Rethinking State Tracking in Recurrent Models Through Error Control Dynamics

Jiwan Chung, Heechan Choi, Seon Joo Kim
Yonsei University
jiwan.chung.research@gmail.com
https://jiwanchung.github.io/rethinking-state-tracking/

Abstract

The theory of state tracking in recurrent architectures has predominantly focused on expressive capacity: whether a fixed architecture can theoretically realize a set of symbolic transition rules. We argue that equally important is error control, the dynamics governing hidden-state drift along the directions that distinguish symbolic states. We prove that affine recurrent networks, a class of models encompassing State-Space Models and Linear Attention, cannot correct errors along state-separating subspaces once they preserve state representations. Consequently, practical affine trackers do not learn robust state tracking; rather, they learn finite-horizon solutions governed by accumulated state-relevant error. We characterize the mechanics of this failure, showing that tracking remains readable only while the accumulating within-class spread remains small relative to the between-class separation. We demonstrate empirically on group state-tracking tasks that this breakdown is predictable: tracking collapses when the distinguishability ratio crosses the readability threshold. Across trained models, the point of this crossing predicts the horizon at which downstream accuracy fails. These results establish that robust state tracking is determined not only by an architecture's theoretical expressivity but crucially by its error control.

1 Introduction

The theory of state tracking in recurrent architectures has been predominantly a theory of expressivity: which symbolic transition rules can a fixed architecture in principle realize [Merrill et al., 2024, Sarrof et al., 2024, Grazzi et al., 2025, Karuvally et al., 2025, Shakerinava et al., 2026]. We argue that equally important is *error control*, the dynamics governing hidden-state drift along the directions that distinguish symbolic states. We prove that affine recurrences, a class that includes State-Space Models (SSMs) [Gu et al., 2022] as well as Linear Attention [Katharopoulos et al., 2020], cannot correct hidden-state drift along state-separating subspaces once they preserve state representations exactly.

In practice the two requirements diverge. Recent literature documents this gap: input-dependent complex-diagonal SSMs sufficient for S_3 at depth two fail to learn solutions that generalize beyond the training length [Shakerinava et al., 2026], and diagonal selective SSM variants can fit regular-language emulation at training lengths while collapsing under length extrapolation [Terzic et al., 2025a]. The same pattern surfaces within an architecture's own claimed task scope: AUSSM, provably sufficient for Abelian groups via unit-modulus rotations [Karuvally et al., 2025, Shakerinava et al., 2026], tracks C_2 and C_6 unevenly in our experiments. Across recurrent architectures developed for long-context sequence modeling [Gu and Dao, 2024, Lahoti et al., 2026, Karuvally et al., 2025], expressive capacity does not predict state-tracking robustness.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

In this work, we study error control as the missing axis in recurrent state tracking. We first show that affine recurrent models cannot correct symbolic-state drift once they preserve state representations (Section 3.1). State-dependent return maps escape this obstruction and can selectively contract symbolic-subspace drift; we verify which canonical activations realize this correction (Appendix E.1). We then characterize the finite horizons that affine trackers sustain without state-dependent correction. Their failures are governed by accumulated state-relevant error: tracking remains readable while within-class spread remains small relative to between-class separation, and breaks down once this ratio crosses the readability threshold (Section 3.2).

We evaluate this account on a set of group state-tracking tasks. Performance exhibits a systematic separation: state-dependent models maintain tracking over the longest tested horizons, whereas affine models lose accuracy at different horizons (Section 4.1). This variation is central to our analysis: affine trackers are not distinguished only by whether they fail, but by how long they sustain tracking under repeated recurrence.

Our diagnostics give a consistent error-dynamics explanation. Perturbation recovery shows that state-dependent models selectively contract injected hidden-state errors, whereas affine models do not (Section 4.2). The absence of selective contraction need not cause immediate failure: the distinguishability ratio $q(t) = R(t)/M(t)$ tracks how affine models gradually exhaust a finite horizon as within-class spread approaches between-class separation (Section 4.3), and subspace decomposition localizes this spread along the symbolic subspace $\mathcal{U}$, where affine return maps cannot contract errors (Section 4.4). The point at which $q(t)$ first crosses the readability threshold, denoted T_{cross}, quantitatively predicts the downstream max-passing length across affine sweeps on S_3 (Figure 4), confirming the finite-horizon mechanism of Corollary 1.

Together, these results establish that robust state tracking is determined not only by an architecture's theoretical expressivity but crucially by its error control dynamics.

2 Background

2.1 Recurrent models

We provide a taxonomy of recurrent models explored in this work, from SSMs to general RNNs as shown in Table 1. We begin by introducing a common recursive form.

Definition 1 (Recursive layer). A d-dimensional recursive layer is a parameterized function that takes as input a sequence $x_t \in \mathcal{X}$ and produces outputs $y_t \in \mathcal{Y}$ via the recurrence

$$h_t = \phi\Big(g(h_{t-1}, x_t) \odot \big(A(x_t)h_{t-1}\big) + b(x_t)\Big), \tag{1}$$

$$y_t = \mathrm{dec}(h_t, x_t), \tag{2}$$

where $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}\}$, $h_t \in \mathbb{F}^d$ is the latent state, $A(x_t) \in \mathbb{F}^{d\times d}$ is the state-transition matrix, $b(x_t) \in \mathbb{F}^d$ is the input-dependent injection term, $g : \mathbb{F}^d \times \mathcal{X} \to \mathbb{F}^d$ is a state-dependent modulation, $\phi : \mathbb{F}^d \to \mathbb{F}^d$ is an optional output nonlinearity, and $\mathrm{dec} : \mathbb{F}^d \times \mathcal{X} \to \mathcal{Y}$ is a decoder.

Equation (1) isolates four conceptually distinct ingredients: state transition $A(x_t)h_{t-1}$, input injection $b(x_t)$, state-dependent modulation $g(h_{t-1}, x_t)$, and output activation function ϕ. Different model classes arise by constraining or removing these ingredients.

SSMs such as S4 [Gu et al., 2022] lie in the affine-in-state regime with $g \equiv \mathbf{1}$ and $\phi(z) = z$, using structured transition operators A. Mamba [Gu and Dao, 2024] makes transition parameters input-adaptive, i.e., $A = A(x_t)$ and $b = b(x_t)$. Mamba-3 [Lahoti et al., 2026] and AUSSM [Karuvally et al., 2025] further increase the expressivity of this family through complex-valued state-space dynamics. More general linear recurrent models allow non-diagonal or matrix-valued transitions $A(x_t)$, as in DeltaNet [Yang et al., 2024] and DeltaProduct [Siems et al., 2025]. Conventional RNNs [Elman, 1990] introduce a nonlinear activation ϕ, while gated models [Hochreiter and Schmidhuber, 1997, Cho et al., 2014] introduce state-dependent gating, which we capture conceptually through the multiplicative modulation $g(h_{t-1}, x_t)$. Refer to Appendix H for details.

Dynamics	Model	Transition A	Field
Affine	Mamba [Gu and Dao, 2024]	diagonal	real
	Mamba-3 [Lahoti et al., 2026]	diagonal	complex
	AUSSM [Karuvally et al., 2025]	diagonal (unitary)	complex
	Simple AUSSM [Shakerinava et al., 2026]	diagonal (unitary)	complex
	Negative Mamba [Grazzi et al., 2025]	diagonal (signed)	real
	Linear RNN	dense	real
	Token-gated RNN	dense, input-gated	real
State-dependent	tanh RNN [Elman, 1990]	dense	real
	State-gated RNN	dense, state-gated	real

Table 1: **Recurrent models categorized by properties of the state-transition matrix A.** A model is *affine* when the state Jacobian $\partial h_t/\partial h_{t-1}$ does not depend on h_{t-1}, and *state-dependent* otherwise. *Transition* A describes the structure of the linear part of the recurrence; *Field* indicates whether A is real- or complex-valued. Full operator definitions in Appendix H.

2.2 State tracking and groups

State tracking is the problem of maintaining a latent representation of a symbolic state that evolves under an input sequence. Let G be a finite state space and let $\mathcal{T} : G \times \mathcal{X} \to G$ be a transition rule. Given $g_0 \in G$ and inputs $x_1, \ldots, x_L$, the symbolic trajectory is

$$g_t = \mathcal{T}(g_{t-1}, x_t), \qquad t = 1, \ldots, L.$$

A model receives the sequence online and must maintain enough information in its hidden state h_t to recover g_t at each step.

A convenient class of state-tracking tasks is given by finite groups. A *group* is a set G with an associative binary operation, an identity element, and inverses. When inputs x_t are drawn from generators $\Sigma \subset G$, the transition is group multiplication, $g_t = g_{t-1} \cdot x_t$. The target is the running product $y_t = x_1 \cdot x_2 \cdots x_t$ after each input. See Rotman [1995] for more details. We evaluate models on several groups that vary in compositional structure:

Parity and cyclic groups (C_k). Cyclic groups represent modular counting and are generated by a single element; C_2 is the parity task. All cyclic groups are Abelian, so reordering input group elements does not change the final product.

Symmetric groups (S_k). The symmetric group S_k consists of all permutations of k elements. For $k \geq 3$, S_k is non-Abelian, so input order changes the resulting state. Thus S_3 is the smallest symmetric group where order-sensitive composition is unavoidable.

Example 1 (S_3)**.** *Let S_3 be the set of permutations of $\{1, 2, 3\}$, and let the input tokens be generators (12) and (23). Starting from the identity $g_0 = e$, the sequence $(12), (23), (12)$ yields*

$$g_1 = (12), \qquad g_2 = (123), \qquad g_3 = (13).$$

The task is to output the running product after each token.

3 Error control in state tracking

Prior work often studies recurrent architectures through *expressivity*: whether a continuous-state model can realize a symbolic transition rule. For long-horizon state tracking, however, exact realization on clean trajectories is not enough. A robust tracker must also correct hidden-state perturbations that move it toward an incorrect symbolic state.

3.1 Exact affine tracking cannot correct state error

Let G be a finite symbolic state space, with each $g \in G$ carrying a fixed reference hidden-state representation $c_g^\star \in \mathbb{F}^d$. For a sequence $s = x_1 \cdots x_T$, let $F_s := F_{x_T} \circ \cdots \circ F_{x_1}$ be the induced hidden-state map, where F_x is the update (1) under input x. We focus on *state-preserving* sequences, whose symbolic action $\mathcal{T}_s$ is the identity: $\mathcal{T}_s(g) = g$ for all $g \in G$. Any exact realization must therefore return every $c_g^\star$ to itself, $F_s(c_g^\star) = c_g^\star$ for all $g \in G$.

The perturbations that matter most for symbolic tracking are those that move a hidden state toward competing representations $c_{g'}^\star$. These directions span the *symbolic subspace*

$$\mathcal{U} := \operatorname{span}\{c_g^\star - c_{g'}^\star : g, g' \in G\}. \tag{3}$$

Thus the directions that separate symbolic states are also the directions along which errors appear.

Theorem 1 (Affine neutrality on the symbolic subspace)**.** *Let s be a state-preserving sequence with non-degenerate representations ($c_g^\star \neq c_{g'}^\star$ for $g \neq g'$), and suppose the induced return map is affine, $F_s(h) = A_s h + b_s$ for all hidden states $h \in \mathbb{F}^d$. If $F_s(c_g^\star) = c_g^\star$ for all $g \in G$, then*

$$A_s|_{\mathcal{U}} = I.$$

Theorem 1 shows that once an affine return map fixes every symbolic state exactly, it has no freedom left to shrink the directions that separate those states. For any $g \in G$ and perturbation $\delta \in \mathcal{U}$,

$$F_s(c_g^\star + \delta) - F_s(c_g^\star) = \delta.$$

Thus symbolic realization and symbolic correction are incompatible on $\mathcal{U}$: exact affine models may preserve every $c_g^\star$, but they cannot create a restoring attractor along the directions that matter for symbolic discrimination. Proofs are in Appendix D, which also extends the theorem to non-minimal realizations, where several hidden states represent one symbolic state (Appendix D.3).

State-dependent error correction. On the other hand, a state-dependent return map can fix each representation $c_g^\star$ without being neutral around it. Writing a perturbed state as $c_g^\star + p$ with $p \in \mathcal{U}$, the relevant local map is $p \mapsto F_s(c_g^\star + p) - c_g^\star$. If its Jacobian at $p = 0$ has norm strictly below one uniformly over g, then nearby symbolic-subspace errors contract and every $c_g^\star$ is locally attracting. Thus state dependence does not guarantee correction, but it permits the state-conditioned perturbation contraction that affine return maps cannot realize; Appendix E.1 works out which choices of nonlinearity ϕ deliver this Jacobian-contraction condition operationally.

3.2 Accumulated error controls finite-horizon tracking

Theorem 1 does not imply immediate failure: affine return dynamics cannot generically remove errors along state-separating directions. The finite-horizon question is how long the learned symbolic states remain distinguishable under repeated reuse.

Let $c_g(t) := \mathbb{E}[h_t \mid g_t = g]$ denote the empirical centroid of the trained model's hidden states with symbolic state g at step t; unlike the fixed reference $c_g^\star$, it may drift over the rollout. Let $W_{\text{out}} \in \mathbb{F}^{|G| \times d}$ be the linear readout the classifier reads from. Define readout-space quantities

$$R(t) := \mathbb{E}\big[\|W_{\text{out}}(h_t - c_{g_t}(t))\|_2\big], \quad M(t) := \min_{g \neq g'} \|W_{\text{out}}(c_g(t) - c_{g'}(t))\|_2, \quad q(t) := R(t)/M(t).$$

$R(t)$ is the within-class spread the decoder sees, $M(t)$ is the between-class separation, and $q(t)$ is the distinguishability ratio. We call the nearest-centroid bound $\tau = \frac{1}{2}$ the *readability threshold*: symbolic states remain readable while $q(t) < \tau$. Further, let $P_{\mathcal{U}}$ denote the orthogonal projection onto the symbolic subspace $\mathcal{U}$. For an exact state-preserving return map $F_s(h) = A_s h + b_s$, Theorem 1 implies $A_s|_{\mathcal{U}} = I$.

		C_2		C_6		S_3	
Model	Dynamics	L1	L2	L1	L2	L1	L2
Mamba	Affine	✗	60	✗	60	✗	✗
Mamba-3	Affine	200	300	100	100	✗	60
AUSSM	Affine	1000	✗	200	100	✗	✗
Simple AUSSM	Affine	300	400	100	100	60	100
Negative Mamba	Affine	1000	1000	100	200	100	200
Linear RNN	Affine	✗	100	✗	60	✗	✗
Token-gated RNN	Affine	1000	700	300	400	500	1000
tanh RNN	State-dependent	1000	1000	1000	1000	1000	1000
State-gated RNN	State-dependent	1000	1000	1000	1000	1000	1000

Table 2: **Model performance on state-tracking tasks.** Values indicate the max-passing length mp, the largest evaluation length at which test accuracy remains $\geq 90\%$. The maximum training length was 60; cells displaying 60 denote models that survive only at the curriculum length. ✗ denotes $\mathrm{mp} = 0$ (the curriculum did not converge). L1 and L2 denote one-layer and two-layer recurrent stacks.

Corollary 1 (Finite-horizon error accumulation)**.** *Let the trained return-cycle tracker be $\tilde{F}_s = F_s + \varepsilon$, where F_s is the exact state-preserving affine return map considered in Theorem 1. Along a return cycle, let h_t be the trained model's hidden state at step t and $c^\star_{g_t}$ the fixed reference representation of its ground-truth symbolic state g_t; define $e_{\mathcal{U}}(t) := P_{\mathcal{U}}(h_t - c^\star_{g_t})$ and $\eta_t := P_{\mathcal{U}}\varepsilon(h_t)$. Then*

$$e_{\mathcal{U}}(t) = e_{\mathcal{U}}(0) + \sum_{j=0}^{t-1} \eta_j.$$

Thus any coherent residual component accumulates linearly. If, over the relevant horizon, the projected residuals have a nonzero average drift $t^{-1}\sum_{j<t}\eta_j \approx \bar{\eta} \neq 0$ and $M(t) \approx M > 0$, then $q(t)$ crosses a fixed threshold τ on the scale

$$T_{\mathrm{cross}} \approx \frac{\tau M}{\|W_{\mathrm{out}}\bar{\eta}\|}.$$

Proof in Appendix D.4. Empirically, the affine models we test (Section 4.3) trace out two trajectories of $q(t)$ consistent with this picture: *saturation*, where $q(t)$ sits above τ from the first few steps because $R(0)/M(0)$ is already large, and *climb*, where $q(t)$ starts below τ and grows linearly until the crossing, exactly the regime in which the T_{cross} estimate above applies.

4 Experiments

Models. We evaluate recurrent architectures spanning a spectrum from SSMs to gated RNNs. The *affine* models are: *Mamba* [Gu and Dao, 2024], a selective SSM; *Mamba-3* [Lahoti et al., 2026], a more expressive SSM variant; *AUSSM* [Karuvally et al., 2025], an adaptive unitary SSM; *Simple AUSSM* [Shakerinava et al., 2026], an ablated AUSSM variant; *Negative Mamba* [Grazzi et al., 2025], a Mamba variant with signed transition factors; *Linear RNN*, a dense real-valued linear recurrence; and *Token-gated RNN*, a gated recurrence whose gate depends only on the input x_t, so the update remains affine in h_{t-1}. The *state-dependent* models are: *tanh RNN*, a standard Elman RNN [Elman, 1990] with $\tanh$ activation; and *State-gated RNN*, a simplified gated recurrence whose gate depends on both h_{t-1} and x_t. Appendix H gives the full operator definitions.

Tasks. We evaluate on three group state-tracking tasks of increasing difficulty: parity C_2, the cyclic group C_6, and the symmetric group S_3. Each task requires tracking the running group product $g_t = g_{t-1} \cdot x_t$ from input elements sampled uniformly from the group. Detailed explanation and examples are in Appendix G.

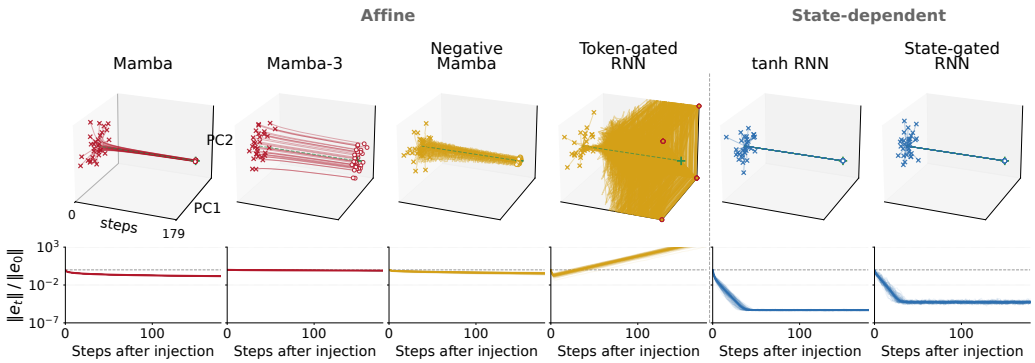

Figure 1: **Perturbation recovery after noise injection on S_3.** Top: error trajectories in a fixed 2D PCA basis of the injected errors; $\times$ marks the injected error, $\circ$ the endpoint (filled when clipped off scale), and the green dashed line and $+$ zero error. Bottom: normalized error magnitude $\|e_t\|/\|e_{t_0}\|$ over time, median and interquartile range over trials; thin lines show individual trials, and the dashed line a ratio of 1. Affine models show either global decay (Mamba, Mamba-3, and Negative Mamba) or expansion (Token-gated RNN), while state-dependent models (tanh RNN and State-gated RNN) show strong, selective error contraction.

Experimental details. All models are trained with curriculum learning up to sequence length 60. We evaluate extrapolation at lengths $\{100, 200, \ldots, 1000\}$ and report the maximum length at which test accuracy remains at least 90%. For each model and task, we run a grid search over state dimension, learning rate, learning-rate schedule, and three random seeds, reporting the best-performing configuration (full grid in Table 4). We evaluate both single-layer and two-layer recurrent stacks, denoted L1 and L2. Additional training and evaluation details are provided in Appendix C.

4.1 State-tracking performance

Table 2 reveals a clear dichotomy in state-tracking robustness: state-dependent models (tanh RNN and State-gated RNN) reliably track symbolic states up to the maximum tested length of 1000 tokens across all three tasks (C_2, C_6, and S_3), whereas affine models are generally unstable, with a few exceptions: Negative Mamba on C_2, and Token-gated RNN on C_2 and S_3. The results match Theorem 1: recurrent operators without state-dependent transitions lack robust error correction.

At the same time, the results show that some affine operators can remain on track well beyond the training length of 60 tokens. For example, Negative Mamba reaches 1000 on C_2, and Token-gated RNN reaches 1000 on C_2 (L1) and S_3 (L2), with shorter horizons of 500 on S_3 (L1) and 400 on C_6 (L2). These cases show that affine dynamics can approximate correction over finite horizons, and in two settings extend tracking out to the maximum tested length.

4.2 Error control behavior

Next, we test each model's error control dynamics, as predicted by Theorem 1. We inject a hidden-state perturbation and measure whether the error is propagated or reduced.

Metric. Error correction is operationally the decay of an injected perturbation under propagation. We inject Gaussian noise at step $t_0 = 20$ and compare the perturbed rollout with a clean rollout under the same input sequence i. Given the stepwise hidden states $h_{i,t}^{\text{pert}}$ and $h_{i,t}^{\text{clean}}$, we measure

$$e_{i,t} = h_{i,t}^{\text{pert}} - h_{i,t}^{\text{clean}}, \qquad \text{ratio}_{i,t} = \frac{\|e_{i,t}\|_2}{\|e_{i,t_0}\|_2}.$$

We track the full hidden-state difference rather than its projection onto $\mathcal{U}$, since the goal here is to characterize each model's overall response to perturbation; symbolic-subspace dynamics are addressed separately in Section 4.4.

Results. Figure 1 shows the accumulated response to injected perturbations. A clear dichotomy emerges. State-dependent models (tanh RNN, State-gated RNN) collapse $\|e_t\|/\|e_{t_0}\|$ by several

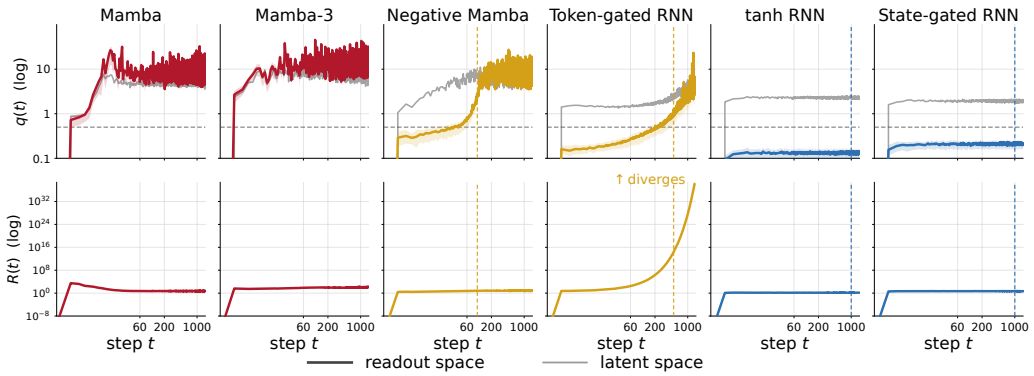

Figure 2: **Distinguishability ratio $q(t)$ over rollouts on S_3.** Top: $q(t) = R(t)/M(t)$ from Section 3.2, with the dashed line marking the nearest-centroid bound $q(t) = 1/2$; gray curves show the latent-space $q(t)$. Bottom: $R(t)$ on log-log axes. Vertical dashes mark each model's one-layer S_3 max-passing length from Table 2. Means over $N = 200$ rollouts; IQR shown as same-color band. Affine models show either immediate saturation (Mamba, Mamba-3) or gradual climb (Negative Mamba and Token-gated RNN), while state-dependent models (tanh RNN and State-gated RNN) remain low.

orders of magnitude within tens of steps and hold near the floor. The affine SSMs (Mamba, Mamba-3, Negative Mamba) instead contract errors through their global diagonal decay α_t with $|\alpha_t| < 1$, at per-step rates $\rho_{\text{step}} < 1$. This indicates global dissipation rather than conditional error correction for the affine models.

Token-gated RNN amplifies perturbations ($\rho_{\text{step}} > 1$), with $\|e_t\|/\|e_{t_0}\|$ reaching orders of magnitude above one. This follows from $h_t = g(x_t) \odot Wh_{t-1} + W_x x_t + b$, where $g(x_t) = \sigma(W_g x_t + b_g)$: because the gate depends only on x_t, clean and perturbed rollouts share the same gates, so errors follow $e_t = g(x_t) \odot We_{t-1}$. Mamba variants have the same cancellation but dissipate through $|\alpha_t| < 1$; Token-gated RNN instead relies on a dense W with spectral radius ≥ 1 to keep group states separable, which also amplifies errors.

4.3 State separation over rollouts

Here, we directly put the framework of Section 3.2 to test. Section 3.2 describes two failure modes for an approximate affine tracker: *saturation*, where $q(t)$ sits above the readability threshold τ from the start, or *climb*, where $q(t)$ starts below τ and crosses it at T_{cross}. We measure $q(t)$ across rollouts and inspect how each architecture's trajectory unfolds against these predictions.

Metric. At each step t we form time-current centroids $c_g(t) := \mathbb{E}_i[h_{i,t} \mid g_{i,t} = g]$, the per-step mean of the hidden state over $N = 200$ rollouts whose oracle symbol at t is g. We then measure the distinguishability ratio $q(t) = R(t)/M(t)$ from Section 3.2: $R(t)$ is the empirical mean over rollouts of $\|W_{\text{out}}(h_{i,t} - c_{g_{i,t}}(t))\|_2$, and $M(t)$ is the smallest pairwise centroid distance in readout space, $\min_{g \neq g'} \|W_{\text{out}}(c_g(t) - c_{g'}(t))\|_2$. Thus $q(t)$ measures within-class spread in units of the smallest inter-class margin. For a nearest-centroid decoder, $q(t) < 0.5$ means that a state at the average distance $R(t)$ is still closer to its own centroid than to any competitor.

Results. Figure 2 supports the two structural predictions of Theorem 1 and Corollary 1. In the top row, state-dependent transitions (tanh RNN and State-gated RNN) maintain $q(t) < 0.5$ throughout rollout, whereas affine transitions eventually cross the nearest-centroid bound, consistent with the no-correction obstruction of Theorem 1.

Within the affine class, the trajectories instantiate the two finite-horizon alternatives of Section 3.2. Mamba and Mamba-3 are already above the nearest-centroid bound at the start of extrapolation, corresponding to saturation: the state clouds are immediately too wide relative to their separation. Negative Mamba and Token-gated RNN follow the climb regime: they start below the bound, remain readable for a finite horizon, and cross only after repeated rollout accumulates readout-space spread.

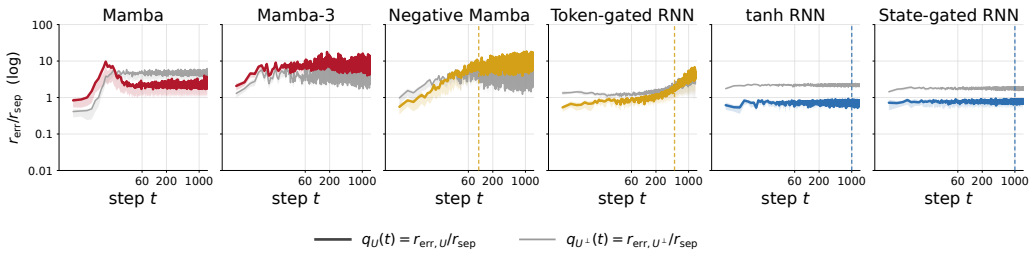

Figure 3: **Subspace decomposition of within-class spread.** For each architecture, we decompose within-class spread into the symbolic-subspace component $q_{\mathcal{U}}(t) = r_{\mathrm{err},\mathcal{U}}(t)/r_{\mathrm{sep}}(t)$ (color) and the orthogonal component $q_{\mathcal{U}^\perp}(t) = r_{\mathrm{err},\mathcal{U}^\perp}(t)/r_{\mathrm{sep}}(t)$ (gray), both on a log scale. Vertical dashes mark each model's one-layer S_3 max-passing length from Table 2.

In both climb cases, the crossing precedes the corresponding mp in Table 2, consistent with a readability-based failure criterion.

The bottom row shows that the climb regime can arise through different dynamics of spread and separation. Token-gated RNN grows R together with M, keeping the latent-space ratio comparatively stable. Negative Mamba instead directly bounds R through its diagonal transition parameterization, yielding a slower climb despite the same affine no-correction constraint.

4.4 Error decomposition along the symbolic subspace

We next ask whether the deviation lies in the symbolic subspace $\mathcal{U}$ from Equation (3), where symbolic errors appear and affine return dynamics cannot generically contract perturbations (Theorem 1). We therefore decompose the within-class deviation into $\mathcal{U}$ and $\mathcal{U}^\perp$ components.

Metric. At step t, let $\delta_{i,t} := h_{i,t} - c_{g_{i,t}}(t)$ be the per-rollout deviation from the time-current centroid, and let $P_{\mathcal{U}}(t)$ project onto the span of centroid differences $\{c_g(t) - c_{g'}(t)\}$. Define the root-mean-square spreads

$$r_{\mathrm{err},\mathcal{U}}(t) := \sqrt{\mathbb{E}_i \|P_{\mathcal{U}}(t)\,\delta_{i,t}\|_2^2}, \qquad r_{\mathrm{err},\mathcal{U}^\perp}(t) := \sqrt{\mathbb{E}_i \|\delta_{i,t}\|_2^2 - r_{\mathrm{err},\mathcal{U}}(t)^2},$$

and the inter-centroid scale $r_{\mathrm{sep}}(t) := \min_{g \neq g'} \|c_g(t) - c_{g'}(t)\|_2$. We report $q_{\mathcal{U}}(t) = r_{\mathrm{err},\mathcal{U}}(t)/r_{\mathrm{sep}}(t)$ and $q_{\mathcal{U}^\perp}(t) = r_{\mathrm{err},\mathcal{U}^\perp}(t)/r_{\mathrm{sep}}(t)$, which split the within-class spread into state-separating and orthogonal components. RMS aggregation preserves the per-rollout Pythagorean identity at the population level.

Results. Figure 3 shows how the spread is distributed across $\mathcal{U}$ and $\mathcal{U}^\perp$. For Negative Mamba and Token-gated RNN, $q_{\mathcal{U}^\perp}$ is larger than $q_{\mathcal{U}}$ early in rollout, indicating that most spread initially lies outside the state-separating directions. Near each model's max-passing length the ordering reverses: $q_{\mathcal{U}}$ catches up to and exceeds $q_{\mathcal{U}^\perp}$. Thus, finite-horizon failure is associated not merely with growth of spread, but with its shift into $\mathcal{U}$, the subspace where affine return dynamics cannot generically contract perturbations.

State-dependent models (tanh RNN and State-gated RNN) show the complementary pattern: $q_{\mathcal{U}}$ remains suppressed while the larger component lies in $\mathcal{U}^\perp$. State-dependent transitions selectively prevent spread along the state-separating directions, supplying the conditional correction unavailable to affine return maps under Theorem 1. Mamba and Mamba-3 are saturated from early rollout, so there is no meaningful subspace-dominance transition to analyze.

4.5 Further analysis

Additional models and tasks. We report additional results in Appendices F.1 and F.2.

Correlation between T_{cross} and downstream performance. Corollary 1 predicts that the first nearest-centroid crossing, $T_{\mathrm{cross}} = \min\{t : q(t) \geq 0.5\}$ (Appendix C.2), should track how long

Type	Operator ϕ	S_3
	Affine	✗
norm	LayerNorm	✗
	sphere projection	✗
nonlinear	$\tanh$	1000
	ReLU	1000
	max	1000
	min	1000
	GroupSort $k = 2$	1000

Table 3: **Many nonlinear operators support robust tracking.** On S_3 in the single-layer setting, diverse nonlinear activations reach the maximum tested length, whereas affine and normalization-only variants fail. Refer to Appendix E.1 for interpretation.

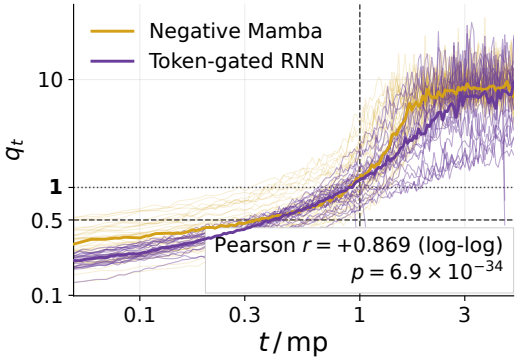

Figure 4: **Readability collapse coincides with downstream failure.** On S_3, the distinguishability ratio $q(t)$ is plotted against failure-normalized time t/mp. Thin lines are individual runs, thick lines the median of each architecture, and both axes are logarithmic. The dotted reference at $q(t) = 1$ marks where within-class spread equals the smallest inter-class margin. The dashed lines mark the threshold $q(t) = 1/2$ and $t = \mathrm{mp}$. Pearson $r = 0.87$ on $\log T_{\mathrm{cross}}$ vs $\log \mathrm{mp}$.

an affine tracker remains usable. Figure 4 supports this: across S_3 models with $\mathrm{mp} \geq 60$ that cross, T_{cross} strongly correlates with downstream max-passing length on a log-log scale ($r = +0.87$, $p < 10^{-30}$).

The trained readout fails later than the nearest-centroid bound suggests: although the nearest-centroid bound is $q(t) < 0.5$, failure empirically aligns closer to $q(t) = 1$, where within-class spread matches between-class separation. At $t = \mathrm{mp}$, the median $q(t)$ is 0.97 (95% bootstrap CI $[0.84, 1.08]$). T_{cross} remains predictive because both thresholds are driven by the accumulated within-class spread predicted by Corollary 1.

Nonlinear activation type. Theorem 1 identifies state-dependent transitions as the key ingredient for error correction, not a specific nonlinear implementation. To test this, we fix the Elman recurrence $h_t = \phi(W_h h_{t-1}, W_x x_t + b)$ and vary only the nonlinear activation ϕ.

Table 3 shows that several distinct state-dependent operators succeed, including standard pointwise activations, pointwise $\max / \min$, and GroupSort [Anil et al., 2019]. In contrast, whole-vector normalization operators fail despite being nonlinear. Thus the relevant distinction is not the activation family itself, but whether the induced Jacobian can modulate symbolic directions in a state-dependent way. We defer the operator-level Jacobian analysis to Appendix E.1.

C_2 is a weak test of correction. Several affine models reach the maximum tested length on C_2 because parity can remain readable under neutral oscillation, without genuine error correction. With $C_2 = \{e, a\}$, the affine involution $F_a(h) = -h + (c_e^\star + c_a^\star)$ swaps any two distinct fixed references and satisfies $F_a^2 = \mathrm{id}$. For a symbolic-subspace perturbation,

$$F_a(c_g^\star + \delta) = c_{g \cdot a}^\star - \delta, \qquad F_a^2(c_g^\star + \delta) = c_g^\star + \delta,$$

so the error flips sign but is not removed. Binary decoding can still remain correct while this oscillation stays within the readout margin. Thus C_2 is an order-two edge case: it tests whether an uncorrected error stays within the readout margin, not active correction of symbolic-subspace drift. See Appendix E.2.

5 Conclusion

Recurrent state tracking is usually framed as expressivity: which symbolic transitions an architecture can represent. We show that this is incomplete: robust tracking also requires controlling errors

accumulated under repeated reuse. Affine models cannot correct errors on the state-separating subspace, because preserving every symbolic state forces the identity action on the directions that separate the states (Theorem 1). Thus approximate affine trackers fail when accumulated error overtakes the state margin, not when expressivity runs out. Empirically, affine models saturate or climb past the readability threshold, whereas state-dependent models retain conditional contraction and track far beyond the training length. Overall, state tracking is limited not only by what a model can represent, but by whether it can correct the errors accumulated over time.

Therefore, future work on architectures for state tracking could explore state-dependent correction, for example through nonlinear RNNs with parallel training [Gonzalez et al., 2024, Danieli et al., 2026] or hybrids that interleave affine layers with attention or nonlinear layers [Kimi Team et al., 2025]; a state-dependent update gives up the exact parallel scan that makes affine models efficient. Limitations are discussed in Appendix A.

Acknowledgments and Disclosure of Funding

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. RS-2020-II201361, Artificial Intelligence Graduate School Program (Yonsei University); No. RS-2022-II220113, Developing a Sustainable Collaborative Multi-modal Lifelong Learning Framework) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-00554790).

References

Cem Anil, James Lucas, and Roger Grosse. Sorting out Lipschitz function approximation. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 291–301, 2019.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

Federico Danieli, Pau Rodríguez, Miguel Sarabia, Xavier Suau, and Luca Zappella. ParaRNN: Unlocking parallel training of nonlinear RNNs for large language models. In *International Conference on Learning Representations*, 2026.

M. Reza Ebrahimi, Michaël Defferrard, Sunny Panchal, and Roland Memisevic. On the "induction bias" in sequence models. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 27550–27568, 2026.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

Mahan Fathi, Clement Gehring, Jonathan Pilault, David Kanaa, Pierre-Luc Bacon, and Ross Goroshin. Course correcting Koopman representations. In *International Conference on Learning Representations*, 2024.

Xavier Gonzalez, Andrew Warrington, Jimmy T. H. Smith, and Scott W. Linderman. Towards scalable and stable parallelization of nonlinear RNNs. In *Advances in Neural Information Processing Systems*, volume 37, pages 5817–5849, 2024.

Riccardo Grazzi, Julien Siems, Arber Zela, Jörg K. H. Franke, 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 *First Conference on Language Modeling*, 2024.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022.

Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal state spaces are as effective as structured state spaces. In *Advances in Neural Information Processing Systems*, volume 35, pages 22982–22994, 2022.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Arjun Karuvally, Franz Nowak, T. Anderson Keller, Carmen Amo Alonso, Terrence J. Sejnowski, and Hava T. Siegelmann. Bridging expressivity and scalability with adaptive unitary SSMs. In *Advances in Neural Information Processing Systems*, volume 38, pages 22017–22057, 2025.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165, 2020.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, T. Y. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi Linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

Aakash Lahoti, Kevin Y. 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.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 35492–35506, 2024.

Antonio Orvieto, Samuel L. Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 26670–26698, 2023.

Joseph J. Rotman. *An Introduction to the Theory of Groups*, volume 148 of *Graduate Texts in Mathematics*. Springer, 4th edition, 1995.

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, pages 41202–41241, 2024.

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*, volume 38, pages 153738–153782, 2025.

Aleksandar Terzic, Michael Hersche, Giacomo Camposampiero, Thomas Hofmann, Abu Sebastian, and Abbas Rahimi. On the expressiveness and length generalization of selective state space models on regular languages. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 20876–20884, 2025a.

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*, volume 38, pages 83072–83111, 2025b.

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, pages 115491–115522, 2024.

Appendix Overview

- **Appendix A:** Limitations
- **Appendix B:** Related work
- **Appendix C:** Experimental details
- **Appendix D:** Proofs
- **Appendix E:** Further discussion
- **Appendix F:** Additional empirical results
- **Appendix G:** Examples of state-tracking tasks
- **Appendix H:** Model details

A Limitations

This work isolates how recurrent dynamics manage error during symbolic state tracking. Our results do not imply that affine recurrences fail in-domain, nor that they are unsuitable for sequence modeling in general. They show that, when a model must reuse symbolic states beyond the training horizon, affine return maps cannot provide the state-dependent error dynamics needed to keep accumulated drift from erasing state separation.

Our experiments are restricted to finite-group state tracking, the canonical testbed used in prior theoretical work on recurrent state tracking [Merrill et al., 2024, Sarrof et al., 2024, Shakerinava et al., 2026]. The main experiments use C_2, C_6, and S_3, and the appendix extends the suite to $C_2 \times C_4$ and A_4; the affine/state-dependent dichotomy persists across all five groups tested (Tables 2 and 6). This restriction is deliberate. The tasks are difficult enough to expose failure, yet simple enough that architectures do not collapse into uniform failure. This separation makes differences in correction dynamics directly observable, whereas richer benchmarks would add confounds and may obscure the mechanism behind failure. The capability these tasks isolate, maintaining and updating an exact latent state over long horizons, is one that many real-world problems also require, such as tracking variables and scopes in long-form code generation or character identities in a video narrative. Group tracking is thus a necessary test for such problems, not a sufficient one. Language-like tasks such as bracket matching, code execution, or board-game move sequences are a natural next step; they need an exactly identifiable intermediate state and little confounding from parsing or task knowledge, and we leave them to future work.

On the theory side, Theorem 1 assumes an exact affine state-preserving return map, while the experiments train approximate models. Corollary 1 bridges the two through the residual ε, but its drift $\bar{\eta}$ depends on the action of A_s on $\mathcal{U}^{\perp}$, which the theorem leaves unconstrained, so it cannot be recovered from a trained model; we therefore test the corollary only through its observable consequences. The theorem itself covers point-valued representations and, through Appendix D.3, non-minimal realizations whose decoder classes have distinct preserved centroids, but not decoder classes with coinciding centroids, transient hidden states, or arbitrary decoder equivalence classes.

Two further limitations concern the experiments. Table 2 reports the best max-passing length across seeds, which does not show how reliably it is reached; Appendix F.4 reports the mean and standard deviation across seeds. In addition, our analysis is restricted to recurrent state-tracking models, following prior work on recurrent state tracking [Merrill et al., 2024, Sarrof et al., 2024, Shakerinava et al., 2026]. Attention-based models are not recurrent state-update models, so the hidden-state error analysis does not apply to them; Appendix F.6 reports their downstream state-tracking performance under the same task settings for comparison.

B Related work

Recurrent models. RNNs model sequential data through recurrent hidden-state updates. Elman [1990] introduced an RNN with a nonlinear activation, while LSTM [Hochreiter and Schmidhuber, 1997] and GRU [Cho et al., 2014] address vanishing gradients with gating mechanisms, yet their nonlinear recurrence limits parallelization. Recent works therefore revisit linear recurrent architectures for better scalability. LRU [Orvieto et al., 2023] uses diagonal linear dynamics, while DeltaNet

[Yang et al., 2024] adopts delta-rule-based linear recurrence for efficiency. Parallel to these, SSMs characterize temporal dynamics by discretizing continuous-time differential equations into linear recurrences. Similarly, SSM-based models use linear state updates and often employ structured transition matrices for efficiency [Gupta et al., 2022, Gu et al., 2022]. Mamba [Gu and Dao, 2024] introduces input-dependent selective updates; more recently, Mamba-3 [Lahoti et al., 2026] extends this line with complex-valued states and MIMO formulations to improve efficiency and modeling capacity.

Model expressivity and state tracking. Prior works study model expressivity to characterize state-tracking ability in sequence models. Merrill et al. [2024] show via circuit complexity that linear SSMs with input-independent or diagonal transition matrices, commonly used in recent SSM models, lie in L-uniform TC^0, similar to Transformers. Thus, they cannot solve NC^1-hard state-tracking problems such as S_5 permutation composition, whereas a single-layer nonlinear RNN can. Sarrof et al. [2024] show that non-negative diagonal SSMs cannot solve parity, revealing a limitation beyond circuit complexity. Grazzi et al. [2025] extend this analysis to non-diagonal linear recurrent models, showing that parity requires an eigenvalue outside $[0, \infty)$, such as a negative one, and that counting modulo m, for m not a power of two, requires transition products with non-real eigenvalues.

Karuvally et al. [2025] introduce AUSSM, an input-dependent complex-valued diagonal SSM with unit-modulus eigenvalues. They show that AUSSM can simulate cyclic groups, and that combining it with Mamba enables solvable group recognition, maximizing the expressivity of diagonal SSMs. Shakerinava et al. [2026] further study the expressivity of input-dependent complex-valued diagonal (DCD) SSMs. They show that single-layer DCD SSMs cannot solve state-tracking problems over non-Abelian groups, while multi-layer DCD SSMs can track solvable groups with subnormal series length bounded by the model depth. Empirically, however, they also show that DCD SSMs often fail to learn length-generalizing solutions, even for S_3 state tracking, despite their theoretical expressivity. Motivated by this gap, our work analyzes why such models fail to learn these solutions in practice.

Long-horizon drift in learned latent dynamics. Fathi et al. [2024] study long-horizon drift in Koopman autoencoders, where repeated linear evolution moves the latent state away from the manifold of valid encoded observations, and correct it by periodically re-encoding the state. We study drift away from the correct symbolic state: a re-encoded state can be valid yet represent the wrong symbol, so re-encoding controls errors in our sense only if it restores the correct symbolic state.

C Experimental details

Compute. All training and analysis runs use one NVIDIA RTX A6000 (48 GB) or RTX 3090 (24 GB) GPU per run. The runs of the main grid (Table 4) total about $1{,}400$ GPU-hours; a run takes on average from under a minute (tanh RNN) to about two hours (AUSSM). Preliminary experiments not reported here (earlier model variants, other perturbation magnitudes and injection steps, and secondary diagnostics) used a comparable amount of compute; none of them changed the split between affine and state-dependent models.

Statistical reporting. Aggregated curves in Figure 2 show means over $N = 200$ rollouts, and those in Figure 3 show root-mean-square spreads over the same rollouts, both with per-step IQR bands. The perturbation-recovery panels in Figure 1 use medians over $n = 200$ injection trials. The T_{cross} correlation in Section 4.5 reports the Pearson coefficient with its p-value, together with a 95% bootstrap confidence interval on the median $q(t)$ at the max-passing length. The max-passing lengths in Table 2 follow the convention of Shakerinava et al. [2026] in reporting the best across three seeds rather than averaging; Appendix F.4 reports the mean and standard deviation across seeds.

Asset licenses. All architectures evaluated in this work are credited via the citations in Appendices F.1 and H. We use PyTorch, NumPy, Matplotlib and the official Mamba implementation under their published open-source licenses. The group state-tracking sequences and labels are generated synthetically by our code.

LLM usage. Large language models were used to assist with adapting reference evaluation code to our experimental setup, generating visualization scripts, and checking factual consistency of the exposition and analytic derivations.

C.1 Training and grid search

All recurrent models are trained from scratch on group state tracking. Each model uses the canonical layer of Section 2.1 inside a pre-norm residual block, an embedding width $d_{\text{model}} = 698$ matched to the parameter budget of Shakerinava et al. [2026], and a linear readout $W_{\text{out}} \in \mathbb{R}^{|G| \times d_{\text{model}}}$. Inputs are i.i.d. token sequences $x_1, \ldots, x_T$ drawn uniformly from G. The model predicts the running product at every step and is trained with cross-entropy.

Optimization uses AdamW with weight decay 0.01 and batch size 256. At each curriculum stage, we regenerate $10\,000$ training sequences and $2\,000$ test sequences. The curriculum starts at $T = 2$ and doubles whenever test accuracy is at least 0.95 for five consecutive epochs, up to $L_{\max} = 60$.

After training, we freeze the model and evaluate it on $2\,000$ fresh sequences for each length in $\{100, 200, \ldots, 1000\}$. The max-passing length is the largest evaluated length with test accuracy at least 0.90. If training reaches $L_{\max} = 60$ but no generalization length passes, we set it to 60; if the curriculum does not converge, we set it to 0 and print it as ✗.

We grid-search the state dimension d_{state}, the learning rate, the scheduler and the seed over the values in Table 4, following Shakerinava et al. [2026]. A *grid-best* checkpoint denotes the run with the largest max-passing length across all hyperparameter settings and seeds, with ties broken by the final test accuracy.

C.2 Per-experiment specification

State-tracking performance (Table 2). The reported number in each cell is the max-passing length of the grid-best checkpoint for that model, group and depth, evaluated exactly as described above. No further hyperparameters are introduced.

Error-correction probe (Figure 1). We use S_3 with one layer and each model's grid-best checkpoint. For each of $N = 200$ fresh sequences of length $T = 200$ we run a clean rollout $h^{\text{clean}}_{i,t}$ and a perturbed rollout $h^{\text{pert}}_{i,t}$ that is identical up to step $t_0 = 20$, where we inject i.i.d. Gaussian noise of standard deviation $\sigma = 10^{-2}$ into the recurrent state of the first layer (the state that the recurrence propagates, not the residual stream). We then track $e_{i,t} = h^{\text{pert}}_{i,t} - h^{\text{clean}}_{i,t}$ for $t \geq t_0$ and report the median of $\|e_{i,t}\|_2 / \|e_{i,t_0}\|_2$. The per-step contraction rate ρ_{step} quoted in the main text is the $(T - t_0 - 1)$-th root of the median final ratio. The 3-D trajectory panels project the error onto its leading two PCA components computed at t_0.

State separation over rollouts (Figure 2). On S_3 with one layer and the grid-best checkpoints, we evaluate $N = 200$ fresh sequences of length $T_{\max} = 1500$, well beyond $L_{\max} = 60$. At every step t we group the rollouts by their oracle symbol and take per-class means to obtain a time-resolved centroid $c_g(t) = \mathbb{E}_i[h_{i,t} \,|\, g_{i,t} = g]$. The reported quantities are $R(t), M(t), q(t)$ from Section 3.2, instantiated as $R(t) = \mathbb{E}_i \|W_{\text{out}}(h_{i,t} - c_{g_{i,t}}(t))\|_2$ and $M(t) = \min_{g \neq g'} \|W_{\text{out}}(c_g(t) - c_{g'}(t))\|_2$. The figure shows readout-space versions as the primary curves and the latent-space versions (without W_{out}) as gray overlays. Vertical markers reproduce each model's max-passing length from Table 2.

Crossing time (Figure 4). To ignore brief spikes, we compute $q(t)$ from 50-step moving averages of R and M, and take T_{cross} as the first step after the training length from which $q(t) \geq 0.5$ holds for 50 steps. The median $q(t)$ at mp in Section 4.5 uses the same averages; Figure 4 plots the unsmoothed ratio.

Symbolic-subspace decomposition (Figure 3). We use the rollouts and checkpoints of Figure 2. At each step t we form the centered centroid matrix $\widetilde{C}(t) \in \mathbb{R}^{|G| \times d_{\text{model}}}$ and take the top $k = |G| - 1$ right singular vectors as the orthonormal basis $P_{\mathcal{U}}(t)$ of the symbolic subspace. The within-class deviation $\delta_{i,t} = h_{i,t} - c_{g_{i,t}}(t)$ splits into its squared component in the symbolic subspace, $\|P_{\mathcal{U}}(t)^\top \delta_{i,t}\|^2$, and the orthogonal remainder, $\|\delta_{i,t}\|^2 - \|P_{\mathcal{U}}(t)^\top \delta_{i,t}\|^2$. We aggregate each part across rollouts as a root mean square and normalize by the latent inter-centroid scale $r_{\text{sep}}(t) = \min_{g \neq g'} \|c_g(t) - c_{g'}(t)\|_2$.

Nonlinear activation type (Table 3). We fix the recurrence $h_t = \phi(W_h h_{t-1} + W_x x_t + b)$ on S_3 with one layer and vary only ϕ: an affine baseline (identity), two whole-vector normalizations

Architecture	
d_{model}	698
depth L	$\{1, 2\}$
readout	$W_{\text{out}} \in \mathbb{R}^{\lvert G\rvert \times d_{\text{model}}}$
Optimization	
optimizer	AdamW, weight decay 0.01
batch size	256
total max epochs (across all curriculum stages)	500
gradient clipping	norm 1
cosine schedule	decays to 10^{-5} over 500 epochs
reduce on plateau	halves the learning rate after 10 epochs without a gain in test accuracy
Curriculum	
start length	2
max training length $L_{\max}$	60
promotion threshold	test accuracy ≥ 0.95 for 5 epochs
training / test sequences per stage	10 000 / 2 000 (regenerated per stage)
Length-generalization evaluation	
evaluated weights	best reached in training: longest curriculum length, then highest test accuracy
evaluation lengths	$\{100, 200, \ldots, 1000\}$
sequences per length	2 000
passing threshold	test accuracy ≥ 0.90
max-passing length	largest evaluation length still passing; $L_{\max}$=60 if curriculum converged but no evaluation length passed; 0 if curriculum failed
Grid search (81 runs per model, group and depth)	
d_{state}	$\{32, 64, 128\}$
learning rate	$\{10^{-4}, 5\times10^{-4}, 10^{-3}\}$
scheduler	fixed, cosine, reduce on plateau
seeds	3
grid-best selection	largest max-passing length, then final test accuracy

Table 4: **Shared training and grid-search configuration.** Used for every checkpoint reported in the paper unless explicitly overridden in the per-experiment specification.

(LayerNorm, sphere projection $h \mapsto h/\|h\|$), pointwise nonlinearities ($\tanh$, ReLU), pointwise pair operators ($\max, \min$), and GroupSort with group size $k = 2$ [Anil et al., 2019]. The identity and $\tanh$ entries are the Linear RNN and tanh RNN of Table 2, with the full grid; each other operator uses one configuration ($d_{\text{state}} = 64$, learning rate 10^{-3}, cosine schedule) with three seeds. Each cell reports the best max-passing length; Appendix E.1 analyzes the per-operator Jacobians.

D Proofs

D.1 Proof of Theorem 1

Let s be a state-preserving sequence and assume the induced return map is affine, $F_s(h) = A_s h + b_s$. By the exact-preservation hypothesis of Theorem 1, $F_s(c_g^\star) = c_g^\star$ for every $g \in G$. For any pair $g, g' \in G$,

$$A_s(c_g^\star - c_{g'}^\star) = F_s(c_g^\star) - F_s(c_{g'}^\star) = c_g^\star - c_{g'}^\star.$$

The vectors $\{c_g^\star - c_{g'}^\star\}_{g,g' \in G}$ span $\mathcal{U}$ by (3), so A_s acts as the identity on $\mathcal{U}$, establishing $A_s|_{\mathcal{U}} = I$. □

D.2 Proof of perturbation neutrality

We verify the consequence stated in Section 3.1: under the hypotheses of Theorem 1, every perturbation $\delta \in \mathcal{U}$ is transported unchanged by the return map. Continuing under the affine return assumption, let $g \in G$ and $\delta \in \mathcal{U}$. Then

$$F_s(c_g^\star + \delta) - F_s(c_g^\star) = A_s(c_g^\star + \delta) + b_s - (A_s c_g^\star + b_s) = A_s\delta = \delta,$$

where the last equality uses $A_s|_{\mathcal{U}} = I$ from Theorem 1. Thus the perturbation along $\mathcal{U}$ is preserved exactly under the return map. □

D.3 Extension to non-minimal realizations

Theorem 1 assumes one reference representation $c_g^\star$ per symbolic state. A trained network may instead realize a symbolic state by several hidden states, all decoded to the same symbol, which may move or cycle among themselves under the dynamics, so that $F_s(h) = h$ need not hold for any of them. Fix a state-preserving sequence s and, for each $g \in G$, let μ_g be a distribution over the hidden states that represent g. Suppose the return map preserves it, $(F_s)_\sharp \mu_g = \mu_g$, and let $m_g := \mathbb{E}_{h\sim\mu_g}[h]$ be its centroid. Because $F_s(h) = A_s h + b_s$ is affine,

$$F_s(m_g) = F_s\big(\mathbb{E}_{\mu_g}[h]\big) = \mathbb{E}_{\mu_g}[F_s(h)] = \mathbb{E}_{(F_s)_\sharp\mu_g}[h] = m_g.$$

Hence, for any $g, g' \in G$, $A_s(m_g - m_{g'}) = F_s(m_g) - F_s(m_{g'}) = m_g - m_{g'}$, so $A_s u = u$ for every $u \in \mathcal{U}_{\mathrm{nm}} := \mathrm{span}\{m_g - m_{g'} : g, g' \in G\}$. The conclusion of Theorem 1 therefore holds on the subspace separating the preserved class centroids, provided $m_g \neq m_{g'}$ for $g \neq g'$. This covers the recurrent part of a finite non-minimal realization, in which several hidden states represent the same symbolic state and move within its decoder class while the class centroid stays fixed.

Limitation. The argument does not cover arbitrary decoder equivalence classes. If two symbolic states have the same centroid, $m_g = m_{g'}$, their centroid difference does not separate them and the argument says nothing about the directions that do. Transient hidden states are not covered until they enter a preserved class. The theorem therefore directly covers point-valued representations $c_g^\star$ and extends to non-minimal realizations with distinct preserved class centroids.

D.4 Proof of Corollary 1

Affine neutrality on the symbolic subspace. Let $F_s(h) = A_s h + b_s$. Since F_s is the exact state-preserving affine return map considered in Theorem 1, its linear part satisfies

$$A_s|_{\mathcal{U}} = I.$$

Thus, along directions that distinguish symbolic states, the exact affine return map has no contracting homogeneous component. Along a return cycle, $g_{t+1} = g_t$, and exact state preservation gives $F_s(c_{g_t}^\star) = c_{g_t}^\star$.

Projected error recurrence. Using $\tilde{F}_s = F_s + \varepsilon$, the trained update satisfies

$$h_{t+1} = \tilde{F}_s(h_t) = F_s(h_t) + \varepsilon(h_t).$$

The projected deviation from the fixed reference representation $c_{g_t}^\star$ of the current symbolic state (unchanged along the return cycle, since $g_{t+1} = g_t$) is therefore

$$\begin{aligned} e_{\mathcal{U}}(t+1) &= P_{\mathcal{U}}(h_{t+1} - c_{g_t}^\star) \\ &= P_{\mathcal{U}}\big(F_s(h_t) - F_s(c_{g_t}^\star)\big) + P_{\mathcal{U}}\varepsilon(h_t). \end{aligned}$$

The first term is the effect of the exact affine return map on the current deviation, while the second term is the projected approximation residual of the trained tracker. Since F_s is affine,

$$F_s(h_t) - F_s(c_{g_t}^\star) = A_s(h_t - c_{g_t}^\star).$$

Restricting to the projected dynamics inside $\mathcal{U}$, and using $A_s|_{\mathcal{U}} = I$, the exact affine part preserves the current symbolic-subspace deviation rather than reducing it. Hence

$$e_{\mathcal{U}}(t+1) = e_{\mathcal{U}}(t) + \eta_t, \qquad \eta_t := P_{\mathcal{U}}\varepsilon(h_t).$$

Unrolling this recurrence gives

$$e_{\mathcal{U}}(t) = e_{\mathcal{U}}(0) + \sum_{j=0}^{t-1} \eta_j.$$

Thus projected residuals are accumulated, not corrected, by the affine return dynamics. Here $e_{\mathcal{U}}(0)$ is the error in the hidden state handed to the recurrence, for example by an encoder or from a compressed prefix, and is separate from the transition residual ε; exact affine return dynamics preserve it, $e_{\mathcal{U}}(t) = e_{\mathcal{U}}(0)$.

Linear accumulation under coherent residual drift. The previous identity does not require residuals to grow linearly; it only shows that they enter additively. Linear growth occurs when the residuals have a coherent component over the relevant horizon. If

$$\frac{1}{t} \sum_{j<t} \eta_j \approx \bar{\eta} \neq 0,$$

then

$$\sum_{j<t} \eta_j \approx t\bar{\eta}, \qquad e_{\mathcal{U}}(t) \approx e_{\mathcal{U}}(0) + t\bar{\eta}.$$

The fixed initial error is lower order relative to the growing drift, so the readout-visible component has scale

$$R(t) \approx \|W_{\text{out}}(t\bar{\eta})\| = t\|W_{\text{out}}\bar{\eta}\|.$$

Crossing scale. If the between-state separation remains approximately stable, $M(t) \approx M > 0$, then

$$q(t) = \frac{R(t)}{M(t)} \approx \frac{t\|W_{\text{out}}\bar{\eta}\|}{M}.$$

The crossing time is obtained by solving $q(t) = \tau$, which yields

$$T_{\text{cross}} \approx \frac{\tau M}{\|W_{\text{out}}\bar{\eta}\|}.$$

This shows that the finite horizon is controlled by the competition between the stable separation scale M and the rate of coherent readout-visible drift $\|W_{\text{out}}\bar{\eta}\|$. □

E Further discussion

E.1 Per-operator Jacobian analysis

Section 3.1 identifies a sufficient condition for state-dependent error correction: the return-map Jacobian on $\mathcal{U}$ has norm strictly below one uniformly over the references $c_g^\star$. The activation ϕ in the canonical form is what gives a return map state-dependent Jacobians at all, so the operational question is which choices of ϕ can deliver this contraction. Table 3 reports that pointwise activations and pair operators support state tracking on S_3 while whole-vector normalizations do not, despite both classes being nonlinear. The distinction is visible in the local Jacobian of each operator. Writing the recurrence as $h_t = \phi(p_t)$ with pre-activation $p_t = W_h h_{t-1} + W_x x_t + b$, the Jacobian wrt h_{t-1} at a fixed reference $c_g^\star$ is

$$J_\phi(c_g^\star) = \frac{\partial \phi}{\partial p}\Big|_{p_t} W_h.$$

Whether the resulting linear map can act differently across different references $c_g^\star, c_{g'}^\star$, and in particular contract along the symbolic subspace $\mathcal{U} = \text{span}\{c_g^\star - c_{g'}^\star\}$, depends entirely on $\partial\phi/\partial p$.

Pointwise scalar activations: state-dependent diagonal. For $\tanh$, $\partial\phi/\partial p = \text{diag}(1 - \tanh^2(p_t))$, which depends elementwise on p_t and therefore on h_{t-1}. ReLU is the binary case $\partial\phi/\partial p = \text{diag}(\mathbf{1}[p_t > 0])$. In both cases the diagonal can scale entries below 1 at some references and not others, so the return-map Jacobian on $\mathcal{U}$ can contract, exactly the conditional restoration that Theorem 1 forbids in the affine case.

Max and min: state-dependent masks. $\max$ and $\min$ take, in each coordinate, either the recurrent term $W_h h_{t-1} + b_h$ or the input term $W_x x_t + b$, so

$$J_{\max}(h_{t-1}) = \mathrm{diag}(s_t)\, W_h,$$

where $s_t \in \{0,1\}^d$ marks the coordinates in which the recurrent term is selected ($\min$ is analogous). The mask depends on h_{t-1}, so the return-map Jacobian on $\mathcal{U}$ can contract at some references and not at others.

GroupSort: piecewise permutations. `GroupSort` with $k = 2$ sorts each pair of entries of p_t. Its Jacobian is a *permutation matrix* that varies with h_{t-1}, so the composed return map mixes $\mathcal{U}$ directions differently at different references. Because the permutation is also 1-Lipschitz in each linear region, the model can implement strict contraction without the gain saturating.

Whole-vector normalizations: nearly identity on $\mathcal{U}$. For LayerNorm with mean μ and variance σ^2,

$$\frac{\partial \phi}{\partial p}\Big|_{p_t} = \frac{1}{\sigma}\Big(I - \tfrac{1}{d}\mathbf{1}\mathbf{1}^\top - \tfrac{1}{d}\widetilde{p}_t\widetilde{p}_t^\top\Big),$$

where $\widetilde{p}_t = (p_t - \mu)/\sigma$. The bracketed projector-like operator (identity minus mean projection minus a rank-1 correction) mixes $\mathcal{U}$ directions only through the rank-1 piece $\widetilde{p}_t\widetilde{p}_t^\top/d$, whose contribution to a generic direction in $\mathcal{U}$ is $O(1/d)$ and so cannot deliver a fixed-margin contraction once $d \gg |G|$; the global prefactor $1/\sigma$ is state-dependent but acts isotropically and cannot discriminate between $\mathcal{U}$ directions. The unit-sphere projection $h \mapsto h/\|h\|$ is similar: its Jacobian $J(h) = (I - \hat{h}\hat{h}^\top)/\|h\|$ is, up to the state-dependent isotropic factor $1/\|h\|$, an orthogonal projector that does not encode per-direction state information. We therefore expect both operators to behave like affine maps on $\mathcal{U}$ at the per-state level, leaving the obstruction of Theorem 1 intact; this is an intuition rather than a formal lemma.

In short, success on the nonlinearity probe depends on whether $\partial\phi/\partial p$ encodes *per-direction* state information. Pointwise and pair operators do; norm operators do not.

E.2 The C_2 edge case

Table 2 shows that parity (C_2) is the only group on which several affine models, notably Negative Mamba and Token-gated RNN, reach the maximum tested length. This success does not imply genuine correction of symbolic-subspace error. Instead, C_2 is unusually tolerant of neutral oscillation: errors can persist and flip sign while remaining within the binary readout margin.

Neutral oscillation in C_2. Let $C_2 = \{e, a\}$ with fixed references $c_e^\star, c_a^\star \in \mathbb{F}^d$, $c_e^\star \neq c_a^\star$, and define

$$F_a(h) = -h + (c_e^\star + c_a^\star), \qquad F_e(h) = h.$$

Then $F_a(c_e^\star) = c_a^\star$, $F_a(c_a^\star) = c_e^\star$, and $F_a^2 = \mathrm{id}$, so the parity transition is realized exactly by an affine involution. For a perturbation $\delta \in \mathcal{U}$,

$$F_a(c_g^\star + \delta) = c_{g\cdot a}^\star - \delta, \qquad F_a^2(c_g^\star + \delta) = c_g^\star + \delta.$$

Thus the perturbation is transported as an oscillation rather than contracted. The prediction can nevertheless remain correct as long as the oscillation stays inside the nearest-centroid margin.

Tolerance shrinks with cycle order. This margin advantage is largest for C_2. In the canonical regular C_k geometry, with

$$c_i^\star = r(\cos(2\pi i/k), \sin(2\pi i/k)),$$

adjacent references are separated by

$$\|c_{i+1}^\star - c_i^\star\| = 2r\sin(\pi/k).$$

Nearest-centroid decoding therefore tolerates only a perturbation on the order of

$$r\sin(\pi/k) \approx \frac{\pi r}{k}$$

before confusing neighboring states. Equivalently, the angular decision sector has width $2\pi/k$, so the tolerated phase error is only π/k. Thus the neutral oscillation that can be harmless for C_2 becomes increasingly fragile as the cycle order grows.

Relation to affine neutrality. This behavior is consistent with Theorem 1. A state-preserving affine return map acts as the identity on $\mathcal{U}$, so it cannot contract symbolic-subspace perturbations. In C_2, the binary margin can hide this lack of contraction over the tested horizons. For larger cycles, the margin is smaller and the return map must remain tightly aligned with the cyclic geometry of the references; residual phase or orbit mismatch is transported rather than corrected.

Return-map gain diagnostic. A practical diagnostic is the spectral radius of $A_s|_{\mathcal{U}}$ over short state-preserving sequences s. Gain near 1, together with non-degenerate centroids, indicates that errors persist without being contracted. Gain > 1 indicates amplification, while gain < 1 can reflect either centroid collapse or genuinely state-dependent correction. On C_2, Negative Mamba and Token-gated RNN have median return-map gain near 1, consistent with the neutral oscillatory route.

E.3 Effect of task complexity

The number of symbolic states enters Corollary 1 only implicitly, through the minimum readout-space separation M. To make this dependence explicit, fix a step t in the regime $M(t) \approx M$, and let $z_g = W_{\mathrm{out}} c_g(t)$, $\bar{z} = |G|^{-1} \sum_g z_g$, $r = \dim \mathrm{span}\{z_g - \bar{z}\} \geq 1$, and $B = \max_g \|z_g - \bar{z}\|_2$. With $M = \min_{g \neq g'} \|z_g - z_{g'}\|_2$, the balls of radius $M/2$ around the z_g are disjoint and lie in the ball of radius $B + M/2$ around $\bar{z}$ within the r-dimensional affine hull of the z_g. Comparing volumes gives $|G|(M/2)^r \leq (B + M/2)^r$, and therefore

$$M \leq \frac{2B}{|G|^{1/r} - 1}.$$

Hence, under a fixed effective rank r and a bounded readout-space radius B, the upper bound on the achievable minimum separation decreases as the number of symbolic states grows. Since Corollary 1 predicts $T_{\mathrm{cross}} \approx \tau M / \|W_{\mathrm{out}} \bar{\eta}\|$, this correspondingly lowers the best separation-limited crossing scale when the residual drift is held approximately fixed.

This does not make T_{cross} monotone in $|G|$: a model can raise its effective rank or scale, and harder transitions can change $\bar{\eta}$. In the corollary, $|G|$ enters only through M.

F Additional empirical results

F.1 Additional models

Table 5 extends Table 2 to four additional architectures absent from the main table: PD-SSM [Terzic et al., 2025b], DeltaNet [Yang et al., 2024], DeltaProduct [Siems et al., 2025], and Low-Rank RNN, a tanh RNN whose recurrent matrix has rank 2, alongside the models from Table 2.

F.2 Additional tasks

Table 6 extends Table 2 to two additional groups: the Abelian group $C_2 \times C_4$ and the non-Abelian alternating group A_4. Both are evaluated on the five recurrent architectures that exhibit non-trivial behavior in Table 2: tanh RNN [Elman, 1990], State-gated RNN, Token-gated RNN, Negative Mamba, and Mamba-3 [Lahoti et al., 2026].

F.3 Per-architecture spread at a fixed step

Figure 3 displays per-step root-mean-square spreads and Figure 1 per-step medians, both with IQR bands over $N = 200$ rollouts. Table 7 tabulates median, Q_1–Q_3, and max over the same rollouts at a single representative step t_{eval}, on the same grid-best checkpoints used in the figures. The metric definitions match the figures: per-rollout latent $\|P_{\mathcal{U}} \delta\|$ divided by the minimum class-pair separation for $r_{\mathrm{err},\mathcal{U}} / r_{\mathrm{sep}}$; unprojected per-rollout error norm for $\|e\| / \|e_{t_0}\|$. We pick t_{eval} as the max-passing length when the model learned the task and the curriculum length 60 otherwise, since the trajectories of failed models diverge numerically beyond that point.

Family	Model	C_2		C_6		S_3	
		L1	L2	L1	L2	L1	L2
Diagonal SSM	Mamba	✗	60	✗	60	✗	✗
	Negative Mamba	1000	1000	100	200	100	200
Complex SSM	Mamba-3	200	300	100	100	✗	60
	AUSSM	1000	✗	200	100	✗	✗
	Simple AUSSM	300	400	100	100	60	100
Sparse SSM	PD-SSM	300	600	400	300	100	60
Delta-rule	DeltaNet	✗	✗	✗	60	✗	✗
	DeltaProduct	✗	✗	✗	60	✗	100
Linear RNN	Linear RNN	✗	100	✗	60	✗	✗
Input-gated RNN	Token-gated RNN	1000	700	300	400	500	1000
State-dependent	tanh RNN	1000	1000	1000	1000	1000	1000
	Low-Rank RNN (r=2, tanh)	1000	1000	✗	1000	1000	1000
	State-gated RNN	1000	1000	1000	1000	1000	1000

Table 5: **Max-passing length of additional models.** We extend Table 2 to additional existing architectures: PD-SSM [Terzic et al., 2025b], DeltaNet [Yang et al., 2024], DeltaProduct [Siems et al., 2025], and a rank-2 tanh RNN. The results show that learned solutions need not realize the full expressivity available to the architecture.

	$C_2 \times C_4$		A_4	
Model	L1	L2	L1	L2
tanh RNN	1000	1000	1000	1000
State-gated RNN	1000	1000	1000	1000
Token-gated RNN	200	300	500	300
Negative Mamba	60	300	✗	✗
Mamba-3	✗	100	✗	✗

Table 6: **Model performances on additional tasks.** We extend Table 2 to $C_2 \times C_4$ and A_4. Both tasks exhibit trends consistent with the main results.

F.4 Seed variability

Table 2 reports the best max-passing length across seeds, following Shakerinava et al. [2026]; Table 8 reports the mean and standard deviation across seeds. The state-dependent models reach 1000 on every seed in all six settings, whereas most affine models that extrapolate beyond the training length vary across seeds.

F.5 Sensitivity of T_{cross} to the threshold and the measurement space

Table 9 recomputes T_{cross} for other thresholds τ and in latent space, without W_{out}. In readout space the correlation with the max-passing length barely changes for τ from 0.5 to 4, so the result does not hinge on $\tau = 1/2$. Latent-space $q(t)$ is larger, so every run crosses $\tau \leq 1$ on the first step after the training length; at thresholds matched to that scale the correlation returns. We use readout space because $\tau = 1/2$ is the nearest-centroid bound in the space the decoder reads.

F.6 Transformer baselines

Full-attention Transformers do not maintain a recurrent hidden state, so our error analysis does not apply to them. For comparison, Table 10 reports the max-passing length of a causal Transformer with rotary position embeddings and a learned beginning-of-sequence token, under the same tasks, curriculum, depths and hyperparameter grid as Table 2. It reaches the training length on C_2 and C_6 but does not extrapolate beyond it, and it does not complete the curriculum on S_3, whereas both

Model	Max-passing length	t_{eval}	$r_{\text{err},\mathcal{U}}/r_{\text{sep}}$ (latent)	$\|e\|/\|e_{t_0}\|$
Mamba	✗	60	1.69 [1.20, 2.27] / 8.21	0.26 [0.25, 0.26] / 0.26
Mamba-3	✗	60	9.62 [7.04, 11.92] / 19.32	0.94 [0.94, 0.94] / 0.95
Negative Mamba	100	100	3.55 [2.79, 6.03] / 26.29	0.47 [0.43, 0.50] / 0.64
Token-gated RNN	500	500	1.39 [1.00, 2.00] / 7.31	4.0×10^{11} [2.6×10^{11}, 6.4×10^{11}] / 2.4×10^{12}
tanh RNN	1000	1000	0.65 [0.55, 0.83] / 1.27	4.9×10^{-6} [4.6×10^{-6}, 5.2×10^{-6}] / 6.5×10^{-6}
State-gated RNN	1000	1000	0.67 [0.52, 0.87] / 1.65	3.9×10^{-5} [3.5×10^{-5}, 4.4×10^{-5}] / 6.2×10^{-5}

Table 7: **Per-architecture error bars for Figure 3 and Figure 1 at t_{eval} on S_3 with one layer, grid-best checkpoints.** Each cell shows median $[Q_1, Q_3]$ / max over $N = 200$ rollouts. t_{eval} is the model's max-passing length when it learned the task, or the curriculum length 60 when it did not. $r_{\text{err},\mathcal{U}}/r_{\text{sep}}$: per-rollout $\|P_{\mathcal{U}}\delta\|$ divided by the minimum class-pair separation (latent). $\|e\|/\|e_{t_0}\|$: per-rollout perturbation-error ratio from the probe of Figure 1, at step t_{eval}.

Model	C_2 L1	C_2 L2	C_6 L1	C_6 L2	S_3 L1	S_3 L2
Mamba	0 ± 0	60 ± 0	0 ± 0	60 ± 0	0 ± 0	0 ± 0
Mamba-3	167 ± 58	140 ± 139	100 ± 0	100 ± 0	0 ± 0	20 ± 35
AUSSM	533 ± 451	0 ± 0	133 ± 58	67 ± 58	0 ± 0	0 ± 0
Simple AUSSM	500 ± 436	187 ± 186	100 ± 0	87 ± 23	40 ± 35	87 ± 23
Negative Mamba	1000 ± 0	1000 ± 0	100 ± 0	133 ± 58	100 ± 0	133 ± 58
Linear RNN	0 ± 0	100 ± 0	0 ± 0	20 ± 35	0 ± 0	0 ± 0
Token-gated RNN	1000 ± 0	300 ± 346	233 ± 58	200 ± 173	367 ± 115	467 ± 462
tanh RNN	1000 ± 0	1000 ± 0	1000 ± 0	1000 ± 0	1000 ± 0	1000 ± 0
State-gated RNN	1000 ± 0	1000 ± 0	1000 ± 0	1000 ± 0	1000 ± 0	1000 ± 0

Table 8: **Mean $\pm$ standard deviation of the max-passing length over three seeds.** Each entry uses the hyperparameter setting (state dimension, learning rate and scheduler) behind the value in Table 2; when several settings reach it, the one with the highest mean. Failure (✗) counts as 0.

state-dependent models reach 1000 on all three tasks. On S_3 it fits the training sequences without generalizing to held-out ones, consistent with reports that Transformers need far more data than recurrent models to learn state tracking [Ebrahimi et al., 2026].

G Examples of state-tracking tasks

G.1 Parity (C_2)

The parity task is equivalent to modulo 2 counting, representing the state transitions within the cyclic group C_2. The algebraic structure of this group is detailed in the Cayley table (Table 11), while an example of parity tracking is provided in Example 2.

Example 2 (C_2)**.** *Let $C_2 = \{0, 1\}$ with transition $g_t = g_{t-1} + x_t \pmod 2$, where each input token $x_t \in \{0, 1\}$. Starting from $g_0 = 0$, the input sequence $1, 1, 0$ produces*

$$g_1 = 1 \text{ (Odd)}, \qquad g_2 = 0 \text{ (Even)}, \qquad g_3 = 0 \text{ (Even)}.$$

The task is to output the running sum modulo 2 at every step.

G.1.1 Sketch of an affine recurrent model for tracking C_2

As demonstrated by Sarrof et al. [2024], Grazzi et al. [2025], solving the parity task with an affine recurrent model with real diagonal transitions, such as Mamba, requires the transition matrix to have at least one negative eigenvalue. To fulfill this condition and track the state of the C_2 group, we can

Space		$\tau = 0.25$	$\tau = 0.5$	$\tau = 1$	$\tau = 2$	$\tau = 4$
Readout	r	$+0.32$	$+0.87$	$+0.90$	$+0.88$	$+0.89$
	n	113	107	98	98	83
Latent	r	–	–	–	$+0.77$	$+0.93$
	n	113	113	113	101	74

Table 9: **Sensitivity of T_{cross} to the threshold τ and the measurement space.** r is the log-log Pearson correlation between T_{cross} and the max-passing length over the n S_3 runs with $\text{mp} \geq 60$ that cross. –: every run crosses on the first step after the training length, so T_{cross} is constant. $\tau = 1/2$ in readout space is the setting of the main text.

	C_2		C_6		S_3	
Model	L1	L2	L1	L2	L1	L2
Transformer	60	60	60	60	✗	✗

Table 10: **Max-passing length of a causal Transformer.** The model uses rotary position embeddings and a learned beginning-of-sequence token and is selected as in Table 2. 60 denotes a model that passes only at the training length; ✗ denotes failure to complete the curriculum.

design a minimalist 1-dimensional recurrent model as follows:

$$h_t = A(x_t)h_{t-1}, \quad h_0 = 1 \tag{4}$$

$$A(x_t) = \begin{cases} 1 & \text{if } x_t = 0 \\ -1 & \text{if } x_t = 1 \end{cases} \tag{5}$$

Here, $h_t \in \{1, -1\}$ represents the internal state encoding the cumulative parity at time step t, initialized at $h_0 = 1$ corresponding to the identity element. The input-dependent transition parameter $A(x_t)$ applies the required negative eigenvalue (-1) whenever an active token ($x_t = 1$) is encountered. Consequently, each occurrence of $x_t = 1$ inverts the sign of the hidden state, alternating between the two states of C_2. At the end of the sequence, the final state h_T determines the parity: $h_T = 1$ indicates an even number of ones, whereas $h_T = -1$ indicates an odd number.

G.2 Cyclic group (C_3)

More generally, tracking operations within a cyclic group C_k is equivalent to modulo k counting, which tracks the cumulative sum of inputs wrapping around a finite set of k distinct states in an Abelian (commutative) group. For the specific case of C_3, this corresponds to modulo 3 counting. The algebraic structure of C_3 is detailed in the Cayley table (Table 12), while an example of this state tracking is provided in Example 3.

Example 3 (C_3). *Let $C_3 = \{0, 1, 2\}$ with transition $g_t = g_{t-1} + x_t \pmod 3$, where each input token $x_t \in \{0, 1, 2\}$. Starting from $g_0 = 0$, the input sequence $1, 2, 1$ produces*

$$g_1 = 1, \qquad g_2 = 0, \qquad g_3 = 1.$$

The task is to output the running sum modulo 3 at every step.

G.2.1 Sketch of an affine recurrent model for tracking C_3

As proven by Grazzi et al. [2025], a linear recurrent model can count modulo k (for non-power-of-two k, such as $k = 3$) only if its transition matrix possesses at least one eigenvalue with a non-zero imaginary part ($\lambda \notin \mathbb{R}$). To fulfill this requirement and track the state of the cyclic group C_3, the model's capacity must be extended beyond the real number line to the complex domain. Specifically, we can formulate a minimalist 1-dimensional complex-valued recurrent model utilizing the 3rd roots

$\cdot$	0	1
0	0	1
1	1	0

Table 11: **Cayley table of C_2.**

$\cdot$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

Table 12: **Cayley table of C_3.**

of unity:

$$h_t = A(x_t)h_{t-1}, \quad h_0 = 1 + 0j \tag{6}$$

$$A(x_t) = \exp\left(j\frac{2\pi}{3}x_t\right) \tag{7}$$

Here, $h_t \in \mathbb{C}$ represents the internal complex state at time step t, initialized at $h_0 = 1 + 0j$, which corresponds to the identity element (zero rotation). The input $x_t \in \{0, 1, 2\}$ indicates the degree of cyclic shift. The transition parameter $A(x_t)$ acts as a phase modulator, shifting the phase of the hidden state by exactly $120°$ ($\frac{2\pi}{3}$ radians) multiplied by the input x_t. By operating on the unit circle in the complex plane, the model avoids exponential decay or explosion. At the end of the sequence, the final state h_T encodes the modulo 3 sum of the inputs: $h_T = 1$ indicates 0, $h_T = e^{j2\pi/3}$ indicates 1, and $h_T = e^{j4\pi/3}$ indicates 2.

G.3 Symmetric group (S_3)

The symmetric group S_k comprises all possible permutations of a set containing k distinct elements. The order of the group, representing the total number of permutations, is given by $|S_k| = k!$. While S_1 and S_2 are Abelian (commutative), S_k exhibits non-commutative properties for all $k \geq 3$. Notably, S_3 is the smallest non-Abelian symmetric group, consisting of the six elements $\{e, (12), (23), (13), (123), (132)\}$. Example 4 shows the non-Abelian nature of S_3.

In this set, e denotes the identity element, which represents the permutation where all elements remain in their original positions. The cycle notation $(a\,b)$ represents a swapping that interchanges the positions of elements a and b while leaving the third element unchanged. In contrast, a 3-cycle such as $(a\,b\,c)$ denotes a cyclic permutation where the elements are shifted in a closed loop: a moves to b, b moves to c, and c returns to a. The Cayley table of symmetric group S_3 is given in Table 13.

$\cdot$	e	(12)	(13)	(23)	(123)	(132)
e	e	(12)	(13)	(23)	(123)	(132)
(12)	(12)	e	(132)	(123)	(23)	(13)
(13)	(13)	(123)	e	(132)	(12)	(23)
(23)	(23)	(132)	(123)	e	(13)	(12)
(123)	(123)	(13)	(23)	(12)	(132)	e
(132)	(132)	(23)	(12)	(13)	e	(123)

Table 13: **Cayley table of S_3.** The entry in row a and column b is $a \cdot b$, the permutation that applies b first and then a.

Input Token	Decomposition	Input Indicators		Layer States	
		$x_t^{(1)}$ (C_2)	$x_t^{(2)}$ (C_3)	$y_t^{(1)}$	$h_t^{(2)}$
e	s^0r^0	0	0	1	1
(123)	s^0r^1	0	1	1	$e^{j2\pi/3}$
(132)	s^0r^2	0	2	1	$e^{j4\pi/3}$
(12)	s^1r^0	1	0	-1	1
(23)	s^1r^1	1	1	-1	$e^{-j2\pi/3}$
(13)	s^1r^2	1	2	-1	$e^{-j4\pi/3}$

Table 14: **Mapping of S_3 elements.** The table demonstrates the bijective relationship between the input tokens, the decomposed indicators, and the internal model states $(y_t^{(1)}, h_t^{(2)})$.

Example 4 (Non-Abelian Property of S_3)**.** *To illustrate the non-Abelian nature of S_3, we compare two sequences with identical tokens in different orders. Let the input tokens be the generators $t_1 = (12)$ and $t_2 = (23)$.*

1. *Applying (12) followed by (23):*

$$g_0 = e \xrightarrow{(12)} g_1 = (12) \xrightarrow{(23)} g_2 = (123)$$

2. *Applying (23) followed by (12):*

$$g_0' = e \xrightarrow{(23)} g_1' = (23) \xrightarrow{(12)} g_2' = (132)$$

Since the final states differ ($g_2 \neq g_2'$), the group S_3 is non-Abelian.

G.3.1 Sketch of an affine recurrent model for tracking S_3

Here, we revisit the theoretical result of Shakerinava et al. [2026]. Refer to the paper for a detailed analysis. Recall that any operation in S_3 can be constructed using two fundamental generators: a transposition $s = (12)$, which swaps two elements, and a rotation $r = (123)$, which shifts the elements cyclically. These generators correspond to the parity group C_2 and the cyclic group C_3, respectively. To process these operations, we assume each input token at time t can be decomposed into two corresponding attributes: a C_2 indicator $x_t^{(1)} \in \{0, 1\}$ and a C_3 indicator $x_t^{(2)} \in \{0, 1, 2\}$. The model tracks the overall group state through the following layers:

Layer 1: C_2 parity tracker. The first layer operates as a 1-dimensional real-valued SSM that tracks the cumulative parity of the swapping operations, modeling the C_2 component.

- **State transition:** Let $h_t^{(1)} \in \mathbb{R}$ be the hidden state initialized at $h_0^{(1)} = 1$. The input-dependent transition parameter is defined as $A_t^{(1)} = (-1)^{x_t^{(1)}}$.
- **Update rule:** $h_t^{(1)} = A_t^{(1)} \cdot h_{t-1}^{(1)}$
- **Output:** $y_t^{(1)} = h_t^{(1)} \in \{1, -1\}$. This output indicates whether the current accumulated state is in a normal (1) or flipped (-1) orientation, capturing the C_2 state.

Layer 2: conditional C_3 accumulator. The second layer operates as a 1-dimensional complex-valued SSM that tracks the rotational operations (C_3 component) in the complex plane. The phase shift in this layer is modulated by the output of Layer 1.

- **State transition:** Let $h_t^{(2)} \in \mathbb{C}$ be initialized at $h_0^{(2)} = 1$. The transition parameter $A_t^{(2)}$ is conditioned on the C_2 state $y_t^{(1)}$:

$$A_t^{(2)} = \exp\left(j\frac{2\pi}{3} \cdot x_t^{(2)} \cdot y_t^{(1)}\right) \tag{8}$$

- **Update rule:** $h_t^{(2)} = A_t^{(2)} \cdot h_{t-1}^{(2)}$

In summary, the internal layer states of the proposed 2-layer affine recurrent model track all six distinct elements of S_3. As demonstrated in Table 14, the unique combinations of these states provide a bijective mapping to the group elements without ambiguity. In practical implementations, although models do not explicitly partition these layers, residual connections and high-dimensional state spaces integrate these features, so the final layer can decode such non-commutative dynamics.

H Model details

We map each architecture of Table 2, and related ones, to the canonical form defined in Equation (1) and Equation (2). For each model we specify (i) the **canonical-form** realization, with emphasis on the geometric structure of the transition $\mathbf{A}(x_t)$; (ii) the architectural **family**; and (iii) explicit **violations** of the canonical form, when present.

Choice of main-experiment models. The architectures evaluated in the main experiments (Table 15, highlighted rows) span the canonical form's $(\mathbf{A}, g, \phi)$ axes jointly: diagonal contractive (Mamba), signed diagonal (Negative Mamba), damped complex rotation (Mamba-3), unitary (AUSSM, Simple AUSSM), dense linear (Linear RNN), dense pointwise-nonlinear (tanh RNN), input-gated (Token-gated RNN), and state-gated (State-gated RNN). Each realizes the canonical form, except for one violation documented below (the exponential-trapezoidal input injection of Mamba-3). Architectures outside the main experiments are listed in Table 15 for reference: DeltaNet and DeltaProduct (reported in Appendix F.1) violate the canonical form via matrix-valued state, while S4 is the trivial input-independent case (a valid but degenerate canonical-form realization).

H.1 State-space models

S4 [Gu et al., 2022]. **Canonical form.** Constant transition: $\mathbf{A}(x_t) = (\mathbf{I} - \Delta/2\,\mathbf{A})^{-1}(\mathbf{I} + \Delta/2\,\mathbf{A})$, time- and input-invariant; $\mathbf{A}$ is HiPPO-initialized in Normal-Plus-Low-Rank form for stable long-range dependence. **Family.** Linear time-invariant structured state-space model.

Mamba [Gu and Dao, 2024]. **Canonical form.** Diagonal contractive: $\mathbf{A}(x_t) = \exp(\Delta_t \mathbf{A})$ with real-valued diagonal $\mathbf{A} \prec 0$ and $\Delta_t = \mathrm{softplus}(\Delta_{\mathrm{bias}} + W_\Delta x_t)$; each entry of $\mathbf{A}(x_t)$ lies in $(0, 1)$. **Family.** Diagonal selective state-space model.

Negative Mamba [Grazzi et al., 2025]. **Canonical form.** Signed diagonal: $\mathbf{A}(x_t) = 2\exp(\Delta_t \mathbf{A}) - \mathbf{I}$, sharing the same diagonal $\mathbf{A} \prec 0$ as Mamba; entries of $\mathbf{A}(x_t)$ lie in $(-1, 1)$. **Family.** Diagonal selective state-space model with signed transitions.

Mamba-3 [Lahoti et al., 2026]. **Canonical form.** Damped complex rotation: $\mathbf{A}(x_t) = \exp(\Delta_t \mathbf{A}_t)$ with complex-valued diagonal $\mathbf{A}_t = \mathbf{A}_{\mathrm{re}}(x_t) + i\boldsymbol{\Theta}(x_t)$ (real decay $\mathbf{A}_{\mathrm{re}}(x_t) \prec 0$ and input-dependent rotation frequencies $\boldsymbol{\Theta}(x_t)$); entries of $\mathbf{A}(x_t)$ lie strictly inside the unit disk. **Family.** Complex-diagonal selective state-space model. **Violations.** Exponential-trapezoidal discretization makes the input injection b a linear combination of $\mathbf{B}_{t-1}x_{t-1}$ and $\mathbf{B}_t x_t$, whereas canonical b takes x_t only.

AUSSM [Karuvally et al., 2025]. **Canonical form.** Unitary: $\mathbf{A}(x_t) = \exp(\Delta_t \mathbf{A}_t)$ with real skew-symmetric input-dependent $\mathbf{A}_t$, so $\mathbf{A}(x_t)$ is orthogonal at every step (equivalently, its eigenvalues lie on the complex unit circle). **Family.** Adaptive unitary state-space model.

Simple AUSSM [Shakerinava et al., 2026]. **Canonical form.** Unitary, complex-diagonal: $\mathbf{A}(x_t) = \exp(i\,\boldsymbol{\Lambda}(x_t))$ with input-dependent real $\boldsymbol{\Lambda}(x_t)$, so the diagonal log is purely imaginary and entries of $\mathbf{A}(x_t)$ have unit modulus. AUSSM's input-dependent step size Δ_t is dropped, since it is not needed for representing groups [Shakerinava et al., 2026]. **Family.** Complex-diagonal unitary state-space model.

H.2 Recurrent networks

Linear RNN. **Canonical form.** Constant dense transition: $\mathbf{A}(x_t) = \mathbf{W}_h$, $\phi = \mathrm{id}$, $g \equiv 1$. **Family.** Linear recurrent network.

Type	Model	$g(h_{t-1}, x_t)$	$\phi(\cdot)$	$\mathbf{A}(x_t)$	$b(x_t)$	$\mathrm{dec}(h_t, x_t)$
SSM	S4 [Gu et al., 2022]	1	id	$(\mathbf{I} - \Delta/2 \cdot \mathbf{A})^{-1}$ $\times(\mathbf{I} + \Delta/2 \cdot \mathbf{A})$	$(\mathbf{I} - \Delta/2 \cdot \mathbf{A})^{-1}$ $\times\Delta\mathbf{B}x_t$	$\mathbf{C}h_t + \mathbf{D}x_t$
	Mamba [Gu and Dao, 2024]	1	id	$e^{\Delta_t \mathbf{A}}$	$\Delta_t \mathbf{B}_t x_t$	$\mathbf{C}_t h_t + \mathbf{D}x_t$
	Negative Mamba [Grazzi et al., 2025]	1	id	$2e^{\Delta_t \mathbf{A}} - \mathbf{I}$	$\Delta_t \mathbf{B}_t x_t$	$\mathbf{C}_t h_t + \mathbf{D}x_t$
	Mamba-3 [Lahoti et al., 2026]	1	id	$e^{\Delta_t \mathbf{A}}$	$(1-\lambda_t)\Delta_t e^{\Delta_t \mathbf{A}} \mathbf{B}_{t-1} x_{t-1}$ $+\lambda_t \Delta_t \mathbf{B}_t x_t$	$\mathbf{C}_t h_t + \mathbf{D}x_t$
	AUSSM [Karuvally et al., 2025]	1	id	$e^{\Delta_t \mathbf{A}_t}$	$\Delta_t \mathbf{B}x_t$	$\mathbf{C}h_t + \mathbf{D}x_t$
	Simple AUSSM [Shakerinava et al., 2026]	1	id	$e^{\mathbf{A}_t}$	$\mathbf{B}x_t$	$\mathbf{C}h_t + \mathbf{D}x_t$
RNN	$\tanh$ RNN [Elman, 1990]	1	$\tanh$	$\mathbf{W}_h$	$\mathbf{W}_x x_t + b_h$	h_t
	Linear RNN	1	id	$\mathbf{W}_h$	$\mathbf{W}_x x_t + b_h$	h_t
	Token-gated RNN	$\sigma(\mathbf{W}_g x_t + b_g)$	id	$\mathbf{W}_h$	$\mathbf{W}_x x_t + b_h$	h_t
	State-gated RNN	$\sigma(\mathbf{W}_g x_t + \mathbf{U}_g h_{t-1}$ $+b_g)$	id	$\mathbf{W}_h$	$\mathbf{W}_x x_t + b_h$	h_t
	DeltaNet [Yang et al., 2024]	1	id	$\mathbf{I} - \beta_t k_t k_t^\top$	$\beta_t k_t v_t^\top$	$h_t^\top q_t$
	DeltaProduct [Siems et al., 2025]	1	id	$\prod_{j=n_h}^{1} (\mathbf{I} - \beta_{t,j} k_{t,j} k_{t,j}^\top)$	$\sum_{j=1}^{n_h} \left(\prod_{k=n_h}^{j+1} (\mathbf{I} - \beta_{t,k} k_{t,k} k_{t,k}^\top) \right)$ $\times \beta_{t,j} k_{t,j} v_{t,j}^\top$	$h_t^\top q_t$

Table 15: **Recursive models mapped to the canonical form.** Comparison of various recurrent architectures mapped to our canonical form $h_t = \phi(g_t \odot (\mathbf{A}_t h_{t-1}) + b_t)$. Mamba-3 utilizes an Exponential-Trapezoidal rule where the $b(x_t, x_{t-1})$ depends on both current and previous inputs. To maintain structural consistency, matrix-valued models such as DeltaNet [Yang et al., 2024] and DeltaProduct [Siems et al., 2025] are represented via a transpose transformation $h_t = \mathbf{S}_t^\top$, converting their original right-multiplication state updates into left-multiplication.

tanh RNN [Elman, 1990]. **Canonical form.** Constant dense transition with elementwise nonlinearity: $\mathbf{A}(x_t) = \mathbf{W}_h$, $\phi(z) = \tanh(z)$. The activation makes the per-step Jacobian state-dependent through $\tanh'$. **Family.** Elman recurrent network.

Token-gated RNN. **Canonical form.** Constant dense transition with input-only gate: $\mathbf{A}(x_t) = \mathbf{W}_h$, $g(x_t) = \sigma(\mathbf{W}_g x_t + b_g)$, $\phi = \mathrm{id}$. The gate is independent of h_{t-1}, so the update remains affine in h_{t-1}. **Family.** Input-gated linear recurrent network.

State-gated RNN. **Canonical form.** Constant dense transition with state-and-input gate: $\mathbf{A}(x_t) = \mathbf{W}_h$, $g(h_{t-1}, x_t) = \sigma(\mathbf{W}_g x_t + \mathbf{U}_g h_{t-1} + b_g)$, $\phi = \mathrm{id}$. State dependence in the gate makes the per-step Jacobian state-dependent. **Family.** State-gated recurrent network.

LSTM [Hochreiter and Schmidhuber, 1997]. **Canonical form.** The cell-state update $c_t = f_t \odot c_{t-1} + i_t \odot \tanh(\mathbf{U}_c h_{t-1} + \mathbf{W}_c x_t + b_c)$ realizes the canonical form on c_t, with the forget gate $f_t = \sigma(\mathbf{U}_f h_{t-1} + \mathbf{W}_f x_t + b_f)$ as the state-and-input gate and identity transition ($\mathbf{A} = \mathbf{I}$). **Family.** Long short-term memory. **Violations.** (i) Two coupled state variables (cell state c_t and hidden state h_t); the canonical form has a single state. Consequently, the input injection $i_t \odot \tanh(\mathbf{U}_c h_{t-1} + \mathbf{W}_c x_t + b_c)$ for c_t depends on h_{t-1} (the other state variable), whereas canonical $b(x_t)$ takes only x_t. (ii) Output gating $h_t = o_t \odot \tanh(c_t)$ applies a second nonlinear transformation after the canonical update, outside the $\phi(g \odot \mathbf{A}h_{t-1} + b)$ template.

GRU [Cho et al., 2014]. **Canonical form.** Convex-combination update (Cho's convention): $h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tanh\big(\mathbf{U}(r_t \odot h_{t-1}) + \mathbf{W}x_t\big)$, with update gate $z_t = \sigma(\mathbf{U}_z h_{t-1} + \mathbf{W}_z x_t)$ in the role of the canonical state-and-input gate against identity transition ($\mathbf{A} = \mathbf{I}$). **Family.** Gated recurrent unit. **Violations.** (i) The candidate term $(1 - z_t) \odot \tanh(\cdot)$ plays the role of b but depends on h_{t-1} (via the inner $\mathbf{U}(r_t \odot h_{t-1})$), whereas canonical $b(x_t)$ takes only x_t. (ii) The reset gate $r_t = \sigma(\mathbf{U}_r h_{t-1} + \mathbf{W}_r x_t)$ multiplies h_{t-1} *inside* the candidate, before the transition matrix $\mathbf{U}$; the canonical form admits gating only outside the transition. (iii) The update has the form $g \odot h_{t-1} + (1 - g) \odot \tanh(\cdot)$ rather than $\phi(g \odot \mathbf{A}h_{t-1} + b)$, with the nonlinearity $\tanh$ applied only to the candidate, not to the full update.

DeltaNet [Yang et al., 2024]. **Canonical form.** Generalized rank-1 Householder (delta-rule update): $\mathbf{A}(x_t) = \mathbf{I} - \beta_t k_t k_t^\top$, with sigmoid-gated scalar $\beta_t = \sigma(W_\beta x_t) \in (0, 1)$. The factor reduces to a true Householder reflection only at $\beta_t = 2/\|k_t\|^2$; in general it is a learnable rank-1 perturbation of the identity. **Family.** Linear-attention recurrent network. **Violations.** The state is matrix-valued ($S_t \in \mathbb{R}^{d_v \times d_k}$); we represent it as $h_t = S_t^\top$ to match the canonical left-multiplication form.

DeltaProduct [Siems et al., 2025]. **Canonical form.** Product of n_h generalized Householders: $\mathbf{A}(x_t) = \prod_{j=n_h}^{1} (\mathbf{I} - \beta_{t,j} k_{t,j} k_{t,j}^\top)$, each factor a learnable rank-1 update with sigmoid-gated $\beta_{t,j}$. With sufficient n_h and $\beta_{t,j}$ in the Householder regime, the product can represent any orthogonal matrix (Cartan–Dieudonné); in general it spans a wider set. **Family.** Multi-step linear-attention recurrent network. **Violations.** Same matrix-valued state as DeltaNet.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We demonstrate our idea and contribution in the abstract and Introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We describe the limits in Limitations, including the exact-realization assumptions of the theory, best-of-seed reporting, and the scope of synthetic group tracking.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide the theoretical derivation in Proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We explain our setting, including model architectures and the training recipe, in Experimental details and Model details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our code is publicly available through the project page. Our description in Experimental details is sufficient to reproduce the results.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We illustrate all the details in Experimental details.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Trajectory figures report means (Figure 2), root-mean-square spreads (Figure 3) and medians (Figure 1), with per-step IQR bands over $N = 200$ rollouts; the T_{cross} correlation analysis in Section 4.5 reports a Pearson coefficient with its p-value and a 95% bootstrap CI on the median $q(t)$ at $t = \text{mp}$. Per-architecture error bars (median, Q_1–Q_3, max over rollouts) for the trajectory quantities are tabulated in Appendix F.3 (Table 7). Table 2 follows the literature convention of reporting the best across seeds; Table 8 reports the mean and standard deviation across seeds, and Table 9 the sensitivity of the T_{cross} correlation to the threshold and measurement space.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report our compute resources in Experimental details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work complies with all principles mentioned in the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work is a theoretical and empirical study of recurrent state-tracking dynamics in controlled synthetic tasks. We do not identify direct negative societal impacts beyond the general risks associated with advances in machine learning.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The work poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All architectures used in this work are credited via citations in the main text and in Appendices F.1 and H; Appendix C lists the external libraries.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The code and its documentation are at `https://github.com/JiwanChung/rethinking-state-tracking`.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing or human subject research is conducted in this paper.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No crowdsourcing or human subject research is conducted in this paper.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not an important, original, or non-standard component of the core method; Appendix C notes their use for code and writing assistance.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.