ERASE-THEN-DELTA ATTENTION: DECOUPLING ERASE AND WRITE ADDRESSES IN DELTA-RULE LINEAR ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Delta-rule linear attention improves recurrent memory updates by correcting what is already stored at the current write address before writing new content. However, the active correction is still anchored to that same write address. As a result, stale information stored at a different address cannot be actively removed before new content is written elsewhere. We propose **Erase-then-Delta Attention** (EDA), a memory update rule that decouples where to erase from where to write. The key insight is that recurrent memory models should not only correct the current write, but also selectively suppress outdated memory at an independently chosen address. Concretely, our method first applies a targeted erase step along a learned erase direction, and then performs the standard delta-style corrective write along the current write direction. This preserves the corrective behavior of delta-rule updates while expanding their memory-management capacity. EDA achieves the highest average score in both dense 2.5B and MoE 25B-A2.8B pretraining, with gains persisting after 80B-token long-context midtraining of the MoE models. A compact update analysis and memory-state probes suggest why: EDA keeps the delta-rule corrective write intact while allocating an additional cleanup path most strongly when passive decay is weak. These results suggest that recurrent memory models should decide not only what to write, but also what stale information to erase and where.

1 INTRODUCTION

Autoregressive Transformers (Vaswani et al., 2017) have become the foundation of modern language modeling, in part because softmax-based self-attention enables efficient parallel computation. This mechanism achieves strong performance on in-context learning and long-context retrieval by maintaining an explicit key–value cache. However, it also introduces fundamental bottlenecks at inference time: quadratic time complexity and linearly growing memory overhead that limit scalability for long-sequence tasks and agentic reasoning trajectories. To address these constraints, a growing body of work has explored efficient alternatives that maintain constant memory and $\mathcal{O}(1)$ inference time while preserving the expressive power of attention.

Recurrent models based on linear attention (Katharopoulos et al., 2020) and state space models (Gu et al., 2022; Gu & Dao, 2024) offer a principled solution: they compress contextual information into a fixed-size state, enabling constant memory and linear-time training. Early variants such as Linformer (Wang et al., 2020) and RetNet (Sun et al., 2023) lacked data-dependent memory control and underperformed softmax attention. Subsequent models introduced dynamic gating mechanisms (Yang et al., 2025; Dao & Gu, 2024; Beck et al., 2024), allowing selective forgetting and significantly narrowing the performance gap. However, additive gated updates still write new content into a finite state without explicitly correcting the association currently stored at the write address.

A more recent line of work replaces additive updates with the *delta rule* (Schlag et al., 2021), which treats the recurrent state as a learnable associative memory that corrects itself toward the current key–value mapping. Gated DeltaNet (GDN) (Yang et al., 2025) combines this corrective write with a head-wise forget gate, and recent channel-wise variants further refine this gate into a diagonal decay that gives each key feature its own retention rate (Team et al., 2025). GDN-2 further separates the

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.

scalar delta gate into key-side erase and value-side write gates, but the active edit remains organized around the current write key (Hatamizadeh et al., 2026). We build on this channel-wise gated delta setting, also known as diagonal-plus-low-rank (DPLR), which combines GDN's hardware-efficient delta-rule structure with finer-grained channel-wise forgetting. Despite this progress, a structural limitation remains unaddressed: the active delta correction still uses the current write direction $\mathbf{k}_t$ as its only address. This coupling means the model can only suppress memory at the address it is currently writing to; stale information stored elsewhere must either persist or decay through channel-wise but address-agnostic forgetting.

This limitation has tangible consequences. In language modeling and state-tracking tasks, useful memory updates require not only writing new content but also removing obsolete information that would otherwise interfere with future reads and writes. When the model encounters a situation where earlier information must be invalidated—for example, a variable reassignment, a fact correction, or a context shift—it has no direct mechanism to remove the old content before committing the new one. The core missing capability is therefore not stronger forgetting, but *targeted deletion of outdated memory at an address chosen independently of the current write*.

We address this problem with **Erase-then-Delta Attention** (EDA), a memory update rule that decouples erasure from writing. Instead of tying memory suppression to the current write address, EDA first removes stale content at an independently selected address and then performs the usual delta-style corrective write at the current write address. Intuitively, the erase step actively clears obsolete memory, while the delta step preserves the corrective writing behavior that makes delta-rule models effective. This yields a strictly richer update rule: the model can erase at one address and write at another within the same recurrent step.

We show that this simple modification has three important consequences. First, it provides a cleaner memory-management view of channel-wise gated delta recurrence by separating diagonal decay, independently addressed erasure, and write-coupled correction. Second, empirical analysis reveals that the model learns a near-orthogonal separation between erase and write addressing, indicating that the two operations serve genuinely different roles. Third, language-model pretraining experiments show that EDA improves over a DPLR-style gated delta baseline and compares favorably with several strong update-rule variants.

In summary, we introduce EDA, a gated delta-rule linear-attention update that decouples erase and write addresses while preserving the standard delta corrective write. We analyze the resulting erase-then-delta update and evaluate it through language-model pretraining, long-context evaluation, and memory-state probes, showing that the extra address acts as a conditional cleanup path rather than merely stronger forgetting.

2 PRELIMINARY

We briefly introduce the recurrent memory notation and the channel-wise gated delta update most relevant to our method. The key point is that a diagonal forget gate already provides fine-grained decay, but the active correction and writing remain tied to the same address.

2.1 NOTATION AND LINEAR ASSOCIATIVE MEMORY

We consider a recurrent memory state $\mathbf{S}_t \in \mathbb{R}^{d_k \times d_v}$ updated at each step t. The key $\mathbf{k}_t \in \mathbb{R}^{d_k}$ serves as a write address, the value $\mathbf{v}_t \in \mathbb{R}^{d_v}$ is the content to store, and the query $\mathbf{q}_t \in \mathbb{R}^{d_k}$ reads from memory through $\mathbf{S}_t^\top \mathbf{q}_t \in \mathbb{R}^{d_v}$.

Standard linear attention updates memory additively:

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{k}_t \mathbf{v}_t^\top, \qquad \mathbf{o}_t = \mathbf{S}_t^\top \mathbf{q}_t. \tag{1}$$

This rule is efficient but does not explicitly decide what stale information to suppress.

2.2 Coupled Erasure and Corrective Writing

DeltaNet (Schlag et al., 2021; Yang et al., 2024) replaces additive writing with a corrective update derived from the reconstruction loss

$$\mathcal{L}_t^{\text{delta}}(\mathbf{S}) = \frac{1}{2}\|\mathbf{S}^\top \mathbf{k}_t - \mathbf{v}_t\|^2. \tag{2}$$

Taking a gradient step with learning rate β_t gives

$$\mathbf{S}_t = (\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top)\mathbf{S}_{t-1} + \beta_t \mathbf{k}_t \mathbf{v}_t^\top. \tag{3}$$

Rather than simply accumulating $\mathbf{k}_t \mathbf{v}_t^\top$, DeltaNet first corrects what memory currently returns at address $\mathbf{k}_t$ and then writes the new content at that same address.

Gated DeltaNet (GDN) (Yang et al., 2025) augments this rule with a head-wise scalar forget gate $\alpha_t \in (0, 1)$:

$$\mathbf{S}_t = \alpha_t(\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top)\mathbf{S}_{t-1} + \beta_t \mathbf{k}_t \mathbf{v}_t^\top. \tag{4}$$

Here α_t provides uniform decay within a head, while $(\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top)$ provides address-specific correction. However, the erase-and-write behavior is still coupled: the same key $\mathbf{k}_t$ determines both where memory is strongly modified and where new content is written. As a result, GDN can strongly suppress only the address it is currently writing to.

Following Kimi Delta Attention (KDA) (Team et al., 2025), we use the channel-wise version of this GDN design, replacing the head-wise scalar forget gate with a diagonal decay $\mathbf{D}_t = \mathrm{Diag}(\boldsymbol{\alpha}_t)$:

$$\mathbf{S}_t = (\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top)\mathbf{D}_t\mathbf{S}_{t-1} + \beta_t \mathbf{k}_t \mathbf{v}_t^\top. \tag{5}$$

The diagonal gate gives each key channel its own retention rate and makes the transition compatible with a diagonal-plus-low-rank view. This improves how strongly different channels are preserved or decayed, but it does not change the addressing structure of the delta update itself: the corrective modification is still anchored to the current write key. Therefore, even with channel-wise gating, stale information stored at a different address cannot be explicitly erased before writing new content elsewhere.

GDN-2 addresses a closely related coupling by separating the scalar delta gate into key-side erase and value-side write gates (Hatamizadeh et al., 2026):

$$\mathbf{S}_t = \left(\mathbf{I} - \mathbf{k}_t \widetilde{\mathbf{e}}_t^\top\right)\mathbf{D}_t\mathbf{S}_{t-1} + \mathbf{k}_t \boldsymbol{z}_t^\top, \qquad \widetilde{\mathbf{e}}_t = \boldsymbol{b}_t \odot \mathbf{k}_t, \quad \boldsymbol{z}_t = \boldsymbol{w}_t \odot \mathbf{v}_t. \tag{6}$$

This decouples the channel-wise erase and write strengths inside the delta residual. However, the erase/read direction $\widetilde{\mathbf{e}}_t$ is still constructed from the current write key $\mathbf{k}_t$, and the correction is still committed along $\mathbf{k}_t$. Thus GDN-2 relaxes the gate-level coupling, while the address-level coupling between erasure and writing remains.

This coupling is the limitation we target. If stale information is stored at an address different from the current write address, the diagonal gate can decay feature channels but cannot selectively remove that stale association before writing elsewhere.

2.3 Relation to Recent Delta-Style Variants

Recent linear-recurrent models often improve performance by enriching the transition rule or embedding delta-style memory updates inside stronger architectures. DeltaProduct (Siems et al., 2026) increases transition expressivity through multiple Householder-like factors per step, while RWKV-7 (Peng et al., 2025) and Comba (Hu et al., 2026) adopt richer structured transition parameterizations. Recent hybrid architectures further demonstrate that strong designs built around expressive channel-wise gated delta components can be highly competitive with full attention (Team et al., 2025).

Our goal is different. We do not primarily seek a globally richer transition; instead, we introduce a missing memory-management capability: erasing stale memory at one address before performing the standard delta-style corrective write at another. In that sense, our method is best viewed as orthogonal to transition-enrichment approaches and potentially compatible with stronger channel-wise gated delta backbones.

3 METHOD

3.1 OVERVIEW

Our goal is to extend gated delta-rule linear attention with a missing memory-management capability: selectively deleting stale memory at an address different from the current write address. To do this, we revisit the DPLR-style update rule and identify a structural coupling between active correction and writing. We then introduce **Erase-then-Delta Attention** (EDA), a sequential update rule that adds an independently addressed erase step before the standard delta-style corrective write. This section first formalizes the limitation of the decay-gated delta baseline, then derives the new rule, and finally discusses its algebraic structure and stability properties.

3.2 ERASE-WRITE COUPLING IN GATED DELTA UPDATES

We consider a recurrent memory state $\mathbf{S}_t$ updated by a gated delta rule with diagonal decay:

$$\mathbf{S}_t = (\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top)\mathbf{D}_t \mathbf{S}_{t-1} + \beta_t \mathbf{k}_t \mathbf{v}_t^\top, \qquad \mathbf{D}_t = \mathrm{Diag}(\boldsymbol{\alpha}_t). \tag{7}$$

Here $\mathbf{D}_t$ is a diagonal decay matrix with retention factors $\boldsymbol{\alpha}_t$, β_t controls the delta-style correction strength, $\mathbf{k}_t$ is the current write direction, and $\mathbf{v}_t$ is the value vector written into memory. This update is effective because it is not a naive additive write: after diagonal decay, the factor $(\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top)$ corrects the memory response along the current write direction, and the additive term $\beta_t \mathbf{k}_t \mathbf{v}_t^\top$ writes the new content at the same address.

Equation (7) already contains both fine-grained decay and address-specific correction. The diagonal gate $\mathbf{D}_t$ decides which key channels persist, while the rank-1 term $(\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top)$ induces stronger correction along the current write direction. GDN-2 relaxes the scalar-gate version of this coupling by separating key-side erase and value-side write gates (Hatamizadeh et al., 2026). However, the active correction remains structurally coupled to writing: the edit is still constructed from the current write key, and the correction is still committed along $\mathbf{k}_t$. Consequently, these updates can only strongly suppress memory through the address they are currently writing to.

This coupling is the core limitation we address. If stale information is stored at an address different from the current write direction, the model has no direct mechanism to remove it selectively before performing the current write. Instead, it must rely on the decay gate $\mathbf{D}_t$, which is not tied to a specific stale address, or wait until future writes happen to revisit that address. Our central design question is therefore: can a delta-rule memory model erase at one address and write at another within the same recurrent step?

3.3 ERASE-THEN-DELTA: DECOUPLED ERASE-WRITE ADDRESSING

EDA decouples cleanup from writing by inserting an independently addressed erase operator before the standard delta write:

$$\mathbf{S}_t = (\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top)(\mathbf{I} - \gamma_t \mathbf{e}_t \mathbf{e}_t^\top)\mathbf{D}_t \mathbf{S}_{t-1} + \beta_t \mathbf{k}_t \mathbf{v}_t^\top. \tag{8}$$

The factors in Eq. (8) are applied from right to left. The diagonal decay $\mathbf{D}_t$ first attenuates retained key coordinates, the erase factor $(\mathbf{I} - \gamma_t \mathbf{e}_t \mathbf{e}_t^\top)$ contracts the decayed memory along a learned cleanup address $\mathbf{e}_t$, and the usual delta factor then performs corrective forgetting and writing at the current write key $\mathbf{k}_t$. This order is part of the update rule: for diagonal decay, $\mathbf{D}_t$ generally does not commute with the rank-1 erase operator unless $\mathbf{D}_t$ degenerates to a scalar decay or $\mathbf{e}_t$ lies in an equal-decay subspace.

To see what the new operator actually erases, let

$$\widehat{\mathbf{S}}_t = \mathbf{D}_t \mathbf{S}_{t-1} \tag{9}$$

denote the memory after diagonal decay and before address-selective cleanup. EDA defines the erase address through the online objective

$$\mathcal{L}_t^{\mathrm{erase}}(\widehat{\mathbf{S}}_t) = \frac{1}{2} \|\widehat{\mathbf{S}}_t^\top \mathbf{e}_t\|^2. \tag{10}$$

This objective penalizes the content currently returned when the decayed memory is queried at $\mathbf{e}_t$. A gradient step with learning rate γ_t gives

$$\widetilde{\mathbf{S}}_t = (\mathbf{I} - \gamma_t \mathbf{e}_t \mathbf{e}_t^\top)\widehat{\mathbf{S}}_t, \tag{11}$$

where $\mathbf{e}_t$ is L2-normalized. Thus $\mathbf{e}_t$ is not merely an extra projection: it is the address whose current memory response is explicitly pushed toward zero.

This readout-level view clarifies why the new direction is more targeted than stronger decay. For any query direction $\mathbf{q}$, the erased memory reads out as

$$\widetilde{\mathbf{S}}_t^\top \mathbf{q} = \widehat{\mathbf{S}}_t^\top \mathbf{q} - \gamma_t (\mathbf{q}^\top \mathbf{e}_t) \widehat{\mathbf{S}}_t^\top \mathbf{e}_t. \tag{12}$$

When $\mathbf{q} = \mathbf{e}_t$, Eq. (12) suppresses the response at the erase address by a factor of $1 - \gamma_t$. When $\mathbf{q}$ is orthogonal to $\mathbf{e}_t$, the erase step leaves that readout unchanged before the later delta update. The decay gate $\mathbf{D}_t$ controls retention by key coordinate; in contrast, Eq. (12) subtracts the content currently returned at a learned memory address, scaled by how much the query aligns with that address.

After this cleanup, EDA applies the standard delta-style corrective write to the erased memory:

$$\mathbf{S}_t = (\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top)\widetilde{\mathbf{S}}_t + \beta_t \mathbf{k}_t \mathbf{v}_t^\top. \tag{13}$$

Substituting Eq. (11) into Eq. (13) recovers Eq. (8). The delta correction and write at $\mathbf{k}_t$ are therefore unchanged; the new degree of freedom is that stale memory can be suppressed at $\mathbf{e}_t$ before new content is written at $\mathbf{k}_t$. If $\mathbf{e}_t$ collapses to $\mathbf{k}_t$, EDA reduces to a stronger same-address correction; when the two directions differ, cleanup and writing are no longer forced to use the same address. This also distinguishes EDA from gate-level erase/write separation, where the residual can be reweighted by gates but remains organized around the current write key.

The resulting rule separates memory management into three levels of specificity: diagonal decay through $\mathbf{D}_t$, independent directional erasure through $\gamma_t \mathbf{e}_t$, and write-coupled correction through $\beta_t \mathbf{k}_t$. In this sense, EDA adds the missing degree of freedom needed to suppress stale memory at one address before performing a corrective write at another. Figure 1 illustrates the full EDA layer architecture.

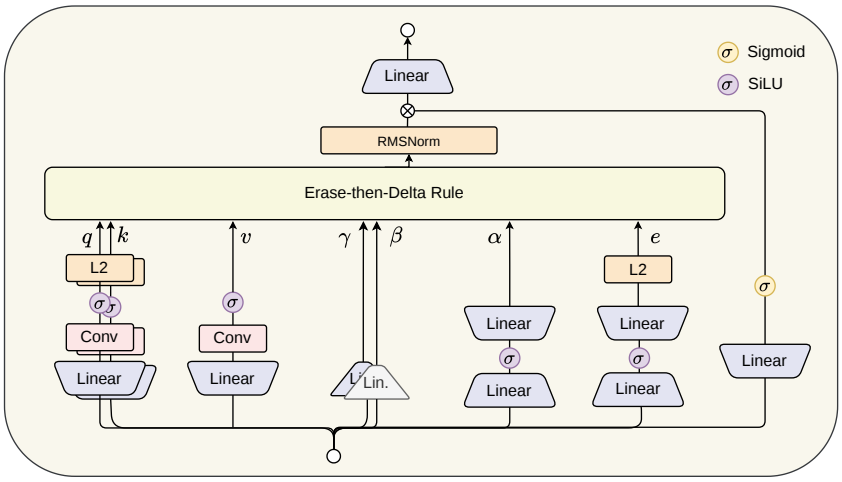

Figure 1: Architecture of an EDA layer. The input is projected into query, key, value, output gate, erase gate (γ), delta gate (β), decay parameters (α), and erase address ($\mathbf{e}$). The query, key, and erase address are L2-normalized; the erase address uses a low-rank projection. All signals feed into the EDA kernel, whose output is normalized and gated before a final linear projection.

Safe gate for bounded decay. The diagonal decay in Eq. (8) is parameterized in log space. Let $\mathbf{D}_t = \mathrm{Diag}(\exp(\boldsymbol{g}_t))$, $\mathbf{A} = \exp(\mathbf{A}_{\log}) > 0$, $\boldsymbol{u}_t = \boldsymbol{a}_t + \boldsymbol{b}_\Delta$, and $\boldsymbol{\Delta}_t = \mathrm{softplus}(\boldsymbol{u}_t)$, where $\boldsymbol{a}_t$ is

the decay projection and $\boldsymbol{b}_\Delta$ is a learned bias. The Mamba2/GDN-style log-space gate (Dao & Gu, 2024; Yang et al., 2025) uses

$$\boldsymbol{g}_t^{\log} = -\mathbf{A} \odot \boldsymbol{\Delta}_t, \tag{14}$$

which guarantees $\exp(\boldsymbol{g}_t) \leq 1$ but leaves the log-decay unbounded below. KDA computes its safe gate as $\boldsymbol{g}_t^{\mathrm{KDA}} = \ell\, \sigma(\mathbf{A} \odot \boldsymbol{u}_t)$ with $\ell < 0$, mapping each log-decay coordinate into $(\ell, 0)$ (Team et al., 2025). EDA instead uses a bounded safe gate with the same lower log-decay limit and maximum value 0:

$$\boldsymbol{g}_t = \ell + (-\ell) \exp\left(-\frac{\mathbf{A}}{|\ell|} \odot \boldsymbol{\Delta}_t \right), \tag{15}$$

where the exponential is applied elementwise. Since $\exp(-x) \in (0, 1]$ for $x \geq 0$, this parameterization keeps $\boldsymbol{g}_t \in (\ell, 0]$ and therefore bounds each decay coordinate by $\exp(\ell) < \alpha_{t,i} \leq 1$.

Comparing Eq. (14) and Eq. (15) shows why this bounded form is useful beyond numerical clipping. The Mamba2/GDN-style log-space gate separates two roles: $\mathbf{A}$ controls the decay magnitude, while $\boldsymbol{\Delta}_t = \mathrm{softplus}(\boldsymbol{u}_t)$ acts as a ReLU-like nonnegative switch for whether a coordinate should decay. Our safe gate preserves this amplitude–switch decomposition near the active region: elementwise, a Taylor expansion around $\Delta_{t,i} = 0$ gives $g_{t,i} = -A_i \Delta_{t,i} + \mathcal{O}(A_i^2 \Delta_{t,i}^2 / |\ell|)$. It therefore behaves like the log-space gate for small decay inputs, but saturates for large inputs instead of driving the log-decay toward $-\infty$. By contrast, the KDA sigmoid gate bounds the log-decay by applying a sigmoid directly to the affine decay signal, so $\mathbf{A}$ mainly changes the sigmoid slope and saturation rather than acting as a separate decay-amplitude parameter. In practice we set $\ell = -5$, making the smallest per-step decay factor $\exp(\ell) \approx 6.7 \times 10^{-3}$, well within the normal range of half-precision formats. This prevents decay factors from becoming subnormal or zero, allowing decay-weighted chunk tensors to remain in half precision and preserving Tensor-Core-friendly dense matrix multiplications.

Cross-term structure and update order. The order in Eq. (8)—erase first, then delta—is essential. Expanding the product of the two rank-1 operators reveals why:

$$(\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top)(\mathbf{I} - \gamma_t \mathbf{e}_t \mathbf{e}_t^\top) = \mathbf{I} - \gamma_t \mathbf{e}_t \mathbf{e}_t^\top - \beta_t \mathbf{k}_t \mathbf{k}_t^\top + \gamma_t \beta_t (\mathbf{k}_t^\top \mathbf{e}_t)\, \mathbf{k}_t \mathbf{e}_t^\top. \tag{16}$$

The final term—the cross-term—is proportional to the cosine similarity $c_t = \mathbf{e}_t^\top \mathbf{k}_t$ between the erase and write directions. It quantifies the "leakage" that occurs when the two directions are not orthogonal: the erase operation can influence the subsequent write-address correction through $\mathbf{k}_t \mathbf{e}_t^\top$. Reversing the order (delta first, then erase) would apply the erase operator after the write, allowing memory cleanup to suppress newly written content. By applying erasure first, our rule ensures that cleanup acts on old content before the new corrective write is committed.

When the model learns a near-orthogonal separation between $\mathbf{e}_t$ and $\mathbf{k}_t$ (mean $|c_t| \approx 0.105$, see Figure 3(c)), the cross-term becomes small and the update is well-approximated by two independent corrections acting on orthogonal subspaces. In this regime, the sequential rule is stable and the first-order effects dominate.

3.4 Parallel Implementation

EDA can reuse existing DPLR chunk-wise kernels by interleaving each original token into two consecutive sub-steps: an erase sub-step that carries the diagonal decay, followed by a delta-write sub-step with identity decay. This transformation reduces Eq. (8) to a standard DPLR recurrence over a doubled sequence without changing its recurrent semantics. The complete derivation, including the WY representation and UT transform, is provided in Appendix C.

4 Experiments

4.1 Experimental Setup

We evaluate EDA under two matched pretraining scales: a dense 2.5B model family and a larger MoE 25B-A2.8B family. The goal is to test whether the proposed erase-then-delta update improves the recurrent component in both a standard dense setting and a sparse-activated large-model setting. Within each scale, the compared models share the same training setup; detailed architecture hyperparameters and parameter counts are listed in Appendix A.

Table 1: Evaluation results after 400B-token pretraining. Values are percentages averaged over two evaluation runs; Avg. is the unweighted mean over the six benchmark columns, ranked before rounding. Within each model family, best results are bold and second-best results are underlined.

Architecture	MMLU	MMLU-Pro	GSM8K	MATH	BBH	EvalPlus	Avg.
Dense 2.5B							
Transformer	**50.11**	**18.11**	20.26	12.16	35.01	30.82	27.75
GDN	49.99	15.76	20.83	12.85	34.99	31.08	27.58
GDN-2	49.90	15.84	<u>21.40</u>	**13.56**	35.18	**32.95**	<u>28.14</u>
KDA	<u>50.03</u>	<u>16.37</u>	21.30	13.09	<u>35.31</u>	30.73	27.81
EDA	49.27	16.04	**23.90**	<u>13.54</u>	**35.42**	<u>32.50</u>	**28.45**
MoE 25B-A2.8B							
GDN	<u>64.75</u>	31.46	**58.89**	<u>31.87</u>	<u>51.88</u>	50.09	48.16
KDA	63.84	<u>33.14</u>	<u>58.61</u>	30.78	51.87	<u>51.60</u>	<u>48.31</u>
EDA	**65.31**	**33.61**	57.71	**33.57**	**52.72**	**53.37**	**49.38**

Compared models. For the dense comparison, we compare a full-attention Transformer baseline with GDN, GDN-2, KDA, and EDA. For the MoE comparison, we compare GDN, KDA, and EDA under the same sparse-activation backbone. Except for the Transformer baseline, all compared linear attention models are hybrid architectures with three linear-attention layers followed by one full-attention Transformer layer, corresponding to a 3:1 linear-to-full attention ratio. This ratio is not tuned specifically for EDA; it follows the common hybrid configuration used in Qwen3.5-style and Kimi Linear architectures (Team, 2025; Team et al., 2025).

Training setup. All models were pretrained for 400B tokens with sequence length 4096 and global batch size 1024. The dense models used a learning rate decayed from 4×10^{-3} to 3×10^{-5}, while the MoE 25B-A2.8B models used a learning rate decayed from 2×10^{-3} to 3×10^{-5}. We additionally report MoE checkpoints after an 80B-token midtraining stage initialized from the 400B-token pretrained MoE checkpoints. The midtraining stage used sequence length 32k.

Evaluation setup. For downstream evaluation, we report six benchmarks: MMLU, MMLU-Pro, GSM8K, MATH, BBH, and EvalPlus. Unless otherwise stated, entries are percentages averaged over two evaluation runs of the same checkpoint, and the Avg. column denotes the unweighted mean over the displayed benchmarks. Brief descriptions of the downstream benchmarks are provided in Appendix B.

4.2 Model Results

At the dense 2.5B scale, EDA achieves the strongest average score among all dense models. Compared with KDA, which shares the same channel-wise gated delta backbone but lacks the independent erase address, EDA increases the Avg. score from 27.81 to 28.45.

The larger MoE 25B-A2.8B setting gives a clearer picture of the scaling behavior. EDA achieves the highest average score and leads on five of the six benchmarks. This larger-scale result suggests that address-level erase/write decoupling provides a broadly useful memory-management degree of freedom: the model can preserve the delta-rule correction at the current write key while using a separate learned address to suppress stale content elsewhere.

4.3 Midtraining Results

Table 2 reports the same benchmark suite after the MoE 25B-A2.8B checkpoints were further trained for 80B tokens at 32k sequence length.

Midtraining tests whether the pretraining-stage advantage survives a harder adaptation setting rather than only appearing at the original 4k training length. After the 80B-token long-context stage, EDA continues to achieve the highest average score, with the best results on MMLU, MMLU-Pro, MATH, and BBH. This persistence is important because long-context midtraining changes the operating

Table 2: Evaluation results for MoE 25B-A2.8B checkpoints after 400B-token pretraining followed by 80B-token midtraining at 32k sequence length. Values are percentages averaged over two evaluation runs; Avg. is the unweighted mean over the six benchmark columns, ranked before rounding. Best results are bold and second-best results are underlined.

Architecture	MMLU	MMLU-Pro	GSM8K	MATH	BBH	EvalPlus	Avg.
GDN	67.43	40.55	75.93	45.94	64.25	50.09	57.37
KDA	67.32	40.60	**76.21**	46.87	65.04	**53.82**	58.31
EDA	**68.12**	**41.71**	75.99	**49.28**	**65.81**	51.78	**58.78**

Table 3: RULER (Hsieh et al., 2024) long-context results for MoE 25B-A2.8B checkpoints after 400B-token pretraining and 80B-token midtraining at 32k sequence length. Values are percentages averaged over four evaluation runs; 64k and 128k are length-extrapolation settings. Avg. is the unweighted mean over the six displayed context lengths. Best results are bold and second-best results are underlined.

Model	4k	8k	16k	32k	64k	128k	Avg.
GDN	92.40	90.09	87.28	**82.15**	67.62	**45.16**	77.45
KDA	93.33	89.29	85.13	79.66	**72.00**	42.70	77.02
EDA	**93.84**	**90.88**	**87.55**	81.52	71.90	44.22	**78.32**

regime of the recurrent state: the model must maintain useful information over longer spans while still removing outdated content that can interfere with later reads.

Combined with the 400B-token pretraining results, the midtraining result strengthens the main conclusion: decoupling erase and write addresses remains useful after the model is further trained for longer contexts, suggesting that the erase path is compatible with, rather than fragile under, subsequent sequence-length adaptation.

4.4 Long-Context Evaluation

We evaluate the midtrained MoE checkpoints on the RULER task from 4k to 128k context length. Since midtraining used 32k sequences, the 64k and 128k settings evaluate length extrapolation beyond the training context. Table 3 reports the RULER score at each context length, aggregated over all sub-tasks and four evaluation runs. EDA outperforms both GDN and KDA in the short-context regime from 4k to 16k, and remains close to the two baselines from 32k to 128k.

4.5 Memory-State Analysis

We analyze gate allocation, raw write-key recall, counterfactual erase directions, and erase/write geometry in Appendix D. Independent erase rises to 69.1% of the contraction-strength allocation for nearly persistent heads, while alternative erase directions increase local readout perturbation by 1.9–3.4$\times$ and the learned erase/write cosine remains near the independent-direction reference. These results support a conditional cleanup mechanism rather than uniformly better historical recall (Figure 3).

4.6 Kernel Efficiency

We compare our EDA kernels with FLA KDA (v0.5.1) and FlashAttention-3 (FA3), a causal softmax reference, using 32 query/key/value heads and $d_k = d_v = 128$. Training/prefill sweep 2k–128k tokens at batch size 1; decoding uses batch sizes 1–32. Inputs are bfloat16. Prefill returns EDA/FLA final states or retains FA3's bfloat16 KV cache. External normalization, convolution, activation, and projection are excluded. Benchmarks were run on NVIDIA H20 GPUs.

EDA has higher training/prefill latency than FLA and lower measured decoding latency, especially at small batches. FA3 is faster at 2k; EDA has lower training/prefill latency from 4k onward. These are isolated core-operator costs. These kernel-level differences have a smaller end-to-end impact when

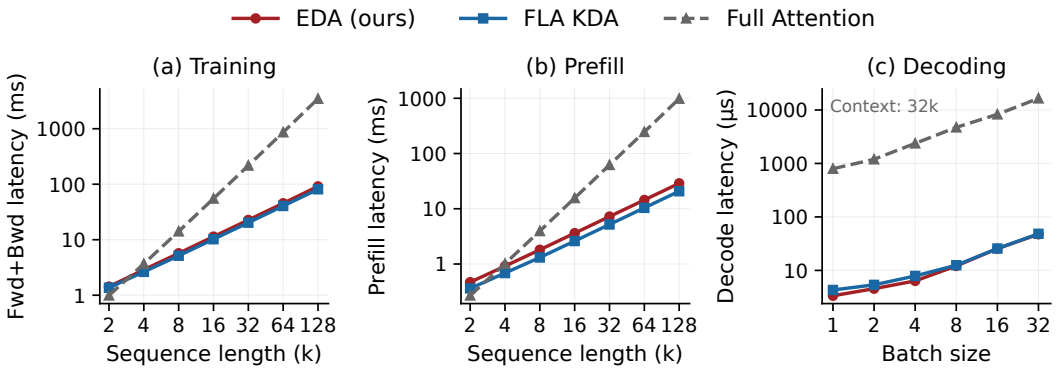

Figure 2: **Kernel latency; lower is better.** (a) Training (forward+backward). (b) Prefill. (c) Single-token decoding with a 32k context. Means over 100 warm-cache CUDA-graph/event samples after 60 warmups.

linear attention accounts for only a small fraction of total runtime, which also includes MoE/MLP blocks and full-attention layers in hybrid models.

5 Related Work

Delta-rule and gated linear memory models. DeltaNet (Schlag et al., 2021; Yang et al., 2024) replaces additive writes with corrections conditioned on memory at the current address, and GDN adds a head-wise forget gate (Yang et al., 2025); channel-wise variants further use diagonal decay (Team et al., 2025). GDN-2 is the closest motivation-side comparison, but it separates key-side erase and value-side write strengths within a residual whose active edit remains at the current write key (Hatamizadeh et al., 2026). EDA instead preserves the corrective write at $\mathbf{k}_t$ while erasing at an independently selected address: GDN-2 decouples how strongly to erase and write, whereas EDA decouples where they occur.

Broader linear-recurrent and hybrid context. DeltaProduct, RWKV-7, and Comba enrich linear-RNN transitions through multiple reflections, diagonal-plus-low-rank gating, or closed-loop output correction (Siems et al., 2026; Peng et al., 2025; Hu et al., 2026). EDA instead adds the specific capability to erase at one address before writing at another. This update is orthogonal to hybrid layouts such as Jamba and Nemotron, which interleave recurrent and full-attention layers (Lieber et al., 2024; Gu et al., 2025), and can replace the channel-wise gated delta component without changing the hybrid schedule.

6 Conclusion

We introduced Erase-then-Delta Attention, which preserves the corrective delta write at the current key while first erasing stale content at an independently selected address. This adds targeted cleanup without changing the role of the original delta-rule update.

EDA achieves the strongest average performance across dense 2.5B and MoE 25B-A2.8B pretraining, with the gain persisting after long-context midtraining. Memory-state probes show that erase is used most when passive decay is weak and that counterfactual directions disturb the readout more, supporting the address-decoupling interpretation.

Limitations. EDA reduces raw write-key recall, and the isolated-kernel microbenchmarks do not establish end-to-end speedups. The current probes also do not causally link individual erase events to prediction improvements.

AI USE STATEMENT

We used generative AI tools to assist with language polishing of the paper and EDA kernel development. The authors take responsibility for the final content of this work.

REFERENCES

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael K Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xLSTM: Extended long short-term memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=ARAxPPIAhq.

Karl Cobbe, Vineet Kosaraju, Mo Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *ArXiv*, abs/2110.14168, 2021. URL https://api.semanticscholar.org/CorpusID:239998651.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=ztn8FCR1td.

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

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Yuxian Gu, Qinghao Hu, Haocheng Xi, Junyu Chen, Shang Yang, Song Han, and Han Cai. Jet-nemotron: Efficient language model with post neural architecture search. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=WZQXaTNYEB.

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Xiaodong Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *ArXiv*, abs/2009.03300, 2020. URL https://api.semanticscholar.org/CorpusID:221516475.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

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

Jiaxi Hu, Yongqi Pan, Jusen Du, Disen Lan, Xiaqiang Tang, Qingsong Wen, Yuxuan Liang, and Weigao Sun. Improving bilinear RNN with closed-loop control. In *Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=jlJaRXDzCE.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and Franccois Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, 2020. URL https://api.semanticscholar.org/CorpusID:220250819.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. Jamba: A hybrid transformer-mamba language model, 2024. URL https://arxiv.org/abs/2403.19887.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=1qvx610Cu7.

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, Nathan Wilce, Johan S. Wind, Tianyi Wu, Daniel Wuttke, and Christian Zhou-Zheng. Rwkv-7 "goose" with expressive dynamic state evolution, 2025. URL https://arxiv.org/abs/2503.14456.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=1b7whO4SfY.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, 2021. URL https://api.semanticscholar.org/CorpusID:235377069.

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. Deltaproduct: Improving state-tracking in linear RNNs via householder products. In *Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=SoRiaijTGr.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *ArXiv*, abs/2307.08621, 2023. URL https://api.semanticscholar.org/CorpusID:259937453.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

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

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

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. URL https://api.semanticscholar.org/CorpusID:13756489.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *ArXiv*, abs/2006.04768, 2020. URL https://api.semanticscholar.org/CorpusID:219530577.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=y10DM6R2r3.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=y8Rm4VNRPH.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=r8H7xhYPwz.

A Model Configurations

The model configurations used in the evaluation are summarized in Tables 4 and 5. All evaluated models use the same vocabulary size (248,320); pretraining used 4096-token sequences, and the MoE midtraining stage used 32k-token sequences. Training used bfloat16 with the AdamW optimizer, SiLU activations in the FFN/MoE blocks, and RMSNorm with $\epsilon = 10^{-6}$. The hybrid models use one full-attention Transformer layer in every four layers, placed after three linear-attention layers. The full-attention layers in both the Transformer baseline and the hybrid models use Gated Attention (Qiu et al., 2025). For parameter alignment, the dense Transformer baseline uses 8/4/4 query/key/value heads in its full-attention layers.

Table 4: Scale-level architecture hyperparameters. "Layers" reports total layers with linear/full-attention counts in parentheses. "Attn/KV" denotes the query and key/value head counts in hybrid full-attention layers. "LA K/V" denotes the number of key/value heads in the linear-attention layers, and "LA dim" denotes their per-head dimensions. For the MoE scale, the FFN/expert column expert width.

Scale	Layers	d_{model}	Attn/KV	LA K/V	LA dim	FFN/expert	MoE routing
Dense 2.5B	24 (18/6)	2048	8/2	8/16	128/128	7424–7488	–
MoE 25B-A2.8B	28 (21/7)	2048	16/2	16/32	128/128	512	256 experts, top-8 activated + 1 shared

Table 5: Total and active parameter counts for evaluated model variants. Dense models activate all parameters; MoE models report both total parameters and the parameters active per token.

Scale	Model	Total params	Active params
Dense 2.5B	Transformer	2.5052B	2.5052B
Dense 2.5B	GDN-2	2.5353B	2.5353B
Dense 2.5B	GDN	2.5035B	2.5035B
Dense 2.5B	KDA	2.5218B	2.5218B
Dense 2.5B	EDA	2.5295B	2.5295B
MoE 25B-A2.8B	GDN	24.5676B	2.7236B
MoE 25B-A2.8B	KDA	24.6324B	2.7885B
MoE 25B-A2.8B	EDA	24.6558B	2.8119B

For parameter efficiency, variants with channel-wise forget gates use rank-16 (per-head) low-rank projections for the gate generator. The EDA MoE configuration uses a rank-16 (per-head) erase-address projection and a safe gate with lower bound -5.

B Evaluation Benchmarks

We evaluate checkpoints on six language-model benchmarks. MMLU measures multitask knowledge across academic and professional subjects (Hendrycks et al., 2020), and MMLU-Pro increases the difficulty with more challenging questions and larger answer sets (Wang et al., 2024). GSM8K evaluates grade-school mathematical reasoning with word problems (Cobbe et al., 2021), while MATH evaluates competition-style mathematical problem solving (Hendrycks et al., 2021). BBH covers difficult reasoning tasks selected from BIG-Bench (Suzgun et al., 2023). EvalPlus evaluates Python code generation with stricter test cases beyond the original HumanEval/MBPP checks (Liu et al., 2023).

C Chunk-wise Parallelization

C.1 EDA with Chunk-wise Parallel

Referring to Eq. (8), the EDA state is multiplied by two rank-1 correction factors per step. To reuse existing DPLR chunk-wise kernels, we interleave the erase and delta sub-steps into a doubled sequence. Let

$$[\mathbf{q}'_\tau, \mathbf{k}'_\tau, \mathbf{v}'_\tau, \beta'_\tau, \boldsymbol{\alpha}'_\tau] = \begin{cases} [\mathbf{0}, \mathbf{e}_t, \mathbf{0}, \gamma_t, \boldsymbol{\alpha}_t], & \tau = 2t-1 \\ [\mathbf{q}_t, \mathbf{k}_t, \mathbf{v}_t, \beta_t, \mathbf{1}], & \tau = 2t \end{cases} \tag{17}$$

Each original step t maps to two sub-steps: the odd sub-step applies the erase operator with decay, and the even sub-step applies the delta correction with identity decay. Here $\mathbf{D}'_\tau = \mathrm{Diag}(\boldsymbol{\alpha}'_\tau)$ and $\boldsymbol{\alpha}_t = \exp(\boldsymbol{g}_t)$, so the corresponding log gates are $\boldsymbol{g}_t$ and $\mathbf{0}$. We can rewrite Eq. (8) as:

$$\begin{aligned} \mathbf{S}_t &= (\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top)(\mathbf{I} - \gamma_t \mathbf{e}_t \mathbf{e}_t^\top)\mathbf{D}_t \mathbf{S}_{t-1} + \beta_t \mathbf{k}_t \mathbf{v}_t^\top \\ &= (\mathbf{I} - \beta_t \mathbf{k}_t \mathbf{k}_t^\top)\mathbf{I}\left((\mathbf{I} - \gamma_t \mathbf{e}_t \mathbf{e}_t^\top)\mathbf{D}_t \mathbf{S}_{t-1} + 0\right) + \beta_t \mathbf{k}_t \mathbf{v}_t^\top \\ &= (\mathbf{I} - \beta'_{2t} \mathbf{k}'_{2t} \mathbf{k}'^\top_{2t})\mathbf{D}'_{2t}\left((\mathbf{I} - \beta'_{2t-1} \mathbf{k}'_{2t-1} \mathbf{k}'^\top_{2t-1})\mathbf{D}'_{2t-1} \mathbf{S}_{t-1} + \beta'_{2t-1} \mathbf{k}'_{2t-1} \mathbf{v}'^\top_{2t-1}\right) + \beta'_{2t} \mathbf{k}'_{2t} \mathbf{v}'^\top_{2t} \end{aligned} \tag{18}$$

Consider one chunk of t original tokens, reindexed locally, with $n = 2t$ sub-steps and incoming state $\mathbf{S}_0$. Let $\mathbf{S}'_j$ denote the state after sub-step j, so $\mathbf{S}'_{2i} = \mathbf{S}_i$. Products of transition matrices are time ordered: $\overleftarrow{\prod}_{j=a}^{b} \mathbf{F}_j = \mathbf{F}_b \cdots \mathbf{F}_a$, with an empty product equal to $\mathbf{I}$. Expanding the recurrence gives

$$\mathbf{S}_t = \underbrace{\left(\overleftarrow{\prod_{i=1}^{n}} \left(\mathbf{I} - \beta'_i \mathbf{k}'_i \mathbf{k}'^\top_i\right) \mathbf{D}'_i\right)}_{:=\mathbf{P}} \mathbf{S}_0 + \underbrace{\sum_{i=1}^{n} \left(\overleftarrow{\prod_{j=i+1}^{n}} \left(\mathbf{I} - \beta'_j \mathbf{k}'_j \mathbf{k}'^\top_j\right) \mathbf{D}'_j\right) \beta'_i \mathbf{k}'_i \mathbf{v}'^\top_i}_{:=\mathbf{H}}. \tag{19}$$

Define $\mathbf{D}'_{a\to b} = \prod_{r=a}^{b} \mathbf{D}'_r$, with $\mathbf{D}'_{a\to b} = \mathbf{I}$ when $a > b$. Following the chunk-wise algorithm of KDA (Team et al., 2025), the **WY representation** is obtained from the following recursions for every sub-step $j = 1, \ldots, n$:

$$
\begin{aligned}
\mathbf{w}_j &= \beta'_j \left(\mathbf{D}'_{1\to j}\mathbf{k}'_j - \sum_{i=1}^{j-1} \mathbf{w}_i \left(\mathbf{k}_i'^{\top} \mathbf{D}'_{i+1\to j}\mathbf{k}'_j \right) \right), \\
\mathbf{u}_j &= \beta'_j \left(\mathbf{v}'_j - \sum_{i=1}^{j-1} \mathbf{u}_i \left(\mathbf{k}_i'^{\top} \mathbf{D}'_{i+1\to j}\mathbf{k}'_j \right) \right), \\
\mathbf{P} &= \mathbf{D}'_{1\to n} - \sum_{i=1}^{n} \mathbf{D}'_{i+1\to n}\mathbf{k}'_i\mathbf{w}_i^{\top}, \\
\mathbf{H} &= \sum_{i=1}^{n} \mathbf{D}'_{i+1\to n}\mathbf{k}'_i\mathbf{u}_i^{\top}.
\end{aligned}
\tag{20}
$$

The propagation from sub-step i starts with $\mathbf{D}'_{i+1}$ because that sub-step's rank-one update is added after applying $\mathbf{D}'_i$.

For the **UT transform**, stack the token vectors as rows of $\mathbf{Q}', \mathbf{K}' \in \mathbb{R}^{n\times d_k}$ and $\mathbf{V}' \in \mathbb{R}^{n\times d_v}$. Likewise, $\mathbf{W}$ and $\mathbf{U}$ have rows $\mathbf{w}_j^{\top}$ and $\mathbf{u}_j^{\top}$. Let $\boldsymbol{\beta}' = (\beta'_1, \ldots, \beta'_n)^{\top}$. The prefix and suffix decay matrices $\mathbf{A}, \mathbf{A}_{\mathrm{tail}} \in \mathbb{R}^{n\times d_k}$ and the triangular transform are

$$
\begin{aligned}
\mathbf{A}_{j,:} &= \mathrm{diag}(\mathbf{D}'_{1\to j})^{\top}, \quad (\mathbf{A}_{\mathrm{tail}})_{j,:} = \mathrm{diag}(\mathbf{D}'_{j+1\to n})^{\top}, \\
\mathbf{X} &= \mathbf{A} \odot \mathbf{K}', & \mathbf{Y} &= \frac{\mathbf{K}'}{\mathbf{A}}, \\
\mathbf{B} &= \mathrm{Diag}(\boldsymbol{\beta}'), & \mathbf{L} &= \mathrm{StrictTril}(\mathbf{B}\mathbf{X}\mathbf{Y}^{\top}), \\
\mathbf{M} &= (\mathbf{I} + \mathbf{L})^{-1}\mathbf{B}, & \mathbf{W} &= \mathbf{M}\mathbf{X}, \qquad \mathbf{U} = \mathbf{M}\mathbf{V}'.
\end{aligned}
\tag{21}
$$

Here $j = 1, \ldots, n$, diag extracts a diagonal as a column vector, and division is elementwise. All decay coordinates are positive, so the division is well-defined. The operator $\mathrm{StrictTril}$ excludes the diagonal; hence $\mathbf{I} + \mathbf{L}$ is unit lower triangular and the transform can be evaluated by triangular solves.

Finally, the chunk state and doubled-sequence readouts are

$$
\begin{aligned}
\mathbf{S}_t &= \mathbf{D}'_{1\to n}\mathbf{S}_0 + (\mathbf{A}_{\mathrm{tail}} \odot \mathbf{K}')^{\top}(\mathbf{U} - \mathbf{W}\mathbf{S}_0), \\
\mathbf{O}' &= (\mathbf{A} \odot \mathbf{Q}')\mathbf{S}_0 + \mathrm{Tril}\left((\mathbf{A} \odot \mathbf{Q}')\mathbf{Y}^{\top}\right)(\mathbf{U} - \mathbf{W}\mathbf{S}_0).
\end{aligned}
\tag{22}
$$

The operator Tril includes the diagonal, so each readout uses the state after its sub-step. The matrix $\mathbf{O}' \in \mathbb{R}^{n\times d_v}$ has zero odd rows, and the original token outputs are its even rows: $\mathbf{o}_i^{\top} = \mathbf{O}'_{2i,:}$. The kernel's default readout scale $d_k^{-1/2}$ multiplies these outputs without affecting the state recurrence. This formulation reduces EDA's two-factor update to the standard DPLR chunk-wise recurrence.

D Memory-State Probe Definitions

This section provides the complete definitions and protocols for the analyses summarized in Section 4.5.

The benchmark gains above do not by themselves explain why an additional erase address helps, since the delta update already has two ways to reduce old content: diagonal decay $\mathbf{D}_t = \mathrm{Diag}(\boldsymbol{\alpha}_t)$ and write-coupled correction $(\mathbf{I} - \beta_t\mathbf{k}_t\mathbf{k}_t^{\top})$. Throughout this subsection we analyze a fixed layer and attention head unless stated otherwise: $\mathbf{k}_t \in \mathbb{R}^{d_k}$ is the L2-normalized write key at token t, d_k is the per-head key dimension, $\mathbf{I} \in \mathbb{R}^{d_k\times d_k}$ is the identity matrix, $\boldsymbol{\alpha}_t \in (0,1]^{d_k}$ is the per-channel retention vector, and $\beta_t \in (0,1)$ is the delta correction gate. We therefore ask a narrower mechanistic question: when the recurrent state must remove stale content, does the model use the new erase address in a way that cannot be explained by these two existing contraction paths alone?

We first measure a *gate-strength allocation*, not exact removed state energy. Recall that the diagonal decay is parameterized in log space as $\mathbf{D}_t = \mathrm{Diag}(\exp(\boldsymbol{g}_t))$, where $\boldsymbol{g}_t \in \mathbb{R}^{d_k}$ is the per-channel log-retention vector. Therefore $\boldsymbol{\alpha}_t = \exp(\boldsymbol{g}_t)$ is the per-channel retention factor applied before the erase and delta operators. For token t and head h, let

$$\bar{\alpha}_{t,h} = \frac{1}{d_k} \sum_{j=1}^{d_k} \alpha_{t,h,j} \tag{23}$$

be the mean retention factor of the diagonal decay within that head, where $\alpha_{t,h,j}$ is the j-th key-channel retention value and the sum averages over all d_k key channels. Below we write $\alpha = \bar{\alpha}_{t,h}$ for compactness. This averaging deliberately collapses the diagonal operator to a scalar summary, so it should not be used to compare the full operator rank or total energy removed by $\mathbf{D}_t$ against the two rank-1 contractions. Its purpose is narrower: under the average retained scale of a head, we ask how the learned gates allocate contraction strength among the decay path, the write-key correction, and the independent erase path. Since both rank-1 operators act after $\mathbf{D}_t$, we define the unnormalized scores

$$b_D = 1 - \alpha, \qquad b_\Delta = \alpha\beta_t, \qquad b_E = \alpha\gamma_t, \tag{24}$$

for diagonal decay, same-address correction, and independent erase, respectively; here $\gamma_t \in (0, 1)$ is the erase gate. For readability, after fixing head h, we omit the head index on $\beta_{t,h}$ and $\gamma_{t,h}$ and write them as β_t and γ_t. Here $1 - \alpha$ is the average decay removal fraction obtained after summarizing the diagonal retention vector by its mean, rather than an exact operator-level decomposition of $\mathbf{D}_t$. We plot the normalized share $q_m = b_m/(b_D + b_\Delta + b_E)$ for mechanism $m \in \{D, \Delta, E\}$. This definition is appropriate for the allocation question because β_t and γ_t are exactly the contraction factors of the two rank-1 readouts, while the multiplier α accounts for the fact that both contractions operate on the state retained after decay. It should not be read as an exact or fully fair state-energy decomposition: the actual content removed also depends on the anisotropic diagonal decay, the current state projections onto $\mathbf{e}_t$ and $\mathbf{k}_t$, and the overlap between the two addresses.

As a boundary check, we also evaluate raw write-key recall, which asks whether older hidden values can be read back from their original write keys. KDA performs better under this strict probe, so EDA's advantage should not be interpreted as uniformly better historical recall. This motivates focusing on cleanup allocation and erase-address structure rather than raw recall alone.

To test whether the learned erase direction is structured, we use two address-level diagnostics. First, we replay the same recurrent states and gates while replacing the actual erase address $\mathbf{e}_t$ with a random unit vector, a head-shuffled learned direction, or the same-address choice $\mathbf{e}_t = \mathbf{k}_t$. Let $\mathbf{o}_t^-$ be the readout immediately before erase and $\delta\mathbf{o}_t$ the erase-induced change. We measure the collateral perturbation $\|\delta\mathbf{o}_t\|_2/\|\mathbf{o}_t^-\|_2$; the raw means are 0.064 for Actual, 0.143 for Random, 0.115 for Shuffle, and 0.223 for $\mathbf{e}_t = \mathbf{k}_t$. Figure 3(b) reports layerwise fold changes relative to Actual. Under the same erase-gate budget, a lower value indicates less disruption to the currently readable state, not an absence of erasure. Second, on GSM8K few-shot prompts we measure $|\cos(\mathbf{e}_t, \mathbf{k}_t)|$ and compare it with the analytic mean for independent random unit directions in $\mathbb{R}^{128}$, matching the per-head key dimension.

Figure 3 summarizes the results. Diagonal decay, same-address correction, and independent erase account for 35.4%, 31.8%, and 32.8% of the global contraction-strength allocation, respectively. When $\bar{\alpha} < 0.3$, decay already supplies most of the contraction, whereas for nearly persistent heads with $\bar{\alpha} \geq 0.9$, independent erase contributes 69.1% and is about $3.0\times$ the same-address correction contribution. This is the regime in which stale content would otherwise persist, so the model assigns the additional cleanup budget to $\gamma_t\mathbf{e}_t$ rather than forcing it through the current write key.

The learned erase direction is also structured at the readout level. Replacing $\mathbf{e}_t$ with random, shuffled, or same-address alternatives increases local readout perturbation by about $2.4\times$, $1.9\times$, and $3.4\times$, respectively, after normalizing each layer by its Actual score. The observed mean $|\cos(\mathbf{e}_t, \mathbf{k}_t)|$ stays around 0.105 across layers, close to the independent-direction reference and far from same-address collapse. Together with the raw-recall boundary check, these probes support the address-decoupling interpretation while clarifying that independent erase is a conditional cleanup mechanism, not a uniformly better historical-recall mechanism.

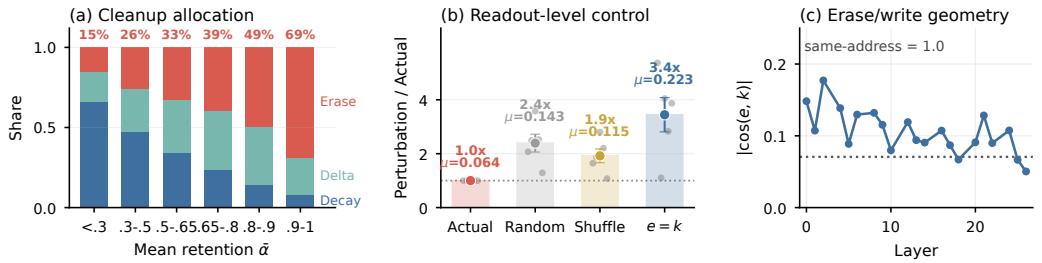

Figure 3: **EDA uses an independent cleanup path.** (a) Gate-strength allocation by mean-retention bin. Independent erase becomes dominant when decay is weak ($\bar{\alpha}$ close to one); red percentages above bars denote the erase share. (b) Under the same erase gates, counterfactual erase directions cause larger local readout perturbations than the learned direction; bars show layerwise fold change relative to Actual, and μ denotes the raw mean perturbation score. (c) The erase address stays close to the independent-direction reference; same-address collapse would give $|\cos(e, k)| = 1$.