer projects tokens into rotation, decay, and write controls; scan preparation (SP) computes the integration coefficients (Figure 1). Parallel projections produce values, keys, queries, and an output gate (Dao & Gu, 2024). Causal convolutions give the values, keys, and queries local context; only values pass through SiLU.

We normalize keys and queries to unit ℓ_2 norm; Mamba-3 uses RMSNorm on its keys and queries (Lahoti et al., 2026). At the output, each head adds the local skip $D\mathbf{u}_n$ to its recurrent readout and applies RMSNorm. The SiLU gate modulates the result before an output projection combines the heads. This follows Mamba's combination of H3 and a gated MLP (Gu & Dao, 2024, Figure 3). Heads begin at fixed coordinate vectors, with zero input and convolution histories (Appendix C).

For associative recall, we introduce CARACAL-Δ, which replaces the additive write with a delta-rule correction after rotating memory (Yang et al., 2024). It writes the prediction error at the addressed key; Appendix C.4 gives the recurrence. We evaluate this variant on MAD; SO3SSD and its benchmarks use the additive recurrence.

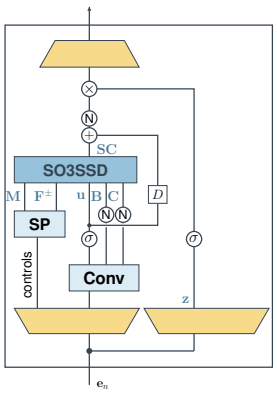

Figure 1: The CARACAL architecture.

5 EXPERIMENTS

We evaluate CARACAL as a *general-purpose token mixer*: its richer state transitions must support hard state tracking (Merrill et al., 2024; Terzic et al., 2025) while preserving memory skills, language quality, and execution speed. Following the protocols of Walker et al. (2025); Shaj et al. (2026), we compare against recent SSM and linear-attention baselines (Siems et al., 2025; Lahoti et al., 2026; Hatamizadeh et al., 2026). To test whether CARACAL meets these combined demands, we ask four questions:

(1) **State tracking.** Can CARACAL solve A_5 composition and finite-state automaton emulation, and generalize to longer sequences?
(2) **Mechanistic architecture design.** How well does a single layer compress and retrieve information on MAD (Poli et al., 2024), and what does the delta rule add?
(3) **Language modeling.** Do stacked layers match SSM and Transformer baselines in perplexity and zero-shot accuracy after pretraining at two scales?
(4) **Parallel training and autoregressive decoding.** How does SO3SSD compare with optimized SSM and attention kernels across sequence lengths and batch sizes?

Full experimental details and additional results appear in Appendix D.

5.1 STATE TRACKING

A_5 and FSA emulation benchmark. We test state tracking through A_5 composition and finite-state automaton (FSA) emulation. The A_5 task predicts the running permutation product, as in Theorem 4. Following the Walker protocol (Walker et al., 2025), we train CARACAL separately at lengths 3–20 and search depths one through four over five seeds. A length is *solved* when validation token accuracy exceeds 90%; we report the smallest successful depth. For FSA emulation (Delétang et al., 2023), two-layer models train on lengths 3–40 and predict the final state at test lengths 40–256.

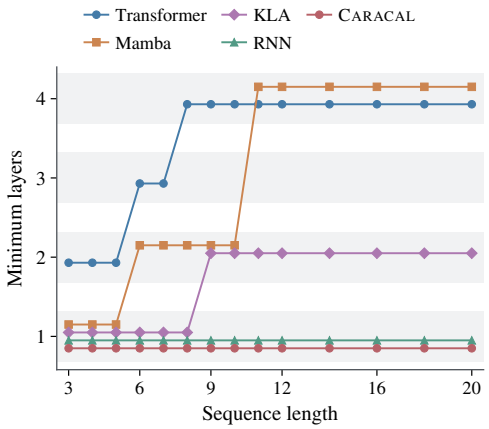

Figure 2: **A_5 state tracking.** Minimum depth at the 90% accuracy threshold. CARACAL uses five seeds under Walker's protocol. Transformer, Mamba, and RNN results are from Merrill et al. (2024); KLA results are from Shaj et al. (2026). Coincident depths are offset vertically.

Discussion. *A single* CARACAL *layer solves A_5 at every tested length from 3 to 20* (Figure 2), with three to five successful seeds and median selected depth one throughout. Appendix D.1.2 reports a separate A_5 length-generalization experiment. On FSA emulation, the two-layer model generalizes beyond the training lengths with near-perfect accuracy on three of four tasks and the highest average accuracy among the evaluated parallel models, including Gated DeltaProduct, GDN2, and Mamba-3 (Table 1). PD-SSM reports 98.8% mean validation accuracy (Terzic et al., 2025); its original implementation's runtime made our five-seed retraining sweep impractical.

5.2 MECHANISTIC ARCHITECTURE DESIGN

Architecture design. Language modeling requires storing and retrieving information from context. MAD (Poli et al., 2024) isolates these skills in six synthetic tasks (Table 2). Following Shaj et al. (2026), we compare single-layer models at width 128.

Table 1: **FSA emulation accuracy (%).** Five-seed test mean ± SD; training lengths 3–40, validation and test lengths 40–256. Bold/underline: best/second-best parallel model.

Model	Cycle nav.	Even pairs	Mod. arith.	Parity	Average
Sequential recurrence					
LSTM	95.76 ± 5.13	100.00 ± 0.00	96.14 ± 8.40	100.00 ± 0.00	97.98 ± 3.31
Parallel sequence models					
SLiCE-DPLR	$\underline{49.34} \pm 7.23$	$\mathbf{100.00} \pm 0.00$	41.38 ± 18.66	91.20 ± 8.66	70.48 ± 7.44
GDN2	41.35 ± 7.26	$\mathbf{100.00} \pm 0.00$	45.07 ± 9.54	93.83 ± 5.01	70.06 ± 1.27
Gated DeltaProduct$[-1, 1]$	44.76 ± 3.49	$\underline{99.97} \pm 0.07$	$\underline{69.05} \pm 15.92$	$\underline{94.03} \pm 4.82$	$\underline{76.95} \pm 4.42$
Mamba-2	49.10 ± 11.53	97.28 ± 5.98	26.12 ± 5.15	53.39 ± 1.91	56.47 ± 3.28
Mamba-3	44.35 ± 9.92	99.44 ± 1.26	46.79 ± 4.10	65.89 ± 10.26	64.12 ± 2.02
Transformer	24.45 ± 0.40	98.18 ± 1.75	21.92 ± 0.27	50.90 ± 0.78	48.86 ± 0.55
CARACAL	$\mathbf{99.99} \pm 0.01$	$\mathbf{100.00} \pm 0.00$	$\mathbf{88.59} \pm 8.25$	$\mathbf{99.98} \pm 0.05$	$\mathbf{97.14} \pm 2.06$
Random chance	20.0	50.0	20.0	50.0	35.0

Discussion. *The delta rule improves compression and selective copying.* CARACAL-Δ leads on both tasks, while GDN2 is strongest on fuzzy recall (Table 2). The gains persist at similar parameter counts: a smaller variant reaches 94.0% compression and 98.8% copying accuracy with 122.9K parameters, compared with GDN2's 116.5K, excluding task embeddings and readouts. This control uses three seeds; the default comparison uses five (Appendix D.2).

Table 2: **Single-layer performance on MAD.** Five-seed mean test accuracy (%) for width-128 layers. Bold/underline: best/second best.

Model	Comp.	Mem.	Ctx.	Noisy	Fuzzy	Copy
Mamba-3	78.5	**100.0**	**100.0**	**100.0**	43.6	26.1
GDN2	88.3	**100.0**	**100.0**	**100.0**	**65.0**	86.1
KLA	80.8	**100.0**	**100.0**	**100.0**	33.5	<u>90.4</u>
CARACAL	<u>90.7</u>	**100.0**	**100.0**	**100.0**	<u>48.0</u>	78.6
CARACAL-Δ	**94.0**	**100.0**	**100.0**	**100.0**	45.8	**98.9**

5.3 LANGUAGE MODELING

Pretraining and zero-shot evaluation. We pretrain CARACAL and four baselines on FineWeb-Edu (Penedo et al., 2024) using the two model scales of Shaj et al. (2026). Within each scale, models use comparable parameter counts and identical token budgets, with the same tokenizer, data order for each seed, and context length of 2,048. We evaluate eight zero-shot tasks using the LM Evaluation Harness (Gao et al., 2024).

Discussion. We obtain the *lowest test perplexity among reported measurements* at both scales. The eight-task zero-shot mean is close to Mamba-3 at 45M and remains competitive at the larger scale (Table 3; Appendix D.3).

5.4 PARALLEL TRAINING AND AUTOREGRESSIVE DECODING

We benchmark temporal operators on one H200 at value width 2,048. SO3SSD uses 96 state coordinates per value; recurrent baselines use 64 or 128. Timings include SO3SSD's control transformations and FOH preparation, with gradients through both. Baselines retain native preparation boundaries. Projections, convolutions, address normalization, and output processing are excluded (Appendix D.4).

Discussion. *SO3SSD leads long-sequence training at batch one.* At 512K tokens, it is $1.44\times$ faster with 31% lower peak allocation than the fastest baseline, Mamba-3 with 64 coordinates. Against 128-coordinate baselines, it leads BF16 decoding at batches 8–128; smaller-state RWKV-7 decodes faster (Figure 3; Appendix Figure 7). At batch eight and 8K tokens, Mamba-3 leads training (Appendix Table 12).

Table 3: **Language modeling.** FineWeb-Edu test perplexity and zero-shot accuracy (%) from the LM Evaluation Harness. Three-seed means at 45M; one seed at 175–180M. Perplexity uses final checkpoints; zero-shot accuracy uses validation-selected checkpoints. Bold/underline: best/second best within each scale; acc_n denotes length-normalized accuracy.

Model	PPL ↓	LAMB. acc↑	HellaS. acc_n ↑	PIQA acc↑	ARC-E acc↑	ARC-C acc_n ↑	WinoG. acc↑	OBQA acc↑	BoolQ acc↑	Average acc↑
45M parameters / 1.8B tokens										
Mamba-3	29.52	17.16	28.38	59.83	**48.58**	**23.49**	52.12	**17.87**	59.63	**38.38**
GDN2	30.65	15.77	28.36	**60.34**	47.91	22.84	51.41	16.60	56.13	37.42
KLA	31.34	15.33	28.30	59.67	47.12	23.21	51.51	16.53	58.38	37.51
Transformer	30.56	**17.71**	27.96	59.10	47.84	23.27	**52.78**	16.07	57.30	37.75
CARACAL	**29.04**	16.77	**28.45**	60.25	47.95	23.46	51.99	17.00	**61.03**	38.36
175–180M parameters / 10.9B tokens										
Mamba-3	–	**29.83**	**35.50**	63.93	55.93	**26.79**	**52.09**	**22.80**	**60.95**	**43.48**
GDN2	18.86	28.22	34.57	63.33	56.40	25.77	51.85	20.00	60.21	42.54
KLA	18.80	28.76	35.01	65.18	56.73	26.71	50.12	21.80	60.21	43.07
Transformer	19.91	26.86	33.19	64.20	55.64	26.54	50.59	19.60	57.49	41.76
CARACAL	**18.09**	27.11	35.27	**65.40**	**56.99**	26.71	49.57	21.60	59.72	42.80

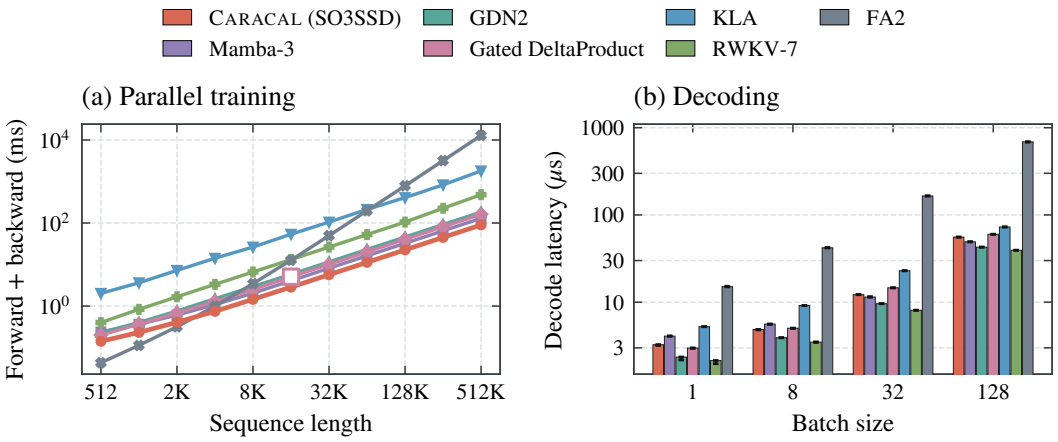

Figure 3: **Training and decoding speed.** (a) Training at batch one; (b) decoding. BF16 operands, FP32 states; state coordinates per value: 96 for CARACAL, 64 for recurrent baselines. Medians with training IQRs (six blocks) and decode ranges (four blocks). Open squares: eager GDP.

6 DISCUSSION AND CONCLUSION

CARACAL meets all four experimental objectives posed in Section 5. A single layer solves the tested A_5 benchmark, matching RNN depth and improving on KLA's reported depth at longer tested lengths. The learned models generalize beyond the training lengths on both group and automaton tasks. FSA test accuracy reaches 97.1%, far above Mamba-3 and GDN2 and close to the LSTM test result (98.0%) and PD-SSM's published validation result (98.8%) (Terzic et al., 2025). On MAD, we lead on compression and selective copying. At both language-model scales, we obtain the lowest measured test perplexity in our comparison and competitive zero-shot accuracy.

Including control preparation, SO3SSD beats every tested baseline in training at 512K tokens and batch one: 1.44× faster with 31% lower peak allocation than the fastest, Mamba-3 with a smaller state. It also decodes 1.34–1.48× faster than 128-coordinate Mamba-3 across both tested precisions and all batches. CARACAL combines noncommuting $\mathrm{SO}(3)$ dynamics, exact integration, and quaternion computation to answer the introduction's question: hard state tracking with SSD's efficiency and strong language modeling and recall.

AI USE STATEMENT

We used generative AI to assist with literature search, coding, and manuscript editing. The authors take responsibility for the final content.

REPRODUCIBILITY STATEMENT

Appendix B collects proofs and derivations; Appendix C specifies the layer and its implementation. Appendix D gives datasets, model configurations, optimization, checkpoint selection, seed-level results, and timing procedures. Build scripts regenerate figures and tables from a frozen snapshot of the experimental records.

REFERENCES

David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.

Kai Biegun, Rares Dolga, Jake Cunningham, and David Barber. RotRNN: Modelling long sequences with rotations. *arXiv preprint arXiv:2407.07239*, 2024.

Guy E. Blelloch. Prefix sums and their applications. Technical Report CMU-CS-90-190, Carnegie Mellon University, 1990.

Jared Boyer, T. Konstantin Rusch, and Daniela Rus. Learning to dissipate energy in oscillatory state-space models. *arXiv preprint arXiv:2505.12171*, 2025.

Christopher M. Brown. Some computational properties of rotation representations. Technical Report TR303, University of Rochester, Department of Computer Science, 1989.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10041–10071. PMLR, 2024.

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and Pedro A. Ortega. Neural networks and the Chomsky hierarchy. In *International Conference on Learning Representations*, 2023.

Jean Gallier. *Geometric Methods and Applications: For Computer Science and Engineering*, volume 38 of *Texts in Applied Mathematics*. Springer, 2001.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, et al. The language model evaluation harness, 2024.

Riccardo Grazzi, Julien Siems, Jörg K. H. Franke, Arber Zela, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear RNNs through negative eigenvalues. In *International Conference on Learning Representations*, 2025.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Conference on Language Modeling*, 2024.

Ali Hatamizadeh, Yejin Choi, and Jan Kautz. Gated DeltaNet-2: Decoupling erase and write in linear attention. *arXiv preprint arXiv:2605.22791*, 2026.

Marlis Hochbruck and Alexander Ostermann. Exponential integrators. *Acta Numerica*, 19:209–286, 2010.

Andrej Karpathy. nanochat: The best ChatGPT that $100 can buy. GitHub, 2025.

Arjun Karuvally, Franz Nowak, T. Anderson Keller, Carmen Amo Alonso, Terrence Sejnowski, and Hava T Siegelmann. Bridging expressivity and scalability with adaptive unitary SSMs. In *Advances in Neural Information Processing Systems*, 2025.

Aakash Lahoti, Kevin Li, Berlin Chen, Caitlin Wang, Aviv Bick, J Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved sequence modeling using state space principles. In *International Conference on Learning Representations*, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In *International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35492–35506. PMLR, 2024.

Richard Montgomery. Gauge theory of the falling cat. In Michael J. Enos (ed.), *Dynamics and Control of Mechanical Systems: The Falling Cat and Related Problems*, volume 1 of *Fields Institute Communications*, pp. 193–218. American Mathematical Society, 1993.

Marc Olive. Effective computation of SO(3) and O(3) linear representation symmetry classes. *Mathematics and Mechanics of Complex Systems*, 7(3):203–237, 2019.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024.

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, Nathan Wilce, Johan S. Wind, Tianyi Wu, Daniel Wuttke, and Christian Zhou-Zheng. RWKV-7 Goose with expressive dynamic state evolution. *arXiv preprint arXiv:2503.14456*, 2025.

Michael Poli, Armin W. Thomas, Eric Nguyen, Pragaash Ponnusamy, Björn Deiseroth, Kristian Kersting, Taiji Suzuki, Brian Hie, Stefano Ermon, Christopher Ré, Ce Zhang, and Stefano Massaroli. Mechanistic design and scaling of hybrid architectures. In *International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2024.

T. Konstantin Rusch and Daniela Rus. Oscillatory state-space models. In *International Conference on Learning Representations*, 2025.

Yash Sarrof, Yana Veitsman, and Michael Hahn. The expressive capacity of state space models: A formal language perspective. In *Advances in Neural Information Processing Systems*, volume 37. Curran Associates, Inc., 2024.

Vaisakh Shaj, Cameron Barker, Aidan Scannell, Andras Szecsenyi, Elliot J. Crowley, and Amos Storkey. Kalman linear attention: Parallel Bayesian filtering for efficient language modelling and state tracking. In *International Conference on Machine Learning*, 2026.

Mehran Shakerinava, Behnoush Khavari, Siamak Ravanbakhsh, and Sarath Chandar. The expressive limits of diagonal SSMs for state-tracking. In *International Conference on Learning Representations*, 2026.

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. DeltaProduct: Improving state-tracking in linear RNNs via Householder products. In *Advances in Neural Information Processing Systems*, 2025.

Jimmy T.H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations*, 2023.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Aleksandar Terzic, Nicolas Menet, Michael Hersche, Thomas Hofmann, and Abbas Rahimi. Structured sparse transition matrices to enable state tracking in state-space models. In *Advances in Neural Information Processing Systems*, 2025.

Benjamin Walker, Lingyi Yang, Nicola Muca Cirone, Cristopher Salvi, and Terry Lyons. Structured linear CDEs: Maximally expressive and parallel-in-time sequence models. In *Advances in Neural Information Processing Systems*, 2025.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Advances in Neural Information Processing Systems*, volume 37. Curran Associates, Inc., 2024.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, 2025.

A ADDITIONAL RELATED WORK

Other structured SSMs. LinOSS and D-LinOSS use fixed oscillator dynamics, with D-LinOSS learning an independent damping time scale (Rusch & Rus, 2025; Boyer et al., 2025). RotRNN combines a learned fixed rotation with scalar decay; its authors propose input-dependent rotations as future work (Biegun et al., 2024). AUSSM makes rotational frequencies input-dependent within a shared eigenbasis (Karuvally et al., 2025). CARACAL also selects the rotation axes, so successive transitions need not share an eigenbasis.

Structure preserved under composition. PD-SSM keeps products tractable through column-one-hot routing and complex diagonal scaling, training the discrete routing with surrogate gradients (Terzic et al., 2025). CARACAL learns real-valued rotation controls. SLiCE keeps a block-diagonal structure whose small dense blocks compose independently. Its authors identify efficient matrix exponentials and scans over these systems as open implementation problems (Walker et al., 2025). We specialize the blocks to damped rotations: the resulting polynomial integration coefficients and orthogonal change of coordinates give an explicit integrator and a scalar SSD reduction. BD-SLiCE's maximal-expressivity theorem assumes unbounded block size; our group-tracking results use three-dimensional rotation blocks.

State tracking and numerical precision. The restrictions on diagonal SSMs arise under different computational models. Finite-precision language analysis shows that nonnegative diagonal recurrences fail on parity (Sarrof et al., 2024), which negative eigenvalues enable (Grazzi et al., 2025). For complex diagonal SSMs, Shakerinava et al. (2026) classify the groups tracked at each depth and obtain only solvable groups without a complexity conjecture. RWKV-7's asymmetric diagonal-plus-rank-one updates recognize every regular language with four layers; Gated DeltaProduct reaches the same language class with language-dependent depth (Peng et al., 2025; Siems et al., 2025). Our analysis constructs finite rotation-group tracking within the layer's parameterization, including positive damping and a linear readout. The numerical guarantee allows precision to grow logarithmically with sequence length (Appendix B.4).

B PROOFS AND DERIVATIONS

We prove the quaternion identity used in the Background, derive the integration coefficients and SO3SSD algorithm, and establish the state-tracking result. All state matrices use the row convention of Eq. (7). Model parameters and numerical implementation details appear in Appendix C.

B.1 ROTATION IDENTITIES

Proof of Proposition 1. The Hamilton product is associative, its norm is multiplicative, and $q \otimes \bar{q} = (\|q\|^2, \mathbf{0})$, as direct expansion verifies (Gallier, 2001, Chapter 9). Unit quaternions therefore form a group with inverse $\bar{q}$. Conjugation preserves pure quaternions and their norms. Its determinant is one by continuity on the unit sphere S^3, while associativity gives $\mathbf{R}(q_2 \otimes q_1) = \mathbf{R}(q_2)\mathbf{R}(q_1)$. Expanding the action gives

$$\mathbf{R}(q)\mathbf{v} = (q_0^2 - \|\mathbf{q}\|^2)\mathbf{v} + 2\mathbf{q}(\mathbf{q}^\mathsf{T}\mathbf{v}) + 2q_0\mathbf{q} \times \mathbf{v}.$$

Substituting $q(\boldsymbol{\omega})$ yields Rodrigues' formula $e^{[\boldsymbol{\omega}]_\times}$, including its limit at zero. Every three-dimensional rotation has an axis and angle, proving surjectivity. A quaternion in the kernel commutes with every pure quaternion; the Hamilton product forces its vector part to vanish. The kernel is thus $\{\pm\mathbf{1}_\mathbb{H}\}$, so each rotation has exactly two representatives, q and $-q$. □

Reflections for a repeated rotation. For a nonidentity $\mathbf{R} \in \mathrm{SO}(3)$, $\mathrm{rank}(\mathbf{R} - \mathbf{I}_3) = 2$. Hence the repeated rotation $\mathbf{P} = \mathrm{diag}(\mathbf{R}, \ldots, \mathbf{R})$ on m triples satisfies $\mathrm{rank}(\mathbf{P} - \mathbf{I}_{3m}) = 2m$. Each generalized Householder factor $\mathbf{H}_j = \mathbf{I} - \beta_j \mathbf{v}_j \mathbf{v}_j^\mathsf{T}$ differs from the identity by rank at most one. Telescoping the product and using subadditivity of rank gives

$$\mathrm{rank}(\mathbf{H}_k \cdots \mathbf{H}_1 - \mathbf{I}) \le \sum_{j=1}^{k} \mathrm{rank}(\mathbf{H}_j - \mathbf{I}) \le k.$$

Thus $k \geq 2m$ to represent $\mathbf{P}$. Two reflections realize each three-dimensional rotation (Grazzi et al., 2025; Siems et al., 2025), so block-supported factors attain $2m$. The count concerns a factorization of this prescribed matrix; sharing its repeated blocks also permits $O(m)$ application.

B.2 Exact first-order-hold integration

Proof of Proposition 2. For value channel j and key triple $\mathbf{B}_{n,k}$, suppress the interval index and set $\mathbf{A} = -\gamma\mathbf{I}_3 + \mathbf{K}$, $\mathbf{K} = [\boldsymbol{\omega}]_\times$, and $\mathbf{v}^- = u_{n,j}\mathbf{B}_{n,k}$, $\mathbf{v}^+ = u_{n+1,j}\mathbf{B}_{n+1,k}$. On the unit interval the input is $\rho[(1-s)\mathbf{v}^- + s\mathbf{v}^+]$. Variation of constants gives

$$\mathbf{x}(1) = e^{\mathbf{A}}\mathbf{x}(0) + \rho \int_0^1 e^{(1-s)\mathbf{A}}[(1-s)\mathbf{v}^- + s\mathbf{v}^+]\,\mathrm{d}s. \tag{12}$$

An interval of length h has the same form after rescaling time, replacing $\mathbf{A}$ by $h\mathbf{A}$ and ρ by $h\rho$. Substituting $r = 1 - s$ identifies the two matrix coefficients as $\rho f_-(\mathbf{A})$ and $\rho f_+(\mathbf{A})$, where

$$f_-(z) = \int_0^1 re^{rz}\,\mathrm{d}r, \qquad f_+(z) = \int_0^1 (1-r)e^{rz}\,\mathrm{d}r. \tag{13}$$

Integration yields Eq. (8) and the value $1/2$ at zero. This is the first-order-hold specialization of exponential integration (Hochbruck & Ostermann, 2010).

For $\Omega = \|\boldsymbol{\omega}\| > 0$, let $\mathbf{a} = \boldsymbol{\omega}/\Omega$ and $\mathbf{J} = [\mathbf{a}]_\times$. The identity $\mathbf{J}^2 = \mathbf{a}\mathbf{a}^\mathsf{T} - \mathbf{I}$ gives

$$e^{r\mathbf{A}} = e^{-\gamma r}\big[\mathbf{a}\mathbf{a}^\mathsf{T} + \cos(\Omega r)(\mathbf{I} - \mathbf{a}\mathbf{a}^\mathsf{T}) + \sin(\Omega r)\mathbf{J}\big].$$

Integrating with the two weights in Eq. (13) gives coefficients $f_\pm(-\gamma)$ on the axis and $\mathrm{Re}\, f_\pm(-\gamma + i\Omega)$, $\mathrm{Im}\, f_\pm(-\gamma + i\Omega)$ on the plane. Substitute $\mathbf{a}\mathbf{a}^\mathsf{T} = \mathbf{I} + \mathbf{K}^2/\Omega^2$ and $\mathbf{J} = \mathbf{K}/\Omega$ to obtain Eq. (9). At $\Omega = 0$, $\mathbf{A} = -\gamma\mathbf{I}$ gives the stated scalar coefficient. Applying this argument to every triple and value channel proves the matrix recurrence. □

Evaluation near zero. The apparent singularities in the scalar formulas are removable:

$$f_-(z) = \sum_{j=0}^{\infty} \frac{z^j}{j!(j+2)}, \qquad f_+(z) = \sum_{j=0}^{\infty} \frac{z^j}{(j+2)!}.$$

For real x, the coefficients of $\mathbf{K}$ and $\mathbf{K}^2$ converge to $f'_\pm(x)$ and $f''_\pm(x)/2$ as $\Omega \to 0$. These series, or the equivalent damped moments in Appendix C.5, avoid subtracting nearly equal quantities and dividing by a small angle.

Half-turn example. For $\gamma = 0$, $\boldsymbol{\omega} = \pi\mathbf{a}$ with $\|\mathbf{a}\| = 1$, and a constant input $\mathbf{v} \perp \mathbf{a}$, Eq. (12) contributes $\rho\int_0^1[\cos(\pi r)\mathbf{v} + \sin(\pi r)\mathbf{a}\times\mathbf{v}]\,\mathrm{d}r = (2\rho/\pi)\mathbf{a}\times\mathbf{v}$. The symmetric endpoint rule gives $\rho(\mathbf{R} + \mathbf{I})\mathbf{v}/2 = \mathbf{0}$.

B.3 SO3SSD: reduction, algorithm, and backward pass

Throughout this subsection, $d_s = 3m$ and the operators have the full-head shapes of Section 4.3.

Proof of Proposition 3. By definition, $\mathbf{Q}_{n+1} = \mathrm{diag}(\mathbf{R}_n, \ldots)\mathbf{Q}_n$, hence $\mathbf{M}_n^\mathsf{T}\mathbf{Q}_{n+1} = \alpha_n\mathbf{Q}_n$. With $\mathbf{h}_0 = \mathbf{0}$ and $\mathbf{h}_{n+1} = \mathbf{F}_n^+\mathbf{B}_{n+1}$, subtract the current write from Eq. (7):

$$\mathbf{S}_{n+1} - \mathbf{u}_{n+1}\mathbf{h}_{n+1}^\mathsf{T} = (\mathbf{S}_n - \mathbf{u}_n\mathbf{h}_n^\mathsf{T})\mathbf{M}_n^\mathsf{T} + \mathbf{u}_n(\mathbf{F}_n^-\mathbf{B}_n + \mathbf{M}_n\mathbf{h}_n)^\mathsf{T}. \tag{14}$$

Right-multiplication by $\mathbf{Q}_{n+1}$ gives $\widehat{\mathbf{S}}_{n+1} = \alpha_n\widehat{\mathbf{S}}_n + \mathbf{u}_n\widehat{\mathbf{B}}_n^\mathsf{T}$. Substitution of $\mathbf{S}_n = \widehat{\mathbf{S}}_n\mathbf{Q}_n^\mathsf{T} + \mathbf{u}_n\mathbf{h}_n^\mathsf{T}$ into the readout proves (i). Unrolling gives

$$\widehat{\mathbf{S}}_{n+1} = \left(\prod_{k=0}^{n}\alpha_k\right)\mathbf{S}_0 + \sum_{j=0}^{n}\left(\prod_{k=j+1}^{n}\alpha_k\right)\mathbf{u}_j\widehat{\mathbf{B}}_j^\mathsf{T}. \tag{15}$$

Multiplying by $\widehat{\mathbf{C}}_{n+1}$, adding $\widehat{D}_{n+1}\mathbf{u}_{n+1}$ and stacking the outputs proves (ii) when $\mathbf{S}_0 = \mathbf{0}$. The first term supplies the usual SSD incoming-state response for nonzero $\mathbf{S}_0$. □

Chunked evaluation. Algorithm 1 separates each chunk's input response from the contribution of its incoming state, following SSD (Dao & Gu, 2024, Appendix C). For a chunk of intervals $s, \dots, e-1$, reset $\pi_s = \mathbf{1}_{\mathbb{H}}$ and $\mathbf{h}_s = \mathbf{0}$, and apply the preceding reduction with origin s. The entire previous history is already in the physical state $\mathbf{S}_s$; resetting $\mathbf{h}_s$ prevents counting its last write twice. Within the chunk, set $\mathbf{h}_{n+1} = \mathbf{F}_n^{+}\mathbf{B}_{n+1}$.

Let $\ell = e - s$. Stack $\widehat{\mathbf{B}}_n^{\mathsf{T}}$ and $\widehat{\mathbf{C}}_{n+1}^{\mathsf{T}}$ as rows of $\widehat{\mathbf{B}}, \widehat{\mathbf{C}} \in \mathbb{R}^{\ell \times d_s}$; $\mathbf{U}^{-}, \mathbf{U}^{+} \in \mathbb{R}^{\ell \times p}$ stack $\mathbf{u}_n^{\mathsf{T}}, \mathbf{u}_{n+1}^{\mathsf{T}}$. Rows are indexed by $n = s, \dots, e-1$. Define $a_n = \prod_{k=s}^{n} \alpha_k$, $b_n = \prod_{k=n+1}^{e-1} \alpha_k$, $\delta = a_{e-1}$, and $L_{nj} = \mathbf{1}_{j \le n} \prod_{k=j+1}^{n} \alpha_k$. The vector $\widehat{\mathbf{d}}$ stacks $\widehat{D}_{n+1}$. All empty products equal one.

Algorithm 1 SO3SSD: quaternion preparation and chunked scalar SSD

Input: Operands of Eq. (7), initial state $\mathbf{S}_0$, and a partition into chunks of length at most L.
Output: $\mathbf{y}_1, \dots, \mathbf{y}_T$ and physical state $\mathbf{S}_T$.

1. **Prepare each chunk independently.** Use the lift in Proposition 1(iii) to scan $\pi_{n+1} = q(\boldsymbol{\omega}_n) \otimes \pi_n$, and scan the damping sums. Part (ii) gives $\mathbf{R}(\pi_n) = \mathbf{R}_{n-1} \cdots \mathbf{R}_s$. Apply $\mathbf{Q}_n^{\mathsf{T}}$ to each triple by conjugation with $\bar{\pi}_n$. Form
$$\widehat{\mathbf{B}}_n = \mathbf{Q}_{n+1}^{\mathsf{T}}(\mathbf{F}_n^{-}\mathbf{B}_n + \mathbf{M}_n\mathbf{h}_n), \quad \widehat{\mathbf{C}}_{n+1} = \mathbf{Q}_{n+1}^{\mathsf{T}}\mathbf{C}_{n+1}, \quad \widehat{D}_{n+1} = D + \mathbf{C}_{n+1}^{\mathsf{T}}\mathbf{h}_{n+1}.$$
2. **Compute local responses independently.** Evaluate the scalar SSD contractions and current-input correction:
$$\mathbf{Y}_{\mathrm{local}} = (\mathbf{L} \odot (\widehat{\mathbf{C}}\widehat{\mathbf{B}}^{\mathsf{T}}))\mathbf{U}^{-} + \mathrm{diag}(\widehat{\mathbf{d}})\mathbf{U}^{+},$$
$$\mathbf{J} = (\mathbf{U}^{-})^{\mathsf{T}}\,\mathrm{diag}(\mathbf{b})\widehat{\mathbf{B}} + \mathbf{u}_e\mathbf{h}_e^{\mathsf{T}}\mathbf{Q}_e.$$
3. **Propagate physical boundary states in chunk order.** Retain each entering state and compute $\mathbf{S}_e = (\delta\mathbf{S}_s + \mathbf{J})\mathbf{Q}_e^{\mathsf{T}}$.
4. **Complete each chunk's outputs.** Once $\mathbf{S}_s$ is available, return the rows of $\mathbf{Y} = \mathbf{Y}_{\mathrm{local}} + \mathrm{diag}(\mathbf{a})\widehat{\mathbf{C}}\mathbf{S}_s^{\mathsf{T}}$.

Equation (15), with origin s, proves the output formula. Its terminal state is $\widehat{\mathbf{S}}_e = \delta\mathbf{S}_s + (\mathbf{U}^{-})^{\mathsf{T}}\,\mathrm{diag}(\mathbf{b})\widehat{\mathbf{B}}$. Restoring the current endpoint and rotating back gives step 3. Induction proves equality with the serial recurrence for arbitrary incoming states and chunk lengths, including partial final chunks. Step 1 supplies the rotated operands for the scalar SSD contractions in step 2. Steps 3–4 propagate boundary states and add their output contributions. Retaining the physical state, convolution histories, and preceding endpoint operands preserves the interval inputs across chunk boundaries. Prefill, continuation, and token-by-token decoding therefore agree in exact arithmetic.

Work and memory. For one head, preparing quaternion prefixes costs $O(T)$; rotating the keys and queries costs $O(Td_s)$ and is shared across all p value channels. Full-state rotations occur only at the K chunk boundaries, with $O(Kpd_s)$ work; uniform chunks of length L give $K = \lceil T/L \rceil$.

The total work is $O(TL(d_s + p) + Tpd_s)$. We retain boundary states and increments and recompute rotation prefixes and key–query products in backward. Including operands and outputs, global storage is $O(T(p + d_s) + Kpd_s)$ per head.

Each interval defines an affine map whose linear part is a scaled rotation. Closure of rotations under multiplication keeps their compositions in this family, so each scan merge costs $O(N)$ for $N = Hpd_s$. A work-efficient tree scan gives the main text's $O(TN)$ work and $O(\log T)$ depth (Blelloch, 1990). The chunked algorithm also has $O(TN)$ work for fixed L; its measured fused schedule uses K ordered boundary transfers, overlapped with independent local work.

Backward through quaternion prefixes. Matrix-product gradients reuse the scalar SSD contractions. The additional operation is differentiating the ordered quaternion scan. Its reverse pass reduces to an additive suffix scan in four coordinates, without retaining a computational graph for every Hamilton multiplication. Write X^{*} for a loss cotangent under the Euclidean or Frobenius inner product.

Proposition 5. *Let $\pi_s = \mathbf{1}_{\mathbb{H}}$ and $\pi_{n+1} = q_n \otimes \pi_n$ for unit quaternions q_n. Let $g_j \in \mathbb{R}^4$ be the cotangent contributed directly to prefix π_j by all its uses, before propagation through other prefixes. Then the tangent cotangent of q_n is*

$$q_n^* = \pi_{n+1} \otimes \eta_n \otimes \bar{\pi}_n, \qquad \eta_n = \sum_{j=n+1}^{e} \bar{\pi}_j \otimes \big(g_j - \langle \pi_j, g_j \rangle \pi_j\big). \tag{16}$$

Proof. For $j \geq n+1$, holding the other interval quaternions fixed gives $\mathrm{d}\pi_j = (\pi_j \otimes \bar{\pi}_{n+1}) \otimes \mathrm{d}q_n \otimes \pi_n$. Multiplication by a unit quaternion is orthogonal, so its adjoint multiplies by the conjugate. Projecting g_j onto the tangent space and summing these adjoints gives Eq. (16). Each summand in η_n is pure imaginary; conjugation by π_n preserves this property, so q_n^* is tangent at q_n. □

For $N(q) = q/\|q\|$, multiplicativity of the quaternion norm gives $N(a \otimes b) = N(a) \otimes N(b)$ in real arithmetic. The pullback of N is $(g - \langle N(q), g \rangle N(q))/\|q\|$. Thus Eq. (16), evaluated on normalized lifts, returns a tangent cotangent; division by the raw interval lift's norm completes its normalization derivative.

Backward first propagates boundary-state cotangents in reverse chunk order. With $\mathbf{W} = \mathbf{Y}^*$, the boundary contribution obeys

$$\mathbf{J}^* = \mathbf{S}_e^* \mathbf{Q}_e, \qquad \mathbf{S}_s^* = \mathbf{W}^{\mathsf{T}} \operatorname{diag}(\mathbf{a}) \widehat{\mathbf{C}} + \delta \mathbf{J}^*. \tag{17}$$

Chunks then differentiate their local contractions independently. Both FOH endpoints and the terminal rotation contribute to g_j in Eq. (16); contributions to repeated uses of a sample are summed at its original index. A scalar suffix sum differentiates cumulative damping. The chain rule through $q(\boldsymbol{\omega}_n)$ and $\mathbf{F}_n^{\pm}$ returns the control gradients, retaining derivatives through incoming states and endpoints. Backward matches forward's asymptotic work and global storage.

B.4 State tracking

Circuit convention. We use polynomial-size Boolean circuit families with no uniformity condition. AC^0 permits constant depth and unbounded-fan-in AND/OR gates; TC^0 also permits majority gates. NC^1 uses bounded-fan-in gates and depth $O(\log T)$. All allow NOT gates, and $\mathrm{AC}^0 \subseteq \mathrm{TC}^0 \subseteq \mathrm{NC}^1$. Completeness is with respect to AC^0 many-one reductions, which may increase the input length polynomially. The A_5 word problem is NC^1-complete (Barrington, 1989). The diagonal-SSM comparison uses Merrill et al. (2024, Theorem 4.4)'s logarithmic-precision arithmetic model. The construction below states separately how mantissa precision and exponent range grow with sequence length.

Proof of Theorem 4. Let $\varrho : G \hookrightarrow \mathrm{SO}(3)$ be faithful. Choose a unit vector $\mathbf{C}$ on none of the axes of the finitely many nonidentity rotations in $\varrho(G)$. Then the unit vectors $\mathbf{v}_g = \varrho(g)^{\mathsf{T}}\mathbf{C}$, $g \in G$, are distinct: equality for g, h would make $\varrho(hg^{-1})$ fix $\mathbf{C}$. For nontrivial G, their minimum distance $\delta = \min_{g \neq h} \|\mathbf{v}_g - \mathbf{v}_h\|_2$ is positive.

Set $p = 3$, $m = 1$, $\mathbf{S}_0 = \mathbf{I}_3$, and $\mathbf{u}_t = \mathbf{0}$. For token g_t, use $\mathbf{R}_{t-1} = \varrho(g_t)$ and $\alpha = 3/4$. The recurrence gives

$$\mathbf{S}_t = \alpha^t \varrho(g_t \cdots g_1)^{\mathsf{T}}, \qquad \mathbf{y}_t = \mathbf{S}_t \mathbf{C} = \alpha^t \mathbf{v}_{g_t \cdots g_1}. \tag{18}$$

Let $\mathbf{W}$ have rows $\mathbf{v}_g^{\mathsf{T}}$. The correct score is α^t and every other score is at most $\alpha^t(1 - \delta^2/2)$, proving the claim in exact arithmetic. The trivial group requires only a constant label.

Layer parameters. One-hot token embeddings let the control projection select a finite table of axes and principal angles. The constant $\gamma = \log(4/3) \in (0, 1/2)$ is admissible in Eq. (20). Zero value projections and zero value-convolution bias eliminate both writes. Set the readout convolution to the identity and every readout-projection column to $\mathbf{C}$; its unit normalization leaves it unchanged. The cyclic initialization in Appendix C.3 gives $\mathbf{S}_0 = \mathbf{I}_3$ when $p = 3, m = 1$. Unit RMS weights and a positive constant output gate multiply $\mathbf{y}_t$ by a positive scalar. The output projection retains its three coordinates, so the same linear classifier also works after these operations. All parameter choices are independent of T.

Precision. Suppose each matrix update incurs error $\|\mathbf{E}_t\|_F \le \eta\alpha\|\widetilde{\mathbf{S}}_{t-1}\|_F$, where $\eta = O(2^{-b})$ for b mantissa bits, including errors in the rotation coefficients. With $\mathbf{X}_t = \alpha^{-t}\widetilde{\mathbf{S}}_t$, orthogonality gives $\|\mathbf{X}_t\|_F \le \sqrt{3}(1+\eta)^t$. Writing $e_t = \|\mathbf{X}_t - \varrho(g_t \cdots g_1)^{\mathsf{T}}\|_F$,

$$e_t \le e_{t-1} + \sqrt{3}\eta(1+\eta)^{t-1}, \qquad e_t \le \sqrt{3}\big((1+\eta)^t - 1\big).$$

For a binary product tree, assume operator-norm error at most $\eta\|A\|_2\|B\|_2$ per multiplication and normalized leaf error at most η. At a node, submultiplicativity gives $1 + e \le (1+\eta)(1+e_{\text{left}})(1+e_{\text{right}})$. A prefix of t leaves has $t-1$ internal nodes, so its Frobenius error is at most $\sqrt{3}((1+\eta)^{2t-1} - 1)$. Thus $b = O(\log(T+1))$ keeps both serial and scanned products within any fixed tolerance through length T.

For a unit vector $\mathbf{v}$, $\|\mathbf{e}\| \le \tau < 1/2$, and $\kappa > 0$, RMS normalization of $\kappa(\mathbf{v}+\mathbf{e})$, divided by $\kappa/\sqrt{\kappa^2/3 + \epsilon}$, differs from $\mathbf{v}$ by at most 4τ. Indeed, the ratio of normalization denominators lies between 1 and $1/\|\mathbf{v}+\mathbf{e}\|$. Choose τ small relative to $\delta^2/2$; errors in the fixed query, gate, and classifier then preserve the positive score margin. Finite rotation groups admit algebraic matrix representatives (Olive, 2019, Section 3), so the finite table of controls can be evaluated to the required precision. The factors α^t and α^{2t} need exponent magnitudes $O(T)$, hence $O(\log(T+1))$ exponent bits to avoid underflow. Both mantissa precision and exponent range grow with T. □

C Model and implementation details

C.1 Learned parameterization

A Caracal layer has H heads, each with p value channels and m three-dimensional state blocks per channel. We describe one head and suppress its index. We set $h = 1$ and select the coefficients on $[t_n, t_{n+1}]$ from the current token embedding $\mathbf{e}_{n+1} \in \mathbb{R}^d$. A learned projection $\mathbf{W} \in \mathbb{R}^{6\times d}$ gives a three-dimensional axis and three scalar controls:

$$(\mathbf{a}_n^{\mathsf{T}}, d_n, r_n, g_n)^{\mathsf{T}} = \mathbf{W}\mathbf{e}_{n+1} + (\mathbf{a}_*^{\mathsf{T}}, d_*, 0, g_*)^{\mathsf{T}}. \tag{19}$$

Here $\mathbf{a}_* \in \mathbb{R}^3$, d_*, and g_* are learned biases. With a learned log-frequency $\nu \in \mathbb{R}$, we define

$$\begin{aligned} \theta_n &= \operatorname{wrap}_\pi(e^\nu(1+r_n)), \quad \boldsymbol{\omega}_n = \theta_n \frac{\mathbf{a}_n}{\|\mathbf{a}_n\|_2}, \\ \gamma_n &= \gamma_{\max}\sigma(d_n), \qquad\qquad \rho_n = \rho_{\max}\sigma(g_n). \end{aligned} \tag{20}$$

The function σ is the logistic sigmoid, and wrap_π selects a principal angle in $[-\pi, \pi]$, preserving the sign of an existing half-turn. The implemented map is piecewise differentiable. FOH integration depends on the selected generator: its input contribution can differ across the $\pm\pi$ branch cut even when the endpoint rotations agree. Axis normalization is regularized near zero.[3] We use $\gamma_{\max} = \rho_{\max} = 1/2$. All state blocks and value channels in the head share the generator $\mathbf{A}_n = -\gamma_n\mathbf{I}_3 + [\boldsymbol{\omega}_n]_\times$. Each value channel therefore has transition $\operatorname{diag}(e^{\mathbf{A}_n}, \ldots, e^{\mathbf{A}_n})$.

The gain ρ_n multiplies both endpoint contributions in Eq. (7) and is selected independently of decay. The readout uses a learned skip scalar D for each head.

C.2 Architecture

The mixer maps token embeddings in $\mathbb{R}^d$ to outputs of the same dimension. Its internal width is $d_{\text{inner}} = Hp$. Let G divide H; each group of H/G heads shares its keys and queries. Bias-free projections produce value and gate vectors in $\mathbb{R}^{Hp}$ and keys and queries in $\mathbb{R}^{3mG}$:

$$\mathbf{v}_n = \mathbf{W}_u\mathbf{e}_n, \qquad \mathbf{z}_n = \mathbf{W}_z\mathbf{e}_n, \qquad \mathbf{B}_n^{\text{proj}} = \mathbf{W}_b\mathbf{e}_n, \qquad \mathbf{C}_n^{\text{proj}} = \mathbf{W}_c\mathbf{e}_n. \tag{21}$$

Each stream except the gate passes through a separate learned causal depthwise convolution of width w, with zero padding before the first token. Only the value branch has a convolution bias and an

[3] The implementation uses $\varepsilon = 10^{-6}$ and $s_n = \max\{\|\mathbf{a}_n\|_2^2, \varepsilon^2\}$, replacing the denominator by $\sqrt{s_n + (\|\mathbf{a}_n\|_2^2 - s_n)(\theta_n/(2\pi))^2}$. This equals $\|\mathbf{a}_n\|_2$ whenever $\|\mathbf{a}_n\|_2 \ge \varepsilon$ and gives $\boldsymbol{\omega}_n = \mathbf{0}$ at $\mathbf{a}_n = \mathbf{0}$.

activation, giving $\mathbf{u}_n = \mathrm{SiLU}(\mathrm{Conv}_u(\mathbf{v})_n)$. The key and query branches have neither biases nor activations; we divide each group's output by its Euclidean norm floored at 10^{-12}. Values and gates are split into H heads of p coordinates. The transition controls remain head-specific and bypass convolution.

For each head, let $\mathbf{y}_n \in \mathbb{R}^p$ denote the linear readout including the skip term in Eq. (7). The head output is

$$\mathbf{o}_n = \left(\boldsymbol{\alpha} \odot \frac{\mathbf{y}_n}{\sqrt{p^{-1}\|\mathbf{y}_n\|_2^2 + \epsilon}} \right) \odot \mathrm{SiLU}(\mathbf{z}_n), \qquad \epsilon = 10^{-5}, \tag{22}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^p$ is learned per head. A bias-free $\mathbf{W}_{\mathrm{out}} \in \mathbb{R}^{d \times Hp}$ projects the concatenated head outputs back to the embedding width. CARACAL-Δ uses the same surrounding operations.

C.3 PARAMETER INITIALIZATION

The mixer uses the following initialization. All parameters remain trainable.

Projections and convolutions. The entries of $\mathbf{W}_u$, $\mathbf{W}_z$, $\mathbf{W}_b$, and $\mathbf{W}_c$ are initially sampled independently from $\mathcal{U}(-d^{-1/2}, d^{-1/2})$. Each row of $\mathbf{W}_b$ and $\mathbf{W}_c$ is then rescaled to Euclidean norm $(3m)^{-1/2}$. The transition-control matrix $\mathbf{W}$ in Eq. (19) is initialized to zero. The entries of $\mathbf{W}_{\mathrm{out}}$ are sampled independently from $\mathcal{U}(-(Hp)^{-1/2}, (Hp)^{-1/2})$. Value-convolution coefficients are sampled independently from $\mathcal{U}(-w^{-1/2}, w^{-1/2})$, with zero bias. Key and query convolutions start as identity maps: their current-token coefficient is one and all older-token coefficients are zero. The skip scalar D and every RMS normalization weight are initialized to one.

Rotation axes. For head $a \in \{1, \ldots, H\}$, let $\zeta_a = 1 - 2(a - 1/2)/H$ and $\psi_a = (a - 1/2)\pi(3 - \sqrt{5})$. The initial axis $\mathbf{a}_*^{(a)} = (\sqrt{1-\zeta_a^2}\cos\psi_a, \sqrt{1-\zeta_a^2}\sin\psi_a, \zeta_a)^\mathsf{T}$ distributes heads over the unit sphere.

Decay horizons and periods. Each head is assigned a decay horizon $\tau_a \in [4, 4096]$ and an oscillation period $P_a \in [4, 256]$, measured in token intervals. The initial parameters are

$$d_*^{(a)} = g_*^{(a)} = \mathrm{logit}\left(\frac{2}{\tau_a}\right), \qquad \nu^{(a)} = \log\frac{2\pi}{P_a}. \tag{23}$$

Together with zero transition-control weights, these give $\gamma_n = \rho_n = 1/\tau_a$ and $\theta_n = 2\pi/P_a$ initially. The decay and write biases are separate parameters despite their equal initial values.

For $H \geq 4$, let $r = \lfloor\sqrt{H}\rfloor$ and $c = \lceil H/r \rceil$. Form the logarithmic grids

$$\tau_i = 4\left(\frac{4096}{4}\right)^{i/(r-1)}, \quad 0 \leq i < r, \qquad P_j = 4\left(\frac{256}{4}\right)^{j/(c-1)}, \quad 0 \leq j < c.$$

Enumerate the pairs (τ_i, P_j) in increasing j, taking i in ascending order for even j and descending order for odd j, and retain the first H pairs. For $H = 2$ or 3, use H logarithmically spaced horizons and periods over the same ranges and circularly shift the period list one position to the right. For $H = 1$, use $(\tau_1, P_1) = (128, 32)$.

Initial recurrent state. For head a and value channel $j \in \{1, \ldots, p\}$, the concatenated state $\mathbf{S}_0^{(a,j)} \in \mathbb{R}^{3m}$ is the coordinate vector with its one at position $1 + (((a-1)p + j - 1) \bmod (3m))$. Previous values, keys, and all convolution histories are zero. CARACAL-Δ initializes its gate-projection weights and bias to zero, giving a gate of $1/2$ for each processed token; its stored previous gate and endpoint write vector are initially zero.

C.4 CARACAL-Δ

The variant applies prediction-error editing after the rotation. Use the endpoint vectors $\mathbf{h}_0 = \mathbf{0}$ and $\mathbf{h}_{n+1} = \mathbf{F}_n^+ \mathbf{B}_{n+1}$ from Proposition 3, and subtract the current endpoint to define $\widetilde{\mathbf{S}}_n = \mathbf{S}_n - \mathbf{u}_n \mathbf{h}_n^\mathsf{T}$.

The base recurrence writes along $\mathbf{F}_n^- \mathbf{B}_n + \mathbf{M}_n \mathbf{h}_n$. We normalize this direction to obtain the delta-rule key:

$$\mathbf{k}_n = \frac{\mathbf{F}_n^- \mathbf{B}_n + \mathbf{M}_n \mathbf{h}_n}{\max\{\|\mathbf{F}_n^- \mathbf{B}_n + \mathbf{M}_n \mathbf{h}_n\|_2, 10^{-12}\}}. \tag{24}$$

Keeping the row-state convention of the main text, the update is

$$\begin{aligned} \overline{\mathbf{S}}_{n+1} &= \widetilde{\mathbf{S}}_n \mathbf{M}_n^{\mathsf{T}}, \\ \widetilde{\mathbf{S}}_{n+1} &= \overline{\mathbf{S}}_{n+1} + \beta_n \left(\mathbf{u}_n - \overline{\mathbf{S}}_{n+1} \mathbf{k}_n \right) \mathbf{k}_n^{\mathsf{T}}. \end{aligned} \tag{25}$$

Here $\beta_n = \sigma(\mathbf{w}_\beta^{\mathsf{T}} \mathbf{e}_n + b_\beta)$, with a learned projection and bias for each head. The correction uses the previous token's value and gate. The current endpoint enters the readout directly:

$$\mathbf{y}_{n+1} = \widetilde{\mathbf{S}}_{n+1} \mathbf{C}_{n+1} + \left(D + \mathbf{C}_{n+1}^{\mathsf{T}} \mathbf{h}_{n+1} \right) \mathbf{u}_{n+1}. \tag{26}$$

For the first interval, $\mathbf{h}_0 = \mathbf{0}$ and $\beta_0 = 0$; both models use the same coordinate initialization. Normalization, gating, and output projection follow Appendix C.2. The effective transition on each transposed state row is $(\mathbf{I}_{3m} - \beta_n \mathbf{k}_n \mathbf{k}_n^{\mathsf{T}}) \mathbf{M}_n$; the rank-one correction can couple distinct three-dimensional blocks. The same quaternion-prefix change of coordinates reduces this to a recurrence with scalar decay and a delta-rule update, which we evaluate with a delta-rule kernel.

C.5 Fused execution and numerical precision

The default CuTe DSL kernels use $(L, p, d_s) = (64, 64, 96)$ and eight warps per block, with 64 KiB of shared memory in forward and 47.8 KiB in local backward. One block processes a head's chunk; one warp prepares the quaternion and damping prefixes. The block publishes its successor state before finishing the readouts, overlapping the next chunk's transfer with local work. Blocks acquire chunks in order; device-scope release/acquire flags synchronize transfers. Every invocation resets the flags, including graph replay. The surrounding layer applies the skip, normalization, gating, and projections separately.

Backward reuses its causal cotangent matrix across three panels of 32 state coordinates. It stores ten lower-triangular 16×16 tiles and retains output accumulators within each warp (Dao, 2023). Boundary states and increments are saved; local causal matrices and rotated operands are recomputed. Other supported dimensions use separate increment, state-transfer, and readout stages.

Mixed precision. Rotation and integration coefficients, recurrent states, state adjoints, and accumulators remain in FP32. An FP32 contraction operand uses a leading BF16 value X_{h} and residual $X_{\mathrm{l}} = \mathrm{RN}_{\mathrm{BF16}}(X - X_{\mathrm{h}})$. The kernels accumulate $XZ \approx X_{\mathrm{h}} Z_{\mathrm{h}} + X_{\mathrm{h}} Z_{\mathrm{l}} + X_{\mathrm{l}} Z_{\mathrm{h}}$ in FP32; a BF16 input needs no residual. Outputs and input cotangents are rounded to their activation dtype.

Coefficient evaluation. For $0 \le \gamma, \rho \le 1/2$ and $\|\boldsymbol{\omega}\| \le \pi$, the fused coefficient kernels use degree-nine polynomials in $\|\boldsymbol{\omega}\|^2$ and moments $M_j = \int_0^1 r^j e^{-\gamma r} \, \mathrm{d}r$. A positive eight-term series seeds M_{21}; the recurrence $M_{j-1} = (e^{-\gamma} + \gamma M_j)/j$ supplies the lower moments. Backward uses $\partial_\gamma M_j = -M_{j+1}$ and one additional moment. These finite polynomial and moment expansions numerically approximate the analytic FOH operators in Appendix B.2 without dividing by small angles or damping. The quaternion exponential also uses degree-nine polynomials, with completed prefixes normalized in FP32 and differentiated as in Appendix B.3. Normalization controls radial drift; angular rounding error remains (Brown, 1989). Damping weights use nonpositive differences of chunk-local prefix sums, avoiding division by accumulated decay.

C.6 Decoding and validation

The decoder applies Eq. (7) with state triples held in registers and FP32 updates and reductions. Its cache contains the physical states, preceding processed keys and values, and the last $w - 1$ raw convolution inputs. Processed keys supply the next interval's $\mathbf{F}_n^-$ write; raw history reconstructs convolution windows. Recurrent work is $\Theta(Hpd_s)$ per token with a cache independent of prefix length. This decoder is inference-only; gradient-enabled calls use the sequence path.

We compare outputs, terminal states, and all recurrence-input gradients against an independent FP64 serial reference, including nonzero incoming states and endpoints. Checks on RTX 3060

and H200 cover lengths $1, 63, 65, 129, 512$, four heads, and one or two address groups. After rounding the reference to the result dtype, the acceptance thresholds for relative RMS and peak error are $(6\times10^{-4}, 2\times10^{-3})$ for FP32 tensors and $(5\times10^{-3}, 3\times10^{-2})$ for BF16 tensors. Both paths receive identical prepared integration coefficients, so these checks assess the recurrence and backward after coefficient preparation. Changing-input graph replays test synchronization and state reuse; continuation checks include the cache as well as outputs. Appendix D.4 gives the timing protocol and performance results.

D Experimental setup and additional results

The four subsections follow the main experiments: state tracking, MAD, language modeling, and kernel performance. Unless specified otherwise, we select checkpoints on validation data and report test means and sample standard deviations across training seeds. Timing intervals summarize repeated measurements of a fixed operator. Tables bold the best mean and underline the second best, including ties at displayed precision; lower is better for perplexity. Per-seed rankings compare the same seed, and the main FSA table ranks parallel models separately from LSTM.

D.1 State tracking

D.1.1 A_5 depth benchmark

We train Caracal following Walker et al. (2025). Each input symbol is one of the 60 elements of A_5, and the target at position t is the ordered product of the first t elements. Figure 2 reports the minimum depth at the 90% accuracy threshold. The Transformer, Mamba, and RNN results are from Merrill et al. (2024); the KLA results are from Shaj et al. (2026).

For Caracal, each length from 3 to 20 has a separate training and validation split. We use Walker's 80% training and 20% validation split, with partition seed 1234. At every optimization step, the batch contains 256 examples of the chosen length and 25 auxiliary examples of length two. We train with AdamW, a 10,000-step linear warmup, cosine decay from 10^{-3} to 10^{-5}, weight decay 0.01, and dropout 0.1, for at most 100,000 steps. Embeddings are exempt from weight decay. Training uses FP32; validation runs every 10,000 steps.

The model uses hidden width 128, 48 state triples per head, head width 32, expansion two, eight input groups, and convolution width four. For each length and seed in $\{0, 1, 2, 3, 4\}$, we test depths $1, 2, 3, 4$ in order and stop at the first whose validation token accuracy exceeds 90%. The plotted depth is the minimum across the five seeds. A single layer succeeds in three to five seeds at every length; both the minimum and median selected depth are one throughout.

D.1.2 A_5 length generalization

For length generalization, we train Caracal, BD-SLiCE, sLSTM, and GDN2 following Walker et al. (2025). For each model and seed, we select the smallest depth exceeding 90% validation token accuracy at length 20, then train a new model on sequences of lengths 3–40. Caracal uses the configuration above; BD-SLiCE uses width 256 and block size four, and sLSTM and GDN2 use width 1,024. The validation set contains 8,192 examples with lengths 40–128. Validation runs every 5,000 steps, and training stops after two consecutive validation accuracies above 99.95% or at 100,000 steps. Other optimizer settings match the fixed-length experiment. We test each selected model on 2,048 independently generated examples at each length in Table 4. Figure 4 and Table 4 report means and standard deviations over three seeds per model.

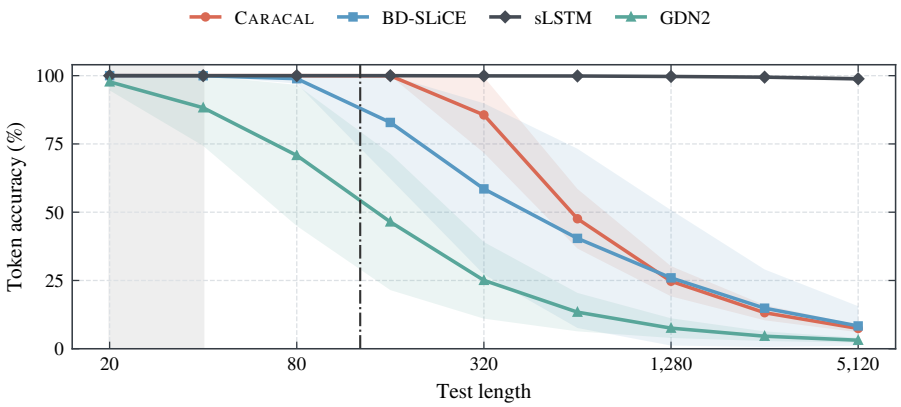

Figure 4: A_5 **length generalization.** Test token accuracy after training on lengths 3–40 (shaded); the dash-dotted line marks the largest validation length, 128. Means and ± one SD are computed over three seeds per model.

Table 4: A_5 **length generalization.** Mean ± SD over three seeds per model for selected depth and test token accuracy (%). Numeric column headings give test lengths. Training lengths are 3–40 and validation lengths are 40–128.

Model	Depth	20	40	80	160
CARACAL	1.0 ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.1
BD-SLiCE	1.0 ± 0.0	**100.0** ± 0.0	**100.0** ± 0.1	98.9 ± 1.8	82.9 ± 20.6
sLSTM	1.0 ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0
GDN2	2.3 ± 0.6	97.7 ± 3.1	88.2 ± 14.0	70.8 ± 25.8	46.4 ± 25.0
Model	**320**	**640**	**1280**	**2560**	**5120**
CARACAL	85.6 ± 13.9	47.6 ± 10.9	24.7 ± 5.5	13.2 ± 2.8	7.4 ± 1.4
BD-SLiCE	58.5 ± 31.3	40.4 ± 32.8	26.0 ± 24.8	14.9 ± 14.2	8.3 ± 7.2
sLSTM	**99.9** ± 0.1	**99.9** ± 0.1	**99.7** ± 0.2	**99.4** ± 0.5	**98.8** ± 0.9
GDN2	25.0 ± 14.0	13.4 ± 7.0	7.5 ± 3.6	4.6 ± 1.8	3.1 ± 0.9

D.1.3 FINITE-STATE AUTOMATON EMULATION

We use four tasks from Delétang et al. (2023), following Walker et al. (2025). Cycle navigation predicts the final position of a walk on a five-state ring; even pairs predicts the parity of changes between adjacent binary symbols. Modular arithmetic evaluates unbracketed expressions modulo five, and parity counts ones modulo two. These tasks score only the final prediction. Models use two layers and train on lengths 3–40. Validation and test each contain 8,192 examples with lengths 40–256. We use batch size 256, AdamW with peak learning rate 2×10^{-3}, 10,000 warmup steps, cosine decay to 10^{-5}, weight decay 0.01, and dropout 0.01. Training ends after two consecutive validation checks above 99.95% or 100,000 updates; checks occur every 5,000 updates. Models use hidden width 128, except Mamba-2 (256) and SLiCE-DPLR (57, rank four). The CARACAL configuration uses 48 state triples per head, head width 32, eight input groups, expansion two, and convolution width four. The four-task average in Table 1 first averages tasks within each seed, then averages across seeds. Gated DeltaProduct uses two Householder factors per token, each with a nontrivial eigenvalue in $[-1, 1]$.

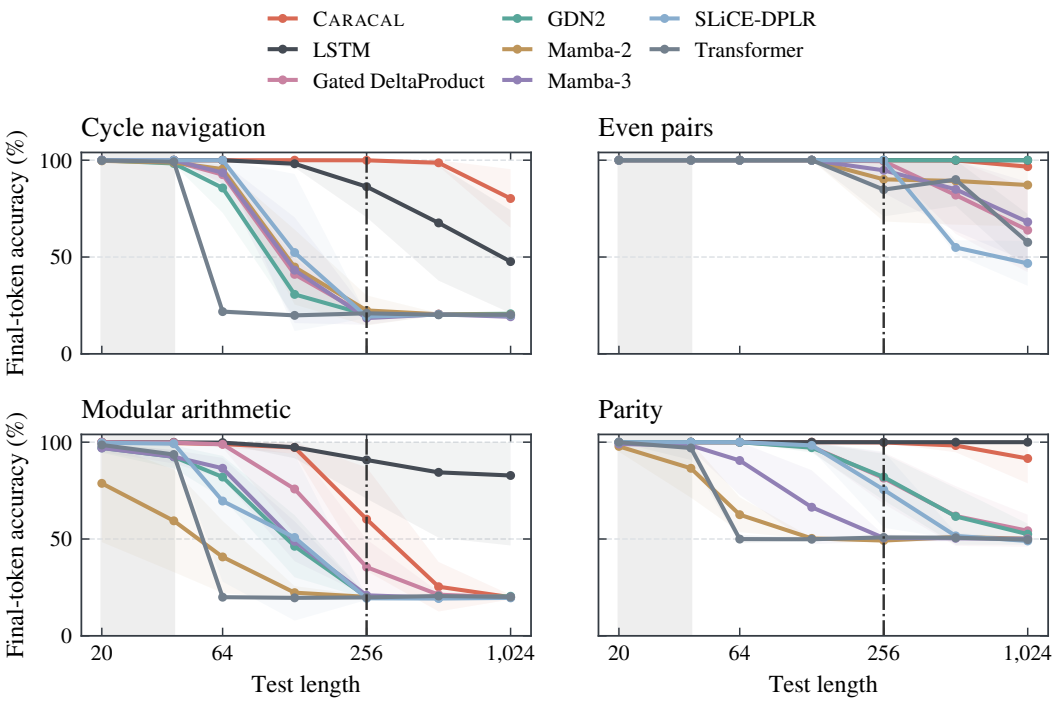

Figure 5: **FSA emulation length generalization.** Final-token test accuracy on 2,048 examples at each length, using the validation-selected checkpoint. Means and ± one SD over five seeds. Shading marks the training range; the dash-dotted line marks the largest validation length.

D.2 ASSOCIATIVE MEMORY ON MAD

Tasks and data. We use the six MAD tasks of Poli et al. (2024) at the settings in Table 5. Each seed determines independent training, validation, and test instances, shared across models. All tasks use 12,800 training examples except memorization, which uses 256; validation and test contain 1,280 examples each. Accuracy is the fraction of correctly predicted scored targets under the task's mask. The compression task reconstructs the input from the final memory through a common task-specific decoder. For fuzzy recall, we reject and resample examples whose final complete query key has no earlier complete binding. Selective-copy distractor positions use a per-example random stream. These choices hold across every model and capacity setting.

Table 5: **MAD task settings.** Recall lengths are input lengths after shifting targets. Noisy recall adds 16 distractor symbols to the 16-symbol vocabulary. Fuzzy recall uses key and value motifs of length three.

Task	Length	Vocabulary	Scored prediction
Compression	32	16	Reconstruct the input from the final state
Memorization	32	256	Recall a dictionary shared across examples
Contextual recall	127	16	Retrieve bindings introduced in context
Noisy recall	127	32	Retrieve bindings with 20% distractors
Fuzzy recall	128	16	Retrieve values for multi-token keys
Selective copying	256	16	Copy 16 tokens past inserted blanks

Architecture and optimization. Each model uses one width-128 layer with the same embedding and readout architecture. CARACAL and CARACAL-Δ use 48 state triples, head width 32, expansion two, and four input groups. The default GDN2 has four 32×32 heads; Mamba-3 uses its SISO block with state size 128 and head width 32. KLA uses state size eight and retains its native SwiGLU channel MLP. All models use FP32 AdamW at learning rate 10^{-3}, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, zero

weight decay, batch size 172, and gradient clipping at five. We validate after each epoch and stop at 750 epochs, after 70 epochs without higher validation accuracy, or after two consecutive epochs at or above 99.9%. We select the highest validation accuracy, breaking ties by lower validation cross-entropy. The main comparison uses seeds 12345–12349.

Table 6: **Full MAD results.** Test accuracy (%), mean ± SD over five seeds. Columns follow Table 5.

Model	Comp.	Mem.	Ctx.	Noisy	Fuzzy	Copy
Mamba-3	78.5 ± 4.8	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	43.6 ± 3.3	26.1 ± 2.3
GDN2	88.3 ± 0.9	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	**65.0** ± 11.2	86.1 ± 1.4
KLA	80.8 ± 1.6	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	33.5 ± 10.8	90.4 ± 0.7
CARACAL	90.7 ± 1.5	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	48.0 ± 4.9	78.6 ± 4.7
CARACAL-Δ	**94.0** ± 0.7	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	45.8 ± 8.0	**98.9** ± 0.8

Capacity controls. We vary CARACAL-Δ's number of state triples over $5, 16, 32, 48$ and GDN2's head configuration over 4×32, 3×64, 6×64, and 9×64. KLA additionally uses an expanded configuration of width 288 inside the mixer and state size 64; the residual width remains 128. The added configurations use seeds 12345–12347 on compression, fuzzy recall, and selective copying, with the same data and optimization recipe. The smallest CARACAL-Δ configuration has 122.9K parameters, excluding the task embedding and output decoder, compared with 116.5K for the default GDN2; both counts include the models' native channel MLPs. It retains the compression and selective-copying gains. Increasing GDN2's head size improves fuzzy recall (Table 7).

Table 7: **MAD capacity comparison.** Test accuracy (%), mean ± SD. Each row's seed count applies to all three tasks. m denotes state triples per head; GDN2 configurations give the number of heads and head dimension, and KLA gives mixer width and state size.

Configuration	Seeds	Compression	Fuzzy recall	Selective copy
CARACAL-Δ, $m = 5$	3	93.96 ± 1.13	32.47 ± 4.65	98.79 ± 0.66
CARACAL-Δ, $m = 16$	3	93.84 ± 0.18	40.50 ± 9.74	98.46 ± 0.84
CARACAL-Δ, $m = 32$	3	**94.28** ± 0.02	56.76 ± 11.45	**99.24** ± 0.73
CARACAL-Δ, $m = 48$	5	94.00 ± 0.66	45.83 ± 7.96	98.90 ± 0.81
CARACAL	5	90.68 ± 1.45	48.01 ± 4.95	78.64 ± 4.66
GDN2	5	88.26 ± 0.87	64.96 ± 11.24	86.15 ± 1.39
GDN2, 3×64	3	88.37 ± 0.69	**94.16** ± 1.70	86.89 ± 0.70
GDN2, 6×64	3	88.64 ± 0.56	92.67 ± 6.50	88.76 ± 5.06
GDN2, 9×64	3	87.97 ± 0.51	85.71 ± 9.60	91.29 ± 1.35
KLA	5	80.83 ± 1.55	33.48 ± 10.82	90.37 ± 0.73
KLA, 288×64	3	80.28 ± 3.84	61.89 ± 3.36	90.06 ± 1.00
Mamba-3	5	78.53 ± 4.77	43.63 ± 3.30	26.13 ± 2.32

D.3 LANGUAGE MODELING

Data and training budget. FineWeb-Edu documents (Penedo et al., 2024) are deduplicated and assigned to disjoint splits by a hash of their text. Documents matching the benchmark exclusion list are removed before corpus construction and tokenizer fitting. We fit a 32,768-token BPE tokenizer on the training split using the `nanochat` tokenizer implementation (Karpathy, 2025). Each document starts with a BOS token; documents are packed into independent 2,048-token windows. Validation contains 4,194,304 tokens and test contains 8,388,608 tokens. At 45M parameters, training uses 13,733 updates of 131,072 tokens; at the larger scale, it uses 20,790 updates of 524,288 tokens. Models share the training-window permutation for a given seed. We use seeds 0–2 at 45M and seed zero at the larger scale.

Architectures. CARACAL, Mamba-3, and GDN2 stack twelve recurrent layers at width 448 or 1,152, with untied token embeddings and output heads. CARACAL uses 32 state triples per head, head width 64, one input group, expansion two, and a width-four convolution. Mamba-3 uses its SISO implementation with state size 96 and head width 64. GDN2 uses seven or eighteen 64×64

heads. KLA retains its native channel MLP; its widths are 496 and 1,360, with MLP hidden widths 128 and 320 and state size 32. The Transformer uses seven attention–MLP layers, rotary position embeddings (Su et al., 2024), head width 64, and MLP widths 1,632 or 3,920. Table 8 reports total and non-embedding parameter counts. These configurations keep total parameter counts comparable while preserving each architecture's form.

Table 8: **Language-model parameter counts.** Non-embedding parameters exclude both the input embedding and output head.

Scale	Model	Total parameters	Non-embedding
45M	Mamba-3	45,238,544	15,878,416
45M	GDN2	45,269,012	15,908,884
45M	KLA	45,386,492	12,880,636
45M	Transformer	45,222,464	15,862,336
45M	CARACAL	45,375,320	16,015,192
175–180M	Mamba-3	175,629,024	100,131,552
175–180M	GDN2	174,797,400	99,299,928
175–180M	KLA	179,673,692	90,544,732
175–180M	Transformer	175,895,424	100,397,952
175–180M	CARACAL	176,882,256	101,384,784

Optimization. We use AdamW (Loshchilov & Hutter, 2019) with $\beta = (0.8, 0.95)$, $\epsilon = 10^{-10}$, weight decay 0.1, gradient clipping at three, and FP32 training. Embeddings, biases, normalization parameters, and recurrence parameters are exempt from weight decay. Rates remain constant for the first 60% of updates and decay linearly to zero over the remaining 40%, without warmup. Table 9 gives the base hidden and embedding rates and chosen multiplier m. CARACAL's peak rates are $m(\eta_h, \eta_e)$. For the baselines, they are $m\sqrt{768/d}\sqrt{B/524288}(\eta_h, \eta_e)$, where d is hidden width and B is the number of training tokens per update.

Table 9: **LM architectures and learning rates.** The width and batch scaling of the nominal rates is specified in the text. The larger model sizes range from approximately 175M to 180M parameters.

Scale	Model	Layers	Width	η_h	η_e	Multiplier
45M	Mamba-3	12	448	0.004	0.3	0.5
45M	GDN2	12	448	0.004	0.3	0.5
45M	KLA	12	496	0.004	0.3	0.5
45M	Transformer	7	448	0.004	0.3	0.5
45M	CARACAL	12	448	0.001	0.05	2.0
175–180M	Mamba-3	12	1152	0.004	0.3	0.5
175–180M	GDN2	12	1152	0.004	0.3	0.5
175–180M	KLA	12	1360	0.004	0.3	0.5
175–180M	Transformer	7	1152	0.004	0.3	1.0
175–180M	CARACAL	12	1152	0.001	0.05	1.0

Evaluation. We report test perplexity at the final training checkpoint. Zero-shot accuracy uses the checkpoint with the lowest validation negative log-likelihood. Validation starts after 13,107,200 training tokens, then runs at multiples of 67,108,864 tokens and at the final update. We use the full evaluation sets of LAMBADA (OpenAI), HellaSwag, PIQA, ARC-Easy, ARC-Challenge, WinoGrande, OpenBookQA, and BoolQ. The evaluation harness (Gao et al., 2024) is version 0.4.14.dev0, revision `d6de81643928`. We use length-normalized accuracy for HellaSwag and ARC-Challenge and accuracy for the other six tasks. The eight-task average weights the tasks equally within each seed. We report three-seed means at 45M. Table 10 gives the individual results and their means and standard deviations.

Table 10: **Language-model results at 45M by seed.** Test perplexity and mean eight-task accuracy, with three-seed means ± SD. Rankings compare models within each column.

	Test PPL ↓				Eight-task average ↑			
Model	Seed 0	Seed 1	Seed 2	Mean ± SD	Seed 0	Seed 1	Seed 2	Mean ± SD
Mamba-3	29.43	29.38	29.74	29.52 ± 0.20	**38.28**	38.00	**38.87**	**38.38** ± 0.45
GDN2	30.78	30.79	30.36	30.65 ± 0.24	37.92	36.36	37.97	37.42 ± 0.92
KLA	31.34	31.16	31.51	31.34 ± 0.17	37.60	37.05	37.87	37.51 ± 0.41
Transformer	30.63	30.52	30.53	30.56 ± 0.06	38.27	37.29	37.70	37.75 ± 0.49
CARACAL	**29.11**	**29.02**	**29.00**	**29.04** ± 0.06	38.05	**38.91**	38.13	38.36 ± 0.48

D.4 KERNEL BENCHMARKS

Operators and configurations. We compare SO3SSD with the released operators of Mamba-3, GDN2, Gated DeltaProduct, KLA, RWKV-7, and FlashAttention-2 (Lahoti et al., 2026; Hatamizadeh et al., 2026; Siems et al., 2025; Shaj et al., 2026; Peng et al., 2025; Dao, 2023). All measurements use the same physical NVIDIA H200 and total value width 2,048 (Table 11). CARACAL uses 96 real state coordinates per value; recurrent baselines use native sizes of 64 and 128. These bracket CARACAL without forcing unsupported or padded shapes. The sequence sweep uses batch one, BF16 operands, and lengths $2^9, \ldots, 2^{19}$; Table 12 tests batch eight at 8K tokens. Decoding uses batches 1, 8, 32, and 128, BF16 or FP32 operands, and FP32 recurrent state. These comparisons fix value width, not model quality.

CARACAL and Mamba-3 share keys and queries across heads. Gated DeltaProduct uses two Householder updates per token. KLA uses its released CUDA scan with its fixed prior for sequences and its single-token update for decoding; BF16 inputs are promoted internally to FP32. FlashAttention-2 uses causal attention, with 2,048 preceding tokens in decoding.

Table 11: **Kernel configurations.** Total value width is 2,048. Slash-separated entries correspond to the 64- and 128-coordinate baseline configurations. KLA counts both carried state components.

Operator	Heads	Head dimension	State per value
CARACAL (SO3SSD)	32	64	3×32
Mamba-3 SISO	32	64	64 / 128
GDN2	32 / 16	64 / 128	64 / 128
Gated DeltaProduct	32 / 16	64 / 128	64 / 128
KLA	–	–	2×32 / 2×64
RWKV-7	32 / 16	64 / 128	64 / 128
FlashAttention-2	32	64	Context length

Table 12: **Training at batch eight.** Forward plus backward in milliseconds at 8K tokens and value width 2,048. Columns specify recurrent baseline state coordinates per value; CARACAL uses 96 in both columns.

Operator	State 64	State 128
CARACAL (SO3SSD)	10.40	10.40
Mamba-3	3.56	7.11
GDN2	20.29	23.99
Gated DeltaProduct	15.97	16.58
KLA	164.91	324.86
RWKV-7	30.38	34.44
FA2	25.91	25.91

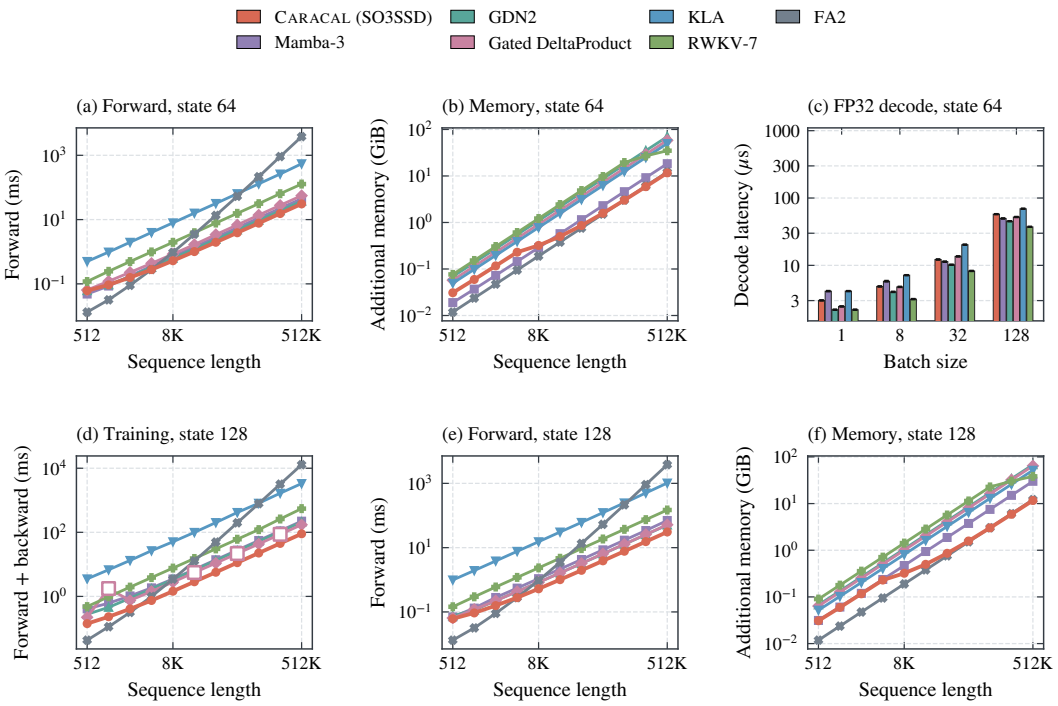

Figure 6: **Additional kernel measurements.** CARACAL uses 96 state coordinates per value throughout. Recurrent baselines use 64 in (a)–(c) and 128 in (d)–(f). Sequence measurements use batch one and BF16 operands; (c) uses FP32 operands. Memory is additional peak training allocation. Medians and uncertainty intervals follow Figure 3. Open squares mark eager GDP timings, which include host overhead. FlashAttention-2 has no FP32-operand decode implementation.

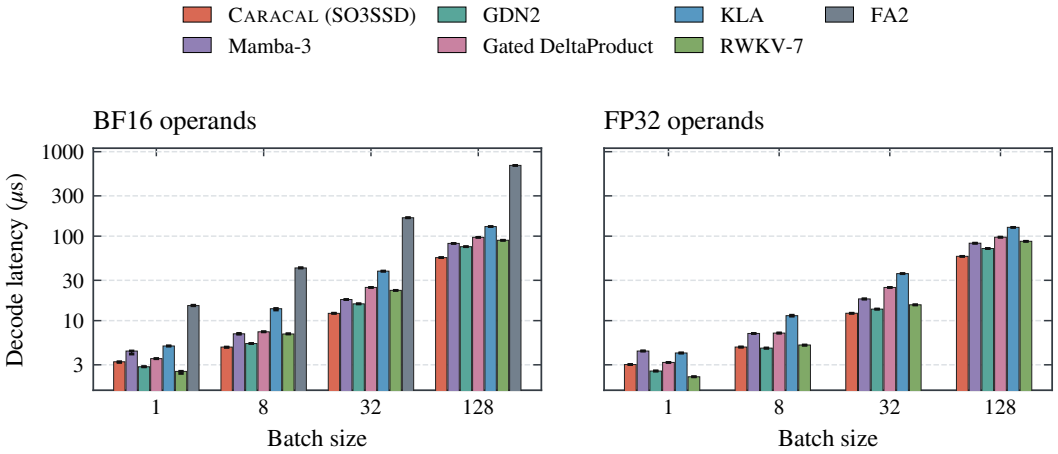

Figure 7: **Decoding with 128-coordinate baselines.** CARACAL retains 96 coordinates per value; recurrent baselines use 128. All recurrent states are FP32. Bars show medians and whiskers the range over four timing blocks. SO3SSD leads at batches 8–128 with BF16 operands and at 32–128 with FP32 operands. The 64-coordinate comparisons appear in Figure 3(b) and Figure 6(c).

Measurement boundary. CARACAL's control transformations, exact FOH preparation, recurrence, and readout are inside the measured call. Training differentiates through all floating temporal inputs and controls. Baselines receive their native inputs, including transformed controls where supported. Dense projections, convolutions, address parameterization and normalization, and output tails are excluded for every model. Initial states are fixed during training. Decoding includes state updates and, for attention, appending the new key and value.

Validated baseline launchers reuse state and output storage where supported. GDP operand packing is performed before decode timing. Mamba-3 retains its phase and rotary work and uses the faster validated launch configuration. KLA is tested eagerly and with whole-call compilation; we report the faster validated configuration. Baseline equations and kernel arithmetic are unchanged.

Timing and memory. Compilation and autotuning precede measurement. CUDA events time captured executions on an otherwise idle GPU. Sequence points contain six timing blocks from one complete sweep; we report medians and interquartile ranges. Decode points contain four randomized-order blocks, each with five event samples over dependent captured calls; we report medians and the range of block medians. Decode measurements use identical implementations on the same day and physical GPU. Host dispatch, graph capture, and optimizer work are outside captured timings.

Memory is measured eagerly before graph capture. Additional peak allocation is the peak above resident temporal inputs, initial states, and output cotangents; it includes preparation intermediates, activations, workspaces, and returned gradients. At 512K tokens, total peak allocation is 16.05 GiB for SO3SSD and 23.19 GiB for 64-coordinate Mamba-3.

Validation and supported configurations. Representative outputs are checked against reference implementations, and all temporal-input and control gradients must be finite. Captured outputs and gradients are compared with eager execution on original and changed inputs. GDP and KLA gradient reductions are also checked against repeated eager execution to distinguish native variability from capture effects. Five GDP training points use eager timing: 16K tokens at state size 64, and 1K, 16K, 64K, and 256K at state size 128. These include host dispatch, are marked by open squares, and are excluded from kernel-speedup claims. All 512K training points use validated graph replay.

RWKV-7 sequences above 128K tokens are partitioned into supported lengths, carrying state and its gradient across boundaries. The adapter is checked against an unpartitioned native call on a supported input. All 288 sequence points and 92 supported decode settings are included. FlashAttention-2 has no native FP32-operand decode kernel.

The environment uses PyTorch 2.9.1, CUDA 12.8, and Triton 3.5.1; FlashAttention-2 is version 2.8.3, FLA is 0.5.1, and KLA is 0.1.0. Mamba-3 decoding uses its CuTe implementation with CUTLASS DSL 4.4.2. SO3SSD measurements use CARACAL's additive recurrence; CARACAL-Δ is evaluated separately on MAD.