GATED SLOT ATTENTION-2: TWO-SIDED ASSOCIATIVE MEMORY CORRECTION IN LINEAR ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Linear attention models have emerged as efficient alternatives to standard attention, but effectively managing their fixed-size recurrent memory remains challenging. To improve memory, recent work has explored two distinct directions: delta-rule variants for precise correction of values associated with keys, and slot-based architectures such as Gated Slot Attention for modeling key and value memories in two stages. We observe that these directions are complementary–the delta rule provides effective memory correction, while the two-stage structure provides a natural way to operate on both sides of an association. Building on this insight, we introduce a new Gated Oja Rule[1] for key-side correction and extend it with decoupled erase and write control to obtain Gated Oja Rule-2. We then introduce Gated Slot Attention-2 (GSA2), which combines Gated Oja Rule-2 for key-side correction with Gated Delta Rule-2 for value-side correction through shared latent slots. We further derive a hardware-efficient chunkwise algorithm for parallel training. Experiments demonstrate that GSA2 consistently improves over strong linear-attention baselines across benchmarks while retaining linear-time sequence modeling and constant-memory recurrent decoding.

1 INTRODUCTION

The Transformer (Vaswani et al., 2017) has become the dominant architecture for large language models and sequence modeling due to the strong expressiveness of self-attention and its natural compatibility with parallel training. However, the computational cost of self-attention grows quadratically with sequence length, while the key-value cache grows linearly during autoregressive decoding. Linear attention (Katharopoulos et al., 2020) offers an appealing alternative by compressing the entire history into a fixed-size recurrent state, enabling linear-time sequence modeling and constant-memory recurrent inference. The central challenge, however, is how to effectively organize and update information within this compressed memory.

Recent progress on linear attention has improved fixed-state memory through data-dependent decay, selective gating, and increasingly expressive state transitions (Sun et al., 2023; Yang et al., 2023; Gu & Dao, 2023; Qin et al., 2024; Dao & Gu, 2024; Hu et al., 2025; Lahoti et al., 2026). Within this landscape, Gated Slot Attention (GSA) (Zhang et al., 2024) stands out as an elegant variant: rather than refining the update of a single recurrent state, it maintains separate key and value memories over a shared set of latent slots. This slot-based factorization provides a natural interface for operating on both sides of an association while retaining a compact recurrent memory.

In parallel, another line of work has focused on improving the memory update itself, with delta-rule models providing an effective mechanism for explicit associative correction. DeltaNet (Yang et al., 2024) corrects the value associated with the current key, Gated DeltaNet (GDN) (Yang et al., 2025) adds adaptive forgetting, Kimi Delta Attention (KDA) (Team et al., 2025) extends decay to the channel level, and Gated DeltaNet-2 (GDN2) (Hatamizadeh et al., 2026) further decouples erasing old content from writing new content. Despite these advances, this line still centers on a single key-addressed associative state whose correction primarily acts on the value side.

[1] Oja's rule is a normalized Hebbian learning rule that augments Hebbian association with a self-normalizing correction term (Oja, 1982); our gated Oja rule adapts this structure to recurrent associative memory.

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.

We observe that these two directions are complementary. GSA provides an explicit two-stage structure spanning both sides, but lacks similarly expressive correction rules; Delta-rule models provide powerful error-correcting memory updates, but primarily correct the value side of an association. This suggests a simple question:

Can we equip both stages of a slot-based associative memory with complementary error-correcting updates?

Building on this insight, we first introduce the **Gated Oja Rule (OJA)**, inspired by Oja's normalized Hebbian update (Oja, 1982), to provide the complementary key-side correction. We then decouple its erase and write operations, yielding **Gated Oja Rule-2 (OJA2)**. Finally, we introduce **Gated Slot Attention-2 (GSA2)**, which combines Gated Oja Rule-2 with the Gated Delta Rule-2 of GDN2 through shared latent slots. The first stage associates keys with latent slots and applies Gated Oja Rule-2 to correct the key side of the association, while the second associates the same slots with values and applies Gated Delta Rule-2 to correct the value side.

Let $\boldsymbol{k}_t$, $\boldsymbol{w}_t$, and $\boldsymbol{v}_t$ denote the key, shared slot-space vector, and value at step t, respectively. GSA2 can then be summarized as

$$\boldsymbol{k}_t \xleftarrow[\text{key-side correction}]{\text{Gated Oja Rule-2}} \boldsymbol{w}_t \xrightarrow[\text{value-side correction}]{\text{Gated Delta Rule-2}} \boldsymbol{v}_t.$$

Compared with GSA, GSA2 retains the compact two-stage memory structure and gated memory control while introducing explicit, side-specific associative correction.

The resulting recurrence remains compatible with efficient parallel training. We reuse the chunkwise formulation of Gated Delta Rule-2 from GDN2 (Hatamizadeh et al., 2026) and derive a corresponding WY/UT-style formulation for Gated Oja Rule-2. Both exploit low-rank state transformations so that intra-chunk computation remains dominated by dense matrix multiplications (Bischof & Van Loan, 1987; Joffrain et al., 2006). GSA2 therefore retains linear-time sequence modeling and constant-size recurrent memory while supporting efficient parallel training.

Empirically, GSA2 performs strongly across language modeling, commonsense reasoning, retrieval, and long-context benchmarks, with particularly clear gains on RULER and LongBench. Ablations show that pairing key-side Oja correction with value-side Delta correction outperforms correcting either side alone or applying the same correction rule to both memories, with the latter resembling multi-stage fast-weight architectures such as Delta MLPs (Irie et al., 2021) and TRELLIS (Karami et al., 2025). GSA2 also preserves favorable recurrent length scaling with only a modest constant throughput overhead.

2 Preliminaries

2.1 Delta-Rule Associative Memory

Additive linear attention continuously superposes new associations in a fixed-size state, which can cause interference as memory becomes crowded. DeltaNet (Yang et al., 2024) instead explicitly corrects the value associated with the current key:

$$\begin{aligned}\mathbf{S}_t &= \mathbf{S}_{t-1} + \beta_t\big(\boldsymbol{v}_t - \mathbf{S}_{t-1}\boldsymbol{k}_t\big)\boldsymbol{k}_t^\top \\ &= \mathbf{S}_{t-1}\big(\mathbf{I} - \beta_t\boldsymbol{k}_t\boldsymbol{k}_t^\top\big) + \beta_t\boldsymbol{v}_t\boldsymbol{k}_t^\top.\end{aligned} \tag{1}$$

where β_t controls the update strength and $\boldsymbol{v}_t - \mathbf{S}_{t-1}\boldsymbol{k}_t$ is the value-prediction error under key $\boldsymbol{k}_t$. Equivalently, Eq. 1 is an online gradient step on

$$\mathcal{L}_t^{\Delta}(\mathbf{S}) = \frac{1}{2}\left|\boldsymbol{v}_t - \mathbf{S}\boldsymbol{k}_t\right|_2^2. \tag{2}$$

Thus, the delta rule uses the key to address memory and corrects its associated value.

Subsequent variants improve this update through adaptive forgetting and finer-grained memory control. In particular, GDN2 (Hatamizadeh et al., 2026) decouples erasing old content from writing new content. Under the state orientation used in this paper, its channel-wise form is

$$\mathbf{S}_t = \mathbf{S}_{t-1}\,\mathrm{Diag}(\alpha_t)\left(\mathbf{I} - (\boldsymbol{b}_t \odot \boldsymbol{k}_t)\boldsymbol{k}_t^\top\right) + (\boldsymbol{c}_t \odot \boldsymbol{v}_t)\boldsymbol{k}_t^\top, \tag{3}$$

where $\boldsymbol{b}_t$ controls erasing along the key-side axis and $\boldsymbol{c}_t$ controls writing along the value-side axis. We reserve $\boldsymbol{w}_t$ for the latent-slot representation introduced later.

2.2 Gated Slot Attention as Two-Stage Associative Memory

GSA (Zhang et al., 2024) factorizes a key–value association through M latent slots. Using our state orientation, $\mathbf{S}_t^{(1)} \in \mathbb{R}^{M \times D_k}$ stores key–slot associations, while $\mathbf{S}_t^{(2)} \in \mathbb{R}^{D_v \times M}$ stores slot–value associations. With slot-wise forgetting gate $\alpha_t \in (0,1)^M$, GSA updates the two memories as

$$\mathbf{S}_t^{(1)} = \mathrm{Diag}(\alpha_t)\mathbf{S}_{t-1}^{(1)} + (\mathbf{1} - \alpha_t)\boldsymbol{k}_t^\top, \tag{4}$$

$$\mathbf{S}_t^{(2)} = \mathbf{S}_{t-1}^{(2)}\,\mathrm{Diag}(\alpha_t) + \boldsymbol{v}_t(\mathbf{1} - \alpha_t)^\top. \tag{5}$$

A query is first mapped to the latent-slot space and then decoded through the value memory:

$$\boldsymbol{o}_t = \mathbf{S}_t^{(2)}\,\mathrm{softmax}\left(\mathbf{S}_t^{(1)}\boldsymbol{q}_t\right). \tag{6}$$

This factorization provides an explicit intermediate slot space between keys and values while retaining compact recurrent memory. Unlike delta-rule models, however, GSA updates both memories through gated accumulation rather than explicit error correction. This distinction motivates the complementary key- and value-side corrections developed next.

3 Gated Slot Attention-2

GSA2 combines a complementary key-side Gated Oja Rule-2 correction with the value-side Gated Delta Rule-2 correction within the two-stage slot memory of GSA. We first derive the key-side update and then instantiate the two corrections through a shared latent-slot representation.

3.1 Gated Oja Rule: Key-Side Correction

The delta rule in Eq. 1 corrects the value predicted from a key. We consider the complementary problem: correcting the key predicted from a value. Specifically, we optimize

$$\mathcal{L}_t^K(\boldsymbol{S}) = \frac{1}{2}\left\|\boldsymbol{k}_t - \boldsymbol{S}^\top\boldsymbol{v}_t\right\|_2^2. \tag{7}$$

One gradient step gives

$$\begin{aligned}\boldsymbol{S}_t &= \boldsymbol{S}_{t-1} + \beta_t\boldsymbol{v}_t\left(\boldsymbol{k}_t - \boldsymbol{S}_{t-1}^\top\boldsymbol{v}_t\right)^\top \\ &= \left(\boldsymbol{I} - \beta_t\boldsymbol{v}_t\boldsymbol{v}_t^\top\right)\boldsymbol{S}_{t-1} + \beta_t\boldsymbol{v}_t\boldsymbol{k}_t^\top.\end{aligned} \tag{8}$$

This left-acting correction has the algebraic form of Oja's normalized Hebbian update (Oja, 1982); we therefore refer to Eq. 8 as the *Oja correction*. In contrast to the delta rule, which uses a key to correct its associated value, the Oja correction uses a value to correct its associated key.

Gated Oja Rule. To support adaptive forgetting, we introduce channel-wise decay $\boldsymbol{\alpha}_t$:

$$\boldsymbol{S}_t = \left(\boldsymbol{I} - \beta_t\boldsymbol{v}_t\boldsymbol{v}_t^\top\right)\mathrm{Diag}(\boldsymbol{\alpha}_t)\boldsymbol{S}_{t-1} + \beta_t\boldsymbol{v}_t\boldsymbol{k}_t^\top. \tag{9}$$

We refer to this recurrence as **OJA Rule**.

Gated Oja Rule-2. Following the erase–write decoupling of GDN2, we further replace the shared scalar β_t with an erase gate $\boldsymbol{b}_t$ and a write gate $\boldsymbol{c}_t$:

$$\boldsymbol{S}_t = \left(\boldsymbol{I} - \boldsymbol{v}_t(\boldsymbol{b}_t \odot \boldsymbol{v}_t)^\top\right)\mathrm{Diag}(\boldsymbol{\alpha}_t)\boldsymbol{S}_{t-1} + \boldsymbol{v}_t(\boldsymbol{c}_t \odot \boldsymbol{k}_t)^\top. \tag{10}$$

We call this variant **OJA2**. Here, $\boldsymbol{b}_t$ controls erasing on the value-side axis, while $\boldsymbol{c}_t$ selects the key channels written into memory. When $\boldsymbol{b}_t = \beta_t\mathbf{1}_{D_v}$ and $\boldsymbol{c}_t = \beta_t\mathbf{1}_{D_k}$, where $\mathbf{1}_{D_k}$ denotes the all-ones vector in $\mathbb{R}^{D_k}$, Eq. 10 reduces to Eq. 9.

Together, OJA2 and GDN2 provide complementary error-correcting updates: OJA2 corrects the key associated with a value, whereas GDN2 corrects the value associated with a key.

Table 1: Comparison of recurrent memory mechanisms under the state orientation used in this paper. Blue terms highlight data-dependent memory control. Mamba-3S and Mamba-3M denote the SISO and MIMO variants of Mamba-3. Delta-rule updates correct values addressed by keys, whereas OJA and OJA2 apply the complementary value-to-key correction. GSA and GSA2 factorize memory into key–slot and slot–value states; GSA2 combines Gated OJA Rule-2 and Gated Delta Rule-2 through the shared latent-slot representation $\boldsymbol{w}_t$.

Model	State capacity	Gated memory update	Output
Decay-Gated Additive Updates			
Mamba-2	$\boldsymbol{S}_t \in \mathbb{R}^{D_v \times D_k}$	$\boldsymbol{S}_t = \alpha_t \boldsymbol{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{S}_t \boldsymbol{q}_t$
Mamba-3S	$\boldsymbol{S}_t \in \mathbb{R}^{D_v \times D_k}$	$\boldsymbol{S}_t = \alpha_t \boldsymbol{S}_{t-1} + \eta_t \boldsymbol{v}_{t-1} \tilde{\boldsymbol{k}}_{t-1}^\top + \zeta_t \boldsymbol{v}_t \tilde{\boldsymbol{k}}_t^\top$	$\boldsymbol{S}_t \boldsymbol{q}_t$
Mamba-3M	$\boldsymbol{S}_t \in \mathbb{R}^{D_v \times D_k}$	$\boldsymbol{S}_t = \alpha_t \boldsymbol{S}_{t-1} + \sum_{r=1}^{R} \left(\eta_{t,r} \boldsymbol{v}_{t-1}^{(r)} \tilde{\boldsymbol{k}}_{t-1}^{(r)\top} + \zeta_{t,r} \boldsymbol{v}_t^{(r)} \tilde{\boldsymbol{k}}_t^{(r)\top} \right)$	$\boldsymbol{S}_t \boldsymbol{q}_t$
Delta-Rule Error-Correcting Updates			
DeltaNet	$\boldsymbol{S}_t \in \mathbb{R}^{D_v \times D_k}$	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} \left(\boldsymbol{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top \right) + \beta_t \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{S}_t \boldsymbol{q}_t$
GDN	$\boldsymbol{S}_t \in \mathbb{R}^{D_v \times D_k}$	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} \alpha_t \left(\boldsymbol{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top \right) + \beta_t \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{S}_t \boldsymbol{q}_t$
KDA	$\boldsymbol{S}_t \in \mathbb{R}^{D_v \times D_k}$	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} \mathrm{Diag}(\boldsymbol{\alpha}_t) \left(\boldsymbol{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top \right) + \beta_t \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{S}_t \boldsymbol{q}_t$
GDN2	$\boldsymbol{S}_t \in \mathbb{R}^{D_v \times D_k}$	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} \mathrm{Diag}(\boldsymbol{\alpha}_t) \left(\boldsymbol{I} - (\boldsymbol{b}_t \odot \boldsymbol{k}_t) \boldsymbol{k}_t^\top \right) + (\boldsymbol{c}_t \odot \boldsymbol{v}_t) \boldsymbol{k}_t^\top$	$\boldsymbol{S}_t \boldsymbol{q}_t$
Oja-Rule Error-Correcting Updates			
OJA	$\boldsymbol{S}_t \in \mathbb{R}^{D_v \times D_k}$	$\boldsymbol{S}_t = \left(\boldsymbol{I} - \beta_t \boldsymbol{v}_t \boldsymbol{v}_t^\top \right) \mathrm{Diag}(\boldsymbol{\alpha}_t) \boldsymbol{S}_{t-1} + \beta_t \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{S}_t \boldsymbol{q}_t$
OJA2	$\boldsymbol{S}_t \in \mathbb{R}^{D_v \times D_k}$	$\boldsymbol{S}_t = \left(\boldsymbol{I} - \boldsymbol{v}_t (\boldsymbol{b}_t \odot \boldsymbol{v}_t)^\top \right) \mathrm{Diag}(\boldsymbol{\alpha}_t) \boldsymbol{S}_{t-1} + \boldsymbol{v}_t (\boldsymbol{c}_t \odot \boldsymbol{k}_t)^\top$	$\boldsymbol{S}_t \boldsymbol{q}_t$
Two-Stage Slot-Based Updates			
GSA	$\boldsymbol{S}_t^{(1)} \in \mathbb{R}^{M \times D_k}$ $\boldsymbol{S}_t^{(2)} \in \mathbb{R}^{D_v \times M}$	$\boldsymbol{S}_t^{(1)} = \mathrm{Diag}(\boldsymbol{\alpha}_t) \boldsymbol{S}_{t-1}^{(1)} + (\boldsymbol{1} - \boldsymbol{\alpha}_t) \boldsymbol{k}_t^\top$ $\boldsymbol{S}_t^{(2)} = \boldsymbol{S}_{t-1}^{(2)} \mathrm{Diag}(\boldsymbol{\alpha}_t) + \boldsymbol{v}_t (\boldsymbol{1} - \boldsymbol{\alpha}_t)^\top$	$\boldsymbol{S}_t^{(2)} \, \mathrm{softmax}(\boldsymbol{S}_t^{(1)} \boldsymbol{q}_t)$
GSA2	$\boldsymbol{S}_t^{(1)} \in \mathbb{R}^{M \times D_k}$ $\boldsymbol{S}_t^{(2)} \in \mathbb{R}^{D_v \times M}$	$\boldsymbol{S}_t^{(1)} = \left(\boldsymbol{I} - \boldsymbol{w}_t (\boldsymbol{b}_t^{(1)} \odot \boldsymbol{w}_t)^\top \right) \mathrm{Diag}(\boldsymbol{\alpha}_t^{(1)}) \boldsymbol{S}_{t-1}^{(1)} + \boldsymbol{w}_t (\boldsymbol{c}_t^{(1)} \odot \boldsymbol{k}_t)^\top$ $\boldsymbol{S}_t^{(2)} = \boldsymbol{S}_{t-1}^{(2)} \mathrm{Diag}(\boldsymbol{\alpha}_t^{(2)}) \left(\boldsymbol{I} - (\boldsymbol{b}_t^{(2)} \odot \boldsymbol{w}_t) \boldsymbol{w}_t^\top \right) + (\boldsymbol{c}_t^{(2)} \odot \boldsymbol{v}_t) \boldsymbol{w}_t^\top$	$\boldsymbol{S}_t^{(2)} \, \mathrm{silu}(\boldsymbol{S}_t^{(1)} \boldsymbol{q}_t)$

3.2 Two-Stage Memory with Shared Slots

We combine the complementary OJA-2 and Gated Delta Rule-2 updates within the two-stage memory structure of GSA. Let $\boldsymbol{S}_t^{(1)} \in \mathbb{R}^{M \times D_k}$ and $\boldsymbol{S}_t^{(2)} \in \mathbb{R}^{D_v \times M}$, and let $\boldsymbol{w}_t \in \mathbb{R}^M$ denote a shared latent-slot representation. It serves as the value-side vector in the first memory and as the addressing vector in the second, thereby coupling the two stages. The first-stage update is

$$\boldsymbol{S}_t^{(1)} = \left(\boldsymbol{I} - \boldsymbol{w}_t (\boldsymbol{b}_t^{(1)} \odot \boldsymbol{w}_t)^\top \right) \mathrm{Diag}(\boldsymbol{\alpha}_t^{(1)}) \boldsymbol{S}_{t-1}^{(1)} + \boldsymbol{w}_t (\boldsymbol{c}_t^{(1)} \odot \boldsymbol{k}_t)^\top. \tag{11}$$

It therefore performs key-side correction between the latent slot and the input key.

The second stage uses the same $\boldsymbol{w}_t$ as the addressing key of a GDN2 memory:

$$\boldsymbol{S}_t^{(2)} = \boldsymbol{S}_{t-1}^{(2)} \mathrm{Diag}(\boldsymbol{\alpha}_t^{(2)}) \left(\boldsymbol{I} - (\boldsymbol{b}_t^{(2)} \odot \boldsymbol{w}_t) \boldsymbol{w}_t^\top \right) + (\boldsymbol{c}_t^{(2)} \odot \boldsymbol{v}_t) \boldsymbol{w}_t^\top. \tag{12}$$

It therefore performs the complementary value-side correction between the shared slot and the value. Together, Eqs. 11 and 12 instantiate the two-sided associative correction introduced in Section 1, with the shared latent slot coupling key-side and value-side memory editing.

The two memories use independent control gates: $\boldsymbol{b}_t^{(1)}, \boldsymbol{b}_t^{(2)} \in [0,1]^M$ are slot-wise erase gates, while $\boldsymbol{c}_t^{(1)} \in [0,1]^{D_k}$ and $\boldsymbol{c}_t^{(2)} \in [0,1]^{D_v}$ are key- and value-channel write gates. Thus, GSA2 extends decoupled erase–write control from a single associative state to both stages of a slot memory.

Finally, GSA2 reads the two memories sequentially:

$$\boldsymbol{h}_t = \mathrm{silu} \left(\boldsymbol{S}_t^{(1)} \boldsymbol{q}_t \right), \qquad \boldsymbol{o}_t = \boldsymbol{S}_t^{(2)} \boldsymbol{h}_t. \tag{13}$$

Unlike GSA, which applies softmax over the intermediate slot activations, GSA2 uses SiLU so that corrected slots can contribute independently rather than compete under a normalized simplex.

Table 1 summarizes the recurrent memory mechanisms considered in this work under a unified state orientation. The comparison highlights the progression from decay-gated additive updates, to Delta- and Oja-based error-correcting updates, and finally to the two-stage slot-based memory of GSA2. In particular, GSA2 combines Gated Oja Rule-2 and Gated Delta Rule-2 through a shared latent-slot representation, yielding explicit correction on both sides of the association.

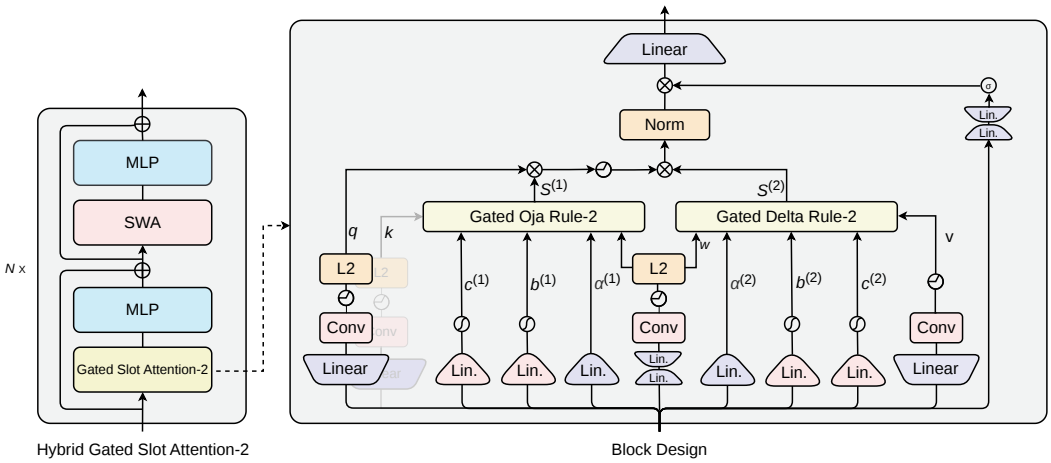

Figure 1: Architecture of **GSA2**. Left: the hybrid backbone interleaves GSA2, MLP, and SWA blocks. Right: the GSA2 token mixer. The query and key branches apply linear projection, short convolution, SiLU, and L2 normalization, while the value branch follows the same pipeline without L2 normalization. The shared-slot branch uses a low-rank two-layer projection followed by short convolution, SiLU, and L2 normalization. Gated Oja Rule-2 and Gated Delta Rule-2 perform key- and value-side correction through shared latent slots, with separate decay, erase, and write gates controlling the two recurrent memories. The output is RMS-normalized, modulated by a sigmoid gate from a low-rank two-layer projection, and linearly projected.

3.3 Hardware-Efficient Chunkwise Training

GSA2 couples Gated Oja Rule-2 with Gated Delta Rule-2, and efficient training therefore requires chunkwise forms for both recurrent stages. For the Gated Delta Rule-2 stage, we directly adopt the hardware-efficient implementation of GDN2 (Hatamizadeh et al., 2026). We therefore focus below on deriving the corresponding chunkwise form for Gated Oja Rule and Gated Oja Rule-2.

The recurrence is sequential tokenwise, with diagonal decay followed by rank-one state transformations. Within each chunk, the accumulated rank-one transitions have the form

$$\prod_{i=1}^{r} \left(\boldsymbol{I} - \boldsymbol{a}_i \boldsymbol{z}_i^\top \right) = \boldsymbol{I} - \boldsymbol{Y}_r \boldsymbol{Z}_r^\top, \tag{14}$$

where the right-hand side is a compact WY representation (Bischof & Van Loan, 1987). The channel-wise decay terms can be absorbed into the corresponding rank-one factors, while the auxiliary matrices can be computed using triangular solves. For Gated Oja Rule-2, the resulting chunk-boundary update can be written as

$$\boldsymbol{S}_{[t+1]} = \mathrm{Diag}\left(\boldsymbol{\alpha}_{[t]}^{1:C} \right) \boldsymbol{S}_{[t]} + \left(\boldsymbol{A}_{[t]}^{i+1 \to C} \odot \boldsymbol{V}_{[t]} \right)^\top \left(\boldsymbol{U}_{[t]} - \boldsymbol{W}_{[t]} \boldsymbol{S}_{[t]} \right), \tag{15}$$

where $\boldsymbol{A}_{[t]}^{i+1 \to C}$ collects the corresponding suffix-decay factors, and $\boldsymbol{U}_{[t]}$ and $\boldsymbol{W}_{[t]}$ are chunk-level auxiliary matrices obtained through the UT transform (Joffrain et al., 2006). The uppercase $\boldsymbol{W}_{[t]}$ denotes this auxiliary matrix and should not be confused with the token-level shared latent-slot representation $\boldsymbol{w}_t$ in GSA2. The corresponding intra-chunk outputs admit an analogous matrix form, so computation within each chunk is dominated by dense matrix multiplications. Detailed WY/UT derivations of Gated Oja Rule and Gated Oja Rule-2 are given in Appendices A.1 and A.2; Eq. 22 gives the base chunkwise form, and Eq. 26 the Gated Oja Rule-2 auxiliary factors.

3.4 Block Design and Model Families

GSA2 token mixer. GSA2 is used as the recurrent token mixer in a standard Transformer-style residual block. Fig. 1 (right) shows its block design. For the two-stage recurrent update, $\{\boldsymbol{q}_t, \boldsymbol{k}_t, \boldsymbol{v}_t\}$ are produced by linear projection, short causal convolution, and SiLU, with L2 normalization applied to $\boldsymbol{q}_t$ and $\boldsymbol{k}_t$ for stability. The shared latent-slot representation $\boldsymbol{w}_t$ is produced by a low-rank projection followed by short causal convolution, SiLU, and L2 normalization. Separate branches produce the stage-specific decay, erase, and write gates $\{\boldsymbol{\alpha}_t^{(i)}, \boldsymbol{b}_t^{(i)}, \boldsymbol{c}_t^{(i)}\}_{i=1}^2$. The recurrent output is RMS-normalized, modulated by a sigmoid output gate, and projected back to the model dimension.

Table 2: Performance on language modeling and zero-shot commonsense reasoning. All non-Transformer hybrid models use SWA. * Recurrent baseline results are taken from GDN2 (Hatamizadeh et al., 2026); OJA, OJA2, GSA, and GSA2 are evaluated in this work.

Model	Language Modeling		Zero-shot Commonsense Reasoning									
	Wiki. ppl ↓	LMB. ppl ↓	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc ↑	OBQA acc ↑	SIQA acc ↑	BoolQ acc ↑	Avg. acc ↑
Recurrent models												
Mamba-2*	16.79	12.38	45.24	72.58	55.51	55.33	70.68	35.26	31.00	40.63	60.19	51.82
Gated DeltaNet*	16.40	11.89	49.62	72.31	56.50	56.75	68.81	35.15	30.20	40.53	58.78	52.07
KDA*	16.81	11.68	48.13	72.09	55.75	55.72	70.83	35.92	30.40	40.99	60.67	52.28
Mamba-3 (SISO)*	16.24	11.84	47.84	73.01	59.39	57.46	72.73	37.37	27.20	**41.97**	**60.80**	53.08
Mamba-3 (MIMO)*	16.17	11.10	49.21	**75.08**	59.51	57.85	**74.12**	**39.16**	28.80	41.81	54.25	53.31
Gated DeltaNet-2*	15.90	11.41	48.09	72.80	56.84	57.85	72.43	38.23	**31.60**	40.58	59.54	53.11
OJA	18.14	14.47	44.05	71.16	54.48	56.27	68.90	33.87	27.60	**41.97**	58.10	50.71
OJA2	18.28	14.84	43.95	71.49	54.18	54.70	68.69	33.79	27.00	40.17	57.37	50.15
Gated Slot Attention	16.52	11.22	49.49	74.05	**60.22**	59.19	73.61	37.80	30.40	39.46	55.35	53.28
Gated Slot Attention-2	**15.69**	**10.68**	**50.07**	73.56	59.28	**60.77**	72.77	38.48	30.00	41.30	60.46	**54.08**
Attention or hybrid models												
Transformer	16.60	12.56	49.08	72.09	57.40	56.67	69.95	35.07	27.20	39.87	59.36	51.85
Mamba-2	**15.85**	13.62	47.37	**73.78**	58.81	58.48	70.83	37.37	29.20	41.15	49.39	51.82
Gated DeltaNet	16.10	12.02	48.69	71.98	57.46	58.96	71.80	35.75	29.20	41.61	57.83	52.59
KDA	16.22	11.49	49.39	73.23	58.95	57.38	71.93	37.20	29.40	41.10	57.58	52.91
Mamba-3 (SISO)	16.00	11.57	49.27	73.34	59.01	58.88	71.17	36.18	28.40	41.61	58.26	52.90
Mamba-3 (MIMO)	15.94	11.75	49.51	72.80	**59.45**	57.93	**72.90**	**37.63**	**30.40**	**41.97**	60.12	53.63
Gated DeltaNet-2	16.26	11.93	49.49	73.01	58.09	57.46	70.83	35.67	29.40	39.97	59.94	52.65
OJA	16.53	12.12	48.92	72.03	58.45	56.67	70.79	34.98	28.60	38.08	60.86	52.15
OJA2	16.33	12.47	48.83	72.47	57.78	56.35	69.95	37.29	26.40	38.84	59.88	51.98
Gated Slot Attention	15.98	11.36	49.52	72.85	58.76	59.35	71.97	36.01	28.20	38.28	59.05	52.67
Gated Slot Attention-2	17.23	**10.80**	**51.76**	72.91	58.92	**59.75**	71.51	37.03	30.20	40.99	**62.54**	**53.96**

Recurrent and hybrid models. We evaluate both recurrent and hybrid GSA2 architectures. The recurrent model alternates GSA2 token mixers with MLP layers under the standard residual backbone. The hybrid model follows the same layer pattern as Gated DeltaNet-2, interleaving GSA2 with Sliding-Window Attention (SWA), as shown in Fig. 1 (left). A repeated hybrid cell consists of a GSA2 token mixer, an MLP layer, an SWA layer, and another MLP layer. GSA2 compresses long-range information into a constant-size recurrent state, while SWA provides exact interactions within a bounded local window. With a fixed window size, the hybrid architecture preserves linear scaling with sequence length and maintains a bounded attention cache.

4 Experiments

Setup. We evaluate each recurrent mixer in both recurrent-only and hybrid settings, with the latter interleaving the recurrent mixer with SWA as described in Section 3.4. We compare against Transformer, Mamba-2, GDN, KDA, Mamba-3 (SISO and MIMO), GDN2, GSA, OJA, and OJA2. Models trained in this work contain approximately 1.3B parameters and are trained on 100B FineWeb-Edu tokens (Penedo et al., 2024). Following GDN2 (Hatamizadeh et al., 2026), we use AdamW with a peak learning rate of 4×10^{-4}, weight decay of 0.1, gradient clipping at 1.0, cosine decay, a 1B-token warm-up, and a global batch size of 0.5M tokens. The training length is 4K tokens, and hybrid models use a 2K SWA window. Additional details are provided in the Appendix A.3.

Language modeling and common-sense reasoning. Table 2 reports WikiText and LAMBADA perplexity (Merity et al., 2016; Paperno et al., 2016), zero-shot LAMBADA accuracy, and the common-sense suite from PIQA through BoolQ (Bisk et al., 2020; Zellers et al., 2019; Sakaguchi et al., 2021; Clark et al., 2018; Mihaylov et al., 2018; Sap et al., 2019; Clark et al., 2019). GSA2 achieves the best common-sense average in both recurrent and hybrid settings while remaining strong on language modeling. The gains are consistent against strong recurrent baselines such as Gated DeltaNet-2, OJA2, and Mamba-3, and persist when combined with SWA.

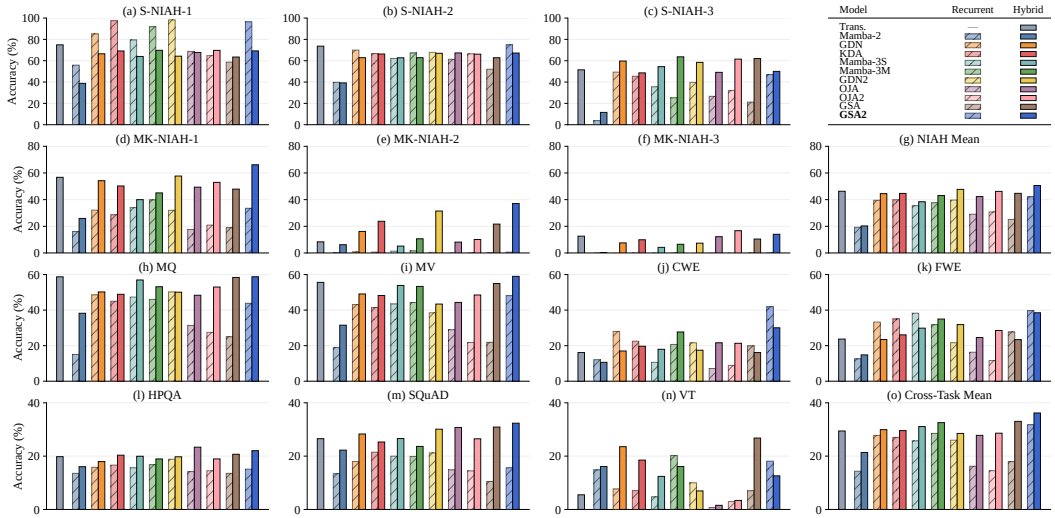

Figure 2: RULER accuracy of recurrent and attention/hybrid models. Each task is averaged over 1K–8K contexts. NIAH Avg. summarizes the 24 S-NIAH and MK-NIAH settings, while Cross-Task Avg. summarizes the 28 settings from MQ, MV, CWE, FWE, HPQA, SQuAD, and VT. Colors denote base models; hatched and solid bars indicate recurrent and attention/hybrid configurations, respectively.

In-context retrieval on real-world tasks. Table 3 reports recall-heavy real-world tasks from (Arora et al., 2024), spanning extraction, question answering, and distractor-rich evidence. GSA2 achieves the best average in the recurrent setting, showing that two-sided correction transfers beyond synthetic retrieval. In the hybrid setting, OJA and OJA2 become particularly competitive. Since these tasks use 2K inputs, matching the SWA window, local attention can directly access most relevant evidence, reducing the burden on the recurrent state.

In-context retrieval on synthetic data. Fig. 2 reports RULER results across NIAH and other long-context tasks (Hsieh et al., 2024). GSA2 achieves the strongest overall retrieval performance in both recurrent and hybrid settings. Its gains over GSA, GDN2, and OJA2 show that two-sided correction is more effective than gated accumulation or correcting either side alone. This supports combining key-side Gated Oja Rule-2 correction with value-side Gated Delta Rule-2 correction in the two-stage memory. Full results are provided in Appendix Tables 6 and 7.

Table 3: Real-world retrieval at 2K context. All non-Transformer hybrid models use SWA.

Model	Real-world Retrieval (2K)						
	SWDE acc ↑	SQD acc ↑	FDA acc ↑	TQA acc ↑	NQ acc ↑	DROP acc ↑	Avg. acc ↑
Recurrent models							
Mamba-2*	17.24	32.38	14.53	58.35	18.91	19.60	26.84
Gated DeltaNet*	17.90	32.67	18.52	59.60	**20.16**	19.69	28.09
KDA*	22.49	35.10	14.90	58.12	19.58	**21.80**	28.67
Mamba-3 (SISO)*	27.36	34.62	10.53	60.37	15.93	21.75	28.43
Mamba-3 (MIMO)*	26.01	34.85	16.97	60.60	15.52	19.65	28.93
Gated DeltaNet-2*	23.65	**36.75**	**19.98**	61.37	19.64	17.87	29.88
OJA	22.14	33.21	7.71	58.77	15.05	18.83	25.95
OJA2	20.61	33.75	12.07	58.18	14.22	18.59	26.24
Gated Slot Attention	17.82	32.47	12.79	61.26	15.81	20.12	26.71
Gated Slot Attention-2	**29.97**	36.29	19.42	**63.09**	17.26	21.51	**31.26**
Attention or hybrid models							
Transformer	41.58	41.12	21.42	63.15	16.79	22.42	34.41
Mamba-2	36.00	36.33	26.13	58.47	18.47	18.74	32.36
Gated DeltaNet	43.11	40.62	29.85	63.45	17.90	20.60	35.92
KDA	42.66	41.05	27.95	62.56	17.90	22.28	35.73
Mamba-3 (SISO)	41.67	40.25	25.50	62.97	18.15	22.09	35.11
Mamba-3 (MIMO)	41.49	40.28	28.58	63.51	17.55	22.86	35.71
Gated DeltaNet-2	44.01	40.75	27.95	61.49	17.58	20.65	35.41
OJA	47.25	41.76	28.86	**64.69**	18.63	**23.77**	**37.49**
OJA2	47.61	40.78	**30.40**	62.91	17.71	23.72	37.19
Gated Slot Attention	35.55	40.35	19.78	63.57	16.69	21.37	32.89
Gated Slot Attention-2	**51.31**	**42.26**	25.05	62.56	**18.78**	20.22	36.70

Long-context evaluation. Table 4 reports LongBench Bai et al. (2024) results on document understanding, summarization, few-shot learning, and code. GSA2 achieves the highest overall average in both recurrent and hybrid settings. Together with RULER, these results show that two-sided associative correction benefits long-context modeling beyond controlled retrieval.

Ablation studies. Table 5 examines the main design choices of GSA2. All ablation variants use the 1.3B model scale and are trained on 100B tokens.

(1) *Complementary two-sided correction.* We first compare the correction rules used in the two recurrent memories. Applying Gated Delta Rule-2 to both memories or Gated Oja Rule-2 to both

Table 4: LongBench performance across five task categories. Avg. is the mean over all 11 tasks. All non-Transformer models in the hybrid group use SWA.

Model	Single-Doc		Multi-Doc		Summ.		Few-shot			Code		Avg.
	MF.	Qasp.	Hotp.	2Wiki.	Gov.	Multi.	SAM.	TREC	Triv.	LCC	Repo.	↑
Recurrent models												
Mamba-2	12.0	5.0	5.2	7.6	8.6	11.4	19.5	14.0	11.2	21.1	21.7	12.5
Gated DeltaNet	**14.2**	**6.4**	5.1	8.2	8.3	12.0	27.5	27.0	**15.1**	23.1	23.8	15.5
KDA	14.0	6.3	6.1	8.0	7.6	10.4	28.8	35.0	14.4	21.6	24.0	16.0
Mamba-3 (SISO)	13.7	5.7	5.8	7.7	8.3	10.7	23.3	27.0	14.2	19.9	22.5	14.4
Mamba-3 (MIMO)	13.1	4.9	**6.4**	8.1	5.1	12.0	24.3	13.5	13.1	**23.5**	24.6	13.5
Gated DeltaNet-2	13.1	6.3	5.9	**10.0**	7.9	11.4	**30.9**	37.0	14.2	22.7	**25.3**	16.8
OJA	13.4	5.4	5.8	8.5	4.7	7.2	17.9	34.0	13.6	21.0	21.4	13.9
OJA2	13.6	5.3	5.7	8.9	8.8	**12.1**	19.5	18.5	14.3	20.8	21.6	13.6
Gated Slot Attention	12.8	6.0	6.1	9.7	6.7	10.2	22.7	19.0	13.8	21.3	21.8	13.6
Gated Slot Attention-2	13.0	5.8	5.5	8.3	**10.0**	11.6	27.8	**46.5**	14.7	22.4	24.2	**17.3**
Attention or hybrid models												
Transformer	6.9	3.7	1.0	1.6	4.5	10.4	2.8	12.5	3.1	16.8	2.3	6.0
Gated DeltaNet	10.3	5.2	2.9	5.0	7.8	11.8	8.5	40.0	5.9	20.5	17.8	12.3
KDA	13.6	6.4	5.5	9.3	9.7	12.6	21.3	34.5	13.3	20.7	22.2	15.4
Mamba-3 (SISO)	7.9	4.9	0.9	3.3	4.3	12.2	8.9	40.0	4.2	16.4	7.8	10.1
Mamba-3 (MIMO)	10.0	5.2	2.4	4.4	8.1	**12.7**	8.9	26.5	6.3	18.6	11.5	10.4
Gated DeltaNet-2	15.1	6.0	5.6	9.6	5.1	10.9	23.0	46.0	13.7	21.2	**24.3**	16.4
OJA	14.2	**6.7**	5.4	9.5	7.7	10.2	27.6	40.0	**15.2**	19.8	22.2	16.2
OJA2	14.0	6.0	5.4	9.4	**10.0**	10.5	28.0	35.5	13.6	21.0	21.7	15.9
Gated Slot Attention	9.6	4.9	2.6	3.7	9.0	12.2	12.9	19.5	5.3	21.4	13.0	10.4
Gated Slot Attention-2	**15.8**	5.7	**6.5**	**10.4**	6.4	8.5	**30.1**	**49.0**	15.0	**22.2**	20.6	**17.3**

Table 5: Ablation study of GSA2. Common. averages LAMBADA accuracy and nine zero-shot commonsense reasoning tasks. S-NIAH, MK-NIAH, and Rem. are averages over the corresponding RULER task groups. Real-world retrieval averages six tasks at 2K context, and LongBench averages 11 tasks.

Ablation configuration		Language modeling		Common.	RULER			Real-world retrieval	LongBench
Stage 1	Stage 2	Wiki.↓	LMB.↓	avg↑	S-NIAH↑	MK-NIAH↑	Rem.↑	avg↑	avg↑
Coupled	Coupled	16.22	10.29	53.25	68.2	9.9	30.0	29.60	17.1
	Decoupled	16.01	10.65	52.46	68.0	10.9	24.7	31.26	14.8
Decoupled	Coupled	16.28	11.25	53.37	65.3	9.8	28.3	30.17	16.2
	Decoupled	**15.69**	10.68	**54.08**	**72.8**	11.4	**31.8**	**31.51**	**17.3**
Gated Delta Rule-2 in both updates		15.92	**10.27**	53.03	72.0	**13.1**	29.5	29.76	16.8
Gated Oja Rule-2 in both updates		16.97	12.46	52.35	60.2	8.4	20.9	30.96	13.8
w/o $1/\sqrt{d}$ Scaling		19.76	16.98	48.50	61.8	8.3	21.8	24.73	13.0
SiLU → Softmax		16.92	11.36	50.66	55.6	9.7	26.1	28.14	14.0
Slots = 64		16.55	11.17	53.48	62.6	8.2	30.0	29.06	16.9
Slots = 256		16.37	11.36	52.87	61.7	11.1	30.4	30.10	15.3

memories is less effective overall than GSA2, which combines key-side Gated Oja Rule-2 with value-side Gated Delta Rule-2 through the shared latent slots. **This supports using complementary correction directions for the semantically distinct key- and value-side memories rather than repeating the same correction rule across both.**

(2) *Standalone correction direction.* When used as standalone recurrent updates, Gated DeltaNet-2 generally outperforms OJA2, as shown in Tables 2, 3, 4, 6 and 7. Thus, the benefit of Gated Oja Rule-2 in GSA2 does not arise from being a uniformly stronger recurrent update. Instead, it is most effective as the complementary key-side correction within the two-stage slot memory, while the Delta rule remains better suited to value-side correction.

(3) *Erase–write decoupling.* Decoupling erase and write in both memories gives the strongest overall performance, while decoupling only one side yields less consistent gains. This shows that independent erase–write control benefits both key- and value-side memories.

(4) *Architectural choices.* Removing the standard $1/\sqrt{d}$ scaling factor or replacing SiLU with Softmax causes broad degradation. Varying the number of latent slots further shows that the default 128-slot configuration provides the best overall balance across the evaluated tasks.

Throughput comparison. Fig. 3 reports single-H20 training throughput for the hybrid 1.30B models under a fixed 16,384-token budget; detailed experimental settings are given in Ap-

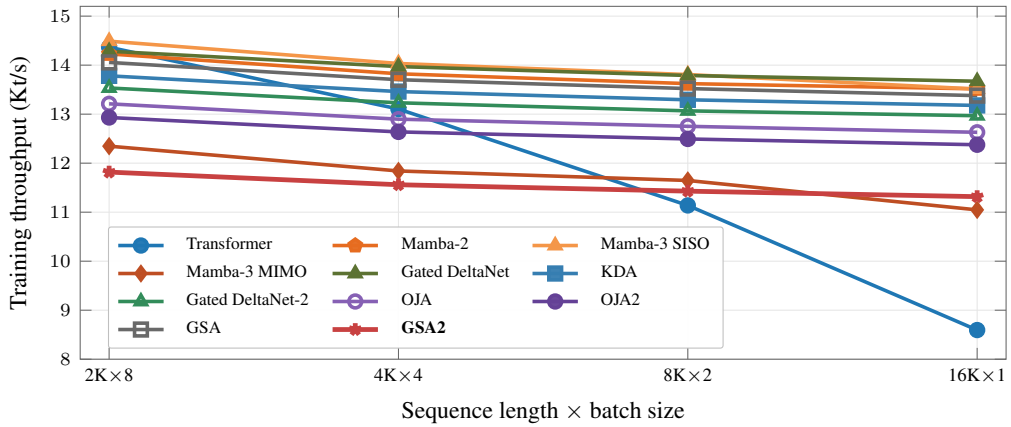

Figure 3: Training throughput of parameter-matched 1.3B models on a single NVIDIA H20 GPU under a fixed 16K-token budget per optimization step. Sequence length increases from 2K to 16K while batch size decreases correspondingly from 8 to 1. The throughput evaluation protocol is detailed in Appendix A.3.3.

pendix A.3.3. GSA2 preserves the near-flat scaling profile of recurrent mixers as sequence length grows, dropping only from 11.82 to 11.32 Kt/s (4.2%), while the Transformer degrades sharply from 14.37 to 8.60 Kt/s (40.2%). OJA, OJA2, and Gated DeltaNet-2 show similarly flat scaling, with throughput reductions of 4.4%, 4.3%, and 4.2%, respectively. Relative to OJA2 and Gated DeltaNet-2, GSA2 is approximately 8.6% and 12.6% slower, respectively, reflecting the additional cost of coupling key-side and value-side error correction through a two-stage slot memory. Thus, GSA2 retains the favorable length-scaling behavior of recurrent attention while paying a moderate constant cost for two-sided associative memory correction.

5 Related Work

Efficient sequence models replace quadratic self-attention with recurrent or linear-time token mixers that maintain fixed-size states. Representative approaches include S4/S5-style state-space models and RetNet (Gu et al., 2021; Smith et al., 2022; Orvieto et al., 2023; Sun et al., 2023), Mamba and its successors (Gu & Dao, 2023; Dao & Gu, 2024; Lahoti et al., 2026), and gated linear RNNs (Yang et al., 2023; Qin et al., 2024). Within this family, Gated Slot Attention (GSA) (Zhang et al., 2024) factorizes associative memory through shared latent slots, maintaining separate key- and value-side memories with gated accumulation. Recurrent states can also be viewed as fast-weight associative memories. Classical work studied error-correcting alternatives to Hebbian accumulation (Gardner, 1988; Prados & Kak, 1989), while modern models instantiate such correction through the delta rule (Irie et al., 2021; Schlag et al., 2021; Yang et al., 2024). Gated DeltaNet (Yang et al., 2025) adds adaptive forgetting, KDA (Team et al., 2025) introduces channel-wise decay, and Gated DeltaNet-2 (Hatamizadeh et al., 2026) decouples erase and write. In this work, GSA2 connects these ideas by introducing complementary Oja- and Delta-based corrections to the key and value sides of GSA.

6 Conclusion

We introduced Gated Slot Attention-2 (GSA2), a two-stage recurrent attention layer that performs complementary associative correction on both sides of a shared latent-slot memory. GSA2 uses Gated Oja Rule-2 to correct key–slot associations in the first stage and Gated Delta Rule-2 to correct slot–value associations in the second, while decoupling erase and write control in both stages. The resulting updates preserve efficient chunkwise training through compact WY representations while retaining fixed-size recurrent memory. At the 1.3B scale, GSA2 achieves strong performance across language modeling, commonsense reasoning, synthetic and real-world retrieval, and general long-context evaluation, with particularly consistent gains on RULER and LongBench. Ablations show that the gains arise from combining complementary correction directions with decoupled memory editing, rather than simply repeating the same update rule across both stages.

AI USE STATEMENT

In this work, we used generative AI tools solely to aid and polish the writing. We did not use generative AI tools for research design, methodology, experiments, data analysis, or the generation of results. All AI-assisted content was reviewed by the authors. We take full responsibility for the final content of this work.

ETHICS STATEMENT

This work studies efficient sequence modeling architectures and does not involve human subjects or the collection of sensitive personal data. Experiments are conducted using publicly available training and evaluation datasets. We are not aware of additional ethical concerns specific to the proposed method beyond those generally associated with large language models.

REPRODUCIBILITY STATEMENT

We provide the model architecture, recurrent update rules, and main training settings in the paper. The appendix provides detailed derivations of the proposed recurrent updates, additional training and evaluation details, and the protocol used for the training-throughput measurements. The complete source code will be made publicly available upon acceptance of the paper.

REFERENCES

Simran Arora, Aman Timalsina, Aaryan Singhal, Benjamin Spector, Sabri Eyuboglu, Xinyi Zhao, Ashish Rao, Atri Rudra, and Christopher Ré. Just read twice: closing the recall gap for recurrent language models. *arXiv preprint arXiv:2407.05483*, 2024.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 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.

Christian Bischof and Charles Van Loan. The wy representation for products of householder matrices. *SIAM Journal on Scientific and Statistical Computing*, 8(1):s2–s13, 1987.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

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, Volume 1 (Long and Short Papers)*, pp. 2924–2936, 2019.

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 preprint arXiv:1803.05457*, 2018.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

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`.

Elizabeth Gardner. The space of interactions in neural network models. *Journal of physics A: Mathematical and general*, 21(1):257–270, 1988.

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. *arXiv preprint arXiv:2111.00396*, 2021.

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

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.

Jiaxi Hu, Yongqi Pan, Jusen Du, Disen Lan, Xiaqiang Tang, Qingsong Wen, Yuxuan Liang, and Weigao Sun. Comba: Improving bilinear rnns with closed-loop control. *arXiv preprint arXiv:2506.02475*, 2025.

Kazuki Irie, Imanol Schlag, Róbert Csordás, and Jürgen Schmidhuber. Going beyond linear transformers with recurrent fast weight programmers. *Advances in neural information processing systems*, 34:7703–7717, 2021.

Thierry Joffrain, Tze Meng Low, Enrique S Quintana-Ortí, Robert van de Geijn, and Field G Van Zee. Accumulating householder transformations, revisited. *ACM Transactions on Mathematical Software (TOMS)*, 32(2):169–179, 2006.

Mahdi Karami, Ali Behrouz, Praneeth Kacham, and Vahab Mirrokni. Trellis: Learning to compress key-value memory in attention models. *arXiv preprint arXiv:2512.23852*, 2025.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Aakash Sunil Lahoti, Kevin Li, Berlin Chen, Caitlin Wang, Aviv Bick, Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved sequence modeling using state space principles. In *International Conference on Learning Representations*, volume 2026, pp. 86173–86202, 2026.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

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 *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2381–2391, 2018.

Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *International conference on machine learning*, pp. 26670–26698. PMLR, 2023.

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, 2016.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

DL Prados and SC Kak. Neural network capacity using delta rule. *Electronics Letters*, 25(3): 197–199, 1989.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. *arXiv preprint arXiv:2404.07904*, 2024.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pp. 9355–9366. PMLR, 2021.

Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

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

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37:115491–115522, 2024.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations*, volume 2025, pp. 29687–29707, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Yu Zhang, Songlin Yang, Ruijie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, et al. Gated slot attention for efficient linear-time sequence modeling. *Advances in Neural Information Processing Systems*, 37:116870–116898, 2024.

A Appendix

A.1 Gated Oja Rule

Throughout this appendix, we use the state orientation $\boldsymbol{S}_t \in \mathbb{R}^{D_v \times D_k}$, with $\boldsymbol{v}_t, \boldsymbol{\alpha}_t \in \mathbb{R}^{D_v}$, $\boldsymbol{k}_t, \boldsymbol{q}_t \in \mathbb{R}^{D_k}$, and scalar β_t. We assume that the decay entries are strictly positive, so that the element-wise divisions used below are well defined.

For the gated Oja rule considered here, we first apply a diagonal value-side decay to the recurrent state, followed by the Oja correction:

$$\begin{aligned}\boldsymbol{S}_t &= \operatorname{Diag}(\boldsymbol{\alpha}_t)\boldsymbol{S}_{t-1} + \beta_t \boldsymbol{v}_t \left(\boldsymbol{k}_t^\intercal - \boldsymbol{v}_t^\intercal \operatorname{Diag}(\boldsymbol{\alpha}_t)\boldsymbol{S}_{t-1}\right) \\ &= \left(\boldsymbol{I} - \beta_t \boldsymbol{v}_t \boldsymbol{v}_t^\intercal\right) \operatorname{Diag}(\boldsymbol{\alpha}_t)\boldsymbol{S}_{t-1} + \beta_t \boldsymbol{v}_t \boldsymbol{k}_t^\intercal .\end{aligned} \tag{16}$$

By partially expanding the recurrence, we have

$$\begin{aligned}\boldsymbol{S}_{[t]}^{r} = &\underbrace{\left(\prod_{i=1}^{r}\left(\boldsymbol{I}-\beta_{[t]}^{i}\boldsymbol{v}_{[t]}^{i}\boldsymbol{v}_{[t]}^{i\intercal}\right)\mathrm{Diag}\left(\boldsymbol{\alpha}_{[t]}^{i}\right)\right)}_{:=\boldsymbol{D}_{[t]}^{r}\ (\text{“pseudo” memory decay})}\boldsymbol{S}_{[t]}^{0} \\ &+\underbrace{\sum_{i=1}^{r}\left(\left(\prod_{j=i+1}^{r}\left(\boldsymbol{I}-\beta_{[t]}^{j}\boldsymbol{v}_{[t]}^{j}\boldsymbol{v}_{[t]}^{j\intercal}\right)\mathrm{Diag}\left(\boldsymbol{\alpha}_{[t]}^{j}\right)\right)\beta_{[t]}^{i}\boldsymbol{v}_{[t]}^{i}\boldsymbol{k}_{[t]}^{i\intercal}\right)}_{:=\boldsymbol{H}_{[t]}^{r}\ (\text{“pseudo” incremental memory})}.\end{aligned} \tag{17}$$

where the matrix products are time ordered, with larger time indices appearing on the left.

For compact notation, we define the element-wise cumulative decay

$$\boldsymbol{\alpha}_{[t]}^{a:b} := \bigodot_{j=a}^{b}\boldsymbol{\alpha}_{[t]}^{j},$$

with $\boldsymbol{\alpha}_{[t]}^{b+1:b} = \mathbf{1}$ for an empty product. We further define the prefix- and suffix-decay matrices $\boldsymbol{A}_{[t]}^{1\to C}, \boldsymbol{A}_{[t]}^{i+1\to C} \in \mathbb{R}^{C\times D_v}$ row-wise as

$$\left[\boldsymbol{A}_{[t]}^{1\to C}\right]_{r,:} = \boldsymbol{\alpha}_{[t]}^{1:r}, \qquad \left[\boldsymbol{A}_{[t]}^{i+1\to C}\right]_{i,:} = \boldsymbol{\alpha}_{[t]}^{i+1:C},$$

for $i, r = 1, \ldots, C$. We also collect the scalar gates within a chunk as

$$\boldsymbol{\beta}_{[t]} := \left[\beta_{[t]}^{1}, \ldots, \beta_{[t]}^{C}\right]^{\intercal}.$$

Then, we employ a WY-style representation Bischof & Van Loan (1987):

$$\begin{aligned}\boldsymbol{D}_{[t]}^{r} &= \mathrm{Diag}\left(\boldsymbol{\alpha}_{[t]}^{1:r}\right) - \sum_{i=1}^{r}\mathrm{Diag}\left(\boldsymbol{\alpha}_{[t]}^{i+1:r}\right)\boldsymbol{v}_{[t]}^{i}\boldsymbol{w}_{[t]}^{i\intercal}, \\ \boldsymbol{H}_{[t]}^{r} &= \sum_{i=1}^{r}\mathrm{Diag}\left(\boldsymbol{\alpha}_{[t]}^{i+1:r}\right)\boldsymbol{v}_{[t]}^{i}\boldsymbol{u}_{[t]}^{i\intercal}.\end{aligned} \tag{18}$$

The corresponding auxiliary vectors are

$$\begin{aligned}\boldsymbol{w}_{[t]}^{r} &= \beta_{[t]}^{r}\left(\mathrm{Diag}\left(\boldsymbol{\alpha}_{[t]}^{1:r}\right)\boldsymbol{v}_{[t]}^{r} - \sum_{i=1}^{r-1}\boldsymbol{w}_{[t]}^{i}\left(\boldsymbol{v}_{[t]}^{i\intercal}\mathrm{Diag}\left(\boldsymbol{\alpha}_{[t]}^{i+1:r}\right)\boldsymbol{v}_{[t]}^{r}\right)\right), \\ \boldsymbol{u}_{[t]}^{r} &= \beta_{[t]}^{r}\left(\boldsymbol{k}_{[t]}^{r} - \sum_{i=1}^{r-1}\boldsymbol{u}_{[t]}^{i}\left(\boldsymbol{v}_{[t]}^{i\intercal}\mathrm{Diag}\left(\boldsymbol{\alpha}_{[t]}^{i+1:r}\right)\boldsymbol{v}_{[t]}^{r}\right)\right).\end{aligned} \tag{19}$$

where $\boldsymbol{\alpha}_{[t]}^{r+1:r} = \mathbf{1}$ denotes the empty product.

To maximize hardware efficiency, we apply the UT transform (Joffrain et al., 2006) to reduce non-matmul FLOPs, which is crucial to enable better hardware utilization during training.

$$\begin{aligned}\boldsymbol{W}_{[t]} &= \boldsymbol{M}_{[t]}\mathrm{Diag}\left(\boldsymbol{\beta}_{[t]}\right)\left(\boldsymbol{A}_{[t]}^{1\to C}\odot\boldsymbol{V}_{[t]}\right), \qquad \boldsymbol{U}_{[t]} = \boldsymbol{M}_{[t]}\mathrm{Diag}\left(\boldsymbol{\beta}_{[t]}\right)\boldsymbol{K}_{[t]}, \\ \boldsymbol{M}_{[t]} &= \left(\boldsymbol{I} + \mathrm{lower}\left(\mathrm{Diag}\left(\boldsymbol{\beta}_{[t]}\right)\left(\boldsymbol{V}_{[t]}\odot\boldsymbol{A}_{[t]}^{1\to C}\right)\left(\boldsymbol{V}_{[t]}\oslash\boldsymbol{A}_{[t]}^{1\to C}\right)^{\intercal}\right)\right)^{-1}.\end{aligned} \tag{20}$$

where $\boldsymbol{M}_{[t]} \in \mathbb{R}^{C\times C}$, $\boldsymbol{W}_{[t]} \in \mathbb{R}^{C\times D_v}$, $\boldsymbol{U}_{[t]} \in \mathbb{R}^{C\times D_k}$, and $\mathrm{lower}(\cdot)$ extracts the strictly lower-triangular part.

The corresponding lower-triangular systems can be efficiently solved through an iterative row-wise approach using forward substitution , with the computation maintained in $\mathrm{float32}$.

Then we have the following vector form:

$$\begin{aligned}
\boldsymbol{S}^r_{[t]} &= \boldsymbol{D}^r_{[t]}\boldsymbol{S}^0_{[t]} + \boldsymbol{H}^r_{[t]} \\
&= \mathrm{Diag}\left(\boldsymbol{\alpha}^{1:r}_{[t]}\right)\boldsymbol{S}^0_{[t]} + \sum_{i=1}^{r} \mathrm{Diag}\left(\boldsymbol{\alpha}^{i+1:r}_{[t]}\right)\boldsymbol{v}^i_{[t]}\left(\boldsymbol{u}^{i\intercal}_{[t]} - \left(\boldsymbol{w}^{i\intercal}_{[t]}\boldsymbol{S}^0_{[t]}\right)\right), \\
\boldsymbol{o}^r_{[t]} &= \boldsymbol{S}^r_{[t]}\boldsymbol{q}^r_{[t]} \\
&= \mathrm{Diag}\left(\boldsymbol{\alpha}^{1:r}_{[t]}\right)\boldsymbol{S}^0_{[t]}\boldsymbol{q}^r_{[t]} + \sum_{i=1}^{r} \mathrm{Diag}\left(\boldsymbol{\alpha}^{i+1:r}_{[t]}\right)\boldsymbol{v}^i_{[t]}\left(\boldsymbol{u}^{i\intercal}_{[t]} - \left(\boldsymbol{w}^{i\intercal}_{[t]}\boldsymbol{S}^0_{[t]}\right)\right)\boldsymbol{q}^r_{[t]}.
\end{aligned} \tag{21}$$

Within chunk $[t]$, we identify $\boldsymbol{S}^0_{[t]} = \boldsymbol{S}_{[t]}$ and $\boldsymbol{S}^C_{[t]} = \boldsymbol{S}_{[t+1]}$. Equivalently, in matrix form:

$$\begin{aligned}
\boldsymbol{S}_{[t+1]} &= \mathrm{Diag}\left(\boldsymbol{\alpha}^{1:C}_{[t]}\right)\boldsymbol{S}_{[t]} + \left(\boldsymbol{A}^{i+1\to C}_{[t]} \odot \boldsymbol{V}_{[t]}\right)^{\intercal}\left(\boldsymbol{U}_{[t]} - \boldsymbol{W}_{[t]}\boldsymbol{S}_{[t]}\right), \\
\boldsymbol{O}_{[t]} &= \boldsymbol{A}^{1\to C}_{[t]} \odot \left(\boldsymbol{Q}_{[t]}\boldsymbol{S}^{\intercal}_{[t]} + \mathrm{Tril}\left(\boldsymbol{Q}_{[t]}\underbrace{\left(\boldsymbol{U}_{[t]} - \boldsymbol{W}_{[t]}\boldsymbol{S}_{[t]}\right)^{\intercal}}_{\text{“pseudo”-key term}}\right)\left(\boldsymbol{V}_{[t]} \oslash \boldsymbol{A}^{1\to C}_{[t]}\right)\right).
\end{aligned} \tag{22}$$

Here, $\mathrm{Tril}(\cdot)$ retains the lower-triangular part including the diagonal.

A.2 Gated Oja Rule-2

Inspired by the erase–write decoupling in GDN2 (Hatamizadeh et al., 2026), Gated Oja Rule-2 replaces the shared scalar gate with a value-side erase gate $\boldsymbol{b}_t \in \mathbb{R}^{D_v}$ and a channel-wise write gate $\boldsymbol{c}_t \in \mathbb{R}^{D_k}$, respectively. Under our $D_v \times D_k$ state orientation, the Oja correction is applied through the value-side direction $\boldsymbol{v}_t$; hence the erase gate modulates $\boldsymbol{v}_t$, whereas the write gate modulates $\boldsymbol{k}_t$:

$$\begin{aligned}
\boldsymbol{S}_t &= \mathrm{Diag}\left(\boldsymbol{\alpha}_t\right)\boldsymbol{S}_{t-1} + \boldsymbol{v}_t\left((\boldsymbol{c}_t \odot \boldsymbol{k}_t)^{\intercal} - (\boldsymbol{b}_t \odot \boldsymbol{v}_t)^{\intercal}\mathrm{Diag}\left(\boldsymbol{\alpha}_t\right)\boldsymbol{S}_{t-1}\right) \\
&= \left(\boldsymbol{I} - \boldsymbol{v}_t(\boldsymbol{b}_t \odot \boldsymbol{v}_t)^{\intercal}\right)\mathrm{Diag}\left(\boldsymbol{\alpha}_t\right)\boldsymbol{S}_{t-1} + \boldsymbol{v}_t(\boldsymbol{c}_t \odot \boldsymbol{k}_t)^{\intercal}.
\end{aligned} \tag{23}$$

By partially expanding the recurrence, we have

$$\begin{aligned}
\boldsymbol{S}^r_{[t]} = &\underbrace{\left(\prod_{i=1}^{r}\left(\boldsymbol{I} - \boldsymbol{v}^i_{[t]}(\boldsymbol{b}^i_{[t]} \odot \boldsymbol{v}^i_{[t]})^{\intercal}\right)\mathrm{Diag}\left(\boldsymbol{\alpha}^i_{[t]}\right)\right)}_{:=\boldsymbol{D}^r_{[t]}\ \text{(“pseudo” memory decay)}}\boldsymbol{S}^0_{[t]} \\
&+ \underbrace{\sum_{i=1}^{r}\left(\left(\prod_{j=i+1}^{r}\left(\boldsymbol{I} - \boldsymbol{v}^j_{[t]}(\boldsymbol{b}^j_{[t]} \odot \boldsymbol{v}^j_{[t]})^{\intercal}\right)\mathrm{Diag}\left(\boldsymbol{\alpha}^j_{[t]}\right)\right)\boldsymbol{v}^i_{[t]}(\boldsymbol{c}^i_{[t]} \odot \boldsymbol{k}^i_{[t]})^{\intercal}\right)}_{:=\boldsymbol{H}^r_{[t]}\ \text{(“pseudo” incremental memory)}}.
\end{aligned} \tag{24}$$

where the matrix products are time ordered, with larger time indices appearing on the left.

We retain the same WY factorization structure of $\boldsymbol{D}^r_{[t]}$ and $\boldsymbol{H}^r_{[t]}$ as in Eq. equation 18. Under the decoupled erase and write gates, however, the auxiliary recurrences in Eq. equation 19 are replaced by

$$\begin{aligned}
\boldsymbol{w}^r_{[t]} &= \mathrm{Diag}\left(\boldsymbol{\alpha}^{1:r}_{[t]}\right)(\boldsymbol{b}^r_{[t]} \odot \boldsymbol{v}^r_{[t]}) - \sum_{i=1}^{r-1}\boldsymbol{w}^i_{[t]}\left(\boldsymbol{v}^{i\intercal}_{[t]}\mathrm{Diag}\left(\boldsymbol{\alpha}^{i+1:r}_{[t]}\right)(\boldsymbol{b}^r_{[t]} \odot \boldsymbol{v}^r_{[t]})\right), \\
\boldsymbol{u}^r_{[t]} &= \boldsymbol{c}^r_{[t]} \odot \boldsymbol{k}^r_{[t]} - \sum_{i=1}^{r-1}\boldsymbol{u}^i_{[t]}\left(\boldsymbol{v}^{i\intercal}_{[t]}\mathrm{Diag}\left(\boldsymbol{\alpha}^{i+1:r}_{[t]}\right)(\boldsymbol{b}^r_{[t]} \odot \boldsymbol{v}^r_{[t]})\right).
\end{aligned} \tag{25}$$

where $\boldsymbol{\alpha}^{r+1:r}_{[t]} = \mathbf{1}$ denotes the empty product.

We collect the erase and write gates within a chunk as

$$\boldsymbol{B}_{[t]} = \begin{bmatrix}(\boldsymbol{b}^1_{[t]})^{\intercal}\\ \vdots \\ (\boldsymbol{b}^C_{[t]})^{\intercal}\end{bmatrix} \in \mathbb{R}^{C\times D_v}, \qquad \boldsymbol{C}_{[t]} = \begin{bmatrix}(\boldsymbol{c}^1_{[t]})^{\intercal}\\ \vdots \\ (\boldsymbol{c}^C_{[t]})^{\intercal}\end{bmatrix} \in \mathbb{R}^{C\times D_k}.$$

To maximize hardware efficiency, we apply the UT transform (Joffrain et al., 2006) to the gate-aware WY recurrences above, reducing non-matmul FLOPs and enabling efficient chunkwise computation:

$$\begin{aligned} \boldsymbol{W}_{[t]} &= \boldsymbol{M}_{[t]} \left(\boldsymbol{A}_{[t]}^{1\to C} \odot (\boldsymbol{B}_{[t]} \odot \boldsymbol{V}_{[t]}) \right), \qquad \boldsymbol{U}_{[t]} = \boldsymbol{M}_{[t]} (\boldsymbol{C}_{[t]} \odot \boldsymbol{K}_{[t]}), \\ \boldsymbol{M}_{[t]} &= \left(\boldsymbol{I} + \operatorname{lower}\left(\left((\boldsymbol{B}_{[t]} \odot \boldsymbol{V}_{[t]}) \odot \boldsymbol{A}_{[t]}^{1\to C} \right) \left(\boldsymbol{V}_{[t]} \oslash \boldsymbol{A}_{[t]}^{1\to C} \right)^{\mathsf{T}} \right) \right)^{-1}. \end{aligned} \tag{26}$$

where $\operatorname{lower}(\cdot)$ extracts the strictly lower-triangular part.

Unlike the Gated Oja Rule, the erase and write operations are controlled by separate gates, so there is no shared gate factor $\operatorname{Diag}(\boldsymbol{\beta}_{[t]})$ that can be factored out from both branches. Instead, the erase gate $\boldsymbol{B}_{[t]}$ is absorbed into the correction-side factors and therefore affects both $\boldsymbol{M}_{[t]}$ and $\boldsymbol{W}_{[t]}$, whereas the write gate $\boldsymbol{C}_{[t]}$ acts only on the write-side factor $\boldsymbol{U}_{[t]}$. With the modified auxiliary recurrences in Eq. equation 25 and the UT factors in Eq. equation 26, the resulting vector and matrix forms retain the same structure as Eqs. equation 21 and equation 22, respectively.

A.3 Experimental Details

A.3.1 Training Details

All models trained in this work contain approximately 1.3B parameters and are trained on 100B tokens from FineWeb-Edu (Penedo et al., 2024). We use the TinyLlama v1.1 tokenizer, which follows the Llama-style SentencePiece BPE vocabulary with 32,000 tokens. No additional padding tokens are introduced, since the vocabulary size is already a multiple of 64.

We train all models using AdamW with $\beta_1 = 0.9$ and $\beta_2 = 0.95$. The peak learning rate is 4×10^{-4}, with weight decay of 0.1 and gradient clipping at 1.0. The learning rate is linearly warmed up over the first 1B training tokens and then decayed with a cosine schedule to 10% of the peak learning rate. The global batch size is approximately 0.5M tokens, corresponding to 128 sequences of length 4,096. Training uses BF16 mixed precision. All models are trained on four compute nodes with eight NVIDIA H20 GPUs per node, for a total of 32 GPUs, using CUDA 12.8.

The training sequence length is 4K tokens. For hybrid models, the recurrent mixer is interleaved with sliding-window attention using a 2K attention window. Unless otherwise stated, all models trained in this work follow the same training recipe.

A.3.2 Benchmarks Details

Language modeling and commonsense reasoning. We evaluate language modeling with perplexity on WikiText and LAMBADA. For zero-shot evaluation, we report LAMBADA accuracy together with PIQA, HellaSwag, WinoGrande, ARC-Easy, ARC-Challenge, OpenBookQA, Social IQa, and BoolQ. The commonsense benchmarks are evaluated using `lm-evaluation-harness` (Gao et al., 2024).

Synthetic long-context retrieval. We evaluate long-context retrieval using RULER (Hsieh et al., 2024). The needle-in-a-haystack evaluation includes the S-NIAH and MK-NIAH task groups at multiple context lengths. We additionally evaluate the remaining RULER tasks, including multi-query (MQ), multi-value (MV), common-word extraction (CWE), frequent-word extraction (FWE), HotpotQA (HPQA), SQuAD, and variable tracking (VT). We follow the official RULER evaluation protocol.

Real-world retrieval. Following Arora et al. (2024), we evaluate real-world retrieval on SWDE, SQuAD, FDA, TriviaQA, Natural Questions (NQ), and DROP. All inputs are truncated to 2K tokens.

Long-context understanding. We evaluate general long-context understanding on 11 tasks from LongBench (Bai et al., 2024). These include MultiFieldQA and Qasper for single-document QA; HotpotQA and 2WikiMultiHopQA for multi-document QA; GovReport and MultiNews for summarization; SAMSum, TREC, and TriviaQA for few-shot evaluation; and LCC and RepoBench-P for code. We report the average over all 11 tasks.

Model	S-NIAH-1				S-NIAH-2				S-NIAH-3				MK-NIAH-1				MK-NIAH-2				MK-NIAH-3				Avg.
	1K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K	8K	↑
Recurrent models																									
Mamba-2	68.6	62.4	41.6	51.0	52.8	73.6	23.0	9.8	0.0	0.0	10.6	4.6	24.8	18.6	14.0	7.2	0.2	0.6	0.0	0.0	**0.0**	**0.0**	**0.0**	**0.0**	19.3
Gated DeltaNet	**100.0**	**100.0**	94.6	46.0	**100.0**	97.2	60.2	22.6	84.6	**69.0**	**37.2**	6.6	41.6	37.8	28.2	21.0	2.4	0.6	0.0	0.0	**0.0**	**0.0**	**0.0**	**0.0**	39.6
KDA	**100.0**	**100.0**	**100.0**	90.0	**100.0**	95.6	50.2	20.8	84.0	57.2	31.2	9.6	49.6	34.8	16.6	13.6	2.4	0.4	0.0	0.0	**0.0**	**0.0**	**0.0**	**0.0**	39.8
Mamba-3 (SISO)	**100.0**	98.0	82.0	37.8	**100.0**	96.6	37.8	14.2	80.4	43.8	14.0	3.8	60.6	37.0	25.6	12.8	3.6	**1.6**	**0.4**	0.0	**0.0**	**0.0**	**0.0**	**0.0**	35.4
Mamba-3 (MIMO)	**100.0**	99.8	98.4	69.4	**100.0**	98.6	55.0	15.6	61.8	24.8	11.8	2.4	**65.8**	**55.8**	25.8	11.8	**5.0**	1.0	0.2	0.0	**0.0**	**0.0**	**0.0**	**0.0**	37.6
Gated DeltaNet-2	**100.0**	**100.0**	**100.0**	**92.8**	99.8	87.2	60.6	23.8	78.4	41.2	26.4	**12.8**	40.8	31.8	**29.6**	**25.8**	1.2	0.2	0.2	**0.2**	**0.0**	**0.0**	**0.0**	**0.0**	39.7
OJA	**100.0**	90.2	61.6	23.2	99.8	96.4	31.6	17.8	67.0	24.2	10.6	4.8	25.6	16.4	16.8	11.8	0.0	0.0	0.0	0.0	**0.0**	**0.0**	**0.0**	**0.0**	29.1
OJA2	99.8	92.4	54.0	12.8	**100.0**	95.2	50.6	20.0	70.8	34.4	20.0	3.4	33.2	20.6	18.6	11.4	0.4	0.0	0.0	0.0	**0.0**	**0.0**	**0.0**	**0.0**	30.7
Gated Slot Attention	98.4	78.4	39.8	18.2	99.0	83.2	19.6	6.4	56.2	20.4	7.2	1.4	27.8	22.0	18.8	7.4	0.6	0.4	0.0	0.0	**0.0**	**0.0**	**0.0**	**0.0**	25.2
Gated Slot Attention-2	**100.0**	**100.0**	99.6	86.0	**100.0**	**99.2**	**70.4**	**30.2**	**92.0**	59.6	27.0	9.6	45.2	39.8	26.0	23.2	1.8	0.6	0.0	**0.2**	**0.0**	**0.0**	**0.0**	**0.0**	**42.1**
Attention or hybrid models																									
Transformer	**100.0**	**100.0**	**99.6**	0.0	**100.0**	**100.0**	**94.6**	0.0	79.2	65.4	**61.2**	0.0	80.2	81.0	**65.8**	0.0	15.8	11.2	6.4	0.0	25.4	**18.8**	**6.2**	0.0	46.3
Mamba-2	70.6	59.8	24.6	0.0	45.2	87.2	24.4	0.0	0.8	0.2	45.2	0.0	41.8	38.0	23.8	0.0	11.6	11.8	1.4	0.0	0.2	0.0	0.0	0.0	20.3
Gated DeltaNet	99.8	98.2	47.8	20.2	99.8	99.8	51.0	0.8	96.0	87.2	55.6	0.0	83.0	85.0	47.2	1.6	40.6	19.6	3.2	1.2	21.0	8.4	0.2	0.6	44.5
KDA	99.6	99.8	53.2	24.0	99.8	99.8	48.8	16.8	86.6	65.2	38.8	3.8	78.6	68.6	40.8	13.0	58.2	32.2	4.4	0.2	26.6	10.4	2.4	0.2	44.7
Mamba-3 (SISO)	**100.0**	**100.0**	53.8	2.2	**100.0**	**100.0**	51.0	0.0	86.2	78.8	52.8	0.0	63.8	54.6	41.6	0.0	11.8	6.8	2.0	0.0	10.8	4.8	1.2	0.0	38.4
Mamba-3 (MIMO)	**100.0**	**100.0**	53.8	**25.2**	**100.0**	**100.0**	51.0	0.4	**98.8**	**98.8**	56.4	0.6	62.8	68.6	45.0	3.8	25.6	12.0	4.8	0.2	18.2	6.8	0.8	0.2	43.1
Gated DeltaNet-2	**100.0**	92.6	48.2	16.2	**100.0**	**100.0**	51.6	16.2	96.2	80.6	47.6	9.4	88.6	83.0	45.8	13.4	55.4	50.2	**20.0**	0.2	21.8	5.8	1.8	0.0	47.7
OJA	**100.0**	**100.0**	51.8	19.2	**100.0**	99.0	50.8	**19.4**	87.6	60.4	38.6	9.8	72.8	57.2	49.2	18.2	22.2	8.6	1.6	0.2	32.6	9.6	5.8	0.8	42.3
OJA2	**100.0**	**100.0**	53.6	**25.2**	**100.0**	**100.0**	51.0	13.6	97.4	81.8	54.4	**12.4**	83.8	68.4	44.8	14.8	26.4	11.2	1.0	1.8	**46.8**	13.2	5.6	**1.2**	46.2
Gated Slot Attention	**100.0**	**100.0**	53.8	0.0	**100.0**	**100.0**	51.0	0.2	97.2	94.2	56.8	0.0	69.2	67.8	54.0	0.6	41.0	34.4	11.2	0.0	30.4	10.6	0.8	0.0	44.7
Gated Slot Attention-2	**100.0**	99.6	52.6	24.6	**100.0**	**100.0**	50.8	18.0	86.0	70.6	36.4	7.0	**92.4**	**94.8**	57.4	**20.0**	**77.4**	**53.0**	14.0	**4.0**	32.8	16.6	5.8	0.8	**50.6**

Table 6: Accuracy on RULER single-needle (S-NIAH) and multi-key needle-in-a-haystack (MK-NIAH) tasks across context lengths. Avg. is computed over all 24 settings. All non-Transformer models in the attention/hybrid group use SWA. Best and second-best results within each model family are bolded and underlined, respectively.

A.3.3 Training Throughput Evaluation

We benchmark end-to-end training throughput on a single NVIDIA H20 GPU under a fixed budget of 16,384 tokens per optimization step. We evaluate sequence lengths of 2K, 4K, 8K, and 16K with corresponding batch sizes of 8, 4, 2, and 1, respectively, without gradient accumulation. All compared models contain approximately 1.30B trainable parameters and use the same BF16 mixed-precision configuration.

Each measured iteration consists of a complete optimization step, including gradient clearing, forward propagation, fused cross-entropy, backward propagation, gradient clipping, and a fused AdamW update. To isolate model computation from the input pipeline, token IDs are sampled uniformly and generated directly on the GPU, excluding tokenization, storage, and data-loading overhead.

For each configuration, we discard 20 warm-up iterations and measure the subsequent 50 iterations. We repeat each benchmark on three different physical H20 GPUs, compute the mean throughput within each run, and report the median of the three run-level means. Throughput is measured in thousands of processed tokens per second (Kt/s).

Model	MQ				MV				CWE				FWE				HPQA				SQuAD				VT				Avg.
	1K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K	8K	↑
Recurrent models																													
Mamba-2	21.3	21.4	14.3	3.3	28.1	27.1	15.3	5.5	17.9	8.4	13.6	8.3	22.1	15.1	9.4	3.7	19.0	13.2	10.8	11.2	17.8	15.8	11.8	8.2	24.8	15.8	18.5	0.2	14.4
Gated DeltaNet	79.7	67.7	**33.0**	**13.9**	**76.5**	49.4	29.4	**17.1**	58.1	20.3	24.7	8.9	40.2	41.2	32.3	19.5	20.4	16.0	12.6	14.2	26.6	17.6	15.2	12.6	4.5	5.9	14.2	6.1	27.8
KDA	**88.9**	70.6	15.7	4.9	70.8	60.2	26.6	8.3	41.3	11.3	23.4	14.1	37.4	38.1	34.1	30.5	23.0	14.8	13.6	**15.2**	30.2	21.6	**17.8**	**16.4**	7.4	3.2	6.4	11.2	27.0
Mamba-3 (SISO)	85.3	69.7	28.1	6.3	70.8	64.7	27.9	10.4	17.6	12.3	11.2	1.5	40.0	38.2	**40.4**	**34.5**	21.4	17.2	10.8	13.0	27.6	20.6	16.4	15.4	3.8	4.0	8.2	2.9	25.7
Mamba-3 (MIMO)	85.0	67.3	24.9	7.0	73.2	**65.8**	29.2	8.9	36.7	15.4	23.9	6.8	39.8	32.5	30.3	24.7	22.4	15.6	14.2	15.0	27.0	20.6	**17.8**	14.2	**36.2**	22.6	**20.3**	1.8	28.5
Gated DeltaNet-2	86.1	**75.0**	32.6	7.1	64.2	60.5	23.8	5.4	33.9	15.4	23.5	13.6	24.1	26.6	17.6	18.3	**27.0**	**18.6**	**14.8**	15.0	**30.6**	**22.8**	16.8	14.6	8.8	6.2	9.5	**15.4**	26.0
OJA	65.5	37.7	16.1	6.3	49.0	38.5	17.1	11.2	10.9	5.8	10.0	1.8	22.3	19.3	11.8	11.9	20.2	15.0	9.2	12.4	21.2	16.2	12.6	9.4	1.6	1.3	0.0	0.0	16.2
OJA2	59.4	32.6	13.2	4.8	44.5	26.7	10.7	5.1	13.8	5.3	11.8	4.8	12.6	16.2	11.6	6.0	19.0	14.2	11.0	13.8	17.4	14.8	13.4	12.2	9.3	1.4	1.0	0.1	14.5
Gated Slot Attention	48.4	33.9	15.7	2.1	39.6	29.9	14.7	3.4	37.1	21.4	10.4	10.6	40.6	29.9	25.2	15.1	17.2	12.6	11.4	13.0	16.0	11.2	7.4	7.4	25.0	3.0	0.1	0.0	17.9
Gated Slot Attention-2	77.0	57.4	30.6	10.4	73.3	60.9	**43.6**	14.7	**66.3**	**41.1**	**40.7**	**19.5**	**42.5**	**44.0**	39.3	33.0	20.0	16.4	11.4	12.8	22.6	15.4	13.6	10.8	30.2	**22.7**	19.2	0.4	**31.8**
Attention or hybrid models																													
Transformer	96.6	90.8	**47.5**	0.0	92.7	81.6	**48.1**	0.0	37.8	8.6	18.1	0.0	47.6	35.7	11.5	0.0	34.8	27.8	**16.4**	0.2	41.6	35.0	**29.4**	0.2	12.2	0.6	9.2	0.0	29.4
Mamba-2	61.2	47.6	44.1	0.0	55.9	44.1	26.1	0.0	28.0	13.2	1.3	0.0	29.9	27.1	2.2	0.0	27.6	24.4	12.0	0.2	38.8	29.2	21.0	0.0	38.8	25.6	0.0	0.0	21.4
Gated DeltaNet	94.3	79.5	26.8	0.3	90.0	77.5	28.4	0.4	48.7	6.7	12.1	0.5	42.5	43.6	7.6	0.1	28.6	26.4	12.2	4.6	45.2	36.8	26.0	5.2	51.5	**42.0**	0.6	0.1	29.9
KDA	92.2	81.8	18.9	2.7	90.5	75.5	23.9	3.0	58.9	10.2	7.2	2.4	45.0	43.9	14.5	0.9	30.2	24.2	14.4	12.6	36.6	30.0	22.0	12.6	42.6	29.9	0.8	0.7	29.6
Mamba-3 (SISO)	97.8	91.0	39.0	0.0	**96.5**	82.7	36.3	0.0	48.3	9.2	9.8	4.5	44.1	46.7	22.1	6.5	34.0	**34.2**	11.6	0.0	43.0	37.0	26.0	0.4	16.0	19.8	13.8	0.0	31.1
Mamba-3 (MIMO)	96.2	89.8	26.6	0.0	95.5	**85.8**	32.0	0.0	54.1	**27.4**	14.5	**14.8**	**53.5**	**49.7**	30.8	5.9	33.2	27.6	11.4	3.6	39.6	32.2	22.4	0.4	16.4	23.6	21.0	**3.4**	32.5
Gated DeltaNet-2	94.0	77.3	20.3	8.7	84.5	57.0	20.9	11.2	51.7	4.9	12.3	0.8	47.0	45.3	16.1	19.0	24.6	24.8	14.2	15.4	42.8	37.4	23.2	**17.0**	13.8	12.2	1.4	0.4	28.5
OJA	87.3	69.2	25.6	**11.3**	86.0	49.8	28.4	13.1	60.1	5.8	16.1	4.5	36.6	36.8	15.1	9.8	32.6	30.2	12.8	**17.8**	42.8	40.0	23.4	16.8	5.2	1.0	0.0	0.0	27.8
OJA2	94.5	82.1	27.1	8.2	84.4	66.4	31.7	11.5	57.1	8.6	17.1	2.5	28.9	32.9	25.7	26.7	29.4	20.8	11.6	14.0	39.6	31.2	20.2	15.0	8.9	4.6	0.0	0.0	28.6
Gated Slot Attention	**98.8**	**95.9**	38.7	0.1	90.9	**87.8**	41.1	0.1	45.1	6.4	1.9	11.1	41.2	45.6	3.7	3.1	34.8	**34.2**	13.8	0.0	**48.4**	**45.4**	28.6	1.2	**53.4**	28.5	**25.2**	0.0	33.0
Gated Slot Attention-2	96.0	87.2	41.3	10.6	91.7	83.0	44.1	**17.1**	**68.5**	25.6	**19.0**	7.1	45.9	42.7	**34.1**	**31.3**	**35.6**	25.6	12.6	14.4	47.6	41.0	24.6	16.2	42.6	6.8	0.8	0.2	**36.2**

Table 7: Accuracy on the remaining RULER tasks across context lengths. Avg. is computed over all 28 settings. All non-Transformer models in the attention/hybrid group use SWA. Best and second-best results within each model family are bolded and underlined, respectively.