RAM: ROUTED ASSOCIATIVE MEMORIES LEARN TO RECALL IN RECURRENT LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recurrent sequence mixers such as Mamba-2, Gated DeltaNet, and Kimi Delta Attention offer a more efficient alternative to softmax attention, but lag behind on in-context recall. On synthetic MQAR, an Online Hebbian MLP achieves information-theoretically optimal storage scaling and matches attention KV-cache capacity with up to $69\times$ fewer state scalars. Yet pretrained recurrent language models recall substantially worse than attention. To study this discrepancy, we introduce noisy MQAR, which combines in-context recall with competing next-token supervision. Using a polynomial-kernel memory family, we identify gradient interference from the background objective and greater parameter sensitivity of low-degree recall margins as two obstacles to learning recall circuits. We address these obstacles with Routed Associative Memory (RAM), which upweights recall-relevant tokens and routes computation across multiple recurrent memories. At 1.4B parameters, RAM closes 56.02% of the average-recall gap to attention with 0.51% more parameters than Gated DeltaNet, while its LM-Evals score is within 0.30 percentage points of the Gated DeltaNet baseline. At 360M parameters, adding n-gram history to the router closes the average-recall gap to attention entirely with two memories and 94.32% of it with eight.

1 INTRODUCTION

Language models (LMs) are increasingly used for long-context tasks such as coding, document question answering, and reasoning over large corpora, all of which require recalling information from across the context. Prior work has found that in-context recall is performed by sequence-mixing layers in LMs, with softmax attention achieving state-of-the-art performance on recall tasks (Arora et al., 2024a; Jelassi et al., 2024; Wen et al., 2024). Recurrent sequence mixers such as Mamba (Gu & Dao, 2024; Dao & Gu, 2024), Gated DeltaNet (Yang et al., 2025), and Kimi Delta Attention (KDA) (Kimi Team, 2025) present a more efficient alternative to softmax attention, and are widely adopted in state-of-the-art open-source language models like Kimi (Kimi Team, 2026) and Qwen (Qwen Team, 2025; 2026). However, despite their efficiency gains, recurrent mixer architectures continue to lag behind softmax attention in recall performance (Arora et al., 2024a;b; Jelassi et al., 2024; Wen et al., 2024).

Associative memory theory (Garcia et al., 2026) suggests that the in-context storage capacity of recurrent mixers should scale linearly (up to log factors) with state-size, at the information-theoretically optimal rate. We find this is the case empirically on synthetic recall tasks (MQAR (Arora et al., 2024a)), with recurrent mixers achieving larger storage capacity than softmax attention for a fixed state-size budget. However, the performance on recall tasks of pretrained recurrent LMs falls behind softmax-attention LMs.

To this end, in this work, we ask: *why does LM pretraining fail to realize the storage capacity of recurrent mixers, and how can understanding this failure guide better training and memory architectures?*

Next-token prediction language modeling is a massively multitask objective: a single model must learn recall alongside many competing prediction tasks (Brown et al., 2020). To study how this competition affects the acquisition of recall capabilities, we introduce *noisy MQAR* (Definition 4.1), a controlled setting that combines in-context recall with a broader noisy background task. Compared with softmax attention, recurrent mixers struggle to learn recall circuits as background supervision

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.

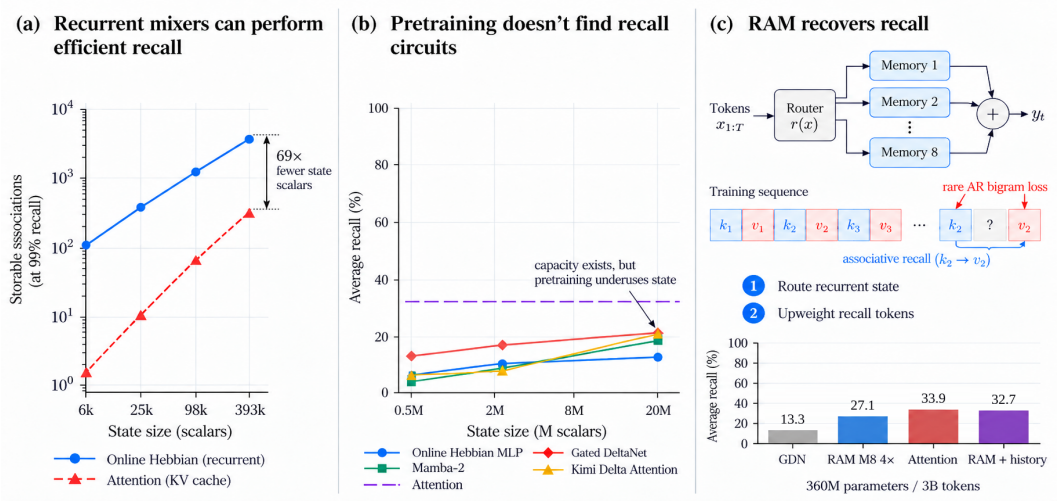


Figure 1: **Recurrent recall capacity, the pretraining gap, and Routed Associative Memory (RAM). (a)** On synthetic associative recall, the Online Hebbian recurrent mixer uses $69\times$ fewer state scalars than attention's KV cache at 99% recall. **(b)** After language-model pretraining, increasing recurrent state size yields limited gains in average recall: at 20M state scalars, the evaluated recurrent mixers reach 12.90%–20.80%, below the attention reference at 32.10%. **(c)** RAM routes tokens across eight recurrent memories and upweights rare associative-recall bigrams during training. At 360M parameters and 3B training tokens, Gated DeltaNet, RAM with eight memories and $4\times$ recall upweighting, attention, and RAM with history-informed routing achieve 13.32%, 27.09%, 33.91%, and 32.74% average recall, respectively.

increases: for example, at 80% background-loss share, Mamba-2 and Gated DeltaNet achieve only 0.30% and 33.50% recall accuracy, respectively, while softmax attention retains 100.00% recall through 98% background-loss share (Figure 3).

This diagnosis motivates two complementary interventions: a router that selects a recurrent memory for each token, separating task selection from task execution; and stronger supervision on recall tokens. We thus combine them in Routed Associative Memory (RAM), a wrapper for recurrent mixers that routes tokens across memories and upweights recall tokens during training. Pretrained at 1.4B parameters on 10B tokens, RAM closes 56.02% of the average-recall gap to softmax attention while adding only 0.51% more parameters than Gated DeltaNet (Tables 1 and 9). Moreover, at 360M parameters, giving the router n-gram statistics of the preceding context fully closes the average-recall gap with softmax attention (Table 2).

Concretely, we make three contributions:

- **We demonstrate that recurrent mixers can store associations at the information-theoretically optimal rate, but fail to do so when pretrained as language models.** We introduce a simple recurrent mixer, *the Online Hebbian MLP*, and use associative memory theory (Garcia et al., 2026) to show that it stores T key–value pairs using $\Theta(T \log T)$ state scalars, matching the information-theoretically optimal rate (Theorem 3.2). Empirically, we show that the storage capacities of Mamba-2 (Dao & Gu, 2024), Gated DeltaNet (Yang et al., 2025), and Kimi Delta Attention (Kimi Team, 2025) grow with state size at a rate consistent with this prediction (Figure 2). Yet at approximately matched parameter counts, pretrained Gated DeltaNet reaches only 13.20% average recall accuracy against softmax attention's 32.10%; a $30.4\times$ increase in recurrent state, with substantially more parameters, raises it to only 20.80% (Table 11).
- **We identify why recurrent mixers fail to learn recall circuits under language-modeling supervision.** To isolate the effect of competing prediction objectives, we introduce *noisy MQAR*, which combines in-context recall with background next-token prediction (Definition 4.1). Recurrent mixers fail to learn recall circuits at lower noise levels than softmax attention: at 80% background loss share, Mamba-2 and Gated DeltaNet reach 0.30% and 33.50% recall accuracy, respectively, while attention maintains 100.00% through 98% background loss share (Figure 3). We attribute

this gap to two effects: destructive interference from the competing gradient and greater sensitivity of low-degree recall margins to parameter perturbations (Section 4.2). Motivated by these findings, we propose two simple fixes: routing across memories and recall-intensive training, and show that they increase recall accuracy on noisy MQAR (Section 4.3 and Figure 4).

- **We introduce Routed Associative Memory (RAM), closing much of the recall gap while largely preserving LM-Evals accuracy.** RAM routes tokens across multiple recurrent memories and upweights recall supervision during training (Section 5.1). At 1.4B parameters and 10B training tokens, it closes 56.02% of the average-recall gap to softmax attention while remaining within 0.30 percentage points of Gated DeltaNet on LM-Evals (Arora et al., 2024b; Gao et al., 2024) and adding only 0.51% more parameters (Tables 1 and 9). RAM's recall scales more strongly with state size than that of prior growing-state recurrent mixers (Figure 5). Finally, we show that at 360M parameters, a history-informed router using n-gram statistics closes 94.32% of the average-recall gap to softmax attention with eight memories, and matches attention with two (Section 5.2 and Table 2).

2 Background and Related Work

2.1 Softmax and Linear Attention

Softmax attention is the workhorse of modern large language models (Vaswani et al., 2017). For a sequence of queries, keys, and values, causal softmax attention computes

$$o_t = \sum_{i \le t} a_{ti} v_i, \qquad a_{ti} = \frac{\exp(q_t^\top k_i/\sqrt{d})}{\sum_{j \le t} \exp(q_t^\top k_j/\sqrt{d})}. \tag{1}$$

The exponential similarity in softmax attention can be interpreted as a kernel and enables precise retrieval across long contexts (Katharopoulos et al., 2020; Arora et al., 2024a; Zhang et al., 2025). However, exact attention requires retaining preceding keys and values in a cache that grows linearly with sequence length.

Linear attention instead uses a separable kernel $K(q,k) = \phi(q)^\top \phi(k)$, allowing the computation to be reorganized as a recurrent update (Katharopoulos et al., 2020):

$$M_t = M_{t-1} + v_t \phi(k_t)^\top, \qquad z_t = z_{t-1} + \phi(k_t), \qquad o_t = \frac{M_t \phi(q_t)}{z_t^\top \phi(q_t)}. \tag{2}$$

These models compress the prefix into fixed-size recurrent statistics (M_t, z_t), yielding constant-state autoregressive inference. However, prior work shows that they struggle to match softmax attention on long-context recall tasks (Arora et al., 2024a;b).

2.2 Kernel Hebbian Memories

We build upon the fact-storage framework of Garcia et al. (2026).

Definition 2.1 (Fact set and recall margin)**.** Let $E = \{v_1, \dots, v_N\} \subset \mathbb{S}^{d-1}$ be a finite vocabulary of value embeddings. A fact set $\mathcal{F} = \{(k_i, v_{f(i)})\}_{i=1}^T$ pairs keys $k_i \in \mathbb{S}^{d-1}$ with values $v_{f(i)} \in E$, where $f(i) \in \{1, \dots, N\}$ specifies the correct value for key i. For a memory read g, define the margin of query k_i and the minimum margin of the fact set by

$$m_i(g; \mathcal{F}) = \min_{j \neq f(i)} \left[\langle g(k_i), v_{f(i)} \rangle - \langle g(k_i), v_j \rangle \right], \qquad \gamma_{\mathcal{F}}(g) = \min_{1 \le i \le T} m_i(g; \mathcal{F}). \tag{3}$$

We say that g stores $\mathcal{F}$ if $\gamma_{\mathcal{F}}(g) > 0$, so the correct value outscores every alternative for every key.

Definition 2.2 (Online Hebbian memory)**.** Let $K_\kappa(q,k) = \langle \phi_\kappa(q), \phi_\kappa(k) \rangle$ be a kernel with feature map $\phi_\kappa : \mathbb{R}^d \to \mathbb{R}^m$. An Online Hebbian memory starts from $M_0 = 0$ and writes each key–value pair (k_t, v_t) to the $d \times m$ state matrix by $M_t = M_{t-1} + v_t \phi_\kappa(k_t)^\top$. Given query q, it outputs:

$$M_T = \sum_{i=1}^T v_{f(i)} \phi_\kappa(k_i)^\top, \qquad g_T(q) = M_T \phi_\kappa(q) = \sum_{i=1}^T v_{f(i)} K_\kappa(k_i, q). \tag{4}$$

Garcia et al. (2026) propose a bilinear MLP construction that implements a kernel Hebbian memory. It achieves asymptotically information-theoretically optimal storage capacity scaling, storing $F = \Theta(W/\log W)$ facts with W finite-precision parameters under isotropic embeddings. In Section 3, we adapt their construction and their margin framework to study the *in-context capacity* of the state matrices in recurrent sequence mixers.

2.3 Related Work

Prior work identifies in-context recall as a major bottleneck for recurrent and other subquadratic sequence mixers. Arora et al. (2024a) isolates recall-intensive tokens as a major source of the quality gap, and recent complementary work studies copying and retrieval limits in recurrent models (Jelassi et al., 2024; Wen et al., 2024). In contrast, we study why recurrent mixers learn effective recall circuits in isolation but fail to acquire them during LM pretraining.

Another line of work increases recurrent state capacity through hierarchical states, routed memories, growing states, or dynamic allocation and caching (Qin et al., 2024; Du et al., 2025; Pan et al., 2025; Guo et al., 2026; Waitz et al., 2026; Wang et al., 2026; Behrouz et al., 2026). We show using associative memory theory that in-context storage capacity of recurrent mixers can scale at an asymptotically information-theoretically optimal rate (Theorem 3.2), but that surprisingly, existing recurrent architectures struggle to realize this recall capacity scaling in LM pretraining. Our analysis of this learning gap motivates RAM, which introduces routing across memories and stronger recall supervision.

3 Quantifying the Storage Capacity of Recurrent Mixers

In this section, we study the theoretical in-context storage capacity of recurrent sequence mixers, how it reflects empirically when they are trained on synthetic recall tasks (MQAR (Arora et al., 2024a)) and when trained on language modeling.

3.1 Recurrent Sequence Mixers Can Do Recall

Our first observation is that recurrent sequence mixers recall key–value pairs at the information-theoretically optimal rate and outperform self-attention on *storage capacity* at a fixed memory budget. We analyze this capacity scaling for an Online Hebbian MLP and also test it empirically for Mamba-2 (Dao & Gu, 2024), Gated DeltaNet (Yang et al., 2025), and Kimi Delta Attention (Kimi Team, 2025).

Definition 3.1 (Online Hebbian MLP)**.** The *Online Hebbian MLP* is the kernel online Hebbian memory of Definition 2.2 with the bilinear feature map

$$\phi(x) = (Ax) \odot (Gx), \qquad A, G \in \mathbb{R}^{m \times d}. \tag{5}$$

Theorem 3.2 (Online Hebbian MLP storage capacity, informal)**.** *Let g be an Online Hebbian MLP sequence mixer (Defn. 3.1) with state matrix $M \in \mathbb{R}^{d \times m}$. Under isotropic keys and values $k_i, v_i \sim \mathrm{Unif}(\mathbb{S}^{d-1})$, g stores T key–value pairs using a state of size*

$$|M| = \Theta(md) = \Theta(T \log T)\,,$$

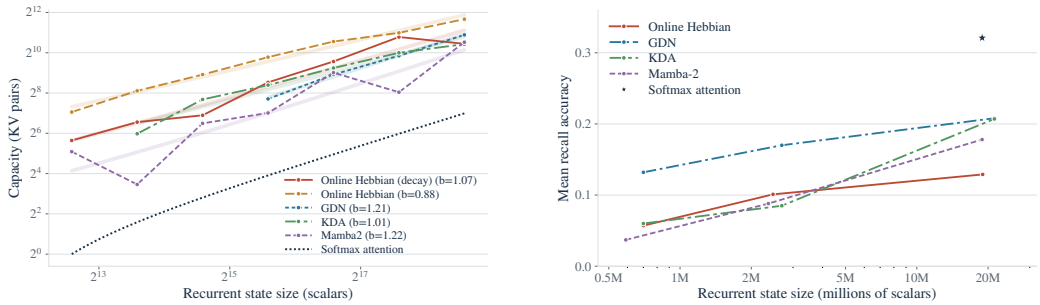
This matches the asymptotically information-theoretically optimal storage-capacity scaling.

See Section C.1 for the formal statement and proof.

We train mixers on MQAR, and report their storage capacity at several state-size budgets, measured as the largest number of key–value pairs recalled at $\geq 99\%$ accuracy ($T@99\%$). The storage capacity of most of these recurrent mixers follows the asymptotically optimal scaling predicted by Theorem 3.2. Moreover, empirically, they obtain up to $\sim 128\times$ larger storage capacity than self-attention at a state-size budget (Figure 2a).

3.2 Recurrent Language Models Fail to Do Recall

While the previous section showed that recurrent mixers can do recall, with even better storage capacity than self-attention, here we find that when recurrent sequence mixers are used inside LMs, they are unable to scale their recall performance with state size the way they do on synthetics.



(a) MQAR (synthetic) **(b)** Pretrained LMs

Figure 2: **Storage capacity of sequence mixers when trained on synthetic recall tasks (MQAR) and on language modeling. (a)** MQAR storage capacity. The evaluated recurrent mixers exhibit the asymptotically optimal storage-capacity scaling of Theorem 3.2. We fit $T \ln T \propto |M|^b$, where $b = 1$ is the predicted rate. Table 3 gives 95% confidence intervals. The self-attention curve shows its KV-cache state size for the same number of associations. **(b)** The same sequence mixers pretrained on the Pile: average recall accuracy (mean of four-task NIAH accuracy, FDA, and SWDE) against recurrent state size $|M|$. Full numbers in Table 11.

We pretrain recurrent language models, varying their state size over $\{|M|, 4|M|, 32|M|\}$, and evaluate on recall tasks: FDA and SWDE (Arora et al., 2024b) and four NIAH tasks from RULER (Hsieh et al., 2024). We find that, unlike synthetic MQAR experiments, the recall accuracy of recurrent mixers within LMs heavily underperforms self-attention at similar state size: Gated DeltaNet reaches 20.80% versus self-attention's 32.10%, a 35.20% relative shortfall (Figure 2.b).

4 Why Recurrent Mixers Struggle to Learn Recall in Pretraining

Section 3 shows that recurrent mixers can learn high-capacity recall circuits when trained directly on recall, but not when pretrained on language modeling. To explain this discrepancy, we introduce *noisy MQAR*, which combines in-context recall with a competing next-token prediction objective. Interestingly, while softmax attention retains perfect recall accuracy at high background loss weight, recurrent mixers increasingly lose their recall capabilities (Figure 3). Our analysis identifies two causes: the background objective interferes with recall learning, and low-degree recall circuits are more sensitive to parameter changes. This motivates two simple interventions to reduce this interference: token-wise routing, to allow the mixer to separate recall from non-recall learning, and recall-intensive training, to strengthen the recall signal.

4.1 Linear Attention Struggles to Learn Recall Circuits under Noise

We formalize competing recall and next-token objectives with the following controlled task.

Definition 4.1 (Noisy MQAR)**.** Noisy multi-query associative recall (MQAR) is a variant of synthetic MQAR (Arora et al., 2024a) that combines in-context recall with a background next-token objective. Each sequence contains F key–value facts, interleaved background tokens ξ_t, and a query tail that asks for stored values:

$$\begin{array}{rcccccc|cccc} & \xi_1 & k_1 & v_1 & \xi_2 & k_2 & v_2 & k_2 & \xi_3 & k_1 & \xi_4 \\ \text{target:} & k_1 & v_1 & \xi_2 & k_2 & v_2 & k_2 & v_2 & k_1 & v_1 & - \\ \text{objective:} & B & B & B & B & B & B & R & B & R & - \end{array}$$

Here, R marks recall supervision, where the model must recall the relevant fact, and B marks ordinary next-token supervision, where the model must output a uniform prior over the keys. We train with:

$$\mathcal{L}_\rho = (1-\rho)\mathcal{L}_R + \rho\mathcal{L}_B, \qquad 0 \le \rho < 1,$$

where $\mathcal{L}_R$ and $\mathcal{L}_B$ are cross-entropy losses over recall and non-recall positions, respectively, and ρ is the background loss weight.

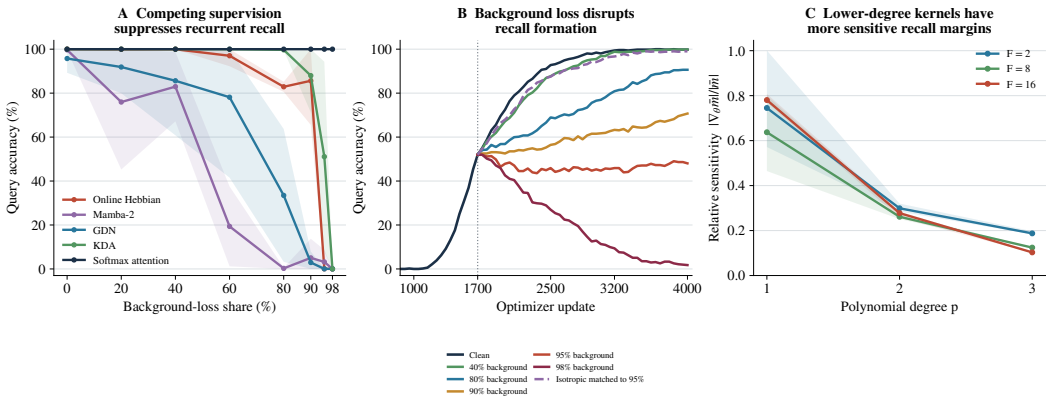


Figure 3: **Competing supervision and recall-margin sensitivity. (a)** As background-loss weight increases accuracy decreases for recurrent mixers. **(b)** Training on noisy MQAR hurts recall circuits, making their solutions diverge to non-recall ones. **(c)** Relative sensitivity of recall margin increases as kernel degree decreases.

Competing supervision suppresses recall for recurrent mixers. We compare Online Hebbian, Mamba-2, GDN, KDA, and softmax attention while varying ρ (Figure 3(a)). Softmax attention retains 100.00% recall accuracy all the way up through $\rho = 0.98$. In contrast, for all recurrent mixers, recall accuracy degrades by $\rho = 0.6$ and reaches zero by $\rho = 0.98$. This suggests that while recurrent mixers can learn recall circuits when trained in isolation, they are less robust to competing objectives than softmax attention.

4.2 Recurrent recall circuits are brittle under perturbations

We next ask two questions: (i) how does the background objective reduce recall across architectures, and (ii) why does the same background objective hurt low-degree recall circuits more strongly?

To isolate these effects, we study a family of online Hebbians (Definition 2.2) with *exact polynomial kernels* of degree p. For a query q, key k, and key dimension d, let $s(q,k) = q^\top k/\sqrt{d}$ and define

$$K_p(q,k) = \left(1 + \frac{s(q,k)}{p}\right)^p, \qquad p \in \mathbb{N}. \tag{6}$$

Intuitively, $p = 1$ corresponds to linear attention, and increasing p makes retrieval increasingly selective, with $p \to \infty$ recovering the exponential kernel underlying softmax attention.

Gradient interference from the competing objective reduces recall margin. To investigate how the competing objective hurts recall, we study how perturbations in the model parameters affect a recall circuit's margins. For a correctly recalled query, let $m_i(\theta) > 0$ denote the query margin of Definition 2.1 at parameters θ, and let $g_B = \nabla_\theta L_B(\theta)$ be the background gradient. Intuitively, a gradient step from the background objective $\Delta\theta = -\eta g_B$ affects the query margin as:

$$\Delta m_i \approx -\eta \, \|g_B\|_2 \underbrace{\|\nabla_\theta m_i(\theta)\|_2}_{\text{parameter sensitivity}} \underbrace{\cos(\nabla_\theta m_i(\theta), g_B)}_{\text{gradient interference}}. \tag{7}$$

In other words, a background gradient step lowers the recall margin when it aligns with the margin's gradient (destructive *gradient interference*), and the magnitude of the change depends on the margin's *parameter sensitivity*.

We train polynomial kernel online Hebbians on noisy MQAR, sweeping the kernel degree $p \to 1$, and measure gradient interference and parameter sensitivity in learned recall circuits as we move from the softmax toward the linear regime. We identify two complementary effects:

- **The background objective creates increasingly destructive gradient interference.** We train online Hebbians on noiseless ($\rho = 0$) MQAR, then partway through the training trajectory,

reintroduce the background objective and continue training. Training with stronger background loss reduces recall accuracy in these continuations. As a control, we compare to introducing matched-magnitude isotropic noise in the model gradients: at $F = 8$, $d = 64$, and $p = 1$, recall accuracy reaches 48.40% with 95% background-loss weight, versus 99.20% under isotropic noise (Figure 3(b)). This suggests the background objective produces structured destructive interference that actively hinders the development of recall circuits.

- **Low-degree recall circuits have higher relative sensitivity at a fixed task load.** We next hold the number of facts F and state dimension d fixed and evaluate the relative sensitivity of the mean recall margin along noiseless MQAR training trajectories, while sweeping $p \in \{1, 2, 3\}$. At $F = 16$ and $d = 64$, linear attention $(p = 1)$ is $2.8\times$ more sensitive than $p = 2$ and $7.6\times$ more sensitive than $p = 3$ (Figure 3(c)). This means that a small parameter change induced by the background objective introduces a larger relative change in the linear attention model's recall margin than in the higher-degree models.

Together, these effects suggest that recurrent mixers struggle to learn recall circuits during pretraining because they are more brittle to small parameter perturbations caused by competing objectives.

4.3 Two Simple Interventions Improve Recall Learning

Motivated by Section 4.2, we propose two simple interventions to reduce the effect of competing supervision on recall learning: *recall-intensive loss* and *routing across recall- and non-recall states*.

- **Recall-intensive training.** High background loss weight reduces the effective fraction of recall data the model sees during training. We thus simply upweight the loss on recall positions by a factor λ. On noisy MQAR, at $F = 256$ and $\rho = 0.98$, $4\times$ recall upweighting retains at least 98% recall accuracy, whereas standard training falls to zero (Figure 4). However, we show that upweighting degrades non-recall perplexity, offering a tradeoff between recall- and non-recall performance (Figure 4).
- **Token-wise routing across multiple states.** Competing objectives force a single recurrent state to perform both in-context recall and other language modeling subtasks. We thus split the single state into two, allowing one state to specialize to recall and the other to background structure, while a router determines which memory each token uses. On noisy MQAR, an oracle router retains 100% recall and low non-recall perplexity through $\rho = 0.98$ (Figure 4).

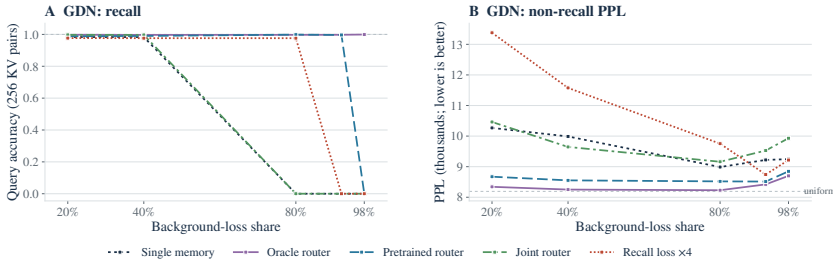


Figure 4: **Recall-intensive training and memory routing improve noisy-MQAR recall in GDN.** At $F = 256$ stored facts, we sweep the background-loss share from $\rho = 0.20$ to $\rho = 0.98$. **(a)** Query accuracy for a single memory, an oracle router, a pretrained router, a jointly trained router, and a single memory with $4\times$ recall-loss weight. **(b)** Perplexity on non-recall positions for the same five interventions; lower is better.

5 RAM: Routed Associative Memory

Section 4 identifies two techniques to improve recall learning in noisy MQAR: recall-intensive training and routing across multiple states. In this section, we extend these techniques to language modeling by introducing *Routed Associative Memory* (RAM). RAM can be applied to existing recurrent mixer architectures and consists of two components: (i) it splits a mixer's recurrent state

into multiple states accessed through learned routers, and (ii) strengthens recall supervision by upweighting rare associative-recall bigrams during training. Applied to Gated DeltaNet, we train a 1.4B-parameter RAM on 10B tokens and show that it closes 56.02% of the average-recall gap to softmax attention while adding only 7.38M parameters (0.51%), relative to Gated DeltaNet, and remaining within 0.30 percentage points of its LM-Evals score (Tables 1 and 9). RAM's recall performance scales better with state size than prior recurrent mixers (Figure 5). Finally, we show at 360M-parameter scale that giving the router access to simple n-gram features allows RAM to fully close the recall gap to self-attention.

5.1 Architecture and Loss Function

RAM applies the two interventions from Section 4.3 to any recurrent mixer: it routes tokens across multiple recurrent states and upweights recall-intensive training positions.

- **Routing across multiple recurrent memories.** Let LA_θ denote a base recurrent mixer with parameters θ. RAM maintains the recurrent state update logic of the base mixer, but splits its recurrent state into M separate recurrent states and combines their outputs using token-dependent write and read weights:

$$\left(u_t^{(m)}, S_t^{(m)}\right) = \mathrm{LA}_{\theta,\Delta\theta_m}\left(x_t, S_{t-1}^{(m)}; w_{t,m}\right), \qquad y_t = \sum_{m=1}^{M} r_{t,m} u_t^{(m)}. \tag{8}$$

  We parameterize the write and read routers as two-layer SiLU MLPs, which produce $w_{t,m}$ and $r_{t,m}$ using top-4 softmax writes and dense softplus reads, respectively. Conceptually, the routers identify which subtask of broader language modeling a token requires, while the selected recurrent memories execute the computation to solve that task.

  Recall-intensive training via rare bigram upweighting. We propose to use rare associative recall (AR) bigrams as a proxy for a recall subtask within broader LM pretraining. Following the procedure from Arora et al. (2024a), we extract tokens completing the first repetition within a sequence of a bigram that occurs fewer than 1,250 times in the first 10B Pile training tokens.[1] During training, we augment the standard next-token prediction loss with a *rare AR bigram upweighting loss*:

$$\mathcal{L}_{\mathrm{RAM}} = \frac{\sum_t \omega_t \ell_t}{\sum_t \omega_t}, \qquad \omega_t = \begin{cases} \lambda, & \text{if } t \text{ completes a rare AR bigram,} \\ 1, & \text{otherwise.} \end{cases} \tag{9}$$

5.2 Results

RAM narrows the recall gap to self-attention while preserving general language modeling capabilities. We pretrain 1.4B-parameter Attention, Gated DeltaNet, and RAM models on 10B tokens, sweeping both the state scaling and the recall loss upweighting. RAM closes 56.02% of the average-recall gap (the mean of FDA, SWDE, and SQuAD accuracy) to self-attention while adding only 0.51% more parameters (7.38M) than the Gated DeltaNet baseline (Table 1). To quantify general language modeling, we report *LM-Evals*, the unweighted mean accuracy across six zero-shot tasks: LAMBADA, HellaSwag, PIQA, Arc-E, Arc-C, and WinoGrande, following (Gao et al., 2024). Across all our RAM models, LM-Evals is at most 0.30 percentage points below the GDN baseline (Table 9).

The RAM loss and architecture are complementary. At the 360M-parameter scale, we sweep $M \in \{2, 4, 8\}$ and $\lambda \in \{1, 2, 4, 8, 16\}$ for 3B training tokens. Increasing λ from 4 to 16 raises Pile perplexity by as much as 2.71 and lowers LM-Evals by as much as 2.12 percentage points. At the default $\lambda = 4$, increasing the number of memories from two to eight doubles FDA from 12.89% to 25.95%, improves Pile perplexity from 10.36 to 10.23, and changes LM-Evals by -0.14 percentage points (panels (a,b) of Figure 5).

RAM's recall performance scales better with state size than existing recurrent mixers. At $S = M = 2, 4, 8$, our $4\times$ model outperforms every evaluated growing-state baseline by at least 1.00, 6.81, and 12.61 FDA-accuracy points, respectively (Figure 5(c)).

[1] Rare AR bigrams comprise only about 3% of all pretraining tokens.

Model	General LM		Recall					
	Pile PPL ↓	LM-Evals (%) ↑	Rare-AR PPL ↓	NIAH (avg) (%) ↑	FDA (%) ↑	SWDE (%) ↑	SQuAD (%) ↑	Avg. recall (%) ↑
Attention	7.05	45.76	2.32	58.65	71.23	72.01	37.47	60.24
GDN / RAM M1 (1×)	7.24	45.14	2.68	31.90	13.88	38.79	29.42	27.36
GDN / RAM M1 (4×)	7.48	44.96	1.91	34.18	30.13	41.76	36.80	36.23
RAM M2 (1×)	7.18	45.22	2.58	32.98	25.86	41.31	31.27	32.81
RAM M2 (4×)	7.47	45.05	1.90	31.05	32.03	40.77	34.72	35.84
RAM M4 (1×)	7.14	45.58	2.54	39.18	36.30	45.36	30.73	37.46
RAM M4 (4×)	7.38	45.34	1.83	48.03	44.10	48.33	41.09	44.51
RAM M8 (1×)	7.14	45.35	2.51	32.80	36.39	49.32	30.80	38.84
RAM M8 (4×)	7.36	44.84	1.81	44.55	49.73	50.14	37.47	45.78

Table 1: **Language modeling and recall at 1.4B parameters (10B tokens).** LM-Evals, NIAH (avg), and Avg. recall average six zero-shot tasks, four RULER needle-retrieval tasks, and FDA/SWDE/SQuAD, respectively.

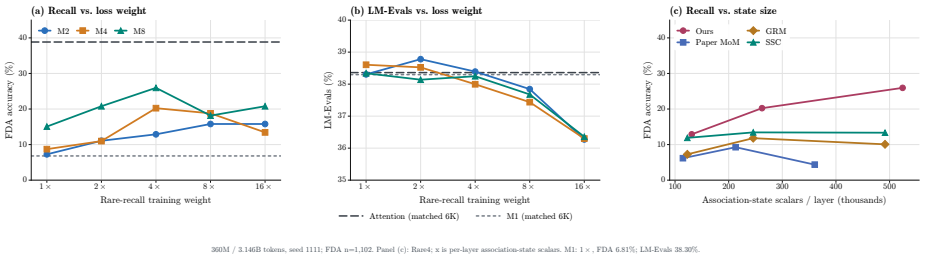


Figure 5: **Recall weighting, general language modeling, and resident state.** Panels (a,b) show RAM at 360M/3B-token sweeping $M = 2, 4, 8$ and rare-recall training weights $1, 2, 4, 8, 16$; panel (c) plots FDA against peak resident association-matrix scalars per layer at $S = M = 2, 4, 8$. For fixed recall upweighting (up through $4\times$), increasing the state size strictly improves recall performance.

n-gram routing closes the remaining recall gap to attention. The preceding models route using only local information from the current token representation. Towards closing the gap to attention, we give the RAM router simple n-gram statistics from the preceding sequence while retaining recurrent memories for task execution (Section A.1). At the 360M-parameter, 3B-token scale, the n-gram router closes 92.08–100.83% of the average-recall gap to softmax attention across $M \in \{1, 2, 4, 8\}$, fully closing it at $M = 2$ (34.08% vs. 33.91%), while matching general language modeling (LM-Evals at most 0.13 percentage points below attention) (Table 2).

Model	General LM		Recall					
	Pile PPL ↓	LM-Evals (%) ↑	Rare-AR PPL ↓	NIAH (avg) (%) ↑	FDA (%) ↑	SWDE (%) ↑	SQuAD (%) ↑	Avg. recall (%) ↑
Attention (1×)	9.97	38.36	3.07	42.05	38.84	39.78	23.12	33.91
GDN / RAM M1 (1×)	10.04	38.30	3.93	15.00	6.81	14.22	18.93	13.32
GDN / RAM M1 (4×)	10.43	38.44	2.56	23.35	10.89	17.10	27.82	18.60
+ n-gram router	10.21	38.47	2.44	23.00	41.20	32.76	26.51	33.49
RAM M2 (4×)	10.36	38.39	2.48	32.93	12.89	19.08	28.28	20.08
+ n-gram router	10.18	38.62	2.34	30.70	38.20	35.28	28.75	34.08
RAM M4 (4×)	10.27	38.00	2.37	43.50	20.24	25.56	28.75	24.85
+ n-gram router	10.16	38.23	2.32	26.95	34.21	34.56	28.08	32.28
RAM M8 (4×)	10.23	38.25	2.31	41.88	25.95	25.47	29.86	27.09
+ n-gram router	10.15	38.63	2.27	29.70	38.57	30.15	29.49	32.74

Table 2: **History-informed routing at 360M parameters (3B tokens).** n-gram-router rows use $4\times$ recall upweighting; metrics follow Table 1; the $1\times$ rows for $M > 1$ are in Table 4. Protocols are in Sections A.3 and A.4.

Reproducibility statement

The model architecture and training objective are described in Sections A.1 and 5, with prior-work baseline adaptations in Section A.2. The empirical appendix documents the experimental setup, evaluation protocols, and additional results (Section B); supporting derivations appear in Section C. We intend to release the code publicly upon acceptance.

AI use statement

Generative AI tools assisted with literature retrieval and discovery, research ideation and execution, and the development and checking of mathematical arguments and proofs. They also assisted with drafting, revising, and polishing parts of this paper. The authors reviewed AI-assisted research and writing, checked the resulting claims and artifacts, and take responsibility for the final content of this work.

Ethics statement

This work studies recall mechanisms in language models using existing training corpora and evaluation benchmarks. We are not aware of ethical issues specific to this study beyond those associated with language-model research generally. Existing corpora may contain biases or sensitive information, and stronger recall may increase the risk of memorizing such information. Our experiments do not assess deployment safety, privacy, or fairness; the reported performance should not be interpreted as evidence of safety in downstream uses.

References

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. In *International Conference on Learning Representations*, 2024a. URL https://arxiv.org/abs/2312.04927.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. In *International Conference on Machine Learning*, 2024b. URL https://arxiv.org/abs/2402.18668.

Ali Behrouz, Zeman Li, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Memory caching: RNNs with growing memory, 2026. URL https://arxiv.org/abs/2602.24281.

Tom B. Brown et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020. URL https://arxiv.org/abs/2005.14165.

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. URL https://arxiv.org/abs/2405.21060.

Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, and Yu Cheng. MoM: Linear sequence modeling with mixture-of-memories, 2025. URL https://arxiv.org/abs/2502.13685.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 2024. URL https://zenodo.org/records/12608602.

Roberto Garcia, Jerry Liu, Ronny Junkins, Sabri Eyuboglu, Atri Rudra, and Christopher Ré. MLPs are hebbians: Constructing efficient fact-storing MLPs for transformers, 2026. URL https://arxiv.org/abs/2607.10034.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2312.00752.

Han Guo, Songlin Yang, Tarushii Goel, Eric P. Xing, Tri Dao, and Yoon Kim. Log-linear attention. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2506.04761.

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://arxiv.org/abs/2404.06654.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. In *International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2402.01032.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, 2020. URL https://arxiv.org/abs/2006.16236.

Kimi Team. Kimi linear: An expressive, efficient attention architecture, 2025. URL https://arxiv.org/abs/2510.26692.

Kimi Team. Kimi K3: Open frontier intelligence. Moonshot AI blog, 2026. URL https://www.kimi.ai/blog/kimi-k3.

Yuqi Pan, Yongqi An, Zheng Li, Yuhong Chou, Ruijie Zhu, Xiaohui Wang, Mingxuan Wang, Jinqiao Wang, and Guoqi Li. Scaling linear attention with sparse state expansion, 2025. URL https://arxiv.org/abs/2507.16577.

Zhen Qin, Songlin Yang, Weigao Sun, Xuyang Shen, Dong Li, Weixuan Sun, and Yiran Zhong. HGRN2: Gated linear RNNs with state expansion. In *Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2404.07904.

Qwen Team. Qwen3-Next: Towards ultimate training & inference efficiency. Qwen blog, 2025. URL https://qwen.ai/blog?id=4074cca80393150c248e508aa62983f9cb7d27cd.

Qwen Team. Qwen3.5: Towards native multimodal agents. Qwen blog, 2026. URL https://qwen.ai/blog?id=qwen3.5.

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

Alexander Waitz, William Hu, Benjamin F. Spector, Atri Rudra, Christopher Ré, and Simran Arora. Cylon: Asynchronous linear attention. In *Conference on Language Modeling*, 2026. URL https://colmweb.org/AcceptedPapers.html.

Xin Wang, Hui Shen, Boyuan Zheng, Xueshen Liu, Minkyoung Cho, Zhongwei Wan, Zesen Zhao, Zhuoqing Mao, Shen Yan, and Mi Zhang. Dynamic linear attention. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2606.10650.

Kaiyue Wen, Xingyu Dang, and Kaifeng Lyu. RNNs are not transformers (yet): The key bottleneck on in-context retrieval, 2024. URL https://arxiv.org/abs/2402.18510.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2412.06464.

Michael Zhang, Simran Arora, Rahul Chalamala, Alan Wu, Benjamin Spector, Aaryan Singhal, Krithik Ramesh, and Christopher Ré. LoLCATs: On low-rank linearizing of large language models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.10254.

A Architecture, Baseline, and Evaluation Details

A.1 Architecture and implementation details

History-informed (n-gram) read router. The router of Section 5.2 changes only the read weights; the write router is unchanged. With write logits $\ell_t^w = W_2\,\mathrm{silu}(W_1 h_t + b_1) + b_2$, the read logits add a residual MLP that also sees a history feature vector g_t:

$$\ell_t^r = \ell_t^w + U_2\,\mathrm{silu}(U_1\,[h_t;\, g_t] + c_1) + c_2, \qquad r_{t,m} = \mathrm{softplus}\big(\ell_{t,m}^r\big)\,, \tag{10}$$

with $U_2 = c_2 = 0$ at initialization, so reads start tied to writes. g_t is a causal function of the token ids $x_{\le t}$ alone, computed once per forward pass and shared by all layers; each layer has its own U_1, and embedding lookups inside g_t are detached. At 360M ($d = 1024$), g_t has 3,382 dimensions:

- the token embedding of x_t (d);
- 12 token-level n-gram scalars: whether the 2/3/4/6/8/12-gram ending at t occurred earlier; log-counts of earlier 4- and 6-gram occurrences; whether the two latest earlier 4-gram occurrences were followed by the same token; and the log-length, log-distance, and existence of the longest repeated suffix match;
- 6 character-level scalars: whether the last 6/10/16/24 characters occurred earlier, the log-distance to that match, and whether it yields a candidate;
- the embeddings (d each) of the token- and character-level induction candidates, i.e. the token that followed the most recent earlier match (zero if none);
- the 18 scalars again at $t-1$ and $t-2$ (36);
- the state of a 256-unit GRU run over projected token embeddings.

The scalars are standardized with a mean and standard deviation measured on the first 32 training sequences and frozen thereafter.

A.2 Prior-work baseline reproduction details

We adapt three published growing-state methods to the common Gated DeltaNet backbone and training protocol. These are architectural reproductions for a matched comparison, rather than reproductions of the original papers' model scales or training runs. The $S = 2, 4, 8$ variants are trained for 3B tokens at the 360M-parameter scale with $4\times$ rare-recall upweighting. They share the evaluation protocol in Section A.3; the comparisons are in Tables 5 and 7.

Memory Caching GRM. Following Memory Caching (Behrouz et al., 2026), we split each 2,048-token sequence into S fixed chronological segments of length $2,048/S$. Each segment starts with a zero Gated DeltaNet association matrix, while shared projections and short-convolution history continue across segment boundaries. Completed states are frozen in value but remain connected to the training graph; the current segment contributes only its causal prefix. We represent each segment by the mean of its mixer inputs and score this summary against a bias-free learned projection of the current token using an unscaled dot product at temperature one. A softmax over all available segments produces scalar weights shared across heads; GRM reads all segment states densely. All states remain accelerator-resident. Autoregressive evaluations recompute the full prefix instead of using an incremental cache.

Memory Caching SSC. SSC uses the same segmentation, zero-initialized segment states, causal current state, and cross-boundary convolution history as GRM (Behrouz et al., 2026). Its segment summary is the mean of normalized effective Gated DeltaNet keys, concatenated across heads. This is a pooling choice allowed by the paper, rather than the literal key sum in its equation: with our initialization and key normalization, sum pooling produced initial route probabilities above 0.97 and stalled training. The same connector and unscaled scores rank all completed segments. SSC selects at most two completed past states, always includes the current online state, and softmax-normalizes over the selected set. We chose two past states because the source paper does not specify k. A dedicated Triton kernel reads only selected past matrices, although every state remains accelerator-resident and every past summary is scored. Training and evaluation use the same hard top-k membership; selected weights and state reads remain differentiable.

Mixture-of-Memories. Our MoM baseline uses routed Gated DeltaNet memories with hard top-2 selection and an always-active shared memory (Du et al., 2025). Selected routed memories receive full updates while unselected memories retain their states. Renormalized top-2 probabilities mix the selected readouts without scaling their update strength; the shared-memory output is added with coefficient one outside that mixture. Following the released implementation, the query, convolutions, decay timescales, output gate, normalization, and output projection are shared. Key, value, decay, and update-strength projections are memory-specific, including for the shared memory. Capacity factor one keeps every token. We isolate each sequence-memory convolutional stream and apply a padding-masked Switch balance loss: the loss is computed per layer from full router probabilities and hard selections, averaged over layers, then weighted by 10^{-3}. The MLP ratio is reduced to two for approximate active-parameter matching. The $S = 2, 4, 8$ variants therefore match recurrent-state budgets, while stored and active parameter counts can differ.

A.3 Language-model evaluation protocol

We evaluate the 360M models after 6,000 training steps (3B tokens) and the 1.4B models after 20,000 steps (10B tokens).

Pile perplexity. At both scales, Pile perplexity uses 1,000 validation sequences of length 2,048, comprising 2,048,000 tokens.

LM-Evals. Following the EleutherAI Language Model Evaluation Harness (Gao et al., 2024), LM-Evals is the unweighted mean of six zero-shot task accuracies, computed before rounding. The LAMBADA OpenAI and WinoGrande columns use raw accuracy; PIQA, HellaSwag, Arc-E, and Arc-C use length-normalized accuracy. These tasks use zero-shot evaluation, a 2,048-token maximum context, and batch size one. For LAMBADA, PIQA, HellaSwag, WinoGrande, Arc-E, and Arc-C, the respective sample counts are 5,153, 1,838, 10,042, 1,267, 2,376, and 1,172.

A.4 Recall evaluation protocol

Rare-AR perplexity. Rare-AR perplexity uses the same Pile validation sequences, which contain 66,394 rare first-recall events.

FDA, SWDE, and SQuAD. FDA, SWDE, and SQuAD use all 1,102, 1,111, and 2,984 examples, respectively. We score case-insensitive answer containment (not SQuAD EM/F1) with a 2,048-token context, batch size one, greedy decoding, at most 48 generated tokens, and a single-newline stop.

NIAH (RULER subset). We report the four needle-in-a-haystack tasks from RULER (Hsieh et al., 2024), rather than its full task suite: `niah_single_1`, `niah_multikey_1`, `niah_multivalue`, and `niah_multiquery`. NIAH (avg) is their unweighted mean accuracy, computed before rounding. Each task uses 250 examples at context length 2,048, batch size one, greedy decoding, and at most 64 generated tokens. An example scores the fraction of its needle values appearing case-insensitively in the generation.

B EMPIRICAL APPENDIX

B.1 MQAR STORAGE-CAPACITY SCALING FITS

For each architecture in Figure 2, we regress $\log(T \ln T)$ on $\log |M|$, where T is the storage capacity at $\geq 99\%$ accuracy ($T@99\%$) and $|M|$ is the recurrent state size in scalars. Theorem 3.2 predicts $|M| = \Theta(T \log T)$, i.e. slope $b = 1$. Fitting $\log T$ directly would bias the slope below 1 because of the log factor. Points with $T@99\% = 0$ carry no scaling information and are excluded; this removes Gated DeltaNet's three smallest state sizes. Confidence intervals are percentile intervals over 2,000 bootstrap resamples of the points. Each point is a single training seed.

Architecture	Points	b	95% CI
Online Hebbian (decay)	7	1.07	[0.90, 1.36]
Online Hebbian	7	0.88	[0.76, 1.03]
Gated DeltaNet	4	1.21	[1.07, 1.42]
Kimi Delta Attention	6	1.01	[0.77, 1.32]
Mamba-2	7	1.22	[0.73, 1.78]

Table 3: **Storage-capacity scaling exponents.** Fitted b in $T \ln T \propto |M|^b$ for each architecture in Figure 2; Theorem 3.2 predicts $b = 1$.

B.2 360M MEMORY-COUNT AND RARE-WEIGHT SWEEP

Architecture	Rare weight	Pile PPL ↓	Rare-AR PPL ↓	FDA ↑	SWDE ↑	SQuAD ↑	NIAH (avg) ↑
Attention	1×	9.97	3.07	38.84%	39.78%	23.12%	42.05%
LogLinear	1×	10.11	3.87	10.71%	15.39%	21.18%	35.05%
GDN / RAM M1	1×	10.04	3.93	6.81%	14.22%	18.93%	15.00%
GDN / RAM M1	4×	10.43	2.56	10.89%	17.10%	27.82%	23.35%
RAM M2	1×	9.96	3.69	7.26%	15.12%	20.38%	8.85%
RAM M2	2×	10.02	3.03	11.07%	19.26%	23.12%	16.28%
RAM M2	4×	10.36	2.48	12.89%	19.08%	28.28%	32.93%
RAM M2	8×	11.21	2.01	15.79%	18.00%	31.87%	40.10%
RAM M2	16×	13.07	1.72	15.79%	20.07%	34.28%	34.23%
RAM M4	1×	9.90	3.51	8.71%	17.10%	20.14%	31.90%
RAM M4	2×	9.97	2.93	10.98%	19.26%	23.73%	39.28%
RAM M4	4×	10.27	2.37	20.24%	25.56%	28.75%	43.50%
RAM M4	8×	11.16	1.95	18.78%	21.87%	30.16%	28.63%
RAM M4	16×	12.85	1.68	13.43%	18.27%	36.16%	34.15%
RAM M8	1×	9.84	3.37	15.06%	22.95%	20.84%	20.95%
RAM M8	2×	9.91	2.83	20.78%	24.48%	26.47%	27.35%
RAM M8	4×	10.23	2.31	25.95%	25.47%	29.86%	41.88%
RAM M8	8×	11.06	1.91	18.15%	24.03%	34.48%	35.55%
RAM M8	16×	12.71	1.65	20.78%	21.33%	35.69%	34.05%
RAM M1 + n-gram router	4×	10.21	2.44	41.20%	32.76%	26.51%	23.00%
RAM M2 + n-gram router	4×	10.18	2.34	38.20%	35.28%	28.75%	30.70%
RAM M4 + n-gram router	4×	10.16	2.32	34.21%	34.56%	28.08%	26.95%
RAM M8 + n-gram router	4×	10.15	2.27	38.57%	30.15%	29.49%	29.70%

Table 4: **Matched 360M-scale language-model and recall results.** All models are trained on 3B tokens; RAM rows vary memory count and rare-recall weight. Evaluation protocols are in Sections A.3 and A.4, with per-task NIAH results in Table 8.

Growing-state baseline comparison. Table 5 compares the matched-state prior-work baselines against the $4\times$ M2/M4/M8 cells from our sweep. Every row was trained for 6,000 steps, or 3B input tokens. Here S denotes the intended number of recurrent states. The comparison is state- and token-matched and approximately parameter-matched.

Table 6 reports LM-Evals on the same factorial checkpoints and matched M1, Attention, and LogLinear controls.

Architecture	Parameters	Pile PPL ↓	Rare-AR PPL ↓	FDA ↑	SWDE ↑	SQuAD ↑
RAM M2	389M	10.36	2.48	12.89%	19.08%	28.28%
RAM M4	389M	10.27	2.37	20.24%	25.56%	28.75%
RAM M8	389M	10.23	2.31	25.95%	25.47%	29.86%
Paper MoM S2	347M	10.81	2.74	6.17%	16.83%	25.54%
Paper MoM S4	365M	10.91	2.69	9.26%	20.16%	26.58%
Paper MoM S8	369M	11.40	2.83	4.36%	15.21%	23.79%
GRM S2	359M	12.19	3.23	7.26%	11.16%	21.05%
GRM S4	359M	12.86	3.40	11.80%	10.44%	23.59%
GRM S8	359M	13.36	3.39	10.07%	12.87%	21.18%
SSC S2	359M	10.71	2.59	11.89%	18.45%	24.73%
SSC S4	359M	10.80	2.63	13.43%	22.50%	28.45%
SSC S8	359M	10.89	2.69	13.34%	18.18%	29.16%

Table 5: **State-matched 360M-scale growing-state baselines.** All models are trained on 3B tokens with $4\times$ rare-recall upweighting. Evaluation protocols are in Sections A.3 and A.4.

Architecture	Rare weight	LAMBADA	PIQA	HellaSwag	WinoGrande	Arc-E	Arc-C	LM-Evals ↑
Attention	1×	30.27	60.01	29.02	50.75	36.70	23.38	38.36
LogLinear	1×	26.47	59.14	28.99	50.99	37.67	22.95	37.70
GDN / RAM M1	1×	27.48	61.26	29.48	51.07	37.37	23.12	38.30
GDN / RAM M1	4×	31.01	60.23	29.38	51.62	36.41	22.01	38.44
RAM M2	1×	29.65	61.92	28.94	49.72	37.50	22.10	38.30
RAM M2	2×	32.02	60.39	29.20	50.75	37.79	22.53	38.78
RAM M2	4×	32.45	60.28	28.44	51.78	36.24	21.16	38.39
RAM M2	8×	31.21	60.12	27.81	48.70	36.45	22.78	37.84
RAM M2	16×	29.36	58.98	26.63	48.38	33.38	20.90	36.27
RAM M4	1×	30.00	61.43	29.25	51.14	37.71	22.10	38.60
RAM M4	2×	31.44	60.66	29.30	49.72	37.75	22.27	38.52
RAM M4	4×	32.16	60.72	28.52	48.86	37.16	20.56	38.00
RAM M4	8×	30.22	60.39	28.09	48.78	35.98	21.16	37.44
RAM M4	16×	26.94	58.49	27.29	47.91	33.46	23.72	36.30
RAM M8	1×	28.57	61.37	29.37	50.67	38.05	22.01	38.34
RAM M8	2×	28.88	60.83	29.14	49.49	36.95	23.55	38.14
RAM M8	4×	30.27	60.66	28.77	49.49	37.08	23.21	38.25
RAM M8	8×	30.37	60.83	27.80	48.46	35.48	23.12	37.68
RAM M8	16×	28.74	58.76	27.04	47.83	33.92	21.84	36.36
RAM M1 + n-gram router	4×	32.39	59.25	28.97	50.67	37.25	22.27	38.47
RAM M2 + n-gram router	4×	32.06	60.83	29.00	50.83	36.66	22.35	38.62
RAM M4 + n-gram router	4×	31.61	60.12	28.57	49.96	37.71	21.42	38.23
RAM M8 + n-gram router	4×	32.18	60.34	28.73	51.85	37.33	21.33	38.63

Table 6: **LM-Evals at 360M parameters.** These zero-shot task accuracies underlie the LM-Evals column of Table 2. Task metrics and sample counts are in Section A.3.

The growing-state baseline LM-Evals use naive evaluation.

Architecture	LAMBADA	PIQA	HellaSwag	WinoGrande	Arc-E	Arc-C	LM-Evals ↑
Paper MoM S2	31.86	60.12	27.97	51.78	33.84	23.04	38.10
Paper MoM S4	31.32	59.85	28.38	49.01	35.73	21.16	37.58
Paper MoM S8	27.91	59.52	27.81	51.46	35.27	21.84	37.30
GRM S2	27.42	58.87	27.14	49.88	34.60	22.53	36.74
GRM S4	25.13	58.27	27.19	49.01	34.47	22.18	36.04
GRM S8	24.49	58.71	27.00	49.64	33.67	21.67	35.86
SSC S2	29.89	60.50	28.21	50.99	35.86	23.21	38.11
SSC S4	30.14	60.55	28.23	50.36	37.33	23.55	38.36
SSC S8	28.60	61.21	28.35	48.54	35.98	22.10	37.46

Table 7: **LM-Evals for state-matched 360M-scale growing-state baselines.** All checkpoints are trained on 3B tokens with $4\times$ rare-recall upweighting. Task metrics and sample counts are in Section A.3.

Architecture	Rare weight	Single	Multi-key	Multi-value	Multi-query	NIAH (avg) ↑
Attention	1×	98.00	27.60	21.30	21.30	42.05
LogLinear	1×	94.40	21.20	13.80	10.80	35.05
GDN / RAM M1	1×	39.20	12.80	2.10	5.90	15.00
GDN / RAM M1	4×	84.00	9.20	0.00	0.20	23.35
RAM M2	1×	23.60	8.40	0.30	3.10	8.85
RAM M2	2×	54.40	4.00	3.70	3.00	16.28
RAM M2	4×	100.00	18.80	2.00	10.90	32.93
RAM M2	8×	96.80	18.80	22.60	22.20	40.10
RAM M2	16×	100.00	19.20	16.70	1.00	34.23
RAM M4	1×	84.80	22.00	7.10	13.70	31.90
RAM M4	2×	99.60	21.20	16.30	20.00	39.28
RAM M4	4×	100.00	27.20	22.90	23.90	43.50
RAM M4	8×	98.80	14.80	0.20	0.70	28.63
RAM M4	16×	92.40	23.60	6.70	13.90	34.15
RAM M8	1×	48.80	18.00	6.60	10.40	20.95
RAM M8	2×	98.80	6.40	3.60	0.60	27.35
RAM M8	4×	100.00	24.40	19.80	23.30	41.88
RAM M8	8×	100.00	25.20	11.10	5.90	35.55
RAM M8	16×	66.00	31.60	15.90	22.70	34.05
RAM M1 + n-gram router	4×	83.60	6.00	0.10	2.30	23.00
RAM M2 + n-gram router	4×	98.80	19.60	0.00	4.40	30.70
RAM M4 + n-gram router	4×	87.60	18.80	0.40	1.00	26.95
RAM M8 + n-gram router	4×	99.20	14.00	0.40	5.20	29.70

Table 8: **Per-task NIAH accuracy at 360M parameters.** Results break down the four tasks averaged in Table 2. The scoring protocol is in Section A.4.

B.3 1.4B-SCALE LANGUAGE-MODEL COMPARISON

Table 1 reports the approximately 1.4B-parameter RRC models trained on 10B tokens. All reported rows use the common evaluation protocols of Sections A.3 and A.4.

The completed M1 $4\times$ row improves rare-first-recall perplexity and FDA over M1 $1\times$, while increasing aggregate Pile perplexity. M4 $1\times$ improves all three metrics over M1 $1\times$; Attention retains the highest FDA accuracy among these evaluated checkpoints.

Table 9 reports LM-Evals for these final checkpoints.

Architecture	Variant	LAMBADA	PIQA	HellaSwag	WinoGrande	Arc-E	Arc-C	LM-Evals ↑
Attention	$1\times$	48.17	67.14	38.82	50.75	44.44	25.26	45.76
GDN / RAM M1	$1\times$	45.78	66.27	38.57	51.14	43.64	25.43	45.14
GDN / RAM M1	$4\times$	45.86	66.32	36.71	52.33	44.23	24.32	44.96
RAM M2	$1\times$	46.46	66.00	38.36	49.72	46.13	24.66	45.22
RAM M2	$4\times$	46.94	66.05	37.90	50.59	44.32	24.49	45.05
RAM M4	$1\times$	46.52	66.92	38.84	51.38	45.16	24.66	45.58
RAM M4	$4\times$	48.88	66.92	36.94	51.38	44.44	23.46	45.34
RAM M8	$1\times$	45.31	66.43	38.70	51.70	44.44	25.51	45.35
RAM M8	$4\times$	46.44	65.94	37.47	52.01	43.48	23.72	44.84

Table 9: **LM-Evals at 1.4B parameters.** These zero-shot task accuracies underlie the LM-Evals column of Table 1. Task metrics and sample counts are in Section A.3.

The single-needle NIAH task is nearly saturated for the 1.4B models; the other three tasks better distinguish their retrieval accuracy.

Architecture	Variant	Single	Multi-key	Multi-value	Multi-query	Mean ↑
Attention	$1\times$	100.00	62.00	42.10	30.50	58.65
GDN / RAM M1	$1\times$	100.00	24.80	0.30	2.50	31.90
GDN / RAM M1	$4\times$	100.00	30.80	4.40	1.50	34.18
RAM M2	$1\times$	100.00	27.60	3.10	1.20	32.98
RAM M2	$4\times$	100.00	21.20	1.00	2.00	31.05
RAM M4	$1\times$	99.60	28.40	21.10	7.60	39.18
RAM M4	$4\times$	100.00	45.60	20.00	26.50	48.03
RAM M8	$1\times$	100.00	28.40	1.00	1.80	32.80
RAM M8	$4\times$	100.00	42.80	18.40	17.00	44.55

Table 10: **Per-task NIAH accuracy at 1.4B parameters.** Results break down the four tasks averaged in Table 1. The scoring protocol is in Section A.4.

B.4 FULL LM RECALL RESULTS

Figure 2(b) plots average recall accuracy against state size; the underlying per-task numbers for all thirteen arms are below.

Table 11: **Recurrent LMs recall far worse than self-attention, and extra state does not close the gap.** 120M models, 3B tokens on the Pile, evaluated at the trained context length of 2048. $|M|$ is the recurrent state size in scalars summed over all layers and heads. NIAH (avg) is the mean of four NIAH tasks. *Self-attention has no fixed state; $|M|$ is its KV cache averaged over the positions of a 2048-token sequence.

Model	Params	State-size $\|M\|$	$\times$	NIAH (avg)	FDA	SWDE	Avg Recall Acc	PPL
Online Hebbian MLP	125.0M	0.70M	1.0	3.00%	2.00%	12.00%	5.70%	12.75
Online Hebbian MLP	128.5M	2.47M	3.5	5.50%	3.20%	21.60%	10.10%	12.27
Online Hebbian MLP	161.6M	18.98M	27.1	8.70%	10.00%	20.00%	12.90%	12.11
Mamba-2	138.6M	0.59M	0.8	0.10%	2.40%	8.80%	3.70%	12.85
Mamba-2	140.4M	2.36M	3.4	4.00%	7.60%	14.80%	8.80%	12.09
Mamba-2	157.0M	18.87M	27.0	16.60%	10.40%	26.40%	17.80%	11.72
Gated DeltaNet	131.0M	0.70M	1.0	19.90%	3.60%	16.00%	13.20%	12.05
Gated DeltaNet	173.7M	2.69M	3.8	15.30%	9.60%	26.00%	17.00%	11.47
Gated DeltaNet	572.1M	21.27M	30.4	22.00%	12.40%	28.00%	20.80%	11.05
Kimi Delta Attn.	126.2M	0.70M	1.0	8.00%	1.60%	8.40%	6.00%	12.35
Kimi Delta Attn.	170.6M	2.69M	3.8	9.20%	3.60%	12.80%	8.50%	11.89
Kimi Delta Attn.	585.8M	21.27M	30.4	28.00%	7.20%	26.80%	20.70%	11.23
Self-attention	123.6M	18.88M*	27.0	28.90%	26.80%	40.40%	32.10%	11.79

Table 12: **Per-task NIAH scores behind the NIAH (avg) column of Table 11.** All at the trained context length of 2048, 250 examples per task, batch size 1, naive decoding – the same evaluation that produced Table 11. NIAH (avg) is the mean of the four NIAH columns. All accuracies are percentages. Longer NIAH lengths (4096, 8192) are omitted because every arm trains at `n_positions` 2048 and scores 0.00% beyond it.

Model	$\|M\|$	single_1	multikey_1	multivalue	multiquery	NIAH (avg)	squad_compl.
Online Hebbian MLP	0.70M	0.80%	8.80%	1.40%	0.90%	3.00%	16.80%
Online Hebbian MLP	2.47M	1.20%	10.40%	8.80%	1.50%	5.50%	25.60%
Online Hebbian MLP	18.98M	4.00%	16.00%	14.00%	0.70%	8.70%	22.40%
Mamba-2	0.59M	0.00%	0.00%	0.00%	0.20%	0.10%	12.80%
Mamba-2	2.36M	0.00%	6.80%	4.00%	5.00%	4.00%	20.40%
Mamba-2	18.87M	42.00%	17.20%	3.30%	4.00%	16.60%	26.40%
Gated DeltaNet	0.70M	75.20%	1.20%	2.90%	0.30%	19.90%	21.60%
Gated DeltaNet	2.69M	58.00%	2.40%	0.60%	0.30%	15.30%	23.20%
Gated DeltaNet	21.27M	58.00%	16.40%	3.30%	10.40%	22.00%	21.20%
Kimi Delta Attn.	0.70M	2.40%	13.60%	9.20%	6.60%	8.00%	20.80%
Kimi Delta Attn.	2.69M	9.20%	10.40%	10.30%	6.80%	9.20%	18.00%
Kimi Delta Attn.	21.27M	95.20%	9.60%	2.90%	4.40%	28.00%	29.20%
Self-attention	18.88M*	74.40%	22.80%	18.30%	0.30%	28.90%	18.00%

C Theory Appendix

C.1 Storage Capacity of the Online Hebbian MLP

Theorem 3.2 follows directly from the fact-storage capacity result of Garcia et al. (2026, Cor. 4.4; formally Cor. B.32), because after T writes the Online Hebbian MLP *is* their bilinear Hebbian MLP construction on the T in-context facts.

Theorem C.1 (Online Hebbian MLP storage capacity, formal)**.** *Let g be the Online Hebbian MLP of Definition 3.1 with state $M \in \mathbb{R}^{d\times m}$ and feature matrices $A, G \in \mathbb{R}^{m\times d}$ with i.i.d. Gaussian entries. Let an episode consist of T key–value pairs with*

$$k_1, \dots, k_T,\ v_1, \dots, v_T \overset{\text{i.i.d.}}{\sim} \mathrm{Unif}(\mathbb{S}^{d-1}), \qquad E = \{v_1, \dots, v_T\},$$

and fix $\delta \in (0,1)$. Under the conditions of Garcia et al. (2026, Cor. B.32) with $F = T$:

(i) ***Sufficiency.*** *There is an absolute constant $C > 0$ such that*

$$|M| = md \ \geq\ C\,T\log(T/\delta) \quad \Longrightarrow \quad \Pr_{\text{episode},\,A,\,G}\big[\,g \text{ stores all } T \text{ pairs}\,\big] \ \geq\ 1-\delta.$$

(ii) ***Necessity.*** *Any mixer that reads through a state of $|M|$ scalars at $O(1)$ bits each, and stores every fact set over T keys and T values, requires $|M| = \Omega(T\log T)$ (Garcia et al., 2026, Thm. 2.4).*

Hence, at fixed δ, the Online Hebbian MLP stores T pairs with a state of size

$$|M| = md = \Theta(T\log T),$$

the information-theoretically optimal rate.

Proof. Step 1: the online state is the offline readout. Unrolling the write rule of Definition 2.2 from $M_0 = 0$ gives, for any query q,

$$M_T = \sum_{i=1}^{T} v_i\,\phi(k_i)^\top = V^\top\Phi, \qquad g_T(q) = M_T\phi(q) = \sum_{i=1}^{T} v_i\,\langle\phi(k_i), \phi(q)\rangle,$$

where $V \in \mathbb{R}^{T\times d}$ and $\Phi \in \mathbb{R}^{T\times m}$ stack the values and featurized keys. This is exactly the bilinear Hebbian MLP $\mathrm{MLP}(q) = B\phi(q)$ with $B = V^\top\Phi$ of Garcia et al. (2026, Eq. (7)–(8)) on $F = T$ facts: our state M_T is their readout B, so $|M| = md = W$, and our margin $\gamma_{\mathcal{F}}(g_T)$ is their minimum margin $\gamma_{\min}$. Their $1/\sqrt{m}$ feature normalization is immaterial, since rescaling ϕ by $c > 0$ rescales every margin by c^2.

Step 2: sufficiency. Isotropic keys and values are their isotropic embeddings setting, so Garcia et al. (2026, Cor. B.32) gives $\gamma_{\min} > 0$, i.e. all T pairs stored, with probability at least $1-\delta$ over the episode and (A, G) whenever $md \geq C\,T\log(T/\delta)$.

Step 3: necessity. The read at the query is a function of the state alone, so the counting argument of Garcia et al. (2026, Thm. 2.4) applies with parameters replaced by state scalars: at most $2^{O(|M|)}$ states must distinguish T^T fact sets, giving $|M| = \Omega(T\log T)$. □