MEMORY BY DESIGN: BAYESIAN LAYERS FOR SEQUENCE MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

To battle softmax attention's linearly growing inference memory cost, current recurrent architectures compress history into a fixed-dimensional state. Their recurrence equations alone, however, leave the underlying assumptions about memory implicit. We introduce *design models*: auxiliary probabilistic models specified by an initial distribution and input-dependent factors encoding memory dynamics and incoming evidence. Once these assumptions are specified, Bayesian filtering determines the recurrent update under the model. Specifically, a linear-Gaussian design model yields the **Bayesian Layer**, which maintains a memory estimate and its covariance via a Kalman update. Resetting this covariance recovers Delta-rule updates, while a different design model yields the additive recurrences of linear attention, GLA, and Mamba-2. These connections make explicit the assumptions underlying familiar memory updates. We illustrate how the geometry of the Bayesian Layer's uncertainty shapes subsequent updates and helps preserve earlier associations. In associative recall tasks, the Bayesian Layer improves recall on longer sequences and with greater overlap between keys than seen during training. We further evaluate its performance on multi-query associative recall (MQAR) and language modeling. Distilling Bayesian Layers into a pretrained Gated DeltaNet improves retrieval accuracy on RULER over a continued-training control.

1 INTRODUCTION

Efficient recurrent sequence layers compress a growing token sequence into a finite-dimensional state that is updated as new inputs arrive. Deciding how to update this state is a central problem in recurrent architecture design: each update must incorporate new information while preserving information from earlier inputs that remains relevant. Linear attention, GLA, Mamba, and Delta-rule layers address this problem through different recurrence equations (Katharopoulos et al., 2020; Yang et al., 2024a; Gu & Dao, 2024; Schlag et al., 2021; Sun et al., 2023; Qin et al., 2024; Liu et al., 2025; Yang et al., 2025; Siems et al., 2025; Peng et al., 2023; 2024; Behrouz et al., 2026; De et al., 2024; Beck et al., 2024), while other work derives such rules from online regression and Bayesian inference (von Oswald et al., 2026; Bonnet et al., 2026; Shaj et al., 2026). The recurrence equations alone, however, do not make explicit the assumptions about what is stored and how new evidence relates to it.

We make these assumptions explicit through a *design model*: an auxiliary probabilistic model of a latent memory state. The model specifies an initial distribution and input-dependent factors that encode state dynamics and incoming evidence. Consequently, Bayesian filtering determines the recurrent update from these assumptions. The belief parameters needed for subsequent updates and readouts form the implemented recurrent state, while a separate query-dependent readout produces the layer output. This framework thus links a recurrent update to explicit assumptions about memory.

The *Bayesian Layer* (BL) instantiates this framework with linear-Gaussian latent-memory dynamics and write likelihood. Its recurrent state contains the posterior mean of a latent memory and a dense covariance that tracks uncertainty over memory directions. Exact filtering uses this covariance to set the direction and gain of each write. Repeated evidence therefore reduces the gain of later corrections along directions that have become less uncertain.

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

The framework also provides a way to relate existing recurrent architectures through the assumptions underlying their memory updates (Table 1). Resetting the Bayesian Layer's predicted covariance to an isotropic matrix at each step recovers Delta-rule updates (Schlag et al., 2021; Yang et al., 2024b; 2025; Kimi Team, 2025). Under a separate design model, exact filtering yields the additive recurrences of linear attention (Katharopoulos et al., 2020), GLA (Yang et al., 2024a), and Mamba-2/SSD (Dao & Gu, 2024). These connections distinguish approximations to the filtering recurrence from differences in the underlying design model.

Our contributions are:

1. We introduce the **design-model framework**, which derives memory updates for recurrent architectures through Bayesian filtering and makes explicit the assumptions behind Delta-rule and additive recurrences (§§2–3).
2. We derive the **Bayesian Layer** and analyze how tracking uncertainty shapes the geometry of memory updates, helping preserve associations under interference (§2).
3. We evaluate the layer on **associative recall tasks, MQAR, and language modeling**, including long-context retrieval after distillation into a pretrained Gated DeltaNet (§4).

2 THE DESIGN-MODEL FRAMEWORK

We write a recurrent sequence layer in terms of its update function f. At each step, the current input $\mathbf{x}_t$ and previous state $\mathbf{h}_{t-1}$ determine the updated state,

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t). \tag{1}$$

Choosing f determines how each new input modifies a finite-dimensional representation of the past. Modern recurrent architectures typically specify the memory-update rule f directly through computational structure that supports parallel training and mechanisms such as gating and error-correcting updates (Dao & Gu, 2024; Yang et al., 2025). These choices shape how information accumulates in memory, but their underlying assumptions can be difficult to identify and compare from the recurrences alone.

We propose making these assumptions the starting point for architectural design, through a probabilistic model of memory, which we call a design model. The model specifies an initial distribution and input-dependent factors that encode memory dynamics and incoming evidence. Bayesian filtering determines the optimal recurrent update under the model (Särkkä, 2013), providing a systematic way to construct updates and interpret their behavior.

Let $\mathbf{S}_t$ denote the latent memory state. A design model specifies an initial distribution $\mu_0(\mathbf{S}_0)$ and nonnegative, input-dependent pairwise potentials $\Psi_t(\mathbf{S}_{t-1}, \mathbf{S}_t; \mathbf{x}_t)$. For each input sequence $\mathbf{x}_{1:t}$, these define

$$\pi_t(\mathbf{S}_{0:t}; \mathbf{x}_{1:t}) = \frac{1}{Z_t(\mathbf{x}_{1:t})}\, \mu_0(\mathbf{S}_0) \prod_{s=1}^{t} \Psi_s(\mathbf{S}_{s-1}, \mathbf{S}_s; \mathbf{x}_s), \tag{2}$$

where $Z_t(\mathbf{x}_{1:t})$ is a strictly positive normalizing constant. The inputs parameterize this distribution; inference is performed over the latent states. The design model does not specify a probability distribution for the raw input sequence $\mathbf{x}_{1:t}$.

Marginalizing the preceding states gives the filtering distribution μ_t and its Bayesian filtering recursion (Murphy, 2023):

$$\mu_t(\mathbf{S}_t) = \int \pi_t(\mathbf{S}_{0:t}; \mathbf{x}_{1:t})\, d\mathbf{S}_{0:t-1} \propto \int \Psi_t(\mathbf{S}_{t-1}, \mathbf{S}_t; \mathbf{x}_t) \mu_{t-1}(\mathbf{S}_{t-1})\, d\mathbf{S}_{t-1}. \tag{3}$$

Once the factors are specified, this recursion determines how each new input updates the belief over memory.

To obtain a layer output from this posterior, we choose a readout function $g(\mathbf{S}_t, \mathbf{x}_t)$ specifying the quantity to estimate from the state. Under squared-error loss, its posterior mean gives the Bayes-optimal output:

$$\mathbf{y}_t = \operatorname*{argmin}_{\mathbf{a}} \; \mathbb{E}_{\mu_t}\left[\|\mathbf{a} - g(\mathbf{S}_t, \mathbf{x}_t)\|^2\right] = \mathbb{E}_{\mu_t}\left[g(\mathbf{S}_t, \mathbf{x}_t)\right]. \tag{4}$$

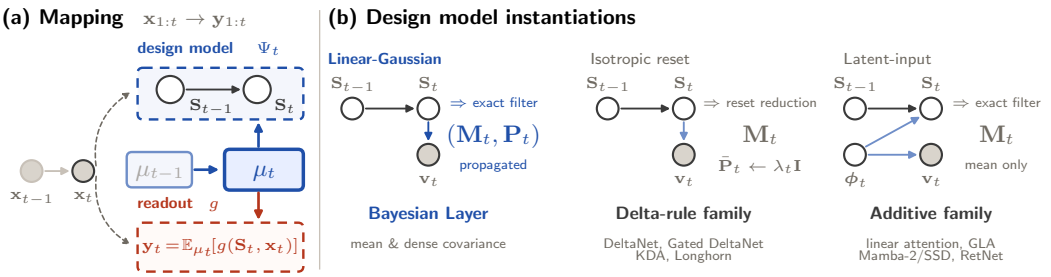


Figure 1: **Design-model framework. (a)** Input parameterizes the design model, Bayesian filtering updates the belief over the latent memory state, and readout produces the layer output. **(b)** Different design models and filtering choices yield different recurrent updates. Exact filtering under the linear-Gaussian model gives the Bayesian Layer; isotropic covariance reset gives the *Delta-rule family*; exact filtering under the latent-input model gives the *additive family*.

Thus the design model determines the filtering update, while the readout specifies what the layer estimates.

With the model and readout specified, a recurrent implementation requires the belief parameters needed for subsequent filtering updates and readouts to admit a closed, fixed-size recurrence. These parameters form the recurrent state $\mathbf{h}_t$ in (1), whose update is determined by filtering.

The architectural choices in this framework are therefore the initial distribution μ_0, the form of the input-dependent pairwise potentials Ψ_t, and the readout g (Figure 1). In the following sections, we illustrate these choices with the Bayesian Layer design model, whose factors encode dynamics and associative relationships, and a latent-input design model, whose factors encode evidence about independent write coefficients.

2.1 The Bayesian Layer design model

Matrix-memory architectures represent key–value associations through a shared linear map, $\mathbf{k} \mapsto \mathbf{S}_t^\top \mathbf{k}$ (Amari, 1972; Yang et al., 2025; von Oswald et al., 2026). We use this representation to construct the *Bayesian Layer design model*, in which each value $\mathbf{v}_t$ is a noisy observation of $\mathbf{S}_t^\top \mathbf{k}_t$ and the memory follows input-dependent dynamics between observations.

Definition 1 (Bayesian Layer design model)**.** *Let $\mathbf{S}_t \in \mathbb{R}^{D\times m}$. Learned functions of the input $\mathbf{x}_t$ produce a key $\mathbf{k}_t = \mathbf{k}(\mathbf{x}_t) \in \mathbb{R}^D$, a value $\mathbf{v}_t = \mathbf{v}(\mathbf{x}_t) \in \mathbb{R}^m$, a transition matrix $\mathbf{A}_t = \mathbf{A}(\mathbf{x}_t) \in \mathbb{R}^{D\times D}$, and noise scales $r_t = r(\mathbf{x}_t) > 0$ and $\ell_t = \ell(\mathbf{x}_t) \geq 0$.*

The initial distribution μ_0 has independent Gaussian columns with mean zero and covariance $\sigma_0^2 \mathbf{I}_D$, where $\sigma_0^2 > 0$. For $\ell_t > 0$, we define

$$\Psi_t(\mathbf{S}_{t-1}, \mathbf{S}_t; \mathbf{x}_t) = \exp\left[-\underbrace{\frac{\|\mathbf{S}_t - \mathbf{A}_t \mathbf{S}_{t-1}\|_F^2}{2\ell_t^2}}_{\textit{dynamics}} - \underbrace{\frac{\|\mathbf{v}_t - \mathbf{S}_t^\top \mathbf{k}_t\|_2^2}{2r_t^2}}_{\textit{incoming evidence}} \right]. \tag{5}$$

Equivalently, the design model has the joint representation

$$p_t(\mathbf{S}_t, \mathbf{v}_t \mid \mathbf{S}_{t-1}) = p_t(\mathbf{S}_t \mid \mathbf{S}_{t-1})\, p_t(\mathbf{v}_t \mid \mathbf{S}_t) \propto \Psi_t(\mathbf{S}_{t-1}, \mathbf{S}_t; \mathbf{x}_t), \tag{6}$$

where the transition has independent Gaussian columns with means $\mathbf{A}_t \mathbf{S}_{t-1}^{(j)}$ and covariance $\ell_t^2 \mathbf{I}_D$, and $p_t(\mathbf{v}_t \mid \mathbf{S}_t) = \mathcal{N}(\mathbf{v}_t; \mathbf{S}_t^\top \mathbf{k}_t, r_t^2 \mathbf{I}_m)$. Here the joint density is evaluated at the supplied key–value pair $(\mathbf{k}_t, \mathbf{v}_t)$, with the input-dependent parameters held fixed.

The two noise scales express different assumptions. The observation variance r_t^2 describes discrepancies between an observed value $\mathbf{v}_t$ and the memory prediction $\mathbf{S}_t^\top \mathbf{k}_t$. The process variance ℓ_t^2 describes deviations of the memory from its expected evolution $\mathbf{A}_t \mathbf{S}_{t-1}$. Thus an imperfect fit to an association and a change in the underlying memory have distinct roles in the model.

For a given input, the dynamics and observations are linear-Gaussian in $\mathbf{S}_t$ even when the learned functions of $\mathbf{x}_t$ are nonlinear. Filtering therefore preserves independent Gaussian columns with a

shared covariance (Appendix B):

$$\mu_t(\mathbf{S}_t) = \prod_{j=1}^{m} \mathcal{N}\left(\mathbf{S}_t^{(j)}; \mathbf{M}_t^{(j)}, \mathbf{P}_t\right). \tag{7}$$

The posterior mean $\mathbf{M}_t \in \mathbb{R}^{D\times m}$ and covariance $\mathbf{P}_t \in \mathbb{R}^{D\times D}$, shared across columns, form the recurrent state of the Bayesian Layer, $\mathbf{h}_t = (\mathbf{M}_t, \mathbf{P}_t)$. Equivalently, $\mathrm{vec}(\mathbf{S}_t) \sim \mathcal{N}(\mathrm{vec}(\mathbf{M}_t), \mathbf{I}_m \otimes \mathbf{P}_t)$.

As in self-attention (Vaswani et al., 2017), we compute a query $\mathbf{q}_t = \mathbf{q}(\mathbf{x}_t) \in \mathbb{R}^D$ from the current input. Here, the query specifies the linear readout $g(\mathbf{S}_t, \mathbf{x}_t) = \mathbf{S}_t^\top \mathbf{q}_t$, so the quantity to estimate is the memory projection along that query. Its posterior mean gives the Bayes-optimal layer output under squared-error loss:

$$\mathbf{y}_t = \mathbb{E}_{\mu_t}\left[\mathbf{S}_t^\top \mathbf{q}_t\right] = \mathbf{M}_t^\top \mathbf{q}_t. \tag{8}$$

Each coordinate of $\mathbf{S}_t^\top \mathbf{q}_t$ has posterior variance $\mathbf{q}_t^\top \mathbf{P}_t \mathbf{q}_t$. Subsequent memory updates use $\mathbf{P}_t$, while the readout uses only the mean. We call the recurrent layer defined by this design model and readout the *Bayesian Layer*.

2.2 The Bayesian Layer recurrence

Under this design model, Bayesian filtering (Särkkä, 2013) gives a closed-form recurrence for the recurrent state $\mathbf{h}_t = (\mathbf{M}_t, \mathbf{P}_t)$. Starting from $\mathbf{M}_0 = \mathbf{0}$ and $\mathbf{P}_0 = \sigma_0^2 \mathbf{I}$ as specified in Definition 1, the same update applies from $t = 1$.

Proposition 1 (Bayesian Layer recurrence)**.** *Under the Bayesian Layer design model (Definition 1) with belief structure* (7)*, the uncertainty-warped key* $\mathbf{u}_t$ *gives the update direction, and the recurrent state* $(\mathbf{M}_t, \mathbf{P}_t)$ *satisfies*

$$\bar{\mathbf{P}}_t = \mathbf{A}_t \mathbf{P}_{t-1} \mathbf{A}_t^\top + \ell_t^2 \mathbf{I}, \qquad \textit{predicted covariance} \tag{9}$$

$$\mathbf{u}_t = \bar{\mathbf{P}}_t \mathbf{k}_t, \qquad \textit{update direction} \tag{10}$$

$$\beta_t = \left(r_t^2 + \mathbf{k}_t^\top \mathbf{u}_t\right)^{-1}, \qquad \textit{innovation precision} \tag{11}$$

$$\mathbf{M}_t = \left(\mathbf{I} - \beta_t \mathbf{u}_t \mathbf{k}_t^\top\right) \mathbf{A}_t \mathbf{M}_{t-1} + \beta_t \mathbf{u}_t \mathbf{v}_t^\top, \qquad \textit{mean update} \tag{12}$$

$$\mathbf{P}_t = \bar{\mathbf{P}}_t - \beta_t \mathbf{u}_t \mathbf{u}_t^\top. \qquad \textit{covariance update} \tag{13}$$

This recurrence is the exact Kalman filter for the design model (Kalman, 1960; Anderson & Moore, 1979; Kailath et al., 2000). It preserves the belief structure in (7), with full posterior covariance $\mathbf{I}_m \otimes \mathbf{P}_t$. The layer therefore propagates one shared $D \times D$ covariance matrix alongside the mean. Appendix B derives the recurrence and establishes this closure.

The predicted covariance maps the key $\mathbf{k}_t$ to the update direction $\mathbf{u}_t$, while β_t scales the correction. These quantities follow from filtering once the learned functions $\mathbf{A}, \ell, \mathbf{k}, \mathbf{v}, r$ are evaluated at $\mathbf{x}_t$ and the previous belief is given. Prediction adds process noise through $\ell_t^2 \mathbf{I}$ and conditioning then reduces covariance by the rank-one term in (13). Both steps preserve positive semidefiniteness.

For efficient computation, we restrict $\mathbf{A}_t$ to be diagonal or blocks of damped rotations; Appendix C gives the full implementation details. Multiplication by $\mathbf{A}_t$ then uses row-wise scaling or transformations within each block. Covariance prediction and conditioning cost $\mathcal{O}(D^2)$, and the mean update costs $\mathcal{O}(Dm)$. The per-step recurrent cost is therefore $\mathcal{O}(D^2 + Dm)$.

Together with the readout $\mathbf{y}_t = \mathbf{M}_t^\top \mathbf{q}_t$, this recurrence gives the key–value–query form common to efficient recurrent layers (Schmidhuber, 1992; Schlag et al., 2021). The key $\mathbf{k}_t$ and value $\mathbf{v}_t$ enter the filtering update, while the query $\mathbf{q}_t$ enters only the readout. Here, the write rule is derived by exact filtering under the design model rather than specified directly. The next section examines how covariance guides memory updates.

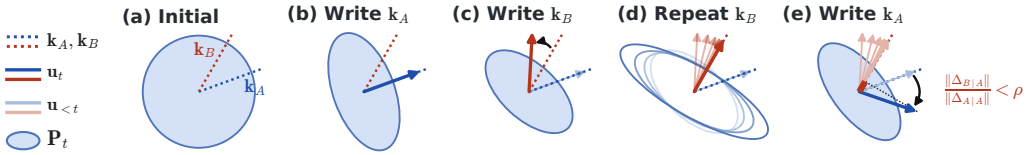


Figure 2: **Covariance-guided memory updates.** Schematic with $D = 2$, $\mathbf{A}_t = \mathbf{I}$, and unit keys with overlap $\rho \approx 0.77$. Ellipses show posterior covariance $\mathbf{P}_t$; dotted and solid lines show keys $\mathbf{k}_t$ and update directions $\mathbf{u}_t$ respectively (faded: earlier). **(a)** Initial covariance. **(b–d)** Writes at $\mathbf{k}_A$, then repeated writes at $\mathbf{k}_B$ (light to dark). **(e)** The next write at $\mathbf{k}_A$; $\|\Delta_{B|A}\|/\|\Delta_{A|A}\| \approx 0.246 < \rho$

2.3 Geometry of Bayesian Layer Memory Updates

The Bayesian Layer updates a memory shared by all keys, so incorporating one association can change the readout at another key. This interference is a known limitation of linear-attention memories (Schlag et al., 2021; Yang et al., 2025). The two-key example in Figure 2 illustrates how propagating covariance can mitigate it.

Let $\bar{\mathbf{v}}_t(\mathbf{k}) = \bar{\mathbf{M}}_t^\top \mathbf{k}$ and $\hat{\mathbf{v}}_t(\mathbf{k}) = \mathbf{M}_t^\top \mathbf{k}$ denote estimated readouts before and after incorporating an observation, with $\bar{\mathbf{M}}_t = \mathbf{A}_t \mathbf{M}_{t-1}$. Before incorporating the observation, each coordinate of the readout $\mathbf{S}_t^\top \mathbf{k}_A$ has variance $\bar{\sigma}_A^2 := \mathbf{k}_A^\top \bar{\mathbf{P}}_t \mathbf{k}_A$. For an observation $(\mathbf{k}_A, \mathbf{v}_A)$, define the *cross-key interference* as the change of the readout at $\mathbf{k}_B$ induced by the mean update in Equation (12):

$$\Delta_{B|A} := \hat{\mathbf{v}}_t(\mathbf{k}_B) - \bar{\mathbf{v}}_t(\mathbf{k}_B) = \frac{\mathbf{k}_B^\top \bar{\mathbf{P}}_t \mathbf{k}_A}{r_t^2 + \bar{\sigma}_A^2} \big(\mathbf{v}_A - \bar{\mathbf{v}}_t(\mathbf{k}_A)\big). \tag{14}$$

Define $\Delta_{A|A}$ analogously as the readout change at $\mathbf{k}_A$ after the same observation. Past observations constrain some memory directions more strongly than others, shaping how predicted covariance $\bar{\mathbf{P}}_t$ maps the incoming key to an update direction (Figure 2b,c).

Consider unit keys $\mathbf{k}_A$ and $\mathbf{k}_B$ with overlap $\rho = \mathbf{k}_A^\top \mathbf{k}_B \in [0, 1)$. Decompose $\mathbf{k}_A$ as

$$\mathbf{k}_A = \rho \mathbf{k}_B + \sqrt{1 - \rho^2}\, \mathbf{k}_\perp,$$

where $\mathbf{k}_\perp$ is a unit vector perpendicular to $\mathbf{k}_B$. The predicted variances along these directions are $\bar{\sigma}_B^2 = \mathbf{k}_B^\top \bar{\mathbf{P}}_t \mathbf{k}_B$ and $\bar{\sigma}_\perp^2 = \mathbf{k}_\perp^\top \bar{\mathbf{P}}_t \mathbf{k}_\perp$. Under identity dynamics and fixed positive noise scales, repeated observations at $\mathbf{k}_B$ reduce $\bar{\sigma}_B^2$ relative to $\bar{\sigma}_\perp^2$ (Figure 2d).

When the next observation arrives at $\mathbf{k}_A$, the covariance maps that key to the write direction $\mathbf{u}_t = \bar{\mathbf{P}}_t \mathbf{k}_A$. When covariance is approximately diagonal in the basis $(\mathbf{k}_B, \mathbf{k}_\perp)$, multiplying by $\bar{\mathbf{P}}_t$ weights the parallel component $\rho \mathbf{k}_B$ by $\bar{\sigma}_B^2$ and the perpendicular component $\sqrt{1 - \rho^2}\, \mathbf{k}_\perp$ by $\bar{\sigma}_\perp^2$. We measure relative cross-key interference with *confusion*, defined for $\Delta_{A|A} \neq \mathbf{0}$ as

$$\text{confusion} := \frac{\|\Delta_{B|A}\|}{\|\Delta_{A|A}\|} = \frac{|\mathbf{k}_B^\top \bar{\mathbf{P}}_t \mathbf{k}_A|}{\mathbf{k}_A^\top \bar{\mathbf{P}}_t \mathbf{k}_A} \approx \frac{\rho\, \bar{\sigma}_B^2}{\rho^2 \bar{\sigma}_B^2 + (1 - \rho^2) \bar{\sigma}_\perp^2}. \tag{15}$$

Repeated observations at $\mathbf{k}_B$ have reduced $\bar{\sigma}_B^2$ relative to $\bar{\sigma}_\perp^2$, and this lowers confusion (Figure 2e). As ρ approaches one, the perpendicular contribution to the denominator shrinks, so a larger variance contrast is needed to keep confusion low. (See Appendix D for derivations and limits.)

Propagating dense covariance can therefore give the memory finer angular resolution: stronger evidence, from repeated or lower-noise observations at a stored key, helps keep the confusion defined in (15) low between more closely aligned keys. The Delta-rule and additive updates considered here instead write along the incoming key, so for a nonzero write, confusion is fixed at the key overlap ρ. Diagonal covariance provides an intermediate case: it can adapt the write direction through coordinate-wise rescaling, but cannot represent correlations between memory coordinates. The next section shows how resetting covariance to an isotropic matrix recovers the Delta-rule family.

3 Recurrent Architecture Families through Design Models

The construction in §2 yields the Bayesian Layer by exact filtering under the Bayesian Layer design model. Resetting predicted covariance in the Bayesian Layer filter gives the *Delta-rule family*. A

Table 1: **Recurrent architectures through design models.** Memory updates grouped by underlying design model. Top: the Bayesian Layer design model (Definition 1) yields the exact Bayesian Layer filter and, with covariance reset, Delta-rule updates (Proposition 2). Bottom: the latent-input model (Equation (17)) yields exact additive updates. $\bar{\mathbf{M}}_t := \mathbf{A}_t\mathbf{M}_{t-1}$, $\bar{\mathbf{v}}_t := \bar{\mathbf{v}}_t(\mathbf{k}_t) = \bar{\mathbf{M}}_t^\top \mathbf{k}_t$, $\mathbf{u}_t := \bar{\mathbf{P}}_t\mathbf{k}_t$, $\beta_t := (r_t^2 + \mathbf{k}_t^\top \mathbf{u}_t)^{-1}$, $\eta_t := \lambda_t/(r_t^2 + \lambda_t\|\mathbf{k}_t\|^2)$, and $\omega_t := \lambda_t/(\lambda_t + r_t^2)$. †The Longhorn row gives its update before diagonal approximation.

Model	Derivation	Recurrence
Bayesian Layer design model (Definition 1): $\mathbf{S}_t \mid \mathbf{S}_{t-1} \sim \mathcal{N}(\mathbf{A}_t\mathbf{S}_{t-1},\, \ell_t^2\mathbf{I}),\ \mathbf{v}_t \mid \mathbf{S}_t \sim \mathcal{N}(\mathbf{S}_t^\top\mathbf{k}_t,\, r_t^2\mathbf{I})$		
Bayesian Layer	exact filter	$\mathbf{M}_t = \bar{\mathbf{M}}_t + \beta_t\mathbf{u}_t(\mathbf{v}_t - \bar{\mathbf{v}}_t)^\top$
DeltaNet (Schlag et al., 2021; Yang et al., 2024b)	reset; $\mathbf{A}_t = \mathbf{I}$	$\mathbf{M}_t = \bar{\mathbf{M}}_t + \eta_t\mathbf{k}_t(\mathbf{v}_t - \bar{\mathbf{v}}_t)^\top$
Gated DeltaNet (GDN) (Yang et al., 2025)	reset; $\mathbf{A}_t = \alpha_t\mathbf{I}$	$\mathbf{M}_t = \bar{\mathbf{M}}_t + \eta_t\mathbf{k}_t(\mathbf{v}_t - \bar{\mathbf{v}}_t)^\top$
KDA (Kimi Linear) (Kimi Team, 2025)	reset; $\mathbf{A}_t = \mathrm{diag}(\boldsymbol{\alpha}_t)$	$\mathbf{M}_t = \bar{\mathbf{M}}_t + \eta_t\mathbf{k}_t(\mathbf{v}_t - \bar{\mathbf{v}}_t)^\top$
Longhorn† (Liu et al., 2025)	per-column reset	$\mathbf{M}_t^{(j)} = \mathbf{M}_{t-1}^{(j)} + \eta_{t,j}\mathbf{k}_t(v_{t,j} - \bar{v}_{t,j})$
Latent-input design model (Definition 2): $\boldsymbol{\phi}_t \sim \mathcal{N}(\mathbf{0},\, \lambda_t\mathbf{I}),\ \mathbf{S}_t = \mathbf{A}_t\mathbf{S}_{t-1} + \mathbf{k}_t\boldsymbol{\phi}_t^\top,\ \mathbf{v}_t \mid \boldsymbol{\phi}_t \sim \mathcal{N}(\boldsymbol{\phi}_t,\, r_t^2\mathbf{I})$		
Linear attention (Katharopoulos et al., 2020)	exact; $\mathbf{A}_t = \mathbf{I}$	$\mathbf{M}_t = \mathbf{M}_{t-1} + \mathbf{k}_t(\omega_t\mathbf{v}_t)^\top$
RetNet (Sun et al., 2023)	exact; $\mathbf{A}_t = \gamma\mathbf{I}$	$\mathbf{M}_t = \gamma\mathbf{M}_{t-1} + \mathbf{k}_t(\omega_t\mathbf{v}_t)^\top$
GLA (diag. form) (Yang et al., 2024a)	exact; $\mathbf{A}_t = \mathrm{diag}(\boldsymbol{\alpha}_t)$	$\mathbf{M}_t = \mathrm{diag}(\boldsymbol{\alpha}_t)\mathbf{M}_{t-1} + \mathbf{k}_t(\omega_t\mathbf{v}_t)^\top$
Mamba-2 / SSD (Dao & Gu, 2024)	equiv. form; $\mathbf{A}_t = a_t\mathbf{I}$	$\mathbf{M}_t = a_t\mathbf{M}_{t-1} + \mathbf{k}_t(\omega_t\mathbf{v}_t)^\top$

different model, the *latent-input design model*, treats observations as evidence about independent write coefficients rather than stored associations. Exact filtering under this model gives the *additive family*. Table 1 summarizes both families; derivations appear in Appendix A.

3.1 Bayesian Layer design model: covariance reset ⇒ Delta-rule family

Proposition 2 (Isotropic-reset reduction)**.** *Under the Bayesian Layer design model (Definition 1), fix a scalar sequence $\lambda_t > 0$ and replace the predicted covariance $\bar{\mathbf{P}}_t$ in the filtering update by $\lambda_t\mathbf{I}$ at every step. Then $\mathbf{u}_t = \lambda_t\mathbf{k}_t$ and the Bayesian Layer update reduces to*

$$\mathbf{M}_t = (\mathbf{I} - \eta_t\mathbf{k}_t\mathbf{k}_t^\top)\,\mathbf{A}_t\,\mathbf{M}_{t-1} + \eta_t\mathbf{k}_t\mathbf{v}_t^\top, \qquad \eta_t := \lambda_t/(r_t^2 + \lambda_t\|\mathbf{k}_t\|^2). \tag{16}$$

Equation (16) has the canonical Delta-rule form: the update corrects the difference between the observed value and the predicted memory readout at $\mathbf{k}_t$. Covariance reset retains Delta-rule correction but discards the covariance accumulated from past observations. Choices of $\mathbf{A}_t$ along the scalar–diagonal–dense hierarchy recover the variants in Table 1.

3.2 Latent-input design model: exact filter ⇒ additive family

We consider a design model that treats each observation as evidence for a new write, independent of stored associations.

Definition 2 (Latent-input design model)**.** *Let the initial memory $\mathbf{S}_0$ have mean $\mathbf{M}_0$. Each observation $\mathbf{v}_t$ provides noisy evidence about a latent write coefficient $\boldsymbol{\phi}_t \in \mathbb{R}^m$. For input-dependent $\lambda_t, r_t > 0$, define*

$$\boldsymbol{\phi}_t \sim \mathcal{N}(\mathbf{0}, \lambda_t\mathbf{I}_m), \qquad \mathbf{v}_t \mid \boldsymbol{\phi}_t \sim \mathcal{N}(\boldsymbol{\phi}_t, r_t^2\mathbf{I}_m), \qquad \mathbf{S}_t = \mathbf{A}_t\mathbf{S}_{t-1} + \mathbf{k}_t\boldsymbol{\phi}_t^\top. \tag{17}$$

All write coefficients and observation noises are mutually independent and independent of $\mathbf{S}_0$.

Given $\mathbf{S}_{t-1}$ and $\boldsymbol{\phi}_t$, the memory $\mathbf{S}_t$ is deterministic. Marginalizing the write coefficient gives a design factor $\Psi_t(\mathbf{S}_{t-1}, \mathbf{S}_t; \mathbf{x}_t)$ within the framework of §2. Taking the posterior expectation of the state update in (17) yields an additive recurrence for the latent memory state's mean (Appendix A.2).

Proposition 3 (Exact additive recurrence)**.** *Under the latent-input design model (Definition 2), the update for the posterior mean $\mathbf{M}_t := \mathbb{E}_{\mu_t}[\mathbf{S}_t]$ is purely additive:*

$$\mathbf{M}_t = \mathbf{A}_t\mathbf{M}_{t-1} + \omega_t\mathbf{k}_t\mathbf{v}_t^\top, \qquad \omega_t := \lambda_t/(\lambda_t + r_t^2). \tag{18}$$

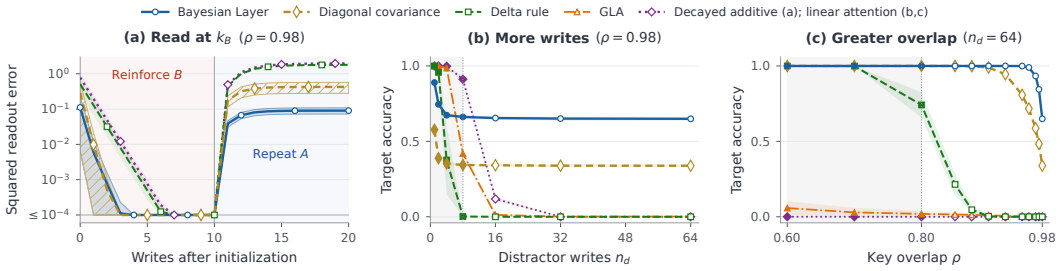


Figure 3: **Associative recall with overlapping keys. (a) Toy experiment without training.** Squared readout error at $\mathbf{k}_B$ during repeated writes at $\mathbf{k}_B$ followed by $\mathbf{k}_A$, with key overlap $\rho = 0.98$. **(b,c) Learned associative recall.** Target recall accuracy versus distractor write count at $\rho = 0.98$ (b) and versus key overlap after 64 distractor writes (c). Bands show ± 1 standard deviation across initial write orderings in (a) and 3 seeds in (b,c). Grey shading and filled markers indicate the training range of the horizontal variable; the fixed variable is outside its training range in both panels.

Each observation contributes a new write without correcting the current memory prediction. The exact mean update and readout require no covariance propagation, although uncertainty in the write coefficients generally leaves nonzero posterior uncertainty over memory (Appendix A.2). Choices of $\mathbf{A}_t$ recover the additive variants in Table 1.

4 Experiments

We test whether propagating covariance helps preserve stored associations under subsequent memory updates. We first use synthetic recall tasks with controlled key overlap (§4.1), then evaluate recall on MQAR and WikiText-103 (§4.2) and retrieval after distilling Bayesian Layers into a pretrained language model (§4.3). Appendix Table 2 summarizes the experimental settings.

4.1 Associative recall with overlapping keys

The analysis in §2.3 predicts that covariance geometry helps preserve associations during writes at nearby keys. We first compare memory recurrences with fixed keys and values, without learned projections. The toy task stores six associations, labeled A–F, with overlap $\rho = 0.98$ between keys A and B while all other keys are orthogonal. After initialization, we repeat $(\mathbf{k}_B, \mathbf{v}_B)$ ten times, followed by ten repetitions of $(\mathbf{k}_A, \mathbf{v}_A)$. Figure 3a shows the squared readout error $\|\hat{\mathbf{v}}_t(\mathbf{k}_B) - \mathbf{v}_B\|^2$, averaged over initial write orderings (Appendix E.1).

Repeated writes at $\mathbf{k}_B$ reduce all methods' mean error below 10^{-4}, with dense and diagonal covariance requiring fewer writes. However, during subsequent writes at $\mathbf{k}_A$, their performance diverges with dense covariance reaching the lowest error plateau, followed by diagonal covariance (Figure 3a). Asymptotic behavior under repeated writes is analyzed in Appendices D.3 and E.2.

We next test whether the covariance advantage persists in learned networks and how it depends on key overlap. We separately train two-layer networks using the Bayesian Layer, its diagonal-covariance variant, Delta-rule (Schlag et al., 2021), linear-attention (Katharopoulos et al., 2020), and GLA (Yang et al., 2024a) updates. Each training sequence contains target–distractor pairs (T_i, D_i) with within-pair key overlap $\mathbf{k}_{T_i}^\top \mathbf{k}_{D_i} = \rho_i$. Keys from different pairs remain orthogonal. Sequences initialize all associations, write repeatedly at each target followed by n_d writes at each distractor, and finally query the target keys. We use the supplied key vectors directly as keys and queries, so learned projections cannot change their overlap. Evaluation uses more distractor writes and greater key overlap than training (Appendices E.3 and E.3.2).

At $\rho = 0.98$, accuracy initially drops for both the Bayesian Layer and its diagonal-covariance variant, then stays nearly constant, reaching 65.0% and 33.9%, respectively at $n_d = 64$. In contrast, the other updates eventually fall to near-zero accuracy. At $n_d = 64$, dense covariance Bayesian Layer also maintains high accuracy over a wider range of key overlap than any baseline (Figure 3c). Both dense

and diagonal variants benefit from covariance propagation, but dense covariance better preserves stored associations.

4.2 Associative recall on MQAR and WikiText-103

The preceding experiment controls key overlap directly. We next ask how the Bayesian Layer performs when the model must learn representations for storing and retrieving associations from tokens. We begin with Multi-Query Associative Recall (MQAR) (Arora et al., 2024a), where sequences present key–value associations and later query their values. We compare two-layer models across widths under a common data and training protocol (Appendix F.1). The baselines are Gated DeltaNet (Yang et al., 2025), Gated DeltaNet-2 (GDN-2) (Hatamizadeh et al., 2026), MesaNet (von Oswald et al., 2026), RWKV-7 (Peng et al., 2025), Palimpsa-D (Bonnet et al., 2026), Mamba-2 (Dao & Gu, 2024), and causal attention (Vaswani et al., 2017) with rotary position embeddings (Su et al., 2024).

Figure 4a reports recall at sequence length 1,024, beyond the training maximum of 256. BL, MesaNet and RWKV-7 achieve similarly high recall at width 256, but BL has higher recall than those two baselines at width 128. BL therefore remains competitive with the strongest recurrent baselines when evaluated on longer sequences containing more associations.

We then ask whether this recall performance extends to language modeling. We train models with approximately 14–17M non-embedding parameters on WikiText-103 (Merity et al., 2017) for 520M tokens each, using 3 training seeds. To isolate the contribution of the update mechanism, we replace the recurrent updates in GDN and RWKV-7 with the Bayesian recurrence while retaining their surrounding projections, normalization, and feedforward layers. We call these the GDN-like and RWKV-like BL variants (Appendix F.2). Following the recall analysis of Arora et al. (2024a), we measure associative recall on predictions completing token pairs that are rare in the training corpus and have appeared within the preceding 1,024 tokens of the same context. Figure 4b groups these predictions by distance to the pair's most recent occurrence.

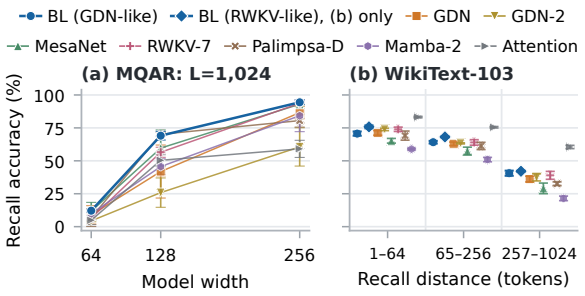


Figure 4: **Associative recall accuracy. (a) MQAR:** versus model width at length 1,024 (training maximum: 256), using GDN-like BL. **(b) WikiText-103:** for repeated rare token pairs, grouped by distance to their most recent occurrence. Means ± 1 standard deviation over 3 seeds.

Both BL variants improve mean associative-recall accuracy over their corresponding GDN and RWKV-7 baselines while maintaining competitive overall perplexity. Their recall advantage is present in both longer-distance bins, and the RWKV-like BL achieves the highest mean accuracy among recurrent models in all three bins. Attention remains strongest on associative recall. The full perplexity and recall results are reported in Table 5.

4.3 Distillation into a pretrained language model

Our last experiment examines whether Bayesian Layers can improve measured retrieval accuracy in a pretrained language model. Building on work that adapts pretrained models through recurrent layer replacements (Wang et al., 2024; Lan et al., 2025), we replace four of a 340M Gated DeltaNet's 24 sequence mixers with Bayesian Layers. Each replacement is trained to match the corresponding frozen mixer's outputs, while the remaining model parameters stay fixed.

To assess whether further training of the original architecture produces comparable improvements, we also continue GDN with next-token cross-entropy, updating the mixers and MLPs at the same four layers. Both procedures use 2k-token SlimPajama sequences and expose each selected mixer to 100M training tokens, totaling 400M when summed across the four mixers. We evaluate three training seeds per procedure on six needle-in-a-haystack tasks from RULER (Hsieh et al., 2024),

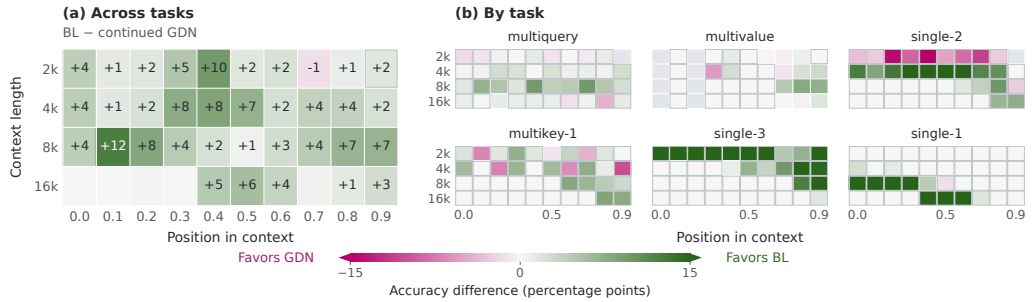


Figure 5: **Long-context retrieval after layerwise distillation.** BL minus continued-GDN retrieval accuracy on RULER, **(a)** aggregated across six needle-in-a-haystack tasks and **(b)** separately for each task. Both methods are adapted on 2k-token sequences. Rows indicate evaluation context length; columns indicate the position of the information to be retrieved, from early to late. Results average 3 training seeds per method on the same evaluation examples. Gray outlines mark cells containing fewer than 30 examples; gray fill indicates empty cells.

varying context length from 2k to 16k and the position of the information to be retrieved. Training details, complete results, and the comparison with frozen GDN appear in Appendix G.

BL achieves higher mean retrieval accuracy than both frozen and continued GDN at each evaluated context length. Across WikiText-103, held-out SlimPajama, and PG-19, perplexity remains close to continued GDN, with increases of 0.3–0.7 points (approximately 3–4%). Averaged across tasks, the retrieval gains over continued GDN span every position bin at 4k and 8k (Figure 5a). The task breakdown shows improvements in both single-target and multi-target retrieval, alongside a regression on single-2 at 2k (Figure 5b). At 8k—four times the adaptation length—BL achieves 25.5% mean accuracy compared with 20.5% for continued GDN.

5 Discussion

Related probabilistic sequence layers. Like Palimpsa (Bonnet et al., 2026) and Kalman Linear Attention (Shaj et al., 2026), the BL treats recurrent computation as probabilistic inference. Its dense key-space covariance retains correlations omitted by the diagonal representations used in their implemented layers. Dense statistics also appear in MesaNet (von Oswald et al., 2026), where they support a reweighted in-context regression solved by conjugate gradient at readout. The BL instead propagates a posterior over an evolving state through structured dynamics and process noise, with readout given by a linear projection of the posterior mean of the latent memory state.

Limitations. Our geometric analysis focuses on Gaussian beliefs. Extending it to the information geometry of other exponential families, drawing on natural-gradient approaches to Bayesian learning, remains future work (Khan & Rue, 2023). Dense covariance adds training computation quadratic in the key dimension and additional activation storage, making low-rank, sparse, or other structured approximations promising future directions. Finally, the readout uses only the posterior mean, leaving the use of propagated uncertainty for downstream losses or calibration unexplored.

Conclusion. Design models connect recurrent sequence architectures to probabilistic assumptions. They place a range of recurrent architectures within a common probabilistic framework, distinguishing changes to the underlying model from approximations to inference. For the Bayesian Layer, our geometric analysis shows how repeated evidence can reshape covariance to reduce confusion between nearby keys. Motivated by this analysis, we examined recall across controlled key geometries, small language models, and distillation into a pretrained GDN. These experiments show improved recall under greater key overlap and more distractor writes than seen during training, higher WikiText-103 associative recall over corresponding GDN and RWKV-7 baselines, and improved RULER retrieval over continued GDN after distillation while retaining similar overall perplexity. Together, the analysis and experiments demonstrate how probabilistic modeling can guide the design of recurrent sequence architectures that use uncertainty geometry to integrate new evidence with stored information.

AI USE STATEMENT

We used generative AI tools to assist with manuscript editing, reviewing mathematical exposition and experimental descriptions, and modifying plotting code and scientific figures. We reviewed the AI-assisted code, checked relevant computations through numerical tests, and consulted related literature to assess the resulting claims and exposition. We take responsibility for the final content of this paper, including all AI-assisted text, claims, and artifacts.

REPRODUCIBILITY STATEMENT

The design models and assumptions are specified in Sections 2 and 3. Appendices A and B derive the filtering updates and architectural connections, Appendix C describes the Bayesian Layer implementation, and Appendices D and E provide the covariance and interference analyses. Appendices E to G document the experimental protocols, including data preparation, model configurations, optimization and tuning settings, training budgets, evaluation procedures, and how results are summarized across training seeds. Each model configuration trained for the reported comparisons is evaluated across at least three training seeds. The supplementary material provides a minimal code package for reproducing the reported experiments.

REFERENCES

Shun-ichi Amari. Learning patterns and pattern sequences by self-organizing nets of threshold elements. *IEEE Transactions on Computers*, C-21(11):1197–1206, November 1972. ISSN 0018-9340,1557-9956. doi: 10.1109/t-c.1972.223477. URL http://dx.doi.org/10.1109/t-c.1972.223477.

Brian D. O. Anderson and John B. Moore. *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, N.J., 1979. ISBN 978-0-13-638122-8.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Y Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. In *International Conference on Learning Representations*, volume 2024, pp. 15664–15730, 2024a.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. In *Proceedings of the 41st International Conference on Machine Learning*, 2024b.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xLSTM: Extended long short-term memory. In *Advances in Neural Information Processing Systems*, volume 37, pp. 107547–107603, 2024.

Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. ATLAS: Learning to optimally memorize the context at test time. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=HV1R5bMSpa.

Djohan Bonnet, Jamie Lohoff, Jan Finkbeiner, Elidona Shiqerukaj, and Emre Neftci. Learning to remember, learn, and forget in attention-based models. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=wKMV7mXjUr.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning*, 2024.

Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, Arnaud Doucet, David Budden, Yee Whye Teh, Razvan Pascanu, Nando de Freitas, and Caglar Gulcehre. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, Tao Zhang, and Yu Cheng. MoM: Linear sequence modeling with mixture-of-memories. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=3PdOq8Rgue.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

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

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Thomas Kailath, Ali H. Sayed, and Babak Hassibi. *Linear estimation*. Prentice Hall, 2000.

Rudolf E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165, 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

Mohammad Emtiyaz Khan and Håvard Rue. The Bayesian Learning Rule. *Journal of Machine Learning Research*, 24(281):1–46, 2023. URL https://jmlr.org/papers/v24/22-0291.html.

Kimi Team. Kimi Linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

Disen Lan, Weigao Sun, Jiaxi Hu, Jusen Du, and Yu Cheng. Liger: Linearizing large language models to gated recurrent structures. In *International Conference on Machine Learning*, 2025.

Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. Longhorn: State space models are amortized online learners. In *International Conference on Learning Representations*, pp. 95419–95434, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023. URL https://probml.github.io/pml-book/book2.html.

Bo Peng, Eric Alcaide, Quentin Gregory Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Nguyen Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan S. Wind, Stanisław Woźniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: Reinventing RNNs for the transformer era. In *Findings of the association for computational linguistics: EMNLP 2023*, pp. 14048–14077, 2023.

Bo Peng, Daniel Goldstein, Quentin Gregory Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Teddy Ferdinan, Kranthi Kiran GV, Haowen Hou, Satyapriya Krishna, Ronald McClelland Jr., Niklas Muennighoff, Fares Obeid, Atsushi Saito, Guangyu Song, Haoqin Tu, Ruichong Zhang, Bingchen Zhao, Qihang Zhao, Jian Zhu, and Rui-Jie Zhu. Eagle and finch: RWKV with matrix-valued states and dynamic recurrence. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=soz1SEiPeq.

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, Nathan Wilce, Johan S. Wind, Tianyi Wu, Daniel Wuttke, and Christian Zhou-Zheng. RWKV-7 "Goose" with expressive dynamic state evolution. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=ayB1PACN5j.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. HGRN2: Gated linear RNNs with state expansion. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=y6SqbJfCSk.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SylKikSYDH.

Simo Särkkä. *Bayesian filtering and smoothing*. Cambridge University Press, 2013. ISBN 9781107619289. URL http://www.worldcat.org/isbn/9781107619289.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, pp. 9355–9366, 2021.

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

Vaisakh Shaj, Cameron Barker, Aidan Scannell, Andras Szecsenyi, Elliot J. Crowley, and Amos Storkey. Kalman Linear Attention: Parallel bayesian filtering for efficient language modeling and state tracking. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=9h7sSJe4jN.

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. DeltaProduct: Improving state-tracking in linear RNNs via Householder products. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=SoRiaijTGr.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R. Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://cerebras.ai/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, June 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

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

Philippe Tillet, H.T. Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Johannes von Oswald, Nino Scherrer, Seijin Kobayashi, Luca Versari, Songlin Yang, Maximilian Schlegel, Kaitlin Maile, Yanick Schimpf, Oliver Sieberling, Alexander Meulemans, Guillaume Lajoie, Rif A. Saurous, Charlotte Frenkel, Razvan Pascanu, Blaise Agüera y Arcas, and João Sacramento. MesaNet: Sequence modeling by locally optimal test-time training. In *International Conference on Learning Representations*, volume 2026, pp. 26397–26447, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/2c3d5a4eff20d24390a445762d61bfbb-Paper-Conference.pdf.

Junxiong Wang, Daniele Paliotta, Avner May, Alexander M. Rush, and Tri Dao. The Mamba in the Llama: Distilling and accelerating hybrid models. In *Advances in Neural Information Processing Systems*, 2024.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *International Conference on Machine Learning*, 2024a.

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, pp. 115491–115522, 2024b.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated Delta Networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, volume 2025, pp. 29687–29707, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/4904fad153f6434a7bcf04465d4be2cc-Paper-Conference.pdf.

A Connections to Recurrent Architectures

We derive the two families summarized in Table 1. Resetting the Bayesian Layer's predicted covariance gives Delta-rule updates. A separate latent-input model gives additive updates by treating each observation as evidence about an independent write coefficient. Throughout, input-dependent model parameters are treated as supplied quantities.

A.1 Covariance reset and Delta-rule updates

In the Bayesian Layer recurrence, replace the predicted covariance by $\bar{\mathbf{P}}_t = \lambda_t \mathbf{I}$, with $\lambda_t > 0$. The gain becomes

$$\beta_t \mathbf{u}_t = \frac{\lambda_t}{r_t^2 + \lambda_t \|\mathbf{k}_t\|^2} \mathbf{k}_t = \eta_t \mathbf{k}_t.$$

Writing $\bar{\mathbf{M}}_t = \mathbf{A}_t \mathbf{M}_{t-1}$, substitution into the mean update gives

$$\mathbf{M}_t = (\mathbf{I} - \eta_t \mathbf{k}_t \mathbf{k}_t^\top) \bar{\mathbf{M}}_t + \eta_t \mathbf{k}_t \mathbf{v}_t^\top. \tag{19}$$

This proves the isotropic-reset reduction in Proposition 2. The resulting Delta-rule update (Schlag et al., 2021; Yang et al., 2025) retains error correction while replacing accumulated covariance with a prescribed isotropic covariance at every step. Choices of $\mathbf{A}_t$ give the identity, scalar-decay, and diagonal-decay variants in Table 1. Allowing column-specific reset scales and observation variances gives the Longhorn update shown in Table 1. Longhorn additionally diagonalizes the rank-one gate in its implemented recurrence (Liu et al., 2025).

A.2 Latent-input model and additive updates

Under the latent-input model of Definition 2, the current write coefficient $\boldsymbol{\phi}_t$ is independent of the previous memory, and $\mathbf{v}_t$ observes only $\boldsymbol{\phi}_t$. Marginalizing the write coefficient gives the design factor

$$\Psi_t(\mathbf{S}_{t-1}, \mathbf{S}_t; \mathbf{x}_t) = \int \delta_{\mathbf{A}_t \mathbf{S}_{t-1} + \mathbf{k}_t \boldsymbol{\phi}^\top}(\mathbf{S}_t)\, \mathcal{N}(\boldsymbol{\phi}; \mathbf{0}, \lambda_t \mathbf{I}_m)\, \mathcal{N}(\mathbf{v}_t; \boldsymbol{\phi}, r_t^2 \mathbf{I}_m)\, d\boldsymbol{\phi}, \tag{20}$$

where $\delta_{\mathbf{A}_t \mathbf{S}_{t-1} + \mathbf{k}_t \boldsymbol{\phi}^\top}$ denotes the point-mass transition kernel at $\mathbf{S}_t = \mathbf{A}_t \mathbf{S}_{t-1} + \mathbf{k}_t \boldsymbol{\phi}^\top$. Together with the initial distribution of $\mathbf{S}_0$, this factor defines the distribution in (2). Substitution into the filtering recursion (3) shows that μ_t is obtained by applying these dynamics to the joint belief

$$\mu_{t-1}(\mathbf{S}_{t-1})\, p_t(\boldsymbol{\phi}_t \mid \mathbf{v}_t).$$

Thus the observation updates the write coefficient while leaving the belief over $\mathbf{S}_{t-1}$ unchanged. Under this design model, Gaussian conditioning (Särkkä, 2013, Lemma A.2) gives

$$\mathbb{E}[\boldsymbol{\phi}_t \mid \mathbf{v}_t] = \omega_t \mathbf{v}_t, \qquad \omega_t = \frac{\lambda_t}{\lambda_t + r_t^2}.$$

Taking the expectation of the memory dynamics under this joint belief yields

$$\mathbf{M}_t = \mathbb{E}_{\mu_t}[\mathbf{S}_t] = \mathbf{A}_t\mathbf{M}_{t-1} + \omega_t\mathbf{k}_t\mathbf{v}_t^\top, \tag{21}$$

establishing Proposition 3.

Defining $\tilde{\mathbf{v}}_t = \omega_t\mathbf{v}_t$ gives the additive-write form

$$\mathbf{M}_t = \mathbf{A}_t\mathbf{M}_{t-1} + \mathbf{k}_t\tilde{\mathbf{v}}_t^\top.$$

For $\mathbf{A}_t = \mathbf{I}$, this is the memory update of linear attention (Katharopoulos et al., 2020). The unit write coefficient can therefore be obtained by absorbing ω_t into the value map, or in the noiseless limit $r_t^2 \to 0$. The linear readout remains $\mathbf{y}_t = \mathbf{M}_t^\top\mathbf{q}_t$. Its evaluation and subsequent mean updates require only $\mathbf{M}_t$, even though uncertainty in the write coefficients generally leaves nonzero posterior uncertainty over memory.

The correspondences in Table 1 concern these recurrent memory updates. Surrounding projections, normalization, additional gates, and feedforward layers require their own specifications.

Distinction from xLSTM's matrix-valued state. The mLSTM of Beck et al. (2024) uses forget and input gates f_t, i_t in the content update

$$\mathbf{C}_t = f_t\mathbf{C}_{t-1} + i_t\mathbf{v}_t\mathbf{k}_t^\top$$

and reads from $\mathbf{C}_t$ directly. This outer-product state stores values and need not be symmetric or positive semidefinite. In the Bayesian Layer, $\mathbf{M}_t$ stores the posterior mean, while the separate posterior covariance $\mathbf{P}_t$ determines the gain and gate. The readout uses only $\mathbf{M}_t$.

B Derivation of the Bayesian Layer Recurrence

We derive Proposition 1 and show that filtering preserves independent Gaussian memory columns with a shared covariance. At each step, the input-dependent model parameters are treated as supplied quantities. The dynamics and observation likelihood in Definition 1 factor across columns, so it suffices to derive the update for one column.

B.1 Filtering recurrence and closure

Prediction. For $t \geq 1$, suppose μ_{t-1} has the belief structure (7). The dynamics preserve independent Gaussian columns, with predicted distribution

$$\mathbf{S}_t^{(j)} \sim \mathcal{N}(\bar{\mathbf{M}}_t^{(j)}, \bar{\mathbf{P}}_t), \qquad \bar{\mathbf{M}}_t = \mathbf{A}_t\mathbf{M}_{t-1}, \qquad \bar{\mathbf{P}}_t = \mathbf{A}_t\mathbf{P}_{t-1}\mathbf{A}_t^\top + \ell_t^2\mathbf{I}.$$

Conditioning. Under the predicted design model, column j and its scalar observation have joint distribution

$$\begin{bmatrix}\mathbf{S}_t^{(j)} \\ v_{t,j}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\bar{\mathbf{M}}_t^{(j)} \\ \mathbf{k}_t^\top\bar{\mathbf{M}}_t^{(j)}\end{bmatrix}, \begin{bmatrix}\bar{\mathbf{P}}_t & \bar{\mathbf{P}}_t\mathbf{k}_t \\ \mathbf{k}_t^\top\bar{\mathbf{P}}_t & r_t^2 + \mathbf{k}_t^\top\bar{\mathbf{P}}_t\mathbf{k}_t\end{bmatrix}\right).$$

Define $\mathbf{u}_t = \bar{\mathbf{P}}_t\mathbf{k}_t$ and $\beta_t = (r_t^2 + \mathbf{k}_t^\top\mathbf{u}_t)^{-1}$; the denominator is positive because $r_t > 0$. Gaussian conditioning gives $\mathbf{M}_t^{(j)} = \bar{\mathbf{M}}_t^{(j)} + \beta_t\mathbf{u}_t(v_{t,j} - \mathbf{k}_t^\top\bar{\mathbf{M}}_t^{(j)})$. Stacking these means and writing the posterior covariance gives

$$\mathbf{M}_t = (\mathbf{I} - \beta_t\mathbf{u}_t\mathbf{k}_t^\top)\bar{\mathbf{M}}_t + \beta_t\mathbf{u}_t\mathbf{v}_t^\top, \qquad \mathbf{P}_t = \bar{\mathbf{P}}_t - \beta_t\mathbf{u}_t\mathbf{u}_t^\top. \tag{22}$$

These are the updates in Proposition 1, with column-wise Kalman gain $\mathbf{K}_t = \beta_t\mathbf{u}_t$.

Closure and initialization. Both the predicted distribution and likelihood factor across columns, so conditioning preserves independence. The posterior covariance is identical for every column, giving

$$\mathrm{Cov}_{\mu_t}[\mathrm{vec}(\mathbf{S}_t)] = \mathbf{I}_m \otimes \mathbf{P}_t.$$

The initial distribution in Definition 1 has this structure with $\mathbf{M}_0 = \mathbf{0}$ and $\mathbf{P}_0 = \sigma_0^2\mathbf{I}_D$. Prediction and conditioning therefore apply from $t = 1$, establishing the belief structure and recurrence for

every subsequent step. Each column retains a dense $D \times D$ covariance, shared exactly under the design-model assumptions. The implemented recurrent state is $\mathbf{h}_t = (\mathbf{M}_t, \mathbf{P}_t)$, while the readout is $\mathbf{M}_t^\top \mathbf{q}_t$.

The same derivation applies when the independent process-noise columns share any covariance $\mathbf{Q}_t \succeq 0$, replacing $\ell_t^2 \mathbf{I}$ by $\mathbf{Q}_t$ in covariance prediction. Appendix C.1 uses this extension for rotational dynamics.

C FILTERING IMPLEMENTATION

We implement the Bayesian Layer using chunks of length L. Each chunk receives the posterior mean and covariance from the preceding chunk, computes its outputs, and returns the updated belief parameters. The implementation reorganizes the recurrence from Appendix B around batched matrix products, compact histories of update directions, and tiled GPU operations. Chunk-boundary state propagation and tiled evaluation also underpin efficient GLA and DeltaNet implementations (Yang et al., 2024a;b). The equations below describe our covariance and mean computations.

C.1 STRUCTURED DYNAMICS AND NOISE

We use diagonal transitions or blocks of damped rotations,

$$\mathbf{A}_t^{(j)} = \rho_{t,j} \begin{bmatrix} \cos\theta_{t,j} & -\sin\theta_{t,j} \\ \sin\theta_{t,j} & \cos\theta_{t,j} \end{bmatrix}, \qquad 0 < \rho_{t,j} \leq 1.$$

Diagonal entries likewise have magnitude at most one. These structures allow multiplication by $\mathbf{A}_t$ through coordinate-wise scaling or transformations within each pair. Throughout this section, D counts real state coordinates, including both coordinates of each rotational block.

For rotational blocks, keys and queries address the real coordinate: their embedded forms are $(k_{t,j}, 0)$ and $(q_{t,j}, 0)$. Below, $\mathbf{k}_t, \mathbf{q}_t \in \mathbb{R}^D$ denote these embedded vectors. We inject process noise into the same coordinates, giving the block covariance

$$\mathbf{Q}_t^{(j)} = \ell_t^2 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}.$$

Rotations subsequently transport this uncertainty between both coordinates. For ordinary real diagonal dynamics, we use $\mathbf{Q}_t = \ell_t^2 \mathbf{I}$, as in the main-text design model. Noise variances are parameterized using softplus with a positive floor.

We use one covariance per head, shared across its value columns, and present the chunk equations with the head index suppressed.

C.2 EXACT CHUNK COMPUTATION

Number the steps within a chunk $1, \ldots, L$, and denote its entry belief parameters by $(\mathbf{M}_0, \mathbf{P}_0)$. Define the forward transition products

$$\Phi_{t \leftarrow s} = \mathbf{A}_t \mathbf{A}_{t-1} \cdots \mathbf{A}_{s+1}, \qquad \Phi_{s \leftarrow s} = \mathbf{I}.$$

Before incorporating observations from the current chunk, the state covariance at step t is

$$\mathbf{\Pi}_t = \Phi_{t \leftarrow 0} \mathbf{P}_0 \Phi_{t \leftarrow 0}^\top + \sum_{s=1}^{t} \Phi_{t \leftarrow s} \mathbf{Q}_s \Phi_{t \leftarrow s}^\top. \tag{23}$$

The first term propagates the entry covariance; the second accumulates process noise. All input-dependent model parameters are treated as supplied quantities.

Gain computation. Each earlier observation contributes a rank-one covariance downdate. Transporting its update direction to step t,

$$\mathbf{w}_{t,s} = \Phi_{t \leftarrow s} \mathbf{u}_s,$$

and unrolling the Kalman covariance recurrence (Särkkä, 2013, Theorem 4.2) gives

$$\bar{\mathbf{P}}_t = \mathbf{\Pi}_t - \sum_{s<t} \beta_s \mathbf{w}_{t,s} \mathbf{w}_{t,s}^\top. \tag{24}$$

Consequently, the next update direction and gain scale can be computed as

$$\mathbf{u}_t = \mathbf{\Pi}_t \mathbf{k}_t - \sum_{s<t} \beta_s \mathbf{w}_{t,s} (\mathbf{w}_{t,s}^\top \mathbf{k}_t), \qquad \beta_t = (r_t^2 + \mathbf{k}_t^\top \mathbf{u}_t)^{-1}. \tag{25}$$

This expression avoids constructing a dense covariance at every position. The entry-covariance contributions to $\mathbf{\Pi}_t \mathbf{k}_t$ are evaluated using a batched multiplication of $\mathbf{P}_0$ by transformed keys, followed by the corresponding forward transforms. The process-noise contributions use the diagonal or rotational block structure. A sequential loop within the chunk then computes $(\mathbf{u}_t, \beta_t)$, retaining the preceding update directions and gain scales.

Mean update and readout. Once these gains are available, the mean update is

$$\bar{\mathbf{M}}_t = \mathbf{A}_t \mathbf{M}_{t-1}, \qquad \mathbf{M}_t = (\mathbf{I} - \beta_t \mathbf{u}_t \mathbf{k}_t^\top) \bar{\mathbf{M}}_t + \beta_t \mathbf{u}_t \mathbf{v}_t^\top. \tag{26}$$

The output uses the posterior mean after incorporating the current observation:

$$\mathbf{y}_t = \mathbf{M}_t^\top \mathbf{q}_t.$$

The kernel processes tiles of value columns, carrying each mean tile through the chunk and emitting its outputs. All value columns within a head reuse the same gain sequence.

Exit covariance. At the final position, collect the transported update directions into

$$\mathbf{W} = \left[\Phi_{L \leftarrow 1} \mathbf{u}_1 \quad \cdots \quad \Phi_{L \leftarrow L} \mathbf{u}_L \right].$$

The covariance passed to the next chunk is

$$\mathbf{P}_L = \mathbf{\Pi}_L - \mathbf{W} \, \mathrm{diag}(\beta_1, \ldots, \beta_L) \, \mathbf{W}^\top. \tag{27}$$

The accumulated downdates are therefore evaluated as a tiled matrix product. This representation preserves the dense covariance: the compact history records its exact within-chunk corrections.

Chunk boundaries are processed sequentially. Within each chunk, parallelism is distributed across sequences, heads, and state or value tiles; the gain and mean passes retain sequential dependencies over the chunk's positions.

C.3 Numerical treatment, backward pass, and cost

The implementation uses forward transition products throughout. For a rotational block,

$$\Phi_{t \leftarrow s}^{(j)} = \exp\left(\sum_{a=s+1}^{t} \log \rho_{a,j} \right) R\left(\sum_{a=s+1}^{t} \theta_{a,j} \right),$$

where $R(\theta)$ is the rotation matrix above. Relative products are evaluated from differences of cumulative log-radii and angles, without explicitly forming inverse transition products. Under the stated parameterization, $\|\Phi_{t \leftarrow s}\|_2 \leq 1$.

We implement the tiled computations in Triton (Tillet et al., 2019). Recurrent quantities and chunk-boundary states are maintained in fp32. The chunk identities above are exact in real arithmetic; floating-point execution additionally floors innovation denominators at 10^{-6}.

A custom backward pass differentiates the entry-covariance products, gain computation, mean/readout recurrence, and exit-covariance update. Training retains chunk-boundary states and compact intermediate histories for backward. Optional checkpointing recomputes these histories and reconstructs intermediate boundaries by replaying chunks from saved states.

For one head with m value coordinates, recurrent-state storage is $\mathcal{O}(Dm + D^2)$. The entry and exit covariance computations cost $\mathcal{O}(LD^2)$ per chunk, the gain-history computation costs $\mathcal{O}(L^2 D)$, and the mean updates and readouts cost $\mathcal{O}(LDm)$. Over a sequence of length T, total forward work is therefore

$$\mathcal{O}\big(T \left[D^2 + LD + Dm \right]\big).$$

For fixed chunk length, this is linear in sequence length. Training activation storage additionally depends on the saved chunk boundaries and histories, and can be reduced through checkpointing.

D MEMORY RESOLUTION AND COVARIANCE GEOMETRY

This section extends the geometry in Section 2.3. We derive how stored evidence sets angular resolution and how covariance geometry controls interference. The calculations and figures use fixed keys and prescribed noise variances, without model training.

D.1 SETUP AND CONFUSION

For the analysis below, we use identity dynamics $\mathbf{A}_t = \mathbf{I}$, real-valued states, unit keys, and constant variances $\ell_t^2 = \ell^2$ and $r_t^2 = r^2$. The memory mean is $\mathbf{M}_t \in \mathbb{R}^{D\times m}$ and the covariance $\mathbf{P}_t \in \mathbb{R}^{D\times D}$ is shared by the m value columns. The prior is $\mathbf{M}_0 = \mathbf{0}$ and $\mathbf{P}_0 = \sigma_0^2\mathbf{I}$ with $\sigma_0^2 = 3$, the value used in the numerical examples. Analytical results retain general $\sigma_0^2 > 0$. As in the main text, $\rho = \mathbf{k}_A^\top \mathbf{k}_B$ is the key overlap. We also use the angle $\theta = \arccos\rho$ to describe angular resolution. We take $0 < \rho < 1$; endpoint formulas are interpreted by continuity. The repeated-key analysis in Appendix D.3 permits a general incoming covariance; the exact eigenvalue formulas here use an isotropic prior.

D.1.1 CROSS-KEY INTERFERENCE AND CONFUSION

Let $\Delta_{B|A}$ be the readout change at $\mathbf{k}_B$ caused by a write of $\mathbf{k}_A$. With identity dynamics, the predicted mean is $\bar{\mathbf{M}}_t = \mathbf{M}_{t-1}$ and the predicted covariance is $\bar{\mathbf{P}}_t = \mathbf{P}_{t-1} + \ell^2\mathbf{I}$. Define $\beta_t = (r^2 + \mathbf{k}_A^\top \bar{\mathbf{P}}_t \mathbf{k}_A)^{-1}$. Then

$$\begin{aligned}\Delta_{B|A} &:= \hat{\mathbf{v}}_t(\mathbf{k}_B) - \bar{\mathbf{v}}_t(\mathbf{k}_B) \\ &= \beta_t\, \mathbf{k}_B^\top \bar{\mathbf{P}}_t \mathbf{k}_A \left(\mathbf{v}_A - \bar{\mathbf{v}}_t(\mathbf{k}_A)\right).\end{aligned} \tag{28}$$

This is the cross-key interference of Equation (14).

Define $\Delta_{A|A}$ as the readout change at $\mathbf{k}_A$ after the same observation. For $\Delta_{A|A} \neq \mathbf{0}$, their shared value error and scalar gain cancel, giving the confusion defined in Equation (15):

$$\text{confusion} := \frac{\|\Delta_{B|A}\|}{\|\Delta_{A|A}\|} = \frac{|\mathbf{k}_B^\top \bar{\mathbf{P}}_t \mathbf{k}_A|}{\mathbf{k}_A^\top \bar{\mathbf{P}}_t \mathbf{k}_A}. \tag{29}$$

The covariance expression remains defined when the value error is zero, although the ratio of readout changes does not. The figures evaluate this expression. For the repeated-key and orthogonal-gap histories below, its numerator before taking the absolute value is nonnegative.

With isotropic covariance, confusion equals ρ. The Delta-rule and additive updates considered in the main text have the same ratio for any nonzero write along the incoming key. Low confusion means that the readout at $\mathbf{k}_B$ changes little relative to the readout at $\mathbf{k}_A$; confusion near one means that the two readout changes have similar magnitudes.

D.2 CONFUSION AFTER REPEATED WRITES

Proposition 4 (Confusion after repeated writes)**.** *Assume identity dynamics and prior covariance $\mathbf{P}_0 = \sigma_0^2\mathbf{I}$ with $\sigma_0^2 > 0$. Let $\mathbf{k}_B$ and $\mathbf{k}_A$ be unit keys with $\mathbf{k}_B^\top\mathbf{k}_A = \rho$. Let $\mathbf{k}_\perp$ be the unit vector in their plane perpendicular to $\mathbf{k}_B$, so $\mathbf{k}_A = \rho\mathbf{k}_B + \sqrt{1-\rho^2}\,\mathbf{k}_\perp$. Write $\mathbf{k}_B$ n times, then $\mathbf{k}_A$ once, using process variance $\ell^2 \geq 0$ and observation variance $r^2 > 0$ at every write.*

The covariance eigenvectors along and orthogonal to $\mathbf{k}_B$ stay fixed. Let σ_B^2 and $\sigma_\perp^2$ denote the posterior variance of each coordinate of the memory readout at $\mathbf{k}_B$ and $\mathbf{k}_\perp$, respectively. Starting from $\sigma_B^2 = \sigma_\perp^2 = \sigma_0^2$, each $\mathbf{k}_B$ write gives

$$\sigma_\perp^2 \leftarrow \sigma_\perp^2 + \ell^2, \qquad \sigma_B^2 \leftarrow \left(\frac{1}{\sigma_B^2 + \ell^2} + \frac{1}{r^2}\right)^{-1}. \tag{30}$$

With $\sigma_B^2, \sigma_\perp^2$ denoting the values after these n writes, the following $\mathbf{k}_A$ write has confusion

$$\text{confusion} = \frac{\rho}{1 + (\kappa - 1)(1-\rho^2)}, \qquad \kappa = \frac{\sigma_\perp^2 + \ell^2}{\sigma_B^2 + \ell^2}. \tag{31}$$

Here $\bar{\sigma}_B^2 = \sigma_B^2 + \ell^2$ and $\bar{\sigma}_\perp^2 = \sigma_\perp^2 + \ell^2$ are the predicted readout variances before the $\mathbf{k}_A$ observation, so $\kappa = \bar{\sigma}_\perp^2/\bar{\sigma}_B^2$.

Proof. **Prediction and precision gain.** Prediction adds ℓ^2 to every covariance eigenvalue without changing the eigenvectors. Conditioning on $\mathbf{k}_B$ then adds $\mathbf{k}_B\mathbf{k}_B^\top/r^2$ to the predicted precision. This adds $1/r^2$ along $\mathbf{k}_B$ and zero in each orthogonal direction, giving Eq. (30). Thus $\sigma_\perp^2 = \sigma_0^2 + n\ell^2$, while $\sigma_B^2 < r^2$ after each write.

Confusion. Write $\mathbf{k}_A = \rho\mathbf{k}_B + \sqrt{1-\rho^2}\,\mathbf{k}_\perp$, where $\mathbf{k}_\perp$ is a unit vector orthogonal to $\mathbf{k}_B$. Before conditioning on $\mathbf{k}_A$,

$$\bar{\mathbf{P}}_t\mathbf{k}_A = \bar{\sigma}_B^2\rho\mathbf{k}_B + \bar{\sigma}_\perp^2\sqrt{1-\rho^2}\,\mathbf{k}_\perp.$$

Substituting into Eq. (29) and dividing numerator and denominator by $\sigma_B^2 + \ell^2$ gives Eq. (31). □

No process noise. When $\ell^2 = 0$, precision gains accumulate directly:

$$\mathbf{P}_n^{-1} = \sigma_0^{-2}\mathbf{I} + \frac{n}{r^2}\mathbf{k}_B\mathbf{k}_B^\top. \tag{32}$$

The posterior variances are therefore

$$\sigma_B^2 = \left(\sigma_0^{-2} + \frac{n}{r^2}\right)^{-1}, \qquad \sigma_\perp^2 = \sigma_0^2. \tag{33}$$

Hence $\kappa = 1 + n\sigma_0^2/r^2$, and Eq. (31) reduces to

$$\text{confusion} = \frac{\rho}{1 + (n\sigma_0^2/r^2)(1-\rho^2)}. \tag{34}$$

Figure 6a plots this zero-noise case for different amounts of stored evidence.

Resolution angle without process noise. For small angles, $\sin\theta \approx \theta$ and $\cos\theta \approx 1$, and Eq. (34) becomes

$$\text{confusion} \approx \frac{1}{1 + (\theta/\theta^*)^2}, \qquad \theta^{*2} = \frac{1}{\kappa - 1} = \frac{r^2}{n\sigma_0^2}. \tag{35}$$

The resolution angle marks a smooth crossover: closer keys have more strongly coupled updates, while farther keys have weaker cross-key interference. It is not a hard boundary between discrete memories.

With process noise. For $\ell^2 > 0$, each write adds uncertainty in all directions, then reduces it along $\mathbf{k}_B$. After $n \geq 1$ writes, Equation (30) gives

$$\begin{aligned} \sigma_\perp^2 &= \sigma_0^2 + n\ell^2, \qquad 0 < \sigma_B^2 < r^2, \\ \kappa &= \frac{\sigma_0^2 + (n+1)\ell^2}{\sigma_B^2 + \ell^2} > \frac{\sigma_0^2 + (n+1)\ell^2}{r^2 + \ell^2}. \end{aligned} \tag{36}$$

The extra ℓ^2 in κ comes from the final $\mathbf{k}_A$ write. At fixed $\ell^2 > 0$ and r^2, repeated writes still build directional contrast: uncertainty orthogonal to the stored key grows without bound, while uncertainty along it remains bounded by r^2. Figure 6b instead fixes $n = 16$ and $r^2 = 0.048$ and varies ℓ^2; in this setup, increasing process noise broadens the angular range of coupled updates.

D.3 Covariance geometry under repeated writes

We now allow nonzero cross-covariance from earlier writes, rather than requiring the repeated key to remain a covariance eigenvector. This extends Appendix D.2 to the histories used in the recall experiment of Appendix E.1. The repeated key remains $\mathbf{k}_B$, followed by an observation at $\mathbf{k}_A$. Assume identity dynamics, unit keys, and fixed $r^2, \ell^2 > 0$. Write $\mathbf{k}_A = \rho\mathbf{k}_B + s\mathbf{k}_\perp$, where $s = \sqrt{1-\rho^2}$, $0 \leq \rho < 1$, and $\mathbf{k}_\perp$ is a unit vector perpendicular to $\mathbf{k}_B$. Before the nth consecutive B-write, define the predicted covariance entries

$$x_n = \mathbf{k}_B^\top\bar{\mathbf{P}}_n\mathbf{k}_B, \qquad c_n = \mathbf{k}_\perp^\top\bar{\mathbf{P}}_n\mathbf{k}_B, \qquad z_n = \mathbf{k}_\perp^\top\bar{\mathbf{P}}_n\mathbf{k}_\perp.$$

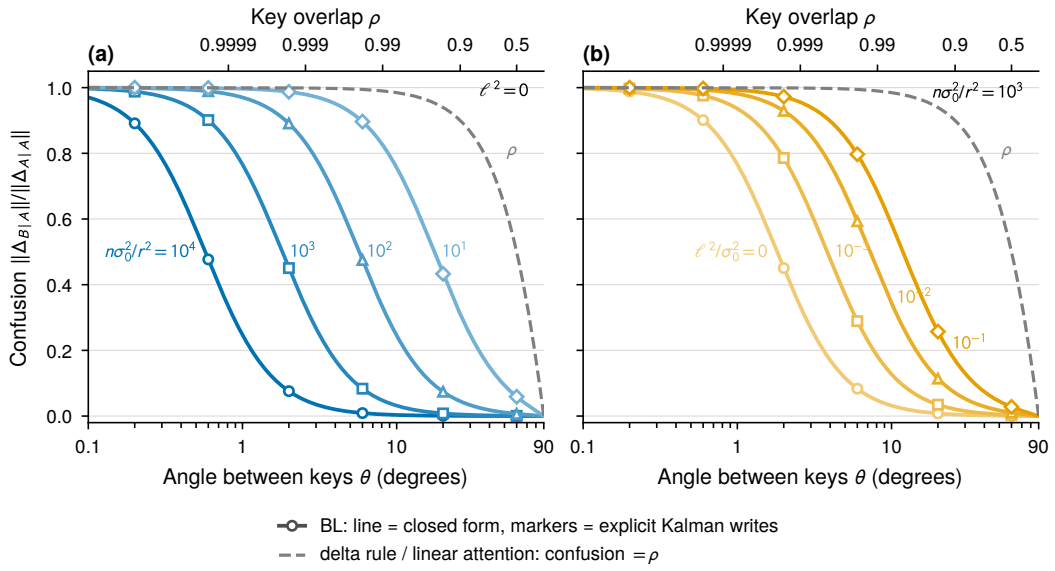


Figure 6: Accumulated evidence and process noise control the angular range of coupled updates. Confusion is the ratio of readout-change magnitudes (Equation (29)), with identity dynamics and $\sigma_0^2 = 3$. (a) No process noise: vary the accumulated evidence $n\sigma_0^2/r^2$; lines show Equation (34). (b) Fixed evidence: $n = 16$ and $r^2 = 0.048$, so $n\sigma_0^2/r^2 = 10^3$. Vary ℓ^2/σ_0^2, adding $\ell^2\mathbf{I}$ before every update during the n writes of $\mathbf{k}_B$ and the final write of $\mathbf{k}_A$. Lines show Equation (31) with eigenvalues from Equation (30). Markers show explicit Kalman updates; dashed curves show the Delta-rule and linear-attention reference $\text{confusion} = \rho$. These are deterministic update calculations, without model training.

Conditioning and prediction give

$$x_{n+1} = \frac{r^2 x_n}{r^2 + x_n} + \ell^2, \tag{37}$$

$$c_{n+1} = \frac{r^2}{r^2 + x_n}\, c_n, \tag{38}$$

$$z_{n+1} = z_n + \ell^2 - \frac{c_n^2}{r^2 + x_n}. \tag{39}$$

Since $x_n \geq \ell^2$, the derivative of the first recurrence is at most $(r^2/(r^2+\ell^2))^2 < 1$. Thus x_n converges to

$$x_* = \frac{\ell^2 + \sqrt{\ell^4 + 4r^2\ell^2}}{2}. \tag{40}$$

The second recurrence contracts c_n geometrically. The cumulative subtraction in the third is therefore finite, giving $z_n = n\ell^2 + O(1)$. Thus the predicted readout variance at $\mathbf{k}_B$ approaches a finite limit, while that at $\mathbf{k}_\perp$ grows. The scalar gain at the repeatedly observed key approaches

$$g^{\text{ss}} = \frac{x_*}{r^2 + x_*}. \tag{41}$$

Since $\mathbf{k}_A^\top \bar{\mathbf{P}}_n \mathbf{k}_A = \rho^2 x_n + 2\rho s c_n + s^2 z_n$, the variance along $\mathbf{k}_A$ increases asymptotically by $(1-\rho^2)\ell^2$ per step. Posterior variances have the same limiting increment because prediction adds ℓ^2 along every unit direction.

For the first subsequent A-write, let x, c, z denote these predicted covariance entries. Assume no coupling between the two-key plane and other coordinates, as in the toy protocol of Appendix E.1. The gain vector $\mathbf{K}_A := \beta_t \bar{\mathbf{P}}_t \mathbf{k}_A$ is

$$\mathbf{K}_A = \frac{(\rho x + sc)\mathbf{k}_B + (\rho c + sz)\mathbf{k}_\perp}{r^2 + \rho^2 x + 2\rho s c + s^2 z}. \tag{42}$$

At fixed $\rho < 1$, increasing the number of preceding B-writes gives

$$\mathbf{K}_A \longrightarrow \frac{\mathbf{k}_\perp}{s}, \qquad \mathbf{k}_A^\top \mathbf{K}_A \longrightarrow 1, \qquad \mathbf{k}_B^\top \mathbf{K}_A \longrightarrow 0. \tag{43}$$

The A-write can therefore correct nearly all of its value error while barely changing the readout at $\mathbf{k}_B$. Greater overlap requires more B-writes for this limit to apply; the finite toy schedule retains cross-key interference.

Update direction and confusion. The covariance reduction at a B-write is

$$\bar{\mathbf{P}}_n - \mathbf{P}_n = \frac{\bar{\mathbf{P}}_n \mathbf{k}_B \mathbf{k}_B^\top \bar{\mathbf{P}}_n}{r^2 + \mathbf{k}_B^\top \bar{\mathbf{P}}_n \mathbf{k}_B} \succeq 0.$$

This compares the beliefs immediately before and after conditioning; process noise prevents a claim of monotone shrinkage across time steps. For the first subsequent A-write, the exact direction in the two-key plane is $(\rho x + sc)\mathbf{k}_B + (\rho c + sz)\mathbf{k}_\perp$. When the cross-covariance terms are negligible, this becomes

$$\mathbf{u}_t = \bar{\mathbf{P}}_t \mathbf{k}_A \approx \rho\, \bar{\sigma}_B^2 \mathbf{k}_B + \sqrt{1-\rho^2}\, \bar{\sigma}_\perp^2 \mathbf{k}_\perp, \tag{44}$$

where $\bar{\sigma}_B^2 = x$ and $\bar{\sigma}_\perp^2 = z$. For $\Delta_{A|A} \neq \mathbf{0}$, the common value error and scalar gain cancel in the ratio of readout-change magnitudes, giving the confusion

$$\frac{\|\Delta_{B|A}\|}{\|\Delta_{A|A}\|} = \frac{|\mathbf{k}_B^\top \bar{\mathbf{P}}_t \mathbf{k}_A|}{\mathbf{k}_A^\top \bar{\mathbf{P}}_t \mathbf{k}_A} = \frac{|\rho x + sc|}{\rho^2 x + 2\rho sc + s^2 z} \approx \frac{\rho\, \bar{\sigma}_B^2}{\rho^2 \bar{\sigma}_B^2 + (1-\rho^2)\, \bar{\sigma}_\perp^2}. \tag{45}$$

The denominator is positive because $\bar{\mathbf{P}}_t \succ 0$. With isotropic covariance, the direction remains parallel to $\mathbf{k}_A$, and confusion equals ρ. In the approximately diagonal regime, a sufficient condition for low confusion is $\rho\, \bar{\sigma}_B^2 \ll (1-\rho^2)\bar{\sigma}_\perp^2$, or $\bar{\sigma}_\perp^2 / \bar{\sigma}_B^2 \gg \rho/(1-\rho^2)$. Thus greater overlap requires a larger variance contrast. For fixed $\rho < 1$, the repeated-write recurrences above give $c \to 0$, bounded x, and growing z, so confusion tends to zero and the update direction approaches the direction perpendicular to $\mathbf{k}_B$.

E Associative Recall with Overlapping Keys

We specify the toy in Figure 3a and derive its readout-error behavior using the covariance analysis in Appendix D.3. We then describe the learned controlled-recall and random-key experiments.

Table 2 summarizes optimization, tuning, and repetition settings across the experiments; individual protocols follow here and in Appendices F and G.

E.1 Toy protocol

Associations and schedule. The toy stores six associations A–F with one-hot values in $\mathbb{R}^6$ and keys in $\mathbb{R}^{16}$:

$$\mathbf{k}_A = \mathbf{e}_1, \qquad \mathbf{k}_B = \rho \mathbf{e}_1 + \sqrt{1-\rho^2}\, \mathbf{e}_2, \qquad (\mathbf{k}_C, \mathbf{k}_D, \mathbf{k}_E, \mathbf{k}_F) = (\mathbf{e}_3, \mathbf{e}_4, \mathbf{e}_5, \mathbf{e}_6),$$

where $\rho = 0.98$. Two initial rounds present each association once, with independent uniform permutations. Ten B-writes then precede ten A-writes, giving 32 writes in total. All methods receive the same sequences; plotted step zero follows the twelve initial writes.

Updates. All methods start from $\mathbf{M}_0 = \mathbf{0}$. BL uses Proposition 1 with $\mathbf{A}_t = \mathbf{I}$, $\mathbf{P}_0 = 3\mathbf{I}$, and $\ell^2 = r^2 = 0.05$. The diagonal variant additionally discards off-diagonal covariance entries after each update. The other recurrences are

$$\mathbf{M}_t = \mathbf{M}_{t-1} + \tfrac{1}{2}\mathbf{k}_t(\mathbf{v}_t - \mathbf{M}_{t-1}^\top \mathbf{k}_t)^\top \qquad \text{(Delta rule)},$$
$$\mathbf{M}_t = \tfrac{1}{2}\mathbf{M}_{t-1} + \tfrac{1}{2}\mathbf{k}_t \mathbf{v}_t^\top \qquad \text{(decayed additive update)}.$$

On unit keys, the Delta rule equals covariance reset with $\bar{\mathbf{P}}_t = 0.05\mathbf{I}$ and $r^2 = 0.05$. The additive baseline here includes decay; the learned linear-attention baseline is undecayed.

Table 2: **Optimization, tuning, budget, and repetition settings.** MQAR learning rates are selected by validation query accuracy; WikiText-103 settings are detailed in Appendix F.2. Both experiments report means and sample standard deviations over three training seeds.

Experiment	Settings
Toy recall with overlapping keys	**Optimization:** None; fixed $r^2{=}\ell^2{=}0.05$. **Tuning:** None. **Budget:** 32 writes per order; 900 order classes. **Repetitions:** Exact order mean ±1 std.
Learned controlled recall	**Optimization:** AdamW; $\beta{=}(0.9, 0.999)$; weight decay 10^{-4}; clipping 1; learning rate 3×10^{-4}. **Tuning:** One shared setting; no per-model tuning. **Budget:** Batch 256; 2,500 updates; query-token cross-entropy. **Repetitions:** 3 seeds; mean ±1 std.
MQAR	**Optimization:** AdamW; $\beta{=}(0.9, 0.95)$; weight decay 0.1; clipping 1; epoch cosine to zero; no warmup. **Tuning:** 4-epoch pilots at width 128; learning rate $\in \{3 \times 10^{-4}, 6 \times 10^{-4}, 10^{-3}\}$. **Budget:** Batch 256; at most 32 epochs; training lengths ≤ 256. **Repetitions:** 3 seeds; mean ±1 std.
WikiText-103	**Optimization:** AdamW; $\beta{=}(0.9, 0.95)$; weight decay 0.01; clipping 1; 20M-token warmup; cosine to $0.1\times$ peak. **Tuning:** Validation-loss learning-rate pilots for baselines; all models use 10^{-3}. **Budget:** Batch 65,536 tokens; 520M tokens; 7,935 updates; sequence length 2,048. **Repetitions:** 3 seeds/model; paired streams.
Layerwise BL distillation	**Optimization:** AdamW; $\beta{=}(0.9, 0.999)$; weight decay 0; clipping 1; learning rate 5×10^{-4}; 25 warmup updates; cosine to $0.1\times$ peak. **Tuning:** Fixed per-layer distillation recipe. **Budget:** 100,007,936 tokens/layer in parallel; 1,526 updates. **Repetitions:** 3 seeds; terminal checkpoints.
GDN continuation	**Optimization:** AdamW; $\beta{=}(0.9, 0.95)$; weight decay 0; clipping 1; learning rate 3×10^{-4}; 50 warmup updates; cosine to $0.1\times$ peak. **Tuning:** Mixers and MLPs at the same four layers; next-token cross-entropy. **Budget:** 100,007,936 tokens; 763 updates; batch 131,072 tokens. **Repetitions:** 3 seeds; terminal checkpoints.

Measurement. After initialization and each subsequent write, we record $E_B(t) = \|\mathbf{M}_t^\top \mathbf{k}_B - \mathbf{v}_B\|^2$, summed over value coordinates. Curves and bands show the mean and one population standard deviation over initial orderings. Only the positions of A and B matter: permuting the four orthogonal identities leaves these readouts unchanged. We therefore average exactly over $(6 \cdot 5)^2 = 900$ equally weighted position pairs. Statistics are computed before clipping values and band boundaries below 10^{-4} to the lowest plotted tick.

E.2 Readout error under subsequent writes

Using the covariance evolution derived in Appendix D.3, let t_0 mark the end of the finite B-write phase and $\mathbf{P}_\mathrm{b} = \mathbf{P}_{t_0}$ its posterior covariance. After n subsequent A-writes, define

$$\mathbf{a}_n = \mathbf{M}_{t_0+n}^\top \mathbf{k}_A, \qquad \mathbf{b}_n = \mathbf{M}_{t_0+n}^\top \mathbf{k}_B.$$

With identity dynamics, an error-correcting update with gain $\mathbf{K}_n$ satisfies

$$\begin{aligned} \mathbf{a}_n &= \mathbf{a}_{n-1} + (\mathbf{k}_A^\top \mathbf{K}_n)(\mathbf{v}_A - \mathbf{a}_{n-1}), \\ \mathbf{b}_n &= \mathbf{b}_{n-1} + (\mathbf{k}_B^\top \mathbf{K}_n)(\mathbf{v}_A - \mathbf{a}_{n-1}). \end{aligned} \tag{46}$$

The main figure evaluates $E_B(t_0+n) = \|\mathbf{b}_n - \mathbf{v}_B\|^2$ for ten A-writes. The following limits describe indefinitely repeated writes from each method's actual boundary state.

Bayesian Layer. For dense covariance and fixed $r^2, \ell^2 > 0$,

$$\begin{aligned} \mathbf{b}_\infty &= \mathbf{b}_0 + \chi_\mathrm{BL}(\mathbf{v}_A - \mathbf{a}_0), \\ \chi_\mathrm{BL} &= \frac{\mathbf{k}_B^\top (\mathbf{P}_\mathrm{b} + x_* \mathbf{I}) \mathbf{k}_A}{\mathbf{k}_A^\top (\mathbf{P}_\mathrm{b} + x_* \mathbf{I}) \mathbf{k}_A}, \end{aligned} \tag{47}$$

where x_* is given by (40).

Proof. Let $p_{A,n} = \mathbf{k}_A^\top \mathbf{P}_{t_0+n} \mathbf{k}_A$ and $w_0 = 1$. The scalar covariance recurrence and innovation contraction are

$$p_{A,n} = \frac{r^2(p_{A,n-1} + \ell^2)}{r^2 + p_{A,n-1} + \ell^2}, \qquad w_n = w_{n-1} \frac{r^2}{r^2 + p_{A,n-1} + \ell^2}, \qquad \mathbf{a}_n - \mathbf{v}_A = w_n(\mathbf{a}_0 - \mathbf{v}_A).$$

Writing $d_n = \mathbf{k}_B^\top \mathbf{P}_{t_0+n} \mathbf{k}_A - \rho p_{A,n}$, the covariance update also gives $d_n = w_n d_0$, since $\mathbf{k}_B - \rho \mathbf{k}_A$ is perpendicular to $\mathbf{k}_A$. Substitution into (46) yields

$$\mathbf{b}_n - \mathbf{b}_0 = \left[\rho(1 - w_n) + d_0 \sum_{j=1}^{n} \frac{w_{j-1}^2}{r^2 + p_{A,j-1} + \ell^2} \right] (\mathbf{v}_A - \mathbf{a}_0).$$

Using the scalar covariance recurrence and $x_*^2 - \ell^2 x_* - r^2 \ell^2 = 0$,

$$\frac{w_{j-1}^2}{r^2 + p_{A,j-1} + \ell^2} = \frac{w_{j-1}^2}{p_{A,j-1} + x_*} - \frac{w_j^2}{p_{A,j} + x_*}. \tag{48}$$

Since $0 \le w_n \le (r^2/(r^2+\ell^2))^n$, the sum telescopes to $1/(p_{A,0} + x_*)$ in the limit. The coefficient becomes $\rho + d_0/(p_{A,0} + x_*) = \chi_\mathrm{BL}$. □

Other updates. For gains parallel to $\mathbf{k}_A$, $\mathbf{b}_n - \mathbf{b}_0 = \rho(\mathbf{a}_n - \mathbf{a}_0)$. The Delta rule uses $\mathbf{K}_n = \frac{1}{2}\mathbf{k}_A$, giving

$$\mathbf{b}_\infty = \mathbf{b}_0 + \rho(\mathbf{v}_A - \mathbf{a}_0). \tag{49}$$

The diagonal variant has the same limit in this toy: $\mathbf{k}_A = \mathbf{e}_1$ keeps its gain parallel to the key, and $\ell^2 > 0$ bounds the scalar gain away from zero. Different boundary readouts $\mathbf{a}_0, \mathbf{b}_0$ account for different plateaus. For the decayed additive update,

$$\mathbf{b}_n = 2^{-n}\mathbf{b}_0 + (1 - 2^{-n})\rho \mathbf{v}_A, \qquad E_B(\infty) = \|\rho \mathbf{v}_A - \mathbf{v}_B\|^2 = 1 + \rho^2, \tag{50}$$

using the distinct one-hot values.

E.3 LEARNED CONTROLLED RECALL

E.3.1 SEQUENCES AND EVALUATION

Each sequence uses $K = 8$ target–distractor pairs (T_i, D_i) in $D = 16$ address dimensions. Here a bar denotes a supplied address, not a predicted quantity:

$$\bar{\mathbf{k}}_{T_i} = \mathbf{e}_{2i-1}, \qquad \bar{\mathbf{k}}_{D_i} = \rho_i \mathbf{e}_{2i-1} + \sqrt{1-\rho_i^2}\,\mathbf{e}_{2i}.$$

Within-pair overlap is ρ_i; different pairs are orthogonal. A fresh random permutation π of $\{1, \ldots, 2K\}$ assigns values $\mathbf{v}_{T_i} = \mathbf{e}_{\pi(2i-1)}$ and $\mathbf{v}_{D_i} = \mathbf{e}_{\pi(2i)}$. This prevents memorizing a fixed address-to-label mapping. Tokens have the form $[\mathtt{type}, \bar{\mathbf{k}}, \mathbf{v}] \in \mathbb{R}^{33}$: a write supplies its one-hot value, while a query supplies zeros in the value coordinates. An explicit write mask freezes each mixer's recurrent state at query positions, so queries read the stored memory without applying state prediction or write updates.

A sequence first writes each identity once, then writes each target $n_b = 4$ times, each distractor n_d times, and finally queries each target once. Writes are independently shuffled within each phase, and query order is random. Training samples $n_d \in \{1, 2, 4, 8\}$ and $\rho_i \sim \mathrm{Uniform}(0.60, 0.80)$. For Figure 3b, evaluation fixes $\rho_i = 0.98$ and sweeps $n_d \in \{1, 2, 4, 8, 16, 32, 64\}$. For panel (c), it fixes $n_d = 64$ and uses a common overlap for all pairs from

$$\rho \in \{0.60, 0.70, 0.80, 0.85, 0.875, 0.90, 0.925, 0.95, 0.96, 0.97, 0.975, 0.98\}.$$

Each condition uses 2,048 sequences per training seed, or 16,384 target queries. Accuracy is the fraction of queries whose predicted label matches the assigned value; chance is $1/(2K) = 6.25\%$. Curves and bands report the arithmetic mean and population standard deviation over three training seeds.

E.3.2 BACKBONE AND MIXER VARIANTS

All models use two residual mixer–MLP blocks with pre-norm RMSNorm, width 64, four heads with key and value dimensions 16, and SwiGLU MLPs of hidden width 128. There are no positional embeddings. A learned map $W_{\mathrm{in}} : \mathbb{R}^{33} \to \mathbb{R}^{64}$ embeds each token into the residual stream. Its address also passes unchanged to every layer and head: $\mathbf{k}_t^{(\ell,h)} = \mathbf{q}_t^{(\ell,h)} = \bar{\mathbf{k}}_t$. Thus learning cannot change the overlap seen by the recurrent update.

Let $\mathbf{z}_t^{(\ell-1)}$ denote the residual-stream input to layer ℓ. Values are $\mathbf{v}_t^{(\ell,h)} = W_V^{(\ell,h)} \mathrm{RMSNorm}(\mathbf{z}_t^{(\ell-1)})$, with $W_V^{(\ell,h)} \in \mathbb{R}^{16\times 64}$. Gates and noise scales use the same normalized input. Each head maintains $\mathbf{M}_t^{(\ell,h)} \in \mathbb{R}^{16\times 16}$ and reads $(\mathbf{M}_t^{(\ell,h)})^\top \bar{\mathbf{k}}_t$ at queries; concatenated head outputs are projected and mapped to logits over the $2K$ labels.

Table 3 specifies the recurrent cores compared in this shared backbone. The Delta-rule label in the main figure denotes the learned DeltaNet-style update; the fixed-reset control appears only in the appendix diagnostics.

Table 3: **Controlled-recall mixer variants.** All variants use the supplied keys and queries.

Mixer	Update specification
Linear attention	Additive update with $\mathbf{A}_t = \mathbf{I}$.
GLA-style	Additive update with $\mathbf{A}_t = \mathrm{diag}(\boldsymbol{\alpha}_t)$.
Delta rule	Error correction with gain $\eta_t \bar{\mathbf{k}}_t$.
Covariance reset	The same correction with fixed gain $\frac{1}{2}\bar{\mathbf{k}}_t$.
Diagonal covariance	Proposition 1, discarding off-diagonal entries of $\mathbf{P}_t$ after each update.
Bayesian Layer	Proposition 1, retaining dense $\mathbf{P}_t$.

Suppressing layer and head indices, let $\tilde{\mathbf{z}}_t = \mathrm{RMSNorm}(\mathbf{z}_t)$ and let σ denote the logistic sigmoid. GLA learns $\boldsymbol{\alpha}_t = \sigma(W_g \tilde{\mathbf{z}}_t + \mathbf{b}_g) \in (0,1)^{16}$; the Delta rule learns $\eta_t = \sigma(\mathbf{w}_\eta^\top \tilde{\mathbf{z}}_t + b_\eta) \in (0,1)$. The reset control uses $\bar{\mathbf{P}}_t = 0.05\mathbf{I}$ and $r^2 = 0.05$. BL and the diagonal variant use identity dynamics and

one covariance per head, with learned input-dependent process variance $\ell_t^2(\tilde{\mathbf{z}}_t) > 0$, fixed $r^2 = 0.05$, and initial covariance $3\mathbf{I}$ at each new sequence.

E.3.3 Optimization

Training minimizes cross-entropy at query positions. We use AdamW with learning rate 3×10^{-4}, $\beta = (0.9, 0.999)$, weight decay 10^{-4}, gradient clipping at norm one, batch size 256, and 2,500 updates. All models use this shared setting without per-model tuning. The random-key experiment below uses the same backbone, optimizer, budget, and three training seeds, with models trained separately.

E.4 Random-key recall

This experiment tests interference beyond the engineered pair structure. Raw addresses are independently sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ in $\mathbb{R}^{64}$ and normalized to unit norm; values are one-hot labels from $\{1, \ldots, 64\}$, re-sampled for each sequence. Each sequence contains n writes followed by eight queries sampled uniformly without replacement from the written keys. Tokens use the same type/address/value format, now with 64 address and 64 value coordinates.

The address pathway bypasses the learned embedding. A shared frozen matrix $R \in \mathbb{R}^{16\times 64}$ with orthonormal rows maps raw addresses to the head dimension, followed by normalization:

$$\tilde{\mathbf{k}}_t = \frac{R\bar{\mathbf{k}}_t}{\|R\bar{\mathbf{k}}_t\|}.$$

Every layer and head uses $\tilde{\mathbf{k}}_t$ as both key and query. Overlap is therefore measured in this projected space. Training uses $n \in \{16, 32, 48, 64\}$; evaluation uses $n \in \{96, 128, 192, 256\}$, corresponding to $n/D \in \{6, 8, 12, 16\}$ with $D = 16$. The ratio n/D indexes the number of associations per key dimension. Figure 7 reports query cross-entropy and accuracy across memory loads. Accuracy is the fraction of queries whose highest-scoring label is correct; chance is $1/64$.

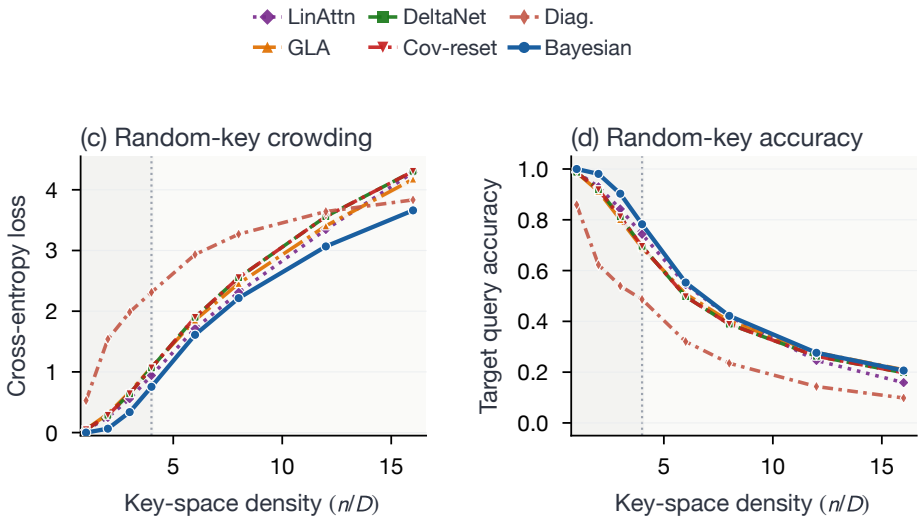


Figure 7: **Random-key crowding.** Query cross-entropy (left) and accuracy (right) versus associations per key dimension. Models are trained separately with $n/D \leq 4$.

F Associative Recall on MQAR and WikiText-103

F.1 MQAR protocol and complete results

Data and evaluation. We use the canonical Zoology MQAR generator (Arora et al., 2024a) with vocabulary size 8,192. Each sequence presents key–value associations and later queries their

values; training minimizes cross-entropy at the query targets. The training pool contains 100,000 examples with sequence length $L = 64$ and $K = 4$ associations, and 20,000 examples for each of $(L, K) = (128, 8)$, $(256, 16)$, $(256, 32)$, and $(256, 64)$. Evaluation uses 1,000 examples per setting, with

$$(L, K) \in \{(64, 4), (64, 8), (64, 16), (128, 32), (256, 64), (512, 128), (1024, 256)\}.$$

We report query accuracy separately for each setting. The final setting appears in Figure 4a; Figure 8 and table 4 give all seven.

Models. All models have two sequence-mixer blocks, tied input and output embeddings, and width $d \in \{64, 128, 256\}$, without intervening MLPs. Head-based mixers use two heads with key and value dimensions $d/2$. Mamba-2 uses expansion factor one, head and state dimensions $d/2$, and two groups for its input and output state projections. Thus width and depth are shared, while the recurrent states and parameter counts follow each architecture. The baselines are Gated DeltaNet (GDN) (Yang et al., 2025), Gated DeltaNet-2 (GDN-2) (Hatamizadeh et al., 2026),[1] MesaNet (von Oswald et al., 2026), RWKV-7 (Peng et al., 2025), Palimpsa-D (Bonnet et al., 2026), Mamba-2 (Dao & Gu, 2024), and causal attention (Vaswani et al., 2017) with rotary position embeddings (Su et al., 2024). For BL, we use identity state dynamics with learned, input-dependent process and observation noise. Process variance is initialized to 0.1. Its GDN-like mixer applies length-four causal convolutions and SiLU to the query, key, and value projections, ℓ_2-normalizes queries and keys, and applies RMS normalization to the readout. This MQAR configuration does not use an output gate; the language model configurations below retain their respective output gates.

Optimization and repetitions. We use AdamW with $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0.1, and gradient clipping at norm one. Training batches contain 256 sequences and evaluation batches contain 32. Learning rates decay to zero with an epoch-level cosine schedule over at most 32 epochs, without warmup. Training stops early if validation query accuracy reaches 0.99, and we retain the checkpoint with the highest validation query accuracy. Four-epoch pilots at width 128 and seed 123 compare peak learning rates $\{3 \times 10^{-4}, 6 \times 10^{-4}, 10^{-3}\}$. We retain 6×10^{-4} for MesaNet and 10^{-3} for the remaining models across widths and seeds. The BL process-noise initialization uses this same learning rate; its noise-projection weights receive a 0.25 learning-rate multiplier. Each reported estimate is the mean over seeds 123, 234, and 345, with sample standard deviation.

F.2 WikiText-103 models and evaluation

Shared training protocol. We train on the official WikiText-103 training split (Merity et al., 2017) using the 32k-BPE `fla-hub/gla-1.3B-100B` tokenizer. Documents begin with a beginning-of-sequence token and are packed into 2,048-token windows. Models have width 512, four mixer–MLP blocks, and tied input and output embeddings. Each run processes 520,028,160 tokens in 7,935 updates of 65,536 tokens, with seeds 0, 1, and 2 and paired token streams across models. We use AdamW with $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0.01, and gradient clipping at norm one. A 20M-token linear warmup precedes cosine decay to 0.1 times the peak learning rate. Baseline learning-rate pilots compare $\{3 \times 10^{-4}, 6 \times 10^{-4}, 10^{-3}\}$ for 20M tokens using validation loss, with a 1M-token pilot warmup. All reported models use a peak learning rate of 10^{-3}; variants that change the surrounding block components retain this rate. Dropout 0.4 is applied to MLP hidden activations. Checkpoints are selected by overall validation loss, using at most 256 validation windows for selection.

GDN-like and RWKV-like BL variants. These names describe the components surrounding the Bayesian recurrence. Both BL variants have four heads with key and value dimensions 128 and use learned diagonal state decay together with input-dependent process and observation noise. Initial decay half-lives span 32 to 4,096 tokens. The initial process and prior variances are calibrated on training windows to give an observation gain of approximately 0.5, with initial observation variance one; eight disjoint training windows check the calibration. The initial prior covariance is then fixed, while the process and observation noise parameters remain trainable.

The GDN-like variant retains GDN's query, key, and value projections, length-four causal convolutions with SiLU, ℓ_2 normalization of queries and keys, and gated RMS-normalized output. It uses

[1] `https://github.com/NVlabs/GatedDeltaNet-2`

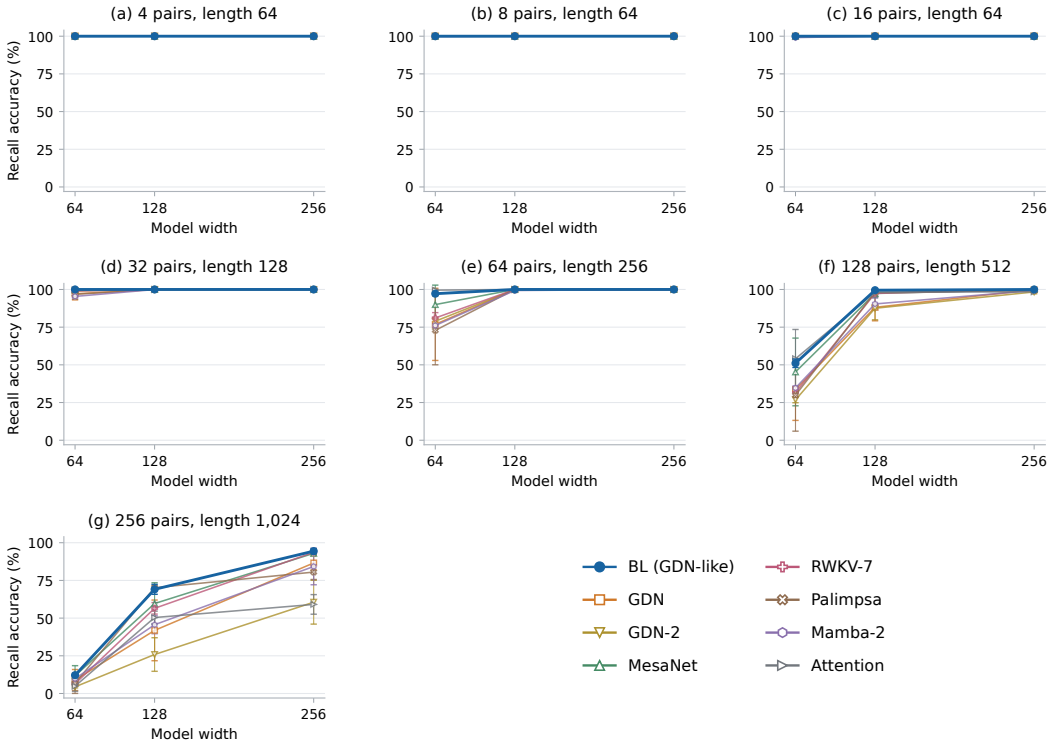


Figure 8: **Complete MQAR evaluation.** Query accuracy as a function of model width for all seven evaluation settings. Titles give the number of key–value associations and sequence length. Models are trained on sequences of length at most 256. Points show means and error bars show one sample standard deviation over three training seeds. The final panel also appears in Figure 4a.

Table 4: **Complete MQAR results.** Query accuracy (%) is reported as mean ± 1 sample standard deviation over seeds 123, 234, and 345. Column headings give the number of associations K and sequence length L.

Model	Width	(K, L)						
		$(4, 64)$	$(8, 64)$	$(16, 64)$	$(32, 128)$	$(64, 256)$	$(128, 512)$	$(256, 1024)$
BL	64	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	97.19 ± 0.57	51.07 ± 2.81	12.14 ± 1.11
	128	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.52 ± 0.20	69.09 ± 3.30
	256	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	94.43 ± 1.70
GDN	64	100.00 ± 0.00	99.99 ± 0.02	99.90 ± 0.12	97.12 ± 4.04	76.85 ± 23.94	33.68 ± 20.53	8.93 ± 6.98
	128	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.88 ± 0.10	88.14 ± 8.93	41.86 ± 20.13
	256	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.85 ± 0.15	86.57 ± 4.60
GDN-2	64	100.00 ± 0.00	99.99 ± 0.01	99.94 ± 0.08	99.01 ± 0.73	78.92 ± 2.48	26.78 ± 1.90	4.41 ± 0.69
	128	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.99 ± 0.01	99.91 ± 0.07	87.53 ± 7.65	25.85 ± 11.14
	256	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.99 ± 0.01	99.93 ± 0.11	98.37 ± 1.24	60.55 ± 14.54
MesaNet	64	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.89 ± 0.17	89.97 ± 12.90	45.29 ± 22.45	11.73 ± 6.73
	128	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	97.34 ± 2.63	59.66 ± 13.79
	256	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.96 ± 0.02	92.97 ± 0.60
RWKV-7	64	100.00 ± 0.00	100.00 ± 0.00	99.98 ± 0.01	99.00 ± 0.55	81.07 ± 3.56	31.57 ± 3.12	6.50 ± 0.99
	128	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	97.41 ± 1.64	56.49 ± 3.82
	256	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.98 ± 0.02	93.67 ± 0.81
Palimpsa	64	99.98 ± 0.03	100.00 ± 0.01	99.93 ± 0.06	96.76 ± 3.18	72.87 ± 22.82	29.87 ± 23.87	6.77 ± 6.70
	128	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.99 ± 0.02	98.20 ± 0.36	70.13 ± 1.50
	256	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.99 ± 0.01	99.25 ± 0.53	80.53 ± 4.91
Mamba-2	64	99.92 ± 0.03	99.74 ± 0.02	99.09 ± 0.27	95.39 ± 0.74	76.11 ± 1.38	34.67 ± 1.28	9.83 ± 1.53
	128	100.00 ± 0.00	99.99 ± 0.01	99.98 ± 0.04	99.94 ± 0.07	99.32 ± 0.61	90.39 ± 3.96	45.61 ± 5.96
	256	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.99 ± 0.00	99.36 ± 0.76	84.25 ± 12.13
Attention	64	99.93 ± 0.05	99.94 ± 0.05	99.89 ± 0.08	99.77 ± 0.23	99.57 ± 0.47	54.04 ± 19.42	4.88 ± 3.38
	128	99.99 ± 0.01	99.98 ± 0.03	100.00 ± 0.00	99.99 ± 0.00	99.99 ± 0.01	98.06 ± 0.66	50.38 ± 4.25
	256	99.91 ± 0.12	99.95 ± 0.03	99.98 ± 0.01	99.99 ± 0.00	99.97 ± 0.03	98.48 ± 0.29	59.08 ± 6.50

RMS normalization around the residual blocks and a SwiGLU MLP of hidden width 1,536. The RWKV-like variant instead retains RWKV-7's two-token mixing, per-head group normalization, local output term, and output gate, with LayerNorm and a squared-ReLU MLP of hidden width 2,048. Both the RWKV-like BL and the RWKV-7 baseline disable cross-layer value sharing. This gives each BL a corresponding baseline with the same surrounding block structure.

Other model configurations. GDN, GDN-2, MesaNet, and Palimpsa-D use four heads of dimension 128 and SwiGLU MLPs of hidden width 1,536. GDN-2 uses independent erase, write, and decay gates. MesaNet uses 30 conjugate-gradient iterations, with its regularizer initialized to one and bounded below by 0.25. Palimpsa-D uses chunk size 16, a rank-32 parameterization of its update control, and fixed prior precision one. Mamba-2 uses expansion factor two, 16 heads with value dimension 64 and state dimension 128, one input/output state-projection group, a length-four causal convolution, and a SwiGLU MLP of hidden width 1,536. Attention uses eight heads of dimension 64, rotary position embeddings, and the same MLP width. These choices retain the architectures' surrounding components; recurrent-state sizes and parameter counts are not constrained to be equal.

Associative-recall evaluation. Following the rare-pair analysis of Zoology and Based (Arora et al., 2024a;b), a target qualifies for associative recall if it completes a token pair whose training-corpus count is at most 16 and whose most recent earlier occurrence is within 1,024 tokens in the same context. The earlier occurrence must be within the same document; tracking resets at document boundaries. Distance is measured to that most recent occurrence, with bins 1–64, 65–256, and 257–1,024 tokens. All models use the same evaluated token positions. The test evaluation contains 325,473 target positions, including 4,301 AR targets; the three distance bins contain 1,669, 1,432, and 1,200 targets, respectively. We compute each metric for each training seed, then report its arithmetic mean and sample standard deviation. In particular, reported mean perplexities average the per-seed perplexities. Table 5 gives the full results.

Table 5: **WikiText-103 test results.** Means ± 1 sample standard deviation over three training seeds. LM and AR perplexity are measured on all evaluated tokens and on associative-recall targets, respectively. The final three columns report AR accuracy by distance to the most recent occurrence. All models use BPE tokens, so perplexities are comparable within this table rather than with word-level WikiText-103 results. RWKV-7 and the RWKV-like BL both disable value sharing.

Model	LM PPL ↓	AR PPL ↓	AR accuracy (%) ↑			
			All	1–64	65–256	257–1,024
BL (GDN-like)	18.94 ± 0.18	8.63 ± 0.55	60.09 ± 1.36	70.66 ± 1.62	64.13 ± 1.00	40.58 ± 1.96
GDN	18.58 ± 0.17	9.29 ± 1.20	58.66 ± 1.42	71.20 ± 1.75	62.92 ± 1.33	36.14 ± 2.18
GDN-2	17.85 ± 0.12	8.20 ± 0.54	60.16 ± 0.73	73.84 ± 0.93	63.24 ± 0.41	37.47 ± 2.75
MesaNet	20.69 ± 0.23	12.78 ± 1.87	52.53 ± 2.64	65.23 ± 1.77	57.38 ± 2.99	29.08 ± 4.02
Palimpsa	18.61 ± 0.20	10.30 ± 1.60	56.34 ± 2.41	69.14 ± 3.47	61.29 ± 2.76	32.61 ± 1.38
Mamba-2 + MLP	18.61 ± 0.05	20.02 ± 1.60	45.82 ± 1.19	58.92 ± 0.66	50.95 ± 1.64	21.47 ± 1.71
BL (RWKV-like)	18.12 ± 0.14	6.37 ± 0.24	63.84 ± 0.63	75.77 ± 1.12	68.13 ± 0.35	42.11 ± 0.38
RWKV-7	17.49 ± 0.06	7.31 ± 0.77	60.86 ± 2.08	73.94 ± 1.65	63.97 ± 2.09	38.97 ± 3.00
Attention	19.35 ± 0.08	3.92 ± 0.18	74.30 ± 0.64	83.20 ± 0.57	75.49 ± 0.55	60.50 ± 1.24

G Distillation and Long-Context Retrieval

Model and layerwise distillation. All variants start from the released 340M Gated DeltaNet checkpoint, `linear-moe-hub/Gated-Deltanet-340M`. It has 24 layers, hidden width 1,024, and four heads of dimension 256. We replace the sequence mixers at zero-based indices $\mathcal{I} = \{5, 11, 17, 23\}$ with Bayesian Layers using rotational dynamics and no short convolution. Each BL head uses 512 real state coordinates and one shared 512×512 covariance matrix. Each replacement is fitted to the corresponding frozen GDN mixer's outputs on frozen teacher inputs:

$$\mathcal{L}^{(\ell)} = \frac{\mathrm{MSE}\big(Y_\ell^{\mathrm{BL}}, Y_\ell^{\mathrm{GDN}}\big)}{\mathrm{Var}\big(Y_\ell^{\mathrm{GDN}}\big) + 10^{-6}} + \frac{0.05}{B} \sum_{b=1}^{B} \Big(1 - \cos\big(y_{\ell,b}^{\mathrm{BL}}, y_{\ell,b}^{\mathrm{GDN}}\big)\Big), \tag{51}$$

Here, $Y_\ell^\bullet$ denotes the full batch of mixer outputs at layer ℓ, and $y_{\ell,b}^\bullet$ denotes sequence b's output flattened over tokens and features. MSE and teacher-output variance are computed over all batch, token, and feature entries, separately at each fitted layer. The cosine term is computed per sequence and averaged over the B sequences in the batch. The MLPs and the remaining backbone parameters are frozen. The four trained replacements are assembled without subsequent joint fine-tuning.

We train each replacement for 1,526 updates on 100,007,936 input tokens, using 2,048-token SlimPajama sequences and a global batch of 65,536 tokens. AdamW uses learning rate 5×10^{-4}, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, no weight decay, gradient clipping at 1, 25 warmup updates, and cosine decay to 0.1 times the peak learning rate. The runs use mixed precision and three training seeds, varying initialization and document order. We evaluate the terminal checkpoint of each run.

Continued-GDN control and token accounting. The continuation starts from the same original GDN checkpoint and updates the mixers and MLPs at the four selected layers with ordinary next-token cross-entropy. All other parameters remain frozen. It uses 763 updates of 131,072 tokens, or 100,007,936 input tokens per run, at sequence length 2,048. AdamW uses learning rate 3×10^{-4}, $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, no weight decay, gradient clipping at 1, 50 warmup updates, and cosine decay to 0.1 times the peak learning rate. We use three training seeds and evaluate terminal checkpoints.

Each selected mixer therefore sees approximately 100M training tokens under either procedure, giving 400M token exposures summed across the four mixers. BL uses a separately shuffled stream at each replacement site. Continued GDN processes one stream through all four selected layers, reusing the corresponding BL seed's layer-5 stream. This matches token exposure per selected mixer; the objectives, trainable parameters, and total computation differ.

Retrieval evaluation. We evaluate six RULER needle-in-a-haystack tasks at context lengths 2,048, 4,096, 8,192, and 16,384. Each task and length has 200 examples, giving 4,800 examples per model and training seed. All methods use the same prompts and evaluation batches. Decoding uses cached

full-context prefill followed by greedy generation, stopping at EOS or 128 generated tokens. We retain RULER's fractional credit for multi-answer tasks. Table 6 reports the full task breakdown. The tables report sample standard deviations over training seeds; the frozen GDN reference is a single checkpoint evaluated on the same examples.

For the heatmaps, we divide the recorded answer position by the prompt length and group examples into ten equal-width position bins. Within each bin, the aggregate panel averages the examples across all six tasks, and each task panel uses only that task's examples. Colors show the difference between the BL seed mean and the indicated reference. Figure 5 compares BL with continued GDN; Figure 9 uses the original frozen checkpoint.

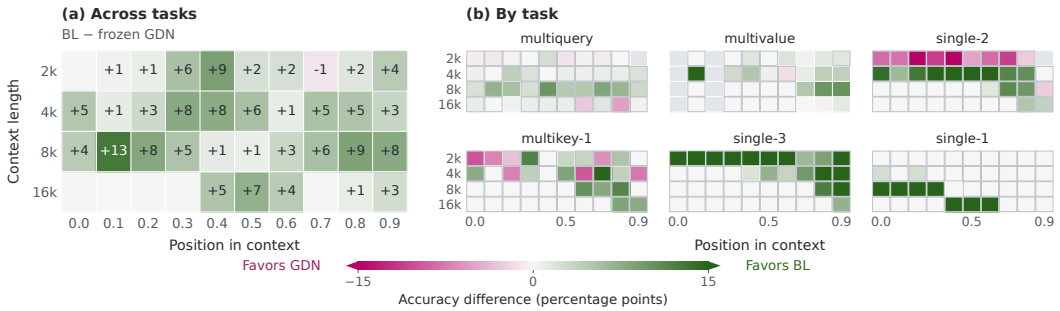


Figure 9: **Long-context retrieval relative to frozen GDN.** BL minus frozen-GDN retrieval accuracy on RULER, **(a)** aggregated across six needle-in-a-haystack tasks and **(b)** separately for each task. BL averages three training seeds; frozen GDN is the original checkpoint. The evaluation examples, layout, and color scale match Figure 5. Gray outlines mark cells containing fewer than 30 examples; gray fill indicates empty cells.

Table 6: **RULER retrieval accuracy (%).** BL and continued GDN report mean ± sample standard deviation over three training seeds; frozen GDN is a single checkpoint. Each task and context length uses the same 200 examples for all methods. The final column averages the six tasks.

Context	Model	Single-1	Single-2	Single-3	Multikey-1	Multivalue	Multiquery	Mean
2048	Frozen GDN	100.00	99.00	26.50	27.50	22.88	24.62	50.08
	Continued GDN	100.00 ± 0.00	97.50 ± 1.80	26.33 ± 3.62	27.00 ± 1.32	23.88 ± 0.54	24.75 ± 0.33	49.91 ± 0.73
	BL	100.00 ± 0.00	89.83 ± 2.52	49.33 ± 9.50	27.00 ± 1.50	24.08 ± 0.14	24.33 ± 0.19	52.43 ± 1.50
4096	Frozen GDN	99.00	50.50	0.50	22.50	16.75	22.75	35.33
	Continued GDN	99.17 ± 0.29	49.00 ± 2.65	1.17 ± 1.15	23.50 ± 0.50	18.29 ± 0.81	22.79 ± 0.19	35.65 ± 0.57
	BL	99.50 ± 0.50	66.50 ± 7.94	5.83 ± 2.75	24.50 ± 0.50	19.54 ± 3.72	24.04 ± 0.19	39.99 ± 2.19
8192	Frozen GDN	76.00	17.50	1.00	11.50	4.75	8.75	19.92
	Continued GDN	75.83 ± 5.86	18.17 ± 1.15	1.00 ± 0.50	12.67 ± 1.26	6.17 ± 1.13	9.21 ± 1.00	20.51 ± 1.27
	BL	88.33 ± 2.02	19.67 ± 0.58	4.17 ± 0.76	15.33 ± 0.58	11.38 ± 1.95	14.17 ± 0.44	25.51 ± 0.41
16384	Frozen GDN	33.50	7.50	2.00	5.00	3.50	5.62	9.52
	Continued GDN	35.17 ± 3.88	7.00 ± 1.32	2.50 ± 0.50	5.00 ± 0.00	3.33 ± 0.19	5.33 ± 0.69	9.72 ± 0.87
	BL	44.17 ± 2.93	8.33 ± 1.04	2.67 ± 0.29	6.17 ± 0.29	3.88 ± 0.12	5.58 ± 0.26	11.80 ± 0.58

Language-model evaluation. Table 7 reports perplexity on WikiText-103 test, held-out SlimPajama, and PG-19 validation. We use the released GDN tokenizer and identical token arrays and window boundaries for all models. WikiText-103 concatenates the nonempty test rows without additional BOS or EOS tokens. SlimPajama uses a held-out pool and the first complete windows reaching at least one million predicted tokens at each context length. PG-19 uses the first 50 validation books with EOS separators, capped at three million tokens. Each corpus is split into nonoverlapping windows, with $L - 1$ next-token targets per window of length L. We compute corpus cross-entropy over all predicted tokens and exponentiate it to obtain each model's perplexity, then report the arithmetic mean and sample standard deviation over training seeds.

Table 7: **Language-model perplexity after adaptation.** BL and continued GDN report mean ± sample standard deviation over three training seeds; frozen GDN is a single checkpoint. All methods use identical token windows within each corpus and context length. Targets gives the number of predicted tokens per model and seed.

Corpus	Context	Frozen GDN	Continued GDN	BL	Targets
WikiText-103	2048	15.942	16.076 ± 0.024	16.684 ± 0.061	315,238
	4096	15.063	15.190 ± 0.027	15.747 ± 0.054	315,315
	8192	14.566	14.697 ± 0.032	15.208 ± 0.052	311,258
	16384	14.438	14.583 ± 0.036	15.054 ± 0.047	311,277
SlimPajama	2048	10.938	10.994 ± 0.005	11.311 ± 0.010	1,000,983
	4096	10.633	10.686 ± 0.005	10.987 ± 0.010	1,003,275
	8192	10.512	10.564 ± 0.005	10.857 ± 0.009	1,007,493
	16384	10.476	10.527 ± 0.005	10.817 ± 0.008	1,015,746
PG-19	2048	20.295	20.651 ± 0.288	21.343 ± 0.080	2,996,808
	4096	19.637	19.973 ± 0.271	20.643 ± 0.070	2,997,540
	8192	19.290	19.617 ± 0.265	20.265 ± 0.066	2,997,906
	16384	19.139	19.463 ± 0.262	20.093 ± 0.063	2,998,089

Existing assets and licenses. The experiments use public research assets only. Zoology/MQAR (Arora et al., 2024a) and RULER (Hsieh et al., 2024) are released under Apache-2.0 licenses. The `linear-moe-hub/Gated-Deltanet-340M` checkpoint (Du et al., 2026; Yang et al., 2025) is released on Hugging Face under Apache-2.0. SlimPajama (Soboleva et al., 2023) is an open dataset release whose documentation lists source-specific terms for its constituent subsets. WikiText-103 (Merity et al., 2017) is distributed under CC-BY-SA-3.0/GFDL terms, and PG-19 (Rae et al., 2020) is released from an Apache-2.0 repository with source texts drawn from Project Gutenberg. We cite the original creators and do not redistribute the underlying assets in this paper.