RECUE: REINSTATEMENT THROUGH QUERY-CUED REHEARSAL FOR FAST WEIGHT MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers retain past key–value pairs explicitly, enabling strong associative retrieval but requiring a growing KV cache. Fast weight models offer a more efficient alternative by compressing the history into a bounded state, but this compression can degrade individual key–value associations through interference. Rehearsing past associations may recover them, yet fast weight updates are non-local and can perturb other associations. We introduce RECUE: Reinstatement through Query-Cued Rehearsal, a training-free method that preserves associations along underrepresented key directions in a bounded cache and selectively rehearses them using the current query. RECUE jointly rehearses the cued associations through ridge regression, applies the resulting correction only to the current read, and immediately restores the original state. On pretrained LaCT and DeltaNet models, RECUE improves associative recall on RULER and improves average performance on LongBench over the base models and query-independent rehearsal. These results show that query-cued rehearsal can recover information lost through fast weight compression.[1]

1 INTRODUCTION

Transformer-based architectures have become the dominant paradigm for sequence modeling, largely due to the strong associative retrieval capability of softmax attention (Vaswani et al., 2017). However, autoregressive attention requires retaining an ever-growing KV cache, motivating efficient alternatives such as linear attention (Katharopoulos et al., 2020; Sun et al., 2023; Yang et al., 2024a;b; 2025; Zhang et al., 2025b; Hatamizadeh et al., 2026), state-space models (Gu et al., 2022; Gu & Dao, 2023; Dao & Gu, 2024; Lahoti et al., 2026), and Test-Time Training (TTT) (Sun et al., 2025; Zhang et al., 2025a; Feng et al., 2026), which compress the history into a bounded recurrent or fast weight state. While this compression enables efficient long-context generation, storing many associations in a shared bounded state introduces *associative interference*, which can degrade the fidelity of previously stored key–value associations.

Recent work mitigates this limitation by augmenting compressed recurrent states with bounded explicit KV memories, enabling direct access to selected past associations through sparse or bounded attention (McDermott et al., 2026; Cui, 2026). These approaches use the cache as a direct retrieval path. An alternative is to restore selected associations through the fast weight memory itself. Since fast weight memories are updated through associative write rules, such reinstatement can naturally be achieved by rehearsing past associations. However, rehearsal is inherently non-local: correcting one association can also interfere with other non-orthogonal reads. This raises a natural question:

> *Can past associations be selectively reinstated while limiting interference?*

We introduce **RECUE: Reinstatement through Query-Cued Rehearsal**. During prefill, RECUE preserves a bounded set of associations whose keys lie along underrepresented directions of the observed key distribution, identified using a regularized centered diagonal Mahalanobis distance. During decoding, the current query cues relevant cached associations. These cued associations are then jointly rehearsed through ridge regression to construct a temporary fast weight correction. The correction is used only for the current read and immediately rolled back. On pretrained LaCT (Zhang

[1] The code will be publicly released.

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.

et al., 2025a) and DeltaNet (Yang et al., 2024b) models, RECUE substantially improves associative recall on RULER (Hsieh et al., 2024), increasing LaCT's 8K single-needle average from 70.27 to 94.13. It also improves the LongBench (Bai et al., 2024) average from 15.03 to 15.71 on LaCT and from 14.21 to 14.86 on DeltaNet without additional training. Our main contributions are:

- We introduce RECUE, a query-cued reinstatement mechanism based on temporary ridge regularized rehearsal and rollback.
- We propose a Mahalanobis-based cache selection criterion that prioritizes underrepresented key directions and provide a ridge leverage interpretation of its geometry.
- We show that RECUE improves associative recall and long-context understanding on both linear and nonlinear fast weight models without additional training.

2 RELATED WORK

Hybrid Compressed and Explicit Memory. Recent work complements lossy recurrent compression with bounded explicit memory. LoLA distributes past associations across a recurrent linear-attention state, a local window, and a sparse KV cache, retaining difficult associations according to self-recall error (McDermott et al., 2026). LTE learns contextualized eviction scores to preserve critical KV pairs for sparse attention (He & Garner, 2025), while HOLA retains exact KV pairs with large delta-rule residuals as a complement to its recurrent state (Cui, 2026). These methods use explicit KV memory to restore a direct attention path to selected associations. RECUE instead uses cached associations to construct a temporary correction to the compressed memory.

Regression-Based Memory. Test-time regression provides a general view of associative memory as fitting a regressor over key–value associations and evaluating it at a query (Wang et al., 2025). MesaNet optimizes such an in-context regression objective toward its local optimum (von Oswald et al., 2026), while Local Linear Attention derives query-local associative prediction from local linear regression (Zuo et al., 2026). RECUE instead uses regression to jointly rehearse cached associations, fitting only the error left by the current fast weight memory to construct a query-conditioned temporary correction rather than reconstructing the associative predictor itself.

Scaling Recurrent Memory. Another line of work increases recurrent memory capacity beyond a single bounded state. Mixture-of-Memories routes tokens across multiple independent memory states (Du et al., 2026), while Log-Linear Attention maintains a logarithmically growing collection of recurrent states (Guo et al., 2026). Memory Caching stores historical recurrent-state checkpoints for later reuse (Behrouz et al., 2026), and MARCH extends this idea with content-routed state anchors (Zhang et al., 2026). Residual Linear Attention instead augments the base memory with an auxiliary recurrent state that accumulates residual prediction errors (Lai et al., 2025). These methods increase recurrent memory capacity, but individual key–value associations may still be lost through compression. RECUE instead preserves a bounded set of associations and selectively reinstates them through a query-conditioned temporary correction.

3 PRELIMINARIES

In this section, we first view self-attention as an explicit associative memory. We then discuss linear attention and Test-Time Training (TTT) as compressed alternatives.

3.1 SELF-ATTENTION AS EXPLICIT MEMORY

Self-attention. Self-attention (Vaswani et al., 2017) operates on an input sequence of N tokens $\mathbf{x} = [x_1, x_2, \ldots, x_N]$, where each token $x_t \in \mathbb{R}^d$. Each token is mapped to a query q_t, a key k_t, and a value v_t through learned projections, where $W_Q, W_K, W_V \in \mathbb{R}^{d\times d}$:

$$q_t = W_Q x_t, \qquad k_t = W_K x_t, \qquad v_t = W_V x_t, \tag{1}$$

$$o_t = \sum_{i=1}^{t} \frac{\exp(q_t^\top k_i/\sqrt{d})}{\sum_{j=1}^{t} \exp(q_t^\top k_j/\sqrt{d})} v_i. \tag{2}$$

Since all key–value pairs are retained explicitly, self-attention enables direct content-based retrieval from past tokens. However, its memory usage grows linearly with the sequence length, motivating compressed-memory alternatives such as linear attention.

3.2 Linear Attention as Compressed Memory

Linear attention. Linear attention (Katharopoulos et al., 2020) reformulates self-attention using a feature map $\phi(\cdot)$ that factorizes the pairwise similarity between queries and keys. Specifically, it replaces the exponential kernel $\exp\left(q_t^\top k_i/\sqrt{d}\right)$ with the dot product $\phi(q_t)^\top \phi(k_i)$. The self-attention output in Eq. 2 can be written as

$$o_t = \frac{\sum_{i=1}^{t} \phi(q_t)^\top \phi(k_i) v_i}{\sum_{j=1}^{t} \phi(q_t)^\top \phi(k_j)} = \frac{\phi(q_t)^\top \left(\sum_{i=1}^{t} \phi(k_i) v_i^\top\right)}{\phi(q_t)^\top \left(\sum_{j=1}^{t} \phi(k_j)\right)} = \frac{\phi(q_t)^\top S_t}{\phi(q_t)^\top z_t}. \tag{3}$$

Thus, instead of explicitly storing all previous key–value pairs, linear attention maintains a compressed state $S_t = S_{t-1} + \phi(k_t)v_t^\top$, which is often referred to as a fast weight (Hinton & Plaut, 1987; Ba et al., 2016; Schlag et al., 2021). A common choice for $\phi(\cdot)$ is the identity feature map, $\phi(x) = x$ (Sun et al., 2023), and the denominator $q_t^T z_t$ is often omitted (Mao, 2022). In this case, the fast weight state and output take the simple form

$$S_t = S_{t-1} + k_t v_t^\top, \qquad o_t = q_t^\top S_t. \tag{4}$$

3.3 Associative Interference and DeltaNet

While linear attention avoids explicitly storing past key–value pairs, this compression comes at the cost of losing individual associations. Unrolling the fast weight gives $S_t = \sum_{i=1}^{t} k_i v_i^\top$. Suppose we query the state using a previously observed key k_j. For normalized keys, the retrieved value $\hat{v}_j = S_t^\top k_j$ becomes

$$\hat{v}_j = \sum_{i=1}^{t} (k_i^\top k_j) v_i = \underbrace{v_j}_{\text{desired association}} + \underbrace{\sum_{i \neq j} (k_i^\top k_j) v_i}_{\text{interference from other associations}}. \tag{5}$$

Eq. 5 shows that the additive update does not preserve each key–value association independently. Unless the stored keys are mutually orthogonal, querying with k_j also retrieves components of the values associated with other keys. As more associations are stored, interference becomes unavoidable due to the limited number of orthogonal directions.

DeltaNet. DeltaNet (Schlag et al., 2021; Yang et al., 2024b) mitigates this interference by replacing the purely additive write with a corrective update. Given a new key–value pair (k_t, v_t), it first reads the value currently associated with k_t from the fast weight, $\hat{v}_t = S_{t-1}^\top k_t$, and updates the state using the prediction error,

$$S_t = S_{t-1} + \beta_t k_t \left(v_t - \hat{v}_t\right)^\top, \qquad 0 \leq \beta_t \leq 1, \tag{6}$$

where β_t controls the update strength. The update in Eq. 6, commonly referred to as the delta rule (Widrow & Hoff, 1960), corrects the memory using the prediction error $v_t - \hat{v}_t$ rather than simply adding $k_t v_t^\top$. When $\|k_t\|_2 = 1, \beta_t = 1$, it exactly corrects the association for the current key, $S_t^\top k_t = S_{t-1}^\top k_t + (v_t - \hat{v}_t) = v_t$.

Rehearsal interference. Importantly, this correction is not local to the association being rehearsed. For any other key k_j, the same update changes its readout according to

$$S_t^\top k_j - S_{t-1}^\top k_j = \beta_t (k_t^\top k_j)(v_t - \hat{v}_t). \tag{7}$$

Hence, rehearsing association t perturbs every non-orthogonal read in proportion to $k_t^\top k_j$ and $v_t - \hat{v}_t$. We call this non-local effect rehearsal interference.

Optimization view. The delta rule can also be viewed as a gradient descent step on an instantaneous key–value regression objective:

$$\mathcal{L}_t(S) = \frac{1}{2}\left\|S^\top k_t - v_t\right\|^2, \qquad \nabla_S \mathcal{L}_t(S) = k_t\left(S^\top k_t - v_t\right)^\top, \tag{8}$$

$$S_t = S_{t-1} - \beta_t \nabla_S \mathcal{L}_t(S)|_{S=S_{t-1}} = S_{t-1} + \beta_t k_t \left(v_t - S_{t-1}^\top k_t\right)^\top. \tag{9}$$

This exactly recovers Eq. 6, showing that DeltaNet can be viewed as online gradient descent on a key–value regression objective, with the fast weight S_t.

3.4 Test-Time Training

Test-Time Training. Test-Time Training (TTT) (Sun et al., 2025) generalizes this optimization-based perspective by parameterizing the fast memory as the weights $\mathbf{W}$ of a small learnable model $f_{\mathbf{W}}$. The fast weights are initialized to a learned state $\mathbf{W}_0$ at the beginning of each sequence and updated online as the sequence is processed, while all other model parameters remain fixed during inference. TTT writes information into the fast weights through an update rule and retrieves it through an output rule. The update rule optimizes a self-supervised objective $\mathcal{L}$, for which a common choice is a key–value association objective. Given a learning rate η_t, the fast weights are updated by gradient descent and subsequently queried with q_t:

$$\mathbf{W}_t = \mathbf{W}_{t-1} - \eta_t \nabla_{\mathbf{W}} \mathcal{L}_t\left(f_{\mathbf{W}}(k_t), v_t\right)|_{\mathbf{W}=\mathbf{W}_{t-1}}, \qquad \text{(update rule)}$$

$$z_t = f_{\mathbf{W}_t}(q_t). \qquad \text{(output rule)}$$

Recent work has further generalized TTT beyond simple gradient descent, for example using momentum (Behrouz et al., 2024) or more sophisticated optimizers such as Muon (Jordan et al., 2024; Behrouz et al., 2025). Large-Chunk TTT (LaCT) (Zhang et al., 2025a) further scales these updates to much larger chunks and combines them with sliding-window attention.

4 ReCue: Query-Cued Reinstatement

ReCue is an inference-time augmentation for fast weight models that selectively reinstates past associations. As shown in Eq. 7, rehearsal is inherently non-local: correcting one association can also affect other non-orthogonal reads. ReCue addresses this issue by using the current query as a retrieval cue to determine which associations in a bounded key–value cache should contribute to the current read. The cued associations are jointly rehearsed to construct a query-dependent correction $\Delta S(q)$. As illustrated in Figure 1, ReCue applies the resulting correction as a temporary patch to the current fast weight state and rolls it back after the read, localizing its effect to the current query:

$$S_{\text{tmp}} = S + \eta \Delta S(q), \qquad o^{\text{ReCue}}(q) = q^\top S_{\text{tmp}}, \qquad S_{\text{tmp}} \xrightarrow{\text{rollback}} S. \tag{10}$$

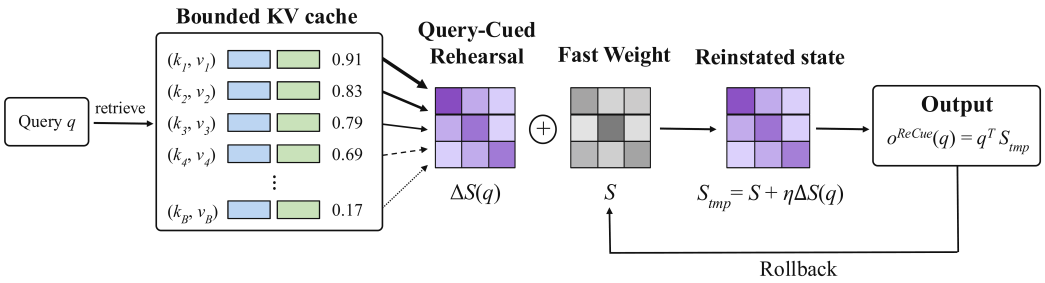


Figure 1: **Overview of ReCue.** The current query cues associations from a bounded KV cache to construct a temporary fast weight correction $\Delta S(q)$.

4.1 CACHE CONSTRUCTION

RECUE maintains a bounded cache $\mathcal{C} = \{(k_i, v_i)\}_{i=1}^{B}$ of associations selected during prefill. In hybrid models with sliding-window attention, $\mathcal{C}$ is maintained separately from the window and is used only as a rehearsal set. Given a fixed cache budget, we prioritize keys along underrepresented directions of the observed key distribution, which receive less geometric support from other keys. We measure this using a regularized centered diagonal Mahalanobis distance.

Let $\mu \in \mathbb{R}^{d_k}$ and $\sigma^2 \in \mathbb{R}^{d_k}$ denote the mean and variance of the prefill keys. We compute the Mahalanobis score and retain the B associations with the largest scores:

$$\bar{\sigma}^2 = \frac{1}{d_k}\sum_{j=1}^{d_k}\sigma_j^2, \qquad s_i = \frac{1}{d_k}\sum_{j=1}^{d_k}\frac{(k_{ij}-\mu_j)^2}{\sigma_j^2 + \lambda_{\text{sel}}\bar{\sigma}^2}, \tag{11}$$

$$\mathcal{I} = \text{TopB}_{i\in\{1,\dots,N\}}\, s_i, \qquad \mathcal{C} = \{(k_i, v_i) \mid i \in \mathcal{I}\}\,. \tag{12}$$

A large s_i indicates an underrepresented key direction. We provide a geometric interpretation of this criterion and its connection to ridge geometry in Appendix A.

4.2 QUERY-CUED REHEARSAL

Given the bounded cache $\mathcal{C} = \{(k_i, v_i)\}_{i=1}^{B}$, we now describe how RECUE constructs the temporary correction $\Delta S(q)$. For the current fast weight state S, we stack the cached keys and values and compute their reconstruction residuals:

$$K = \begin{bmatrix} k_1^\top \\ \vdots \\ k_B^\top \end{bmatrix}, \qquad V = \begin{bmatrix} v_1^\top \\ \vdots \\ v_B^\top \end{bmatrix}, \qquad R = V - KS. \tag{13}$$

For a single cached association, the delta rule reinstates its reconstruction residual $r_i = v_i - S^\top k_i$. Applying the corresponding corrections independently across the cache gives

$$\Delta S_i = k_i r_i^\top, \qquad \sum_{i=1}^{B}\Delta S_i = \sum_{i=1}^{B} k_i r_i^\top = K^\top R. \tag{14}$$

This independent accumulation can cause correlated keys to reinforce shared directions too strongly.

Ridge Rehearsal. Rather than independently accumulating residual writes, we jointly fit the cached reconstruction residuals with a regularized least-squares objective, motivated by the test-time regression view of associative memory (Wang et al., 2025; von Oswald et al., 2026; Ouyang et al., 2026). Unlike methods that fit the associative predictor itself, Ridge Rehearsal fits only the residual left by the current fast weight state:

$$\Delta S_{\text{RR}} = \arg\min_{\Delta S} \|K\Delta S - R\|_F^2 + \lambda\|\Delta S\|_F^2 \tag{15}$$

$$= K^\top\left(KK^\top + \lambda I_B\right)^{-1} R. \tag{16}$$

The inverse Gram matrix accounts for correlations among cached key directions. Ridge Rehearsal therefore jointly corrects the cached associations, but remains query-independent.

Top-K RECUE. We make rehearsal query-dependent by restricting it to the cached associations most relevant to the current query. We compute the cosine similarity between the query and each cached key and select the K most relevant associations. Let $K_q = [k_i^\top]_{i\in\mathcal{I}_q}$ and $R_q = [r_i^\top]_{i\in\mathcal{I}_q}$ denote the selected key and residual matrices, respectively. RECUE then jointly rehearses the selected associations using ridge regression:

$$c_i(q) = \cos(q, k_i), \qquad \mathcal{I}_q = \text{TopK}_{i\in\{1,\dots,B\}}\, c_i(q), \tag{17}$$

$$\Delta S_{\text{TopK}}(q) = \arg\min_{\Delta S} \|K_q\Delta S - R_q\|_F^2 + \lambda\|\Delta S\|_F^2 \tag{18}$$

$$= K_q^\top\left(K_qK_q^\top + \lambda I_K\right)^{-1} R_q. \tag{19}$$

Table 1: Comparison of rehearsal rules, with the Delta Rule as a single-association reference.

Method	**Objective**	**Rehearsal Correction** $\Delta S(q)$
Delta Rule	$\frac{1}{2}\|k_i^\top \Delta S - r_i^\top\|_2^2$	$k_i r_i^\top$
Ridge Rehearsal	$\|K\Delta S - R\|_F^2 + \lambda\|\Delta S\|_F^2$	$K^\top(KK^\top + \lambda I_B)^{-1}R$
Top-K RECUE	$\|K_q\Delta S - R_q\|_F^2 + \lambda\|\Delta S\|_F^2$	$K_q^\top(K_qK_q^\top + \lambda I_K)^{-1}R_q$
Full RECUE	$\|K\Delta S - D(q)R\|_F^2 + \lambda\|\Delta S\|_F^2$	$K^\top(KK^\top + \lambda I_B)^{-1}D(q)R$

Thus, the query determines which associations are rehearsed, while the ridge solve jointly accounts for correlations within the selected rehearsal set.

Full RECUE. Top-K RECUE makes rehearsal query-dependent by selecting a local subset of cached associations. As a dense alternative, Full RECUE retains the entire cache and modulates each cached residual according to its relevance to the current query. Let $c_i(q) = \cos(q, k_i)$ and $c_{\max}(q) = \max_j c_j(q)$. We use an exponential cue centered at the strongest query–key match, with temperature $\tau > 0$ controlling the sharpness of the cue.

$$\rho_i(q) = \exp\left(\frac{c_i(q) - c_{\max}(q)}{\tau}\right), \qquad D(q) = \operatorname{diag}\left(\rho_1(q), \ldots, \rho_B(q)\right). \tag{20}$$

$$\Delta S_{\text{Full}}(q) = \arg\min_{\Delta S} \|K\Delta S - D(q)R\|_F^2 + \lambda\|\Delta S\|_F^2 \tag{21}$$

$$= K^\top\left(KK^\top + \lambda I_B\right)^{-1} D(q)R. \tag{22}$$

The strongest query–key match receives unit weight, while the contribution of other associations decays exponentially with their similarity gap from the strongest cue. Lower temperatures yield sharper cueing. Unlike Top-K RECUE, Full RECUE keeps the regression geometry of the full cache fixed and makes only the residual targets query-dependent. Consequently, the inverse Gram matrix is independent of the query and can be precomputed once for a fixed cache. Table 1 summarizes the four rehearsal rules.

Rehearsal Strength. The amount of reinstatement should depend on how confidently the current query identifies a relevant cached association. We use the margin between the two largest query–key cosine similarities as a training-free gate. Let $c_{(1)}(q)$ and $c_{(2)}(q)$ denote the largest and second-largest values among $\{c_i(q)\}_{i=1}^B$, respectively. We define

$$\eta(q) = c_{(1)}(q) - c_{(2)}(q). \tag{23}$$

A large margin indicates a distinctive retrieval cue and therefore permits stronger reinstatement, whereas a small margin indicates an ambiguous cue and suppresses the correction. This gate introduces no learned parameters or additional training. Given the rehearsal correction $\Delta S(q)$, RECUE temporarily modifies the fast weight state as $S_{\text{tmp}} = S + \eta(q)\Delta S(q)$. The current query is read from S_{tmp}, after which the original state S is restored.

4.3 NONLINEAR FAST WEIGHT MODELS

RECUE also applies to nonlinear fast weight memories whose output is linear in an intermediate representation. We consider a gated MLP fast weight memory, as used in LaCT (Zhang et al., 2025a), with nonlinear feature extractor parameterized by $\theta = \{W_{\text{gate}}, W_{\text{up}}\}$ and linear output projection W_{out}:

$$\phi_\theta(x) = \sigma(W_{\text{gate}}x) \odot (W_{\text{up}}x), \qquad f_{\mathbf{W}}(x) = W_{\text{out}}\phi_\theta(x). \tag{24}$$

During reinstatement, RECUE keeps the nonlinear feature extractor fixed and temporarily patches only W_{out}. For a cached association and the current query, we define $x_i = \phi_\theta(k_i)$ and $x_q = \phi_\theta(q)$, with reconstruction residual $r_i = v_i - W_{\text{out}}x_i$. The rehearsal rules in Section 4.2 apply directly to the nonlinear feature representations. RECUE then temporarily updates the linear readout as

$$W_{\text{out}}^{\text{tmp}} = W_{\text{out}} + \eta\,\Delta W_{\text{out}}(q), \qquad o^{\text{RECUE}}(q) = W_{\text{out}}^{\text{tmp}}x_q. \tag{25}$$

Table 2: RULER benchmark results on Needle-in-a-Haystack (NIAH) tasks. †Transformer results are taken from LaCT (Zhang et al., 2025a).

	S-NIAH-1			S-NIAH-2			S-NIAH-3			Average		
Model	4K	8K	16K	4K	8K	16K	4K	8K	16K	4K	8K	16K
760M params / 40B tokens												
Transformer†	99.2	96.6	85.2	100.0	100.0	85.8	81.0	73.8	74.8	93.40	90.10	81.90
LaCT	99.0	93.0	97.2	98.0	83.8	37.6	55.2	34.0	8.2	84.07	70.27	47.67
+ Ridge Rehearsal	99.4	95.6	97.6	99.0	88.6	42.6	59.8	39.2	9.0	86.07	74.47	49.73
+ Attention Read	**100.0**	96.4	**98.4**	99.6	95.0	42.2	67.6	63.0	11.0	89.07	84.80	50.53
+ Top-16 RECUE	**100.0**	97.2	96.6	99.6	**99.8**	**65.2**	69.8	84.6	**23.6**	89.80	93.87	**61.80**
+ Full RECUE	**100.0**	**97.4**	97.8	**99.8**	**99.8**	58.4	**70.8**	**85.2**	19.2	**90.20**	**94.13**	58.47
1.3B params / 100B tokens												
DeltaNet	100.0	100.0	100.0	76.2	19.8	2.0	54.2	8.6	0.8	76.80	42.80	34.27
+ Ridge Rehearsal	100.0	100.0	100.0	82.6	19.8	1.6	55.6	10.2	0.2	79.40	43.33	33.93
+ Attention Read	100.0	100.0	100.0	89.0	28.8	2.0	**69.2**	12.8	1.6	86.07	47.20	34.53
+ Top-16 RECUE	100.0	100.0	100.0	89.2	**36.8**	**2.4**	61.4	**14.0**	1.4	83.53	**50.27**	34.60
+ Full RECUE	100.0	100.0	100.0	**91.4**	28.6	**2.4**	67.6	13.4	**1.8**	**86.33**	47.33	**34.73**

	MK-NIAH-1			MQ-NIAH			MV-NIAH			Average		
Model	4K	8K	16K	4K	8K	16K	4K	8K	16K	4K	8K	16K
760M params / 40B tokens												
Transformer†	63.8	72.0	71.4	33.40	28.90	24.00	27.95	24.00	20.50	41.70	41.60	38.60
LaCT	76.4	48.6	30.8	46.20	29.65	20.80	34.30	27.85	24.50	52.30	35.37	25.37
+ Ridge Rehearsal	77.0	49.6	30.8	46.45	30.10	19.65	35.15	28.60	23.45	52.87	36.10	24.63
+ Attention Read	79.0	61.4	39.6	**48.10**	34.85	27.20	37.90	34.05	30.15	55.00	43.43	32.32
+ Top-16 RECUE	81.6	69.8	44.0	44.10	**38.25**	29.40	35.80	**37.50**	33.00	53.83	48.52	35.47
+ Full RECUE	**84.2**	**72.4**	**44.2**	47.00	37.60	**29.50**	**39.25**	36.90	**34.75**	**56.82**	**48.97**	**36.15**
1.3B params / 100B tokens												
DeltaNet	25.4	11.6	**2.4**	**27.15**	12.45	1.70	18.15	12.90	4.25	23.57	12.32	2.78
+ Ridge Rehearsal	24.4	10.6	2.0	24.00	14.50	1.45	15.80	13.80	4.55	21.40	12.97	2.67
+ Attention Read	25.2	18.6	1.8	25.20	19.35	2.70	20.55	**22.15**	6.80	23.65	20.03	3.77
+ Top-16 RECUE	**26.2**	**20.6**	**2.4**	25.45	**21.95**	**2.95**	21.30	22.05	**9.50**	24.32	**21.53**	**4.95**
+ Full RECUE	25.6	19.2	2.2	26.95	19.70	2.60	**22.30**	21.55	8.05	**24.95**	20.15	4.28

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Backbones and Checkpoints. We evaluate RECUE on two pretrained fast weight backbones, LaCT (Zhang et al., 2025a) and DeltaNet (Yang et al., 2024b), using official checkpoints without additional training. For LaCT, we use the 760M parameter Muon NoPE PostNorm model, trained on 40B tokens with a 2K token chunk size and a 2K token sliding-attention window. For DeltaNet, we use the 1.3B parameter model trained on 100B tokens.

Configuration. To compare with direct retrieval (McDermott et al., 2026; Cui, 2026), we include Attention Read, which retrieves cached values using attention and adds the retrieved value to the fast weight readout. Full RECUE and Attention Read use temperature $\tau = 0.1$, with a temperature ablation for Full RECUE provided in Appendix B. All methods except Ridge Rehearsal use the margin gate in Eq. 23 to determine the correction strength for each query. Ridge Rehearsal instead uses fixed strengths of $\eta = 0.0045$ for LaCT and $\eta = 0.35$ for DeltaNet, chosen to produce correction magnitudes comparable to Full RECUE. For stability, we clip the norm of the rehearsal correction to at most 25% of the corresponding base output norm. Cache entries are ranked using the centered diagonal Mahalanobis score. We maintain separate caches of capacity B for each layer and head, with $B = 2$K for LaCT and $B = 512$ for DeltaNet. The cache is updated every 2K token chunk for LaCT and every 512 tokens for DeltaNet. At each update, new and cached associations are jointly re-ranked, retaining the top B entries. Detailed implementation and pseudocode for cache updates and RECUE are provided in Appendix E.

Table 3: Performance on LongBench tasks. NQA: NarrativeQA; QQA: Qasper; MFQA: MultiFieldQA; HQA: HotpotQA; 2Wiki: 2WikiMQA; GovRep: GovReport; MNews: MultiNews.

	Single-Doc QA			Multi-Doc QA			Summarization		
Model	NQA	QQA	MFQA	HQA	2Wiki	MuSiQue	GovRep	QMSum	MNews
LaCT	2.36	5.00	13.85	5.98	6.71	**3.80**	11.93	15.18	5.29
+ Ridge Rehearsal	2.74	4.88	14.21	6.32	6.37	3.62	12.38	14.57	5.43
+ Attention Read	2.65	5.15	14.00	5.47	6.54	3.57	12.86	**15.28**	5.65
+ Top-16 RECUE	**2.86**	5.32	13.63	6.17	7.06	3.75	**14.64**	14.30	5.83
+ Full RECUE	2.81	**5.51**	**14.57**	**6.63**	**7.29**	3.70	14.06	14.68	**6.34**
DeltaNet	3.26	3.57	12.93	4.78	8.05	2.01	5.39	11.34	11.10
+ Ridge Rehearsal	3.26	3.46	12.62	4.35	7.03	2.06	5.21	11.23	10.53
+ Attention Read	**4.55**	3.99	14.54	4.46	7.09	2.24	6.07	11.10	15.12
+ Top-16 RECUE	4.01	**4.49**	13.90	3.65	6.17	**2.36**	**7.33**	11.72	**17.88**
+ Full RECUE	3.05	4.04	**14.84**	**5.05**	**8.08**	2.34	6.48	**11.81**	17.07

	Few-shot Learning			Code		Overall
Model	TREC	TriviaQA	SAMSum	LCC	RepoBench-P	Avg.
LaCT	13.00	30.98	27.82	**32.95**	35.52	15.03
+ Ridge Rehearsal	**13.50**	31.95	29.07	32.29	35.64	15.21
+ Attention Read	13.00	30.69	28.93	32.87	**36.18**	15.20
+ Top-16 RECUE	12.50	30.43	29.24	31.61	35.36	15.19
+ Full RECUE	**13.50**	**32.71**	**29.50**	32.47	36.16	**15.71**
DeltaNet	25.50	26.25	**9.09**	41.56	34.05	14.21
+ Ridge Rehearsal	23.50	23.60	6.22	40.67	34.22	13.43
+ Attention Read	24.50	25.56	9.04	**41.70**	**35.17**	14.65
+ Top-16 RECUE	**26.00**	25.82	6.60	38.03	33.42	14.38
+ Full RECUE	25.00	**27.45**	8.63	40.04	34.22	**14.86**

5.2 NEEDLE-IN-A-HAYSTACK TASKS

We evaluate associative recall on Needle-in-a-Haystack tasks from RULER (Hsieh et al., 2024). S-NIAH-1 retrieves a word–number association from a noise context, S-NIAH-2 places the same association in an essay context, and S-NIAH-3 replaces the value with a longer UUID. We further evaluate the multi-association variants MK-NIAH-1, MQ-NIAH, and MV-NIAH, which introduce multiple keys, queries, and values, respectively. Results for all six tasks are reported in Table 2. Across both LaCT and DeltaNet, RECUE generally outperforms both Ridge Rehearsal and direct Attention Read, with larger gains at longer contexts and on multi-association tasks. Top-16 and Full RECUE show complementary strengths across tasks and context lengths.

5.3 LONG CONTEXT UNDERSTANDING TASKS

We use LongBench (Bai et al., 2024) to evaluate whether the retrieval gains of RECUE extend beyond synthetic recall to realistic long-context understanding tasks. On LongBench, Full RECUE achieves the best overall average for both LaCT and DeltaNet, while Top-16 RECUE shows more task-dependent behavior. This suggests that dense query-conditioned rehearsal transfers more robustly to heterogeneous long-context tasks than hard local selection. Further evaluation of general language modeling performance is provided in Appendix C.

5.4 ABLATION STUDIES

Cache selection. We compare our Mahalanobis-based cache policy against random selection and Self-Recall Error (SRE) (McDermott et al., 2026), where $s_i^{\mathrm{SRE}} = \|v_i - \hat{v}_i\|_2^2$. We evaluate LaCT with Full RECUE on S-NIAH-3 16K following Section 5.1. In Table 4, we report accuracy and improvement over the base model, and the Readout Correction Ratio (RCR), defined as $\mathrm{RCR}(q) = \|\eta(q)q^\top \Delta S(q)\|_2 / \|q^\top S\|_2$. As shown in Eq. 7, a rehearsal update ΔS perturbs a

Table 4: Cache-selection ablation on S-NIAH-3 16K.

	Standard (500)			UUID-Protected (445)		
Cache Policy	Accuracy	Δ Base	Readout Correction Ratio	Accuracy	Δ Base	Readout Correction Ratio
Baseline	8.20	–	–	2.47	–	–
Random	5.60	-2.60	0.1305	13.03	+10.56	0.1388
Self-Recall Error	5.40	-2.80	0.2568	9.44	+6.97	0.2585
Mahalanobis (Ours)	**19.20**	**+11.00**	0.1567	**29.21**	**+26.74**	0.1647

non-orthogonal read q through $q^\top \Delta S$, so RCR measures the magnitude of this rehearsal-induced perturbation relative to the original fast weight readout.

We consider two settings. In the standard setting, we evaluate which cache policy is most effective using the setup in Section 5.2, varying only the cache-selection policy over all 500 samples. Mahalanobis achieves the highest accuracy among the evaluated policies. We then ask whether this advantage simply comes from retaining the target UUID more often. To isolate this effect and investigate whether the composition of the remaining cache entries also affects performance, we consider a UUID-protected setting in which the target UUID KV pairs are fixed in the cache and exempted from eviction while each cache policy selects the remaining entries. We evaluate the 445 samples whose UUID span lies outside the final sliding-attention window, ensuring that the target is not directly accessible through local attention. Mahalanobis again achieves the highest recall, indicating that reinstatement depends not only on retaining the target association but also on the composition of the remaining cache. Mahalanobis achieves substantially higher accuracy with a lower RCR than SRE, suggesting that it produces more effective corrections rather than simply larger perturbations.

Query Cueing and Rehearsal Strength. We examine two components of RECUE: query-dependent cueing and adaptive reinstatement strength on S-NIAH-3 8K over 500 examples. First, to isolate the effect of query cueing, we compare Ridge Rehearsal and Attention Read with Full RECUE using a fixed strength, choosing η such that their Readout Correction Ratios are comparable. As shown in Table 5, Full RECUE substantially outperforms both Ridge Rehearsal and Attention Read despite similar correction magnitudes, indicating that query-conditioned rehearsal is important for effective reinstatement. Second, we compare Full RECUE with a fixed strength against the margin gate. The margin gate further improves accuracy, supporting adaptive reinstatement strength based on query distinctiveness.

Table 5: Query cueing and rehearsal strength ablation on S-NIAH-3 at 8K. RCR denotes readout correction ratio.

Method	Strength (η)	Acc.	RCR
Baseline	–	34.0	–
Ridge Rehearsal	0.00468	38.2	0.1732
Attention Read	0.00530	62.4	0.1735
Full RECUE	0.01249	82.6	0.1704
Full RECUE	Margin Gate	**85.2**	0.1726

Using the Same Memory Budget for Attention. We compare RECUE with alternative uses of the same additional memory for attention. Attention Read uses the same cache for direct retrieval. As shown in Tables 2 and 3, both Full RECUE and Top-16 RECUE outperform Attention Read on RULER, while Full RECUE also achieves higher aggregate performance on LongBench. We further compare selective caching against LaCT models pretrained with a larger sliding-attention window as an alternative use of the additional memory budget in Appendix D.

6 CONCLUSION

We introduced RECUE, an inference-time reinstatement mechanism for fast weight models that uses the current query to cue relevant cached associations, jointly rehearses them through ridge regression, and rolls the resulting correction back after the read. Across LaCT and DeltaNet, RECUE improves associative recall on RULER and average performance on LongBench without additional training. Ablations further show that query-conditioned rehearsal, adaptive rehearsal strength, and cache composition all contribute to recall. In particular, the Mahalanobis-based cache criterion outperforms self-recall error and random selection. These results suggest that separating *what to preserve* from *when and what to reinstate* offers a simple way to extend bounded fast weight memories by using the cache for selective reinstatement rather than direct value retrieval.

AI USE STATEMENT

We did not use generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in writing proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement research methods, assist with translation, clean or reformat datasets, support qualitative or thematic data analysis, or interpret research results. We used generative AI tools only for tasks for which disclosure is recommended, specifically to improve the readability and clarity of the manuscript and to assist in searching for and identifying potentially relevant literature for the related work. All AI-assisted edits were reviewed and revised by the authors. Any literature identified with the assistance of generative AI was independently verified by the authors against the original sources before being cited or discussed in the manuscript. We take full responsibility for the final content of this work, including all text, claims, analyses, and citations produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 5.1 describes the model checkpoints, benchmarks, and experimental configuration used in our evaluation. Detailed implementation procedures and pseudocode are provided in Appendix E. We use publicly available checkpoints and benchmarks where applicable and plan to release the implementation of RECUE publicly.

REFERENCES

Jimmy Ba, Geoffrey E. Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. In *Neural Information Processing Systems*, 2016.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, 2024. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Ali Behrouz, Peilin Zhong, and Vahab S. Mirrokni. Titans: Learning to memorize at test time. *ArXiv*, abs/2501.00663, 2024.

Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab S. Mirrokni. Atlas: Learning to optimally memorize the context at test time. *ArXiv*, abs/2505.23735, 2025.

Ali Behrouz, Zeman Li, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Memory caching: RNNs with growing memory. In *International Conference on Machine Learning*, 2026.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020.

LOIC CABANNES, Maximilian Beck, Maria Lomeli, Gergely Szilvasy, Matthijs Douze, Jade Copet, Pierre-Emmanuel Mazare, Gabriel Synnaeve, and Herve Jegou. Short window attention enables long-term memorization. In *International Conference on Learning Representations*, volume 2026, pp. 79759–79774, 2026.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2924–2936, 2019. doi: 10.18653/v1/N19-1300.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018.

Wanyun Cui. A hippocampus for linear attention: An exact memory for what the recurrent state forgets, 2026. URL https://arxiv.org/abs/2607.02303.

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

Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, Tao Zhang, and Yu Cheng. MoM: Linear sequence modeling with mixture-of-memories. In *The Fourteenth International Conference on Learning Representations*, 2026.

emozilla. Long-data-collections-pretrain-without-books, 2023. URL https://huggingface.co/datasets/emozilla/Long-Data-Collections-Pretrain-Without-Books.

Guhao Feng, Shengjie Luo, Kai Hua, Ge Zhang, Wenhao Huang, Di He, and Tianle Cai. In-place test-time training. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2604.06169. Oral Presentation.

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, 07 2024. URL https://zenodo.org/records/12608602.

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

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

Han Guo, Songlin Yang, Tarushii Goel, Eric P. Xing, Tri Dao, and Yoon Kim. Log-linear attention. In *The Fourteenth International Conference on Learning Representations*, 2026.

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

Mutian He and Philip N. Garner. Alleviating forgetfulness of linear attention by hybrid sparse attention and contextualized learnable token eviction. *ArXiv*, abs/2510.20787, 2025.

Geoffrey E. Hinton and David C. Plaut. Using fast weights to deblur old memories. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, pp. 177–186, 1987.

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? *arXiv preprint arXiv:2404.06654*, 2024.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024.

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.

Aakash Lahoti, Kevin Y. Li, Berlin Chen, Caitlin Wang, Aviv Bick, J. Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved sequence modeling using state space principles, 2026. URL https://arxiv.org/abs/2603.15569.

Xunhao Lai, Jialiang Kang, Jianqiao Lu, Tong Lin, and Pengyu Zhao. Enhancing linear attention with residual learning, 2025. URL https://arxiv.org/abs/2509.25223.

Huanru Henry Mao. Fine-tuning pre-trained transformers into decaying fast weights. In *Conference on Empirical Methods in Natural Language Processing*, 2022.

Luke McDermott, Robert W. Heath, and Rahul Parhi. LoLA: Low-rank linear attention with sparse caching. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=3KhDA252y3.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Conference on Empirical Methods in Natural Language Processing*, 2018.

Xuan Ouyang, Zefan Cai, and Junjie Hu. Test-time training with next-token prediction, 2026. URL https://arxiv.org/abs/2606.21803.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 8732–8740, 2020. doi: 10.1609/aaai.v34i05.6399.

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

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (Learn at test time): RNNs with expressive hidden states. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 57503–57522. PMLR, 2025.

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, 2023. URL https://arxiv.org/abs/2307.08621.

Together AI. Long data collections, 2023. URL https://huggingface.co/datasets/togethercomputer/Long-Data-Collections.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 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 Aguera y Arcas, and Joao Sacramento. MesaNet: Sequence modeling by locally optimal test-time training. In *The Fourteenth International Conference on Learning Representations*, 2026.

Ke Alexander Wang, Jiaxin Shi, and Emily B. Fox. Test-time regression: a unifying framework for designing sequence models with associative memory. *ArXiv*, abs/2501.12352, 2025.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4413. URL https://aclanthology.org/W17-4413/.

Bernard Widrow and Marcian E. Hoff. Adaptive switching circuits. In *1960 IRE WESCON Convention Record, Part 4*, pp. 96–104, New York, 1960.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Proceedings of ICML*, 2024a.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *ArXiv*, abs/2406.06484, 2024b.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *Proceedings of ICLR*, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics*, 2019.

Ming Zhang, Kaisen Yang, Shu Yu, Ermo Hua, Ning Ding, Xia Hu, Bowen Zhou, Chaochao Lu, and Youbang Sun. MARCH: Scaling recurrent memory with content-routed state anchors, 2026. URL https://arxiv.org/abs/2608.12435.

Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T Freeman, and Hao Tan. Test-time training done right. *arXiv preprint arXiv:2505.23884*, 2025a.

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

Yifei Zuo, Yutong Yin, Zhichen Zeng, Ang Li, Banghua Zhu, and Zhaoran Wang. Local linear attention: An optimal interpolation of linear and softmax attention for test-time regression. In *The Fourteenth International Conference on Learning Representations*, 2026.

A Geometric Interpretation of Cache Selection

This appendix provides a geometric motivation for the cache-selection criterion in Eq. 11. We first relate the criterion to ridge leverage, and then derive the practical diagonal Mahalanobis score as a lightweight approximation to this geometry. This connection explains why the cache favors key directions that are weakly represented by the observed key distribution.

Connection to Ridge Leverage. Let $x_i = k_i - \mu$ denote the centered prefill keys, and stack them as $X \in \mathbb{R}^{N \times d_k}$. The regularized ridge leverage score of x_i is defined by

$$X = \begin{bmatrix} x_1^\top \\ \vdots \\ x_N^\top \end{bmatrix}, \qquad \ell_i = x_i^\top \left(X^\top X + \gamma I \right)^{-1} x_i. \tag{26}$$

The matrix $X^\top X$ captures the aggregate geometry of the observed keys. A large ℓ_i therefore indicates that x_i lies along a direction that is relatively weakly represented by this geometry, giving it greater influence in the corresponding ridge regression problem.

Diagonal Approximation. To obtain an efficient cache score, we approximate the Gram matrix using only its diagonal statistics. Let σ_j^2 denote the empirical variance of the j-th centered key coordinate and $\bar{\sigma}^2$ the average coordinate variance:

$$\sigma_j^2 = \frac{1}{N} \sum_{i=1}^{N} x_{ij}^2, \qquad \bar{\sigma}^2 = \frac{1}{d_k} \sum_{j=1}^{d_k} \sigma_j^2, \qquad X^\top X \approx N \operatorname{diag}\left(\sigma_1^2, \ldots, \sigma_{d_k}^2 \right). \tag{27}$$

We set $\gamma = N \lambda_{\text{sel}} \bar{\sigma}^2$. The regularized diagonal approximation and its inverse are therefore

$$\begin{aligned} N \operatorname{diag}(\sigma^2) + \gamma I &= N \operatorname{diag}\left(\sigma_1^2 + \lambda_{\text{sel}} \bar{\sigma}^2, \ldots, \sigma_{d_k}^2 + \lambda_{\text{sel}} \bar{\sigma}^2 \right), \\ \left[N \operatorname{diag}(\sigma^2) + \gamma I \right]^{-1} &= \frac{1}{N} \operatorname{diag}\left(\frac{1}{\sigma_1^2 + \lambda_{\text{sel}} \bar{\sigma}^2}, \ldots, \frac{1}{\sigma_{d_k}^2 + \lambda_{\text{sel}} \bar{\sigma}^2} \right). \end{aligned} \tag{28}$$

Substituting this into the ridge leverage score gives

$$\ell_i^{\text{diag}} = x_i^\top \left[N \operatorname{diag}(\sigma^2) + \gamma I \right]^{-1} x_i = \frac{1}{N} \sum_{j=1}^{d_k} \frac{x_{ij}^2}{\sigma_j^2 + \lambda_{\text{sel}} \bar{\sigma}^2} = \frac{1}{N} \sum_{j=1}^{d_k} \frac{(k_{ij} - \mu_j)^2}{\sigma_j^2 + \lambda_{\text{sel}} \bar{\sigma}^2}. \tag{29}$$

Finally, normalizing by the key dimension yields

$$s_i = \frac{N}{d_k} \ell_i^{\text{diag}} = \frac{1}{d_k} \sum_{j=1}^{d_k} \frac{(k_{ij} - \mu_j)^2}{\sigma_j^2 + \lambda_{\text{sel}} \bar{\sigma}^2}, \tag{30}$$

which is exactly the cache-selection score in Eq. 11. Since N/d_k is constant across all keys, ranking by s_i is equivalent to ranking by the diagonal ridge leverage approximation.

Thus, the practical cache criterion can be viewed equivalently as a regularized centered diagonal Mahalanobis score or as a lightweight diagonal approximation to ridge leverage. It favors key directions that are weakly represented by the global key geometry, while avoiding the cost of a full Gram-matrix inverse. This also provides a natural connection to subsequent ridge rehearsal, which uses the Gram geometry of the retained keys to jointly fit their reconstruction residuals.

B Effect of Cue Temperature

Full ReCue controls the sharpness of query-dependent residual cueing through the temperature τ in Eq. 20. Smaller temperatures concentrate the correction on associations most relevant to the current query, while larger temperatures make the cue increasingly uniform across the cache, causing

Table 6: RULER benchmark results on Needle-in-a-Haystack (NIAH) tasks.

Model	S-NIAH-1			S-NIAH-2			S-NIAH-3			Average		
	4K	8K	16K	4K	8K	16K	4K	8K	16K	4K	8K	16K
760M params / 40B tokens												
LaCT	99.0	93.0	97.2	98.0	83.8	37.6	55.2	34.0	8.2	84.07	70.27	47.67
+ Ridge Rehearsal	99.4	96.0	97.6	98.8	88.4	42.4	58.6	37.8	8.4	85.60	74.07	49.47
+ Full RECUE ($\tau = 0.05$)	**100.0**	**97.4**	97.8	**99.8**	**99.8**	**64.2**	**71.6**	**88.6**	**19.8**	**90.47**	**95.27**	**60.60**
+ Full RECUE ($\tau = 0.1$)	**100.0**	**97.4**	97.8	**99.8**	**99.8**	58.4	70.8	85.2	19.2	90.20	94.13	58.47
+ Full RECUE ($\tau = 0.2$)	**100.0**	96.8	**98.0**	99.4	99.4	55.6	70.2	77.6	15.8	89.87	91.27	56.47
+ Full RECUE ($\tau = 0.4$)	**100.0**	96.8	97.6	99.2	96.2	50.8	65.8	63.4	12.8	88.33	85.47	53.73
+ Full RECUE ($\tau = 0.8$)	99.6	96.4	97.4	99.0	93.4	47.8	62.0	50.8	10.0	86.87	80.20	51.73
1.3B params / 100B tokens												
DeltaNet	100.0	100.0	100.0	76.2	19.8	2.0	54.2	8.6	0.8	76.80	42.80	34.27
+ Ridge Rehearsal	100.0	100.0	100.0	85.8	22.8	1.0	52.6	9.6	0.2	79.47	44.13	33.73
+ Full RECUE ($\tau = 0.05$)	100.0	100.0	100.0	91.0	**29.8**	2.2	**67.8**	**13.8**	**1.8**	86.27	**47.87**	34.67
+ Full RECUE ($\tau = 0.1$)	100.0	100.0	100.0	**91.4**	28.6	**2.4**	67.6	13.4	**1.8**	**86.33**	47.33	**34.73**
+ Full RECUE ($\tau = 0.2$)	100.0	100.0	100.0	89.4	28.4	1.4	64.4	**13.8**	1.4	84.60	47.40	34.27
+ Full RECUE ($\tau = 0.4$)	100.0	100.0	100.0	89.0	25.6	1.2	62.0	13.0	1.4	83.67	46.20	34.20
+ Full RECUE ($\tau = 0.8$)	100.0	100.0	100.0	87.2	25.4	1.4	58.0	13.0	0.8	81.73	46.13	34.07

Model	MK-NIAH-1			MQ-NIAH			MV-NIAH			Average		
	4K	8K	16K	4K	8K	16K	4K	8K	16K	4K	8K	16K
760M params / 40B tokens												
LaCT	76.4	48.6	30.8	46.20	29.65	20.80	34.30	27.85	24.50	52.30	35.37	25.37
+ Ridge Rehearsal	77.0	51.0	31.0	45.50	30.30	19.45	34.40	28.30	23.20	52.30	36.53	24.55
+ Full RECUE ($\tau = 0.05$)	82.2	71.0	**44.6**	45.30	36.95	**30.40**	35.95	**36.90**	34.50	54.48	48.28	**36.50**
+ Full RECUE ($\tau = 0.1$)	**84.2**	**72.4**	44.2	47.00	**37.60**	29.50	**39.25**	**36.90**	**34.75**	**56.82**	**48.97**	36.15
+ Full RECUE ($\tau = 0.2$)	81.0	64.6	41.0	**48.25**	36.65	26.20	37.65	34.50	31.50	55.63	45.25	32.90
+ Full RECUE ($\tau = 0.4$)	78.8	58.0	37.2	46.70	34.85	23.10	35.95	32.20	27.55	53.82	41.68	29.28
+ Full RECUE ($\tau = 0.8$)	77.6	54.2	33.0	46.45	32.65	20.65	35.30	30.75	25.80	53.12	39.20	26.48
1.3B params / 100B tokens												
DeltaNet	25.4	11.6	**2.4**	**27.15**	12.45	1.70	18.15	12.90	4.25	23.57	12.32	2.78
+ Ridge Rehearsal	23.4	10.8	1.6	**27.15**	14.65	1.75	19.20	14.15	4.00	23.25	13.20	2.45
+ Full RECUE ($\tau = 0.05$)	**26.0**	**19.8**	2.2	26.95	18.85	2.55	21.90	21.30	7.70	**24.95**	19.98	4.15
+ Full RECUE ($\tau = 0.1$)	25.6	19.2	2.2	26.95	19.70	2.60	**22.30**	21.55	**8.05**	**24.95**	**20.15**	**4.28**
+ Full RECUE ($\tau = 0.2$)	**26.0**	18.2	2.2	26.20	**20.10**	2.90	22.10	**21.70**	7.70	24.77	20.00	4.27
+ Full RECUE ($\tau = 0.4$)	**26.0**	16.4	2.2	26.05	20.00	**2.95**	22.05	20.55	6.80	24.70	18.98	3.98
+ Full RECUE ($\tau = 0.8$)	**26.0**	13.4	2.2	25.90	18.90	2.45	21.15	18.70	5.70	24.35	17.00	3.45

Full RECUE to approach Ridge Rehearsal in Eq. 16. To characterize this transition, we evaluate $\tau \in \{0.05, 0.1, 0.2, 0.4, 0.8\}$ on both LaCT and DeltaNet. We additionally compare against the corresponding base model and Ridge Rehearsal with the same margin gate, allowing us to isolate the contribution of query-dependent residual cueing. All other model, cache, and rehearsal settings are kept identical to those in Section 5.1.

Results. As shown in Table 6, Full RECUE performs best with relatively small temperatures on RULER. For LaCT, $\tau = 0.05$ is strongest on the single-needle tasks, while $\tau = 0.1$ performs slightly better on most multi-association settings. DeltaNet shows a similar pattern, with $\tau = 0.05$ and $\tau = 0.1$ consistently providing the strongest results. Performance generally degrades as τ increases, suggesting that sharper query-dependent cueing is important for associative recall.

Table 7 shows a similar trend on LongBench. Full RECUE with $\tau \in \{0.05, 0.1\}$ achieves the strongest overall performance on both backbones, while larger temperatures generally reduce the average score. As τ increases and the cue becomes more uniform, performance moves closer to that of Ridge Rehearsal. This supports the importance of query-dependent residual cueing rather than uniformly rehearsing the cache.

Overall, Full RECUE is relatively robust for small temperatures. We use $\tau = 0.1$ in the main experiments as a strong and balanced default across RULER and LongBench.

Table 7: Effect of the temperature parameter τ on LongBench tasks. NQA: NarrativeQA; QQA: Qasper; MFQA: MultiFieldQA; HQA: HotpotQA; 2Wiki: 2WikiMQA; GovRep: GovReport; MNews: MultiNews.

	Single-Doc QA			Multi-Doc QA			Summarization		
Model	NQA	QQA	MFQA	HQA	2Wiki	MuSiQue	GovRep	QMSum	MNews
LaCT	2.36	5.00	13.85	5.98	6.71	**3.80**	11.93	**15.18**	5.29
+ Ridge Rehearsal	2.64	4.97	14.52	6.03	6.86	3.63	12.56	14.57	5.77
+ Full RECUE ($\tau = 0.05$)	2.72	**5.55**	14.20	**6.64**	7.23	3.43	**14.41**	14.68	6.33
+ Full RECUE ($\tau = 0.1$)	**2.81**	5.51	14.57	6.63	**7.29**	3.70	14.06	14.68	**6.34**
+ Full RECUE ($\tau = 0.2$)	2.71	5.33	14.30	6.41	7.06	3.45	12.70	14.68	5.97
+ Full RECUE ($\tau = 0.4$)	2.74	4.96	**14.98**	6.22	6.98	3.54	12.55	14.67	5.90
+ Full RECUE ($\tau = 0.8$)	2.67	5.03	14.55	6.13	6.55	3.69	12.78	14.57	5.95
DeltaNet	3.26	3.57	12.93	4.78	8.05	2.01	5.39	11.34	11.10
+ Ridge Rehearsal	**3.63**	3.67	11.81	3.97	7.40	2.34	4.76	10.44	11.11
+ Full RECUE ($\tau = 0.05$)	2.78	4.08	**15.18**	4.81	7.62	**2.46**	6.38	**11.98**	17.03
+ Full RECUE ($\tau = 0.1$)	3.05	4.04	14.84	**5.05**	**8.08**	2.34	6.48	11.81	**17.07**
+ Full RECUE ($\tau = 0.2$)	2.33	**4.50**	13.78	4.70	7.60	2.36	**6.88**	11.81	16.86
+ Full RECUE ($\tau = 0.4$)	3.22	4.35	13.79	4.62	7.94	2.39	6.60	11.08	15.95
+ Full RECUE ($\tau = 0.8$)	2.24	3.61	12.29	3.90	7.70	2.34	5.86	11.24	15.49

	Few-shot Learning			Code		Overall
Model	TREC	TriviaQA	SAMSum	LCC	RepoBench-P	Avg.
LaCT	13.00	30.98	27.82	**32.95**	35.52	15.03
+ Ridge Rehearsal	**13.50**	30.63	29.76	31.87	36.04	15.24
+ Full RECUE ($\tau = 0.05$)	12.50	32.35	28.90	32.64	35.85	15.53
+ Full RECUE ($\tau = 0.1$)	**13.50**	32.71	29.50	32.47	36.16	**15.71**
+ Full RECUE ($\tau = 0.2$)	**13.50**	**33.78**	29.01	32.27	36.52	15.55
+ Full RECUE ($\tau = 0.4$)	**13.50**	31.47	29.58	32.24	36.17	15.39
+ Full RECUE ($\tau = 0.8$)	**13.50**	31.28	**29.94**	32.61	**36.53**	15.41
DeltaNet	**25.50**	26.25	**9.09**	**41.56**	34.05	14.21
+ Ridge Rehearsal	21.50	24.38	5.53	39.12	32.92	13.04
+ Full RECUE ($\tau = 0.05$)	**25.50**	26.64	8.94	40.81	34.10	**14.88**
+ Full RECUE ($\tau = 0.1$)	25.00	**27.45**	8.63	40.04	34.22	14.86
+ Full RECUE ($\tau = 0.2$)	25.00	27.29	8.93	40.41	**34.76**	14.80
+ Full RECUE ($\tau = 0.4$)	23.50	26.19	7.89	39.69	34.04	14.37
+ Full RECUE ($\tau = 0.8$)	23.00	24.96	6.81	39.56	33.26	13.73

C General Language Modeling Performance

To examine whether the associative recall gains from RECUE come at the expense of general language modeling performance, we evaluate the base LaCT and DeltaNet models and their Full RECUE using `lm-evaluation-harness` (Gao et al., 2024). For downstream evaluation, we use ARC-Challenge, ARC-Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), BoolQ (Clark et al., 2019), WinoGrande (Sakaguchi et al., 2020), OpenBookQA (Mihaylov et al., 2018), LAMBADA (Paperno et al., 2016), and SciQ (Welbl et al., 2017). We additionally report perplexity on LAMBADA and WikiText (Merity et al., 2017). We use the same RECUE configuration as in Section 5.1 and report results in Table 8.

Results. Full RECUE largely preserves the general performance of both backbones. For LaCT, the average accuracy remains nearly unchanged, while perplexity is also comparable to the base model. For DeltaNet, Full RECUE improves the average accuracy and reduces perplexity. These results

Table 8: `lm-evaluation-harness` results for LaCT 760M and DeltaNet 1.3B with and without Full RECUE. ARC-C: ARC-Challenge; ARC-E: ARC-Easy; Hella.: HellaSwag; Wino.: WinoGrande; OBQA: OpenBookQA; LAMB.: LAMBADA; Wiki: WikiText; acc: accuracy; acc_n: normalized accuracy; ppl: perplexity (lower is better).

Model	ARC-C (acc)	ARC-E (acc)	Hella. (acc_n)	PIQA (acc_n)	BoolQ (acc)	Wino. (acc)	OBQA (acc)	LAMB. (acc)	SciQ (acc)	LAMB. ppl↓	Wiki ppl↓	Avg. (acc)
LaCT	25.43	**49.96**	41.67	66.00	55.29	51.07	19.20	38.83	90.70	**20.106**	20.949	48.68
+ Full RECUE	**25.51**	49.83	41.67	**66.27**	**55.35**	**51.30**	19.20	**38.93**	90.70	20.111	**20.904**	**48.75**
DeltaNet	24.23	**58.71**	**50.25**	**70.67**	61.16	53.43	19.60	48.42	84.40	12.270	16.665	52.32
+ Full RECUE	**25.17**	57.66	50.09	70.46	**61.53**	**53.59**	**20.80**	**54.98**	**84.50**	**8.614**	**16.573**	**53.20**

suggest that the improvements in associative recall do not come at the cost of broadly degrading general language modeling performance.

D INCREASING THE SLIDING WINDOW CAN HURT LONG-TERM MEMORY

One alternative to maintaining an additional cache is to allocate the same memory budget to a larger sliding-attention window. However, the two serve different roles. Sliding Window Attention (SWA) retains more recent tokens but only extends the boundary of direct attention; associations beyond the window remain inaccessible through the local attention path. In contrast, RECUE uses a bounded cache to preserve selected associations from the entire preceding context and reinstate them when cued by the current query.

Moreover, allocating additional memory to a larger sliding window can be counterproductive for long-term memorization in hybrid architectures. Recent work (CABANNES et al., 2026) shows that larger attention windows can hurt long-context performance by encouraging greater reliance on local attention and less on recurrent memory, whereas shorter attention windows encourage better training of the recurrent component's long-term memory. This motivates using the additional memory budget for selective preservation rather than simply extending the local attention window.

To examine whether this effect also holds for TTT–SWA hybrid models, we pretrain two LaCT models with 2K and 4K sliding-attention windows, respectively, while keeping all other model and training configurations, including the training data, identical. We evaluate both models on the 8K S-NIAH-1/2/3 tasks, separately measuring accuracy for examples whose needle falls inside or outside the final 4K tokens. This split separates cases where the needle remains directly accessible to the 4K attention window from cases that require retrieval beyond the local window. The results are reported in Table 9. We additionally evaluate general language modeling performance using `lm-evaluation-harness` (Gao et al., 2024) to examine whether increasing the SWA window affects overall model quality. These results are shown in Table 10.

Training Setup. We follow the LaCT architecture and training recipe (Zhang et al., 2025a). Both models have 760M parameters and use a LaCT chunk size of 2K. We use momentum with the Muon optimizer, pre-normalization, and RoPE, with a sequence length of 32,768. Each optimizer step processes 1M tokens, and both models are trained for 40,960 steps. The only architectural difference between the two models is the SWA window size, which is set to 2K and 4K, respectively. Since the exact Long-Data-Collections (Together AI, 2023) corpus used by the original LaCT model is no longer publicly available, we instead use a publicly available variant of Long-Data-Collections with the Books portion removed (emozilla, 2023). Since this corpus differs from the data used by the original LaCT models, we use these models only for a controlled comparison of sliding-window sizes, rather than as reproductions of the original LaCT results.

Results. Table 9 shows a clear trade-off between local attention and long-term memory. The SWA-4K model performs substantially better when the needle remains within the final 4K tokens, where it is directly accessible through local attention. However, its accuracy drops to zero across all three tasks once the needle falls outside the 4K window. In contrast, the SWA-2K model retains substantial recall beyond the final 4K tokens, achieving 47.45 and 64.06 accuracy on S-NIAH-1 and S-NIAH-2, respectively. This difference is not accompanied by a degradation in general language

Table 9: Accuracy on 8K S-NIAH tasks, split by whether the needle remains inside the final 4K sliding-attention window.

	S-NIAH-1		S-NIAH-2		S-NIAH-3	
Model	inside 4K	outside 4K	inside 4K	outside 4K	inside 4K	outside 4K
LaCT (SWA-2K)	76.33	47.45	69.61	64.06	43.24	10.29
+ Full RECUE	86.94	78.82	85.16	83.87	50.00	30.88
LaCT (SWA-4K)	100.0	0.0	100.0	0.0	91.22	0.0

Table 10: `lm-evaluation-harness` results for LaCT 760M. Both models use a 2K chunk size and differ only in the SWA window. ARC-C: ARC-Challenge; ARC-E: ARC-Easy; Hella.: HellaSwag; Wino.: WinoGrande; OBQA: OpenBookQA; LAMB.: LAMBADA; Wiki: WikiText; acc: accuracy; acc_n: normalized accuracy; ppl: perplexity (lower is better).

Model	**ARC-C** (acc)	**ARC-E** (acc)	**Hella.** (acc_n)	**PIQA** (acc_n)	**BoolQ** (acc)	**Wino.** (acc)	**OBQA** (acc)	**LAMB.** (acc)	**SciQ** (acc)	**LAMB.** ppl↓	**Wiki** ppl↓	**Avg.** (acc)
SWA-2K	**25.26**	**51.85**	43.55	**67.03**	56.18	**53.99**	17.80	42.46	**90.80**	16.971	20.863	49.88
SWA-4K	23.46	48.74	**44.68**	66.81	**60.86**	53.83	**19.20**	**44.07**	87.90	**16.428**	**19.663**	**49.95**

modeling performance. As shown in Table 10, SWA-2K and SWA-4K achieve nearly identical average accuracy (49.88 vs. 49.95), while SWA-4K achieves lower perplexity on both LAMBADA and WikiText. This suggests that the weaker long-range recall of SWA-4K is not due to poorer overall training, but rather to a greater reliance on the enlarged local attention window. Applying Full RECUE to SWA-2K substantially improves out-of-window recall across all three S-NIAH tasks. Unlike enlarging the SWA window, RECUE recovers associations after they leave the local attention window, supporting selective reinstatement over simply extending local attention.

We hypothesize that this behavior arises from LaCT's outer-loop next-token prediction objective. A larger SWA window exposes more predictive information through local attention, reducing the pressure on the recurrent TTT memory to retain information over long horizons. This may explain why SWA-4K performs well within its window but loses recall beyond it, whereas SWA-2K retains longer-term information. In this setting, enlarging the sliding window may therefore trade the development of long-term recurrent memory for stronger local retrieval, while RECUE preserves the shorter-window memory dynamics and selectively restores associations when they are needed.

E IMPLEMENTATION DETAILS AND PSEUDOCODE

Notation. Let $\mathcal{C}$ denote the bounded association cache and $\mathcal{U}$ the newly eligible token chunk. Let f_W denote the native fast weight read and define

$$\psi_W(x) = \begin{cases} x, & \text{DeltaNet}, \\ \phi_\theta(x), & \text{LaCT}, \end{cases} \qquad \mathrm{unit}(x) = \frac{x}{\max(\|x\|_2, \epsilon)}.$$

For LaCT, ϕ_θ is the gated-MLP representation before the linear output projection. Cache selection and query cueing operate in key space, while ridge regression operates on ψ_W. All operations are performed independently for each layer and head.

Implementation. The cache and reusable ridge map are refreshed whenever a new chunk becomes eligible and reused until the next refresh. Residuals are computed at each refresh from the current native fast weight state. We row-normalize the regression features and use a scale-relative ridge penalty, $\lambda_{\text{eff}} = \lambda_{\text{rr}} \max\left(\frac{\mathrm{tr}(X^\top X)}{d_f}, \epsilon\right)$. The backbone fast weight state continues to follow its native update rule, while RECUE modifies only the current readout.

We use $(B, C_{\text{chunk}}) = (2048, 2048)$ for LaCT and $(512, 512)$ for DeltaNet, with $\lambda_{\text{sel}} = 10^{-3}$, $\lambda_{\text{rr}} = 1$, $\tau = 0.1$, correction cap $\alpha = 0.25$, and $\epsilon = 10^{-6}$. At each refresh, cache statistics are computed over the retained cache and the newly eligible chunk.

Algorithm 1 RECUE cache update and ridge-map preparation

Require: Cache $\mathcal{C}$, newly eligible associations $\mathcal{U}$, current native state W, budget B, λ_{sel}, λ_{rr}
Ensure: Updated cache $\mathcal{C}$ and reusable map $\mathcal{M}$

$\mathcal{A} \leftarrow \mathcal{C} \,\|\, \mathcal{U}$; $n \leftarrow |\mathcal{A}|$
if $n = 0$ **then**
  return $(\varnothing, \varnothing)$
end if
$\mu \leftarrow n^{-1} \sum_{i=1}^{n} k_i$
$\sigma^2 \leftarrow n^{-1} \sum_{i=1}^{n} (k_i - \mu)^{\odot 2}$
$\delta_{\text{sel}} \leftarrow \lambda_{\text{sel}} \max\Big(d_k^{-1} \sum_{j=1}^{d_k} \sigma_j^2, \epsilon\Big)$
for $i = 1, \ldots, n$ **do**
  $s_i \leftarrow d_k^{-1} \sum_{j=1}^{d_k} \frac{(k_{ij} - \mu_j)^2}{\sigma_j^2 + \delta_{\text{sel}}}$
end for
$I \leftarrow \text{TopIndices}(s, \min(B, n))$
$\mathcal{C} \leftarrow [(k_i, v_i)]_{i \in I}$; $m \leftarrow |\mathcal{C}|$
$A \leftarrow \text{rownorm}([k_i^\top]_{(k_i, v_i) \in \mathcal{C}})$ ▷ Cue features
$X \leftarrow \text{rownorm}([\psi_W(k_i)^\top]_{(k_i, v_i) \in \mathcal{C}})$ ▷ Regression features
$R \leftarrow [(v_i - f_W(k_i))^\top]_{(k_i, v_i) \in \mathcal{C}}$ ▷ Reconstruction residuals
$G \leftarrow X^\top X$
$\lambda_{\text{eff}} \leftarrow \lambda_{\text{rr}} \max(\text{tr}(G)/d_f, \epsilon)$
$L \leftarrow \text{Cholesky}(G + \lambda_{\text{eff}} I_{d_f})$
$P \leftarrow \text{CholeskySolve}(L, X^\top)$ ▷ $P \in \mathbb{R}^{d_f \times m}$
$\mathcal{M} \leftarrow (A, P, R)$
return $(\mathcal{C}, \mathcal{M})$

Algorithm 2 Full RECUE read with a reusable cache map

Require: Query q, native fast-memory output o_{base}, native state W, map $\mathcal{M}$, temperature τ, correction cap α
Ensure: Corrected read o; native state W remains unchanged

if $\mathcal{M} = \varnothing$ **then**
  return o_{base}
end if
$(A, P, R) \leftarrow \mathcal{M}$; $m \leftarrow \text{rows}(A)$
$a \leftarrow \text{unit}(q)$; $z \leftarrow \text{unit}(\psi_W(q))$
$c \leftarrow Aa$ ▷ Query–key cosine similarities
$c_{(1)} \leftarrow \max_i c_i$
$\rho_i \leftarrow \exp((c_i - c_{(1)})/\tau)$ for $i = 1, \ldots, m$ ▷ Maximum-relative cue
$b \leftarrow P^\top z$ ▷ Ridge coefficients
$c_{(2)} \leftarrow \text{SecondLargest}(c)$
$\eta \leftarrow \min(1, \max(0, c_{(1)} - c_{(2)}))$
$u \leftarrow \eta R^\top (\rho \odot b)$
$\gamma \leftarrow \min\Big(1, \frac{\alpha \|o_{\text{base}}\|_2}{\max(\|u\|_2, \epsilon)}\Big)$
$o \leftarrow o_{\text{base}} + \gamma u$
return o ▷ Read-only correction implements rollback