MARCH: SCALING RECURRENT MEMORY WITH CONTENT-ROUTED STATE ANCHORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers owe much of their strong long-context retrieval capability to a token-level memory that grows with context length. This flexibility, however, incurs quadratic training cost and a key–value cache that grows linearly during autoregressive inference. Recurrent alternatives offer efficient decoding by compressing the entire history into a fixed-size state, but often underperform on recall-intensive tasks since earlier associations usually get overwritten by subsequent updates, and only the most recent contextual information is retained. In this paper, we introduce **M**emory-**A**nchor **R**outing across **C**ontext **H**istory (**MARCH**), a network architecture that effectively scales state-space models beyond a fixed-size dimension, while maintaining computational efficiency over long-sequences. MARCH periodically caches cumulative recurrent-state checkpoints as *state anchors* and associates each anchor with a compact, content-conditioned anchor key. This lets MARCH maintain a memory bank, which can grow as context length increases, providing a controllable trade-off between historical resolution and memory cost. At each token, MARCH produces an anchor query to attend all causally available state anchors, and the output is calculated as an attention-style aggregation over all historical anchors along the current state. We show that after standard pretraining, MARCH consistently outperforms multiple linear attention variants across commonsense reasoning, LongBench, and in-context retrieval. These results demonstrate that content-routed state caching substantially strengthens recurrent long-range memory while preserving its native computation path.

1 INTRODUCTION

Many real-world applications—including long-document understanding, multi-turn interaction, and in-context learning—require models to integrate information distributed across extended sequences (Bai et al., 2025; Kwan et al., 2024; Zou et al., 2025). Supporting such contexts involves more than simply increasing the number of input tokens: models must retain relevant information across long intervening spans and reliably retrieve it when needed (Liu et al., 2024b; Hsieh et al., 2024). Effective long-context modeling therefore hinges on a model's ability to manage memory—determining what information to preserve, how to represent it, and when to retrieve it (Wang et al., 2023; Behrouz et al., 2025). A useful perspective is to view a sequence model as a memory system with two basic operations: *writing*, which incorporates each new input into memory, and *reading*, which retrieves information relevant to the current input (Behrouz et al., 2025). Under this view, standard self-attention maintains a growing token-level memory in its key–value cache (Vaswani et al., 2017): it writes by appending each new key–value pair without compressing the existing cache, and reads by matching the current query against all stored keys and combining their associated values. This uncompressed, token-level memory provides a direct path to every preceding token, enabling accurate recall of fine-grained details and distant dependencies. Its flexibility, however, entails quadratic computation during training and a key–value cache that grows linearly with sequence length during autoregressive inference, making self-attention increasingly costly as context windows expand.

Linear attention and modern recurrent sequence models make the opposite trade-off (Katharopoulos et al., 2020; Dao & Gu, 2024). They compress the causal prefix into a fixed-size, matrix-valued recurrent state, update this state with each new input, and read from it by applying the current query. This design enables constant-memory recurrent decoding, but the same compressed write

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.

that makes it efficient also limits its long-range memory. At each step, information from the new token is written into a state already shared by the entire history. Much recent work has therefore focused on improving the write operation. Selective state-space models such as Mamba introduce input-dependent state transitions and forgetting (Gu & Dao, 2023), whereas DeltaNet and Gated DeltaNet use data-dependent delta-rule updates to revise existing associations before incorporating new information (Schlag et al., 2021; Yang et al., 2025). These mechanisms improve state tracking and mitigate indiscriminate accumulation, but they do not eliminate the underlying fixed-state bottleneck: the entire history must still share one evolving state, and only its latest version remains available for reading. Indeed, although recent linear recurrent models can match or surpass softmax attention in short-context settings, their performance often degrades as the evaluation context grows (Arora et al., 2024a; Wang et al., 2026). Once an earlier association has been weakened by forgetting or modified by subsequent writes, the model has no direct path to its earlier representation and cannot recover it from the latest state alone.

Recent works have relaxed the fixed-state bottleneck along two broad directions. One approach increases the memory capacity available at each step through partitioned sparse states, large memories with sparse reads and writes, or routed mixtures of independent states (Pan et al., 2025; Cabannes et al., 2026; Du et al., 2025). A second line preserves a temporally structured collection of compressed states through logarithmic hierarchies, adaptive state construction and merging, or recurrent-state caching (Guo et al., 2026; Wang et al., 2026). A concurrent work on Memory Caching (Behrouz et al., 2026) is of particular relevance to this work, proposing to store additional memory modules and, in some variants, also includes a selective retrieval. We provide a case study with oracle diagnostic to illustrate the value of historical access: *adding an earlier state readout with a known needle location raises the 16K S-NIAH-2 score of a pretrained Gated DeltaNet model from 37 to 98 with no additional training* (Appendix E). The diagnostic uses the known needle location; representing and retrieving relevant historical states without it remains the challenge.

In this work, we introduce **M**emory-**A**nchor **R**outing across **C**ontext **H**istory (**MARCH**), a memory-augmented recurrent architecture that enables selective retrieval from earlier versions of recurrent memory. Without modifying the underlying recurrence, MARCH periodically preserves cumulative states as *state anchors*, giving later tokens access to earlier versions of the evolving memory. Each anchor is associated with a compact learned descriptor, allowing the model to route each token to relevant historical states when additional context is needed and combine their contents with the current-state readout. MARCH thereby complements efficient recurrent processing with selective access to preserved historical memory. The resulting mechanism remains causal and is trained end to end with the standard language-modeling objective.

Our main contributions are summarized as follows:

- **State-conditioned anchor keys.** We pair checkpoints of a continuously evolving recurrent state with instances of a shared anchor token. Reading the aligned checkpoint conditions the anchor representation in subsequent layers, yielding compact routing keys that reflect the cached state.
- **Selective historical reading.** Each text token scores anchor keys to retrieve historical state readouts and adds their weighted result to the current-state output. A learned null route lets the model bypass historical retrieval when the current state suffices, while residual fusion preserves the recurrent path.
- **Extensive empirical validation.** We demonstrate consistent improvements over strong recurrent baselines across commonsense reasoning, LongBench, in-context retrieval, and NIAH evaluations, including robust extrapolation beyond the training context length.

2 PRELIMINARIES

Full and Linear Attention as Memory. Let $\mathbf{x}_t \in \mathbb{R}^d$ denote the hidden representation at position t. The corresponding query, key, and value vectors are obtained through learned linear projections:

$$\mathbf{q}_t = \mathbf{W}_q \mathbf{x}_t, \qquad \mathbf{k}_t = \mathbf{W}_k \mathbf{x}_t, \qquad \mathbf{v}_t = \mathbf{W}_v \mathbf{x}_t, \tag{1}$$

where $\mathbf{W}_q, \mathbf{W}_k \in \mathbb{R}^{d_k \times d}$ and $\mathbf{W}_v \in \mathbb{R}^{d_v \times d}$, such that $\mathbf{q}_t, \mathbf{k}_t \in \mathbb{R}^{d_k}$ and $\mathbf{v}_t \in \mathbb{R}^{d_v}$. Following the memory-system perspective adopted in prior work (Behrouz et al., 2025), we view a causal

sequence mixer as an online memory system. Let $\mathcal{M}_t$ denote the memory state after processing the first t tokens, with $\mathcal{M}_0$ denoting its initial state. At each position, the current key–value pair is first written into memory, after which the updated memory is queried using the current query:

$$\mathcal{M}_t = \mathrm{Write}\big(\mathcal{M}_{t-1}; \mathbf{k}_t, \mathbf{v}_t\big), \qquad \mathbf{o}_t = \mathrm{Read}\big(\mathcal{M}_t; \mathbf{q}_t\big), \tag{2}$$

where $\mathbf{o}_t \in \mathbb{R}^{d_v}$ denotes the memory readout at position t. For causal softmax attention, the memory explicitly retains all projected key–value pairs observed up to position t:

$$\mathcal{M}_t = \big(\mathbf{K}_{\le t}, \mathbf{V}_{\le t}\big), \tag{3}$$

where $\mathbf{K}_{\le t} \in \mathbb{R}^{t\times d_k}$ and $\mathbf{V}_{\le t} \in \mathbb{R}^{t\times d_v}$ stack the keys and values row-wise, respectively. Writing appends $(\mathbf{k}_t, \mathbf{v}_t)$ to these matrices, whereas reading performs content-based retrieval:

$$\mathbf{o}_t = \mathbf{V}_{\le t}^\top \mathrm{softmax}\left(\frac{\mathbf{K}_{\le t}\mathbf{q}_t}{\sqrt{d_k}}\right). \tag{4}$$

This explicit storage keeps individual tokens directly retrievable and enables fine-grained retrieval from the entire causal prefix. However, processing a sequence of length T requires $\mathcal{O}(T^2)$ query–key interactions, while autoregressive decoding maintains a key–value cache of size $\mathcal{O}(T(d_k + d_v))$ per attention head (Vaswani et al., 2017).

Linear attention instantiates the memory state $\mathcal{M}_t$ as a fixed-size matrix $\mathbf{S}_t \in \mathbb{R}^{d_v\times d_k}$. Its write and read operations are given by

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t\mathbf{k}_t^\top, \qquad \mathbf{o}_t = \mathbf{S}_t\mathbf{q}_t. \tag{5}$$

Each write therefore adds a rank-one key–value association to the shared matrix, while each read retrieves a query-dependent superposition of the stored values. This enables constant-memory recurrent decoding, but introduces interference as the compressed history grows.

Gated DeltaNet (GDN). To mitigate the interference caused by the additive write rule of linear attention, GDN retains the same state-based read operation but introduces input-dependent retention and a targeted delta-rule write (Yang et al., 2025):

$$\mathbf{S}_t = \alpha_t\mathbf{S}_{t-1} + \beta_t\left(\mathbf{v}_t - \alpha_t\mathbf{S}_{t-1}\mathbf{k}_t\right)\mathbf{k}_t^\top, \qquad \mathbf{o}_t = \mathbf{S}_t\mathbf{q}_t, \tag{6}$$

where $\alpha_t \in (0,1)$ is an input-dependent retention gate, and $\beta_t \in [0,1]$ modulates the strength of the targeted delta update. Despite its more adaptive state dynamics, GDN still compresses the entire causal history into a single fixed-size recurrent state. Because all associations share this evolving state, information weakened or overwritten by subsequent updates has no direct retrieval path, limiting reliable long-context recall.

3 METHOD

In Figure 1, MARCH periodically checkpoints the cumulative recurrent state to form a bank of *state anchors*. Each checkpoint is paired with an anchor token whose hidden representation provides a compact, content-conditioned *routing key*. Content-based routing retrieves historical state readouts and adds them to the current-state output without modifying the underlying recurrent update.

3.1 CONTINUOUS RECURRENT-STATE ANCHORING

Anchor placement. Let $\mathcal{T} = [t_1, \ldots, t_L]$ be a sequence of L text tokens. An anchoring policy specifies an ordered set of text boundaries $\mathcal{B} = \{b_m\}_{m=1}^{M}$, where $0 = b_0 < b_1 < \cdots < b_M \le L$. We insert an anchor position after each boundary:

$$\widehat{\mathcal{T}} = \mathop{\Big\|}_{m=1}^{M}\Big([t_{b_{m-1}+1}, \ldots, t_{b_m}] \,\|\, [\xi_m]\Big) \,\|\, [t_{b_M+1}, \ldots, t_L]. \tag{7}$$

where $\|$ denotes sequence concatenation, and ξ_m denotes the m-th occurrence of a dedicated anchor token. All anchor positions use the same learned input embedding. Text and anchor positions serve different computational roles. Text positions apply the base recurrent update, allowing the matrix-valued state to evolve continuously across anchor boundaries. Immediately after processing t_{b_m},

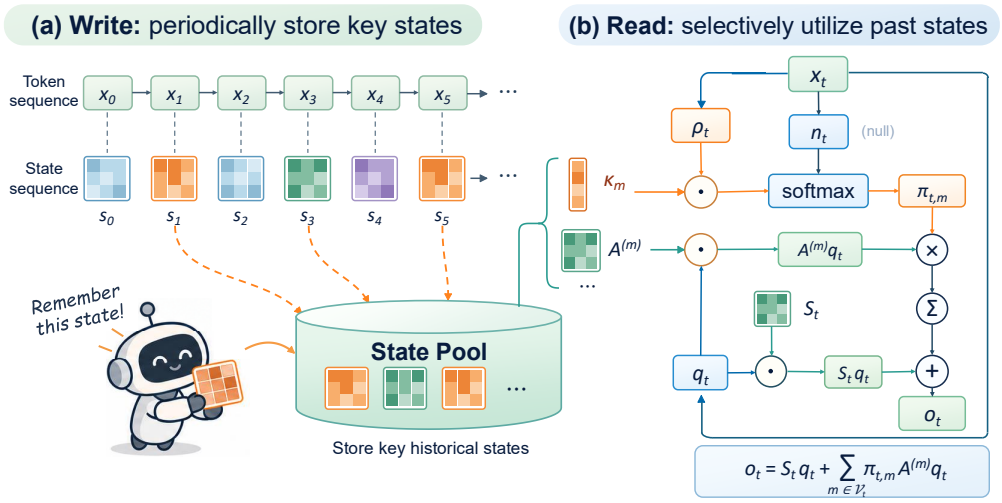

Figure 1: Architecture of MARCH. Left: Periodic checkpoints of the continuously evolving recurrent state form a pool of historical state anchors. Right: A token-dependent routing query matches anchor keys, with a learned null option, to weight historical state readouts. The weighted historical readout is added to the current-state readout.

MARCH checkpoints the resulting cumulative state to form the m-th state anchor. The following anchor position ξ_m does not modify the recurrent state; instead, its hidden representation provides the routing metadata associated with that checkpoint. Thus, each anchor boundary produces two coupled objects: a snapshot of the recurrent memory and a compact representation through which that snapshot can later be retrieved. We formalize these two operations next.

Cumulative recurrent-state checkpointing. At layer ℓ, MARCH leaves the underlying recurrent update unchanged and carries the state $\mathbf{S}_t^{(\ell)} \in \mathbb{R}^{d_v \times d_k}$ continuously across anchor boundaries. At each boundary b_m, it snapshots the current state:

$$\mathbf{A}^{(m,\ell)} = \mathbf{S}_{b_m}^{(\ell)} \in \mathbb{R}^{d_v \times d_k}, \qquad m = 1, \ldots, M. \tag{8}$$

Because the recurrence is not reset between anchor boundaries, $\mathbf{A}^{(m,\ell)}$ encodes the cumulative prefix up to position b_m, rather than only the segment since the preceding boundary. We therefore refer to it as a *state anchor*. The ordered bank $\{\mathbf{A}^{(1,\ell)}, \ldots, \mathbf{A}^{(M,\ell)}\}$ traces the temporal evolution of a single recurrent memory, preserving earlier versions before subsequent decay and delta updates attenuate or modify their contents.

Content-conditioned anchor metadata. Let $\mathbf{u}_m^{(\ell)} \in \mathbb{R}^d$ denote the normalized input at anchor position ξ_m in layer ℓ, and let d_r denote the routing dimension. The anchor reads its aligned state checkpoint and forms a routing key $\boldsymbol{\kappa}_m^{(\ell)} \in \mathbb{R}^{d_r}$ using a dedicated learned projection $\mathbf{W}_\kappa^{(\ell)} \in \mathbb{R}^{d_r \times d}$:

$$\mathbf{q}_m^{(\ell)} = \mathbf{W}_q^{(\ell)} \mathbf{u}_m^{(\ell)}, \quad \mathbf{o}_m^{(\ell)} = \mathbf{A}^{(m,\ell)} \mathbf{q}_m^{(\ell)}, \quad \boldsymbol{\kappa}_m^{(\ell)} = \mathbf{W}_\kappa^{(\ell)} \mathbf{u}_m^{(\ell)}. \tag{9}$$

The aligned readout is incorporated into the anchor position through the standard output projection and residual pathway. Consequently, $\mathbf{u}_m^{(\ell+1)}$ depends on $\mathbf{A}^{(m,\ell)}$, and the routing key produced at layer $\ell + 1$ becomes conditioned on the content retained by the aligned state anchor. Thus, although all anchor positions share the same learned input embedding, they acquire distinct, state-dependent representations after the first layer. This cross-layer construction makes routing explicitly dependent on what each state anchor contains, rather than only on its temporal index.

3.2 Content-Routed Historical Reading

Content-based routing. We omit layer superscripts in this subsection for clarity. For a text token at position t, the causally available state anchors are indexed by $\mathcal{V}_t = \{m \in \{1, \ldots, M\} \mid b_m < t\}$.

MARCH projects the normalized hidden state $\mathbf{x}_t$ into a routing query and scores it against the key of each visible anchor:

$$\boldsymbol{\rho}_t = \mathbf{W}_r \mathbf{x}_t, \qquad a_{t,m} = \boldsymbol{\rho}_t^\top \boldsymbol{\kappa}_m, \quad m \in \mathcal{V}_t. \tag{10}$$

Here $\mathbf{W}_r \in \mathbb{R}^{d_r \times d}$ is the learned routing-query projection, so $\boldsymbol{\rho}_t \in \mathbb{R}^{d_r}$ matches the dimension of $\boldsymbol{\kappa}_m$. To allow the model to bypass historical memory, we augment the visible anchor set with a null option $\varnothing$, whose payload is fixed to zero, $\mathbf{A}^{(\varnothing)} = \mathbf{0}$. Its query-dependent logit is $n_t = \mathbf{w}_\varnothing^\top \mathbf{x}_t + b_\varnothing$. Let $\widetilde{\mathcal{V}}_t = \mathcal{V}_t \cup \{\varnothing\}$ denote the augmented candidate set. We define the logit of each candidate $j \in \widetilde{\mathcal{V}}_t$ as

$$s_{t,j} = \begin{cases} a_{t,j}, & j \in \mathcal{V}_t, \\ n_t, & j = \varnothing, \end{cases} \qquad \pi_{t,j} = \frac{\exp(s_{t,j})}{\sum_{r \in \widetilde{\mathcal{V}}_t} \exp(s_{t,r})}. \tag{11}$$

The routing query $\boldsymbol{\rho}_t$ determines which anchors to retrieve, whereas the state-read query $\mathbf{q}_t$ reads their matrix-valued contents. If no anchor is visible, the null option receives all probability mass.

The aggregation in Equation (11) also admits a sparse Top-K variant: it retains the K highest-scoring visible anchors together with the null option. Applied only at inference, this restriction requires no further training of a dense checkpoint. It connects to hierarchical sparse attention (Lu et al., 2025; Hu et al., 2026) while preserving dense token-level processing rather than pruning text tokens. We evaluate this fixed-checkpoint inference setting in Table 8.

Historical retrieval and residual fusion. Given the routing probabilities, the causally visible state anchors are aggregated into a query-dependent historical state, which is read using the same state-read query as the current state. The resulting historical readout is then added to the current-state readout:

$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t + \sum_{j \in \widetilde{\mathcal{V}}_t} \pi_{t,j} \mathbf{A}^{(j)} \mathbf{q}_t. \tag{12}$$

This additive formulation preserves the original recurrent path and introduces historical retrieval as an auxiliary residual branch, without modifying the underlying recurrent update. Since the routing probabilities directly affect the layer output, the routing queries and anchor-derived keys are optimized end-to-end with the language-modeling objective.

Retrieval cost. Compared with token-level full attention, MARCH routes over $O(L/C)$ recurrent checkpoints, with the interval C controlling the trade-off between historical resolution and cost. Per layer and head, the bank stores $O((L/C)(d_k d_v + d_r))$ elements, and dense historical retrieval costs $O((L^2/C)(d_r + d_k d_v))$ over L tokens, excluding projections and the base recurrence. The quadratic retrieval term is moderated by the checkpoint interval C: with our default $C = 512$, a 64K-token context contains only 128 state anchors. This coarse retrieval granularity substantially reduces the number of query–memory interactions relative to token-level full attention and, together with fused execution, supports the long-context throughput gains observed in Figure 3. Top-K reduces state-read work to $O(LKd_k d_v)$, while scoring all keys costs $O(L^2 d_r/C)$ and the full bank is retained. Fusion reduces intermediate storage and memory traffic without changing these bounds.

3.3 Implementation

We implement MARCH as a two-stage producer–reader computation. Following the hardware-efficient chunkwise formulation of Gated DeltaNet (Yang et al., 2025), the producer processes recurrent updates in blocks amenable to tensor-core acceleration, computes each token's current-state output, and checkpoints the recurrent state at each anchor boundary. The resulting state anchors are consumed by the historical reader. Inspired by the I/O-aware principles of FlashAttention (Dao et al., 2022), the reader jointly tiles query tokens and state anchors, reuses each anchor tile across a block of queries, and fuses routing-score computation, online softmax updates, and the accumulation of weighted state readouts into a streaming reduction. This fused schedule avoids materializing either the dense token-to-anchor routing matrix or the larger tensor of per-anchor candidate readouts, thereby reducing intermediate storage and the associated HBM traffic. As shown in Figure 3, despite the cost of historical retrieval, our fused dense implementation exceeds FlashAttention-2 in throughput at 64K and above and incurs lower core runtime from 32K onward.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Training configuration and baselines. Following Log-Linear Attention (Guo et al., 2026), we pretrain all models from scratch on the same 50B tokens from Long-Data-Collections with 16K sequences and identical optimization settings, detailed in Appendix A. We compare MARCH with Gated DeltaNet (GDN, 793M) (Yang et al., 2025) and its Log-Linear variant (Guo et al., 2026), all using 21 layers, a hidden size of 1536, and six value heads. MARCH uses a routing dimension d_r of 64 and an anchoring interval C of 512 text tokens. We also include a depth-matched 21-layer Transformer (693M) and a 24-layer Transformer (778M) approximately matched to GDN in parameter count; both use a hidden size of 1536, 16 attention heads, and rotary position embeddings (RoPE) (Su et al., 2024) with a base of 500K.

Evaluation tasks. For short-context generalization, we use eight zero-shot commonsense benchmarks: LAMBADA (Paperno et al., 2016), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2020), ARC-Easy and ARC-Challenge (Clark et al., 2018), OpenBookQA (Mihaylov et al., 2018), and CommonsenseQA (Talmor et al., 2019). We additionally evaluate long-context understanding on LongBench (Bai et al., 2024), covering single-document QA, multi-document QA, summarization, and few-shot learning. The long-context retrieval evaluation covers six single-neddle and multi-needle tasks from RULER (Hsieh et al., 2024) at 4K, 8K, and 16K context lengths. Finally, the in-context retrieval suite contains SQuAD (Rajpurkar et al., 2018), TriviaQA (Joshi et al., 2017), SWDE (Lockard et al., 2019), FDA (Arora et al., 2023), Natural Questions (Kwiatkowski et al., 2019), and DROP (Dua et al., 2019). We follow the evaluation protocol of prior work (Wang et al., 2026) and use the LM-Evaluation-Harness (Gao et al., 2021).

Table 1: Zero-shot performance of MARCH and baseline models on eight commonsense reasoning benchmarks. Results are reported using accuracy (`acc`) or normalized accuracy (`acc_n`), as indicated in the column headers; higher is better (↑). The best result among Gated DeltaNet variants in each column is highlighted in bold.

Model	LMB. acc ↑	PIQA acc ↑	Hella. acc ↑	Wino. acc ↑	ARC-e acc_n ↑	ARC-c acc_n ↑	OBQA acc_n ↑	CSQA acc ↑	Avg.
Transformer	49.4	66.5	33.9	52.1	47.8	26.4	32.0	22.1	41.3
w/ *24 Layers*	50.3	67.6	34.4	50.6	46.3	25.8	31.2	24.7	41.4
Gated DeltaNet	48.5	66.1	33.1	50.8	45.9	25.3	30.0	21.1	40.1
w/ *Log-Linear*	47.7	65.7	33.2	51.9	44.3	24.9	30.4	21.7	40.0
w/ *MARCH*	**49.5**	**66.9**	**34.8**	**52.6**	**47.1**	**25.6**	**32.8**	**22.5**	**41.5**

4.2 MAIN RESULTS

Commonsense reasoning. As shown in Table 1, MARCH consistently outperforms both the vanilla and Log-Linear variants of Gated DeltaNet across all eight zero-shot commonsense reasoning benchmarks. It improves the average accuracy from 40.1 and 40.0 to 41.5, respectively, with the largest gain over the vanilla baseline observed on OpenBookQA ($+2.8$ points), aligning with our findings in retrieval tasks presented below. Moreover, MARCH achieves a higher average score than both Transformer baselines, surpassing the standard Transformer on six of eight tasks and the 24-layer Transformer on four. These results indicate that MARCH consistently strengthens the Gated DeltaNet backbone while remaining competitive with comparable full-attention models on short-context language understanding tasks.

Needle-in-a-haystack retrieval. We evaluate long-context associative retrieval using the needle-in-a-haystack (NIAH) suite from RULER (Hsieh et al., 2024), where a model must recover values associated with keys embedded among irrelevant context. All models are trained with a maximum context length of 16K; Figure 2 reports results from 4K to 32K, making 32K a zero-shot length-extrapolation setting. Across the 24 task–length combinations, MARCH outperforms the stronger recurrent baseline in 19 settings and matches it in the remaining five. On the multi-needle tasks, it

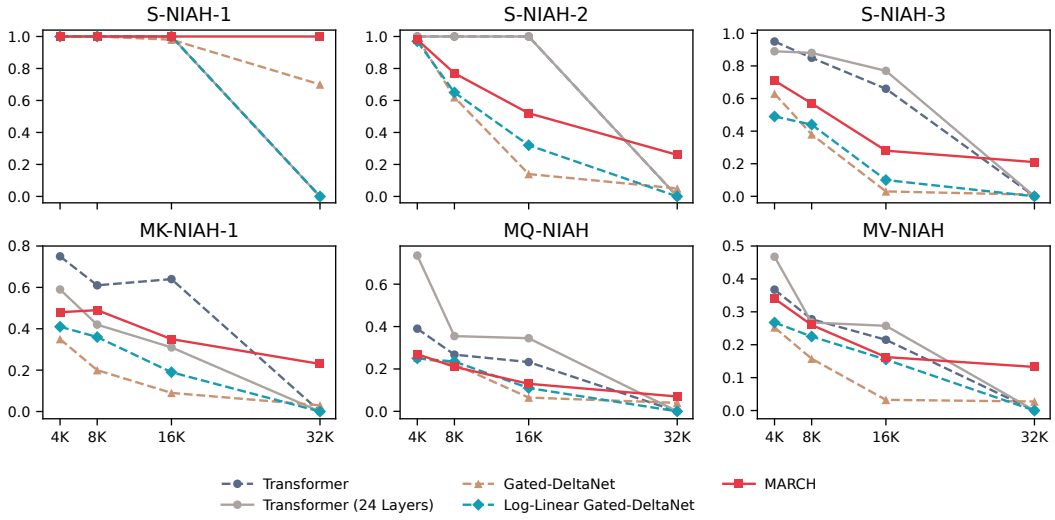

Figure 2: NIAH performance on three single-needle and three multi-needle tasks. The Transformer achieves perfect accuracy on both S-NIAH-1 and S-NIAH-2 at context lengths of 4K, 8K, and 16K.

wins in 11 of 12 settings. At 32K, MARCH achieves the best result on all six tasks, retaining perfect accuracy on S-NIAH-1 and nonzero accuracy on the remaining tasks, whereas both Transformer variants and Log-Linear Gated-DeltaNet score zero throughout. This contrast is consistent with RoPE extrapolation in the Transformers and the state-index-dependent coefficients of Log-Linear Gated-DeltaNet. MARCH instead shares the same content-based router across all anchors, allowing longer contexts to introduce additional anchors without requiring new specific routing parameters.

Table 2: Results on twelve LongBench tasks. The best result among Gated DeltaNet and its variants is shown in bold for each task. The relative average gain over the Gated DeltaNet is shown in green.

	Single-Doc QA			**Multi-Doc QA**			**Summarization**			**Few-shot Learning**			
Model	NQA	QQA	MFQ	HQA	2WM	Mus	GvR	QMS	MNs	TRC	TQA	SSM	**Avg.**↑
Transformer	4.4	4.1	15.9	7.5	9.9	4.1	10.7	11.6	14.5	21.0	33.9	28.3	13.8
w/ *24 Layers*	3.5	11.1	18.3	7.8	9.8	4.1	11.3	12.9	12.8	22.5	47.5	23.2	15.4
Gated DeltaNet	3.0	4.8	13.3	5.6	8.7	2.1	2.6	11.3	13.0	18.0	38.6	21.9	11.9
w/ *Log-Linear*	3.6	6.2	13.6	7.1	8.2	3.3	6.3	13.2	13.5	17.0	32.8	25.1	12.5
w/ MARCH	**4.2**	**7.8**	**14.6**	**7.4**	**11.5**	**4.8**	**8.2**	**17.4**	**14.1**	**19.0**	**43.1**	**26.3**	**14.9** (↑ 25%)

Long-context understanding and in-context retrieval. MARCH outperforms both vanilla and Log-Linear Gated DeltaNet on all 12 LongBench tasks (Table 2). Relative to the stronger recurrent baseline on each task, it improves multi-document QA on MuSiQue from 3.3 to 4.8 and summarization on QMSum from 13.2 to 17.4. On the six in-context retrieval benchmarks (Arora et al., 2024b), MARCH also outperforms both recurrent baselines on every task, with average accuracy increasing from 20.5 for Log-Linear Gated DeltaNet to 23.3, a relative gain of approximately 14% (Table 6 in Appendix B).

Training efficiency. Figure 3 compares the end-to-end throughput and core forward–backward runtime of FlashAttention-2, Gated DeltaNet, dense MARCH, and its Top-4 implementation. Sparse routing becomes increasingly beneficial as the context grows. At 128K tokens, Top-4 MARCH more than doubles the training throughput of dense MARCH and reduces its core runtime by roughly an order of magnitude. It also achieves higher throughput than FlashAttention-2 at this length, although vanilla Gated DeltaNet remains faster because it incurs no historical-retrieval overhead.

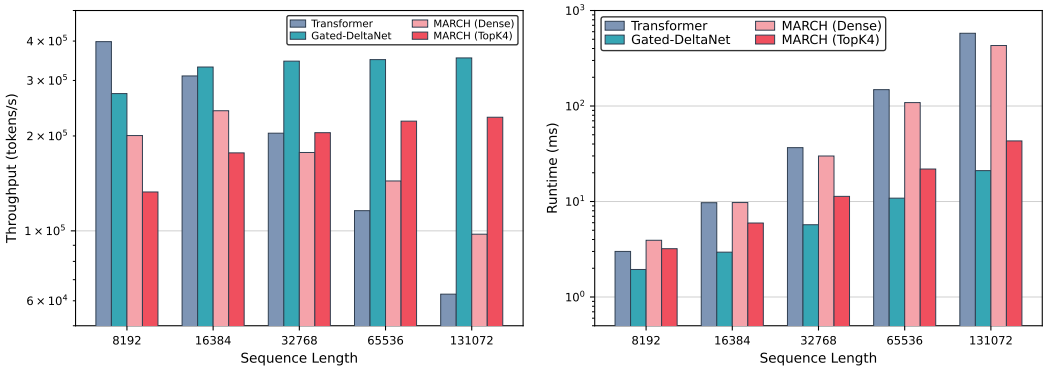

Figure 3: Training efficiency across sequence lengths. Left: end-to-end training throughput in tokens per second (higher is better). Right: forward–backward runtime of the core sequence-mixing operation in milliseconds (lower is better). MARCH (Top-4) retains only the four highest-scoring state anchors for each token and head during historical retrieval.

5 Ablation Studies

Generality across recurrent backbones. We evaluate MARCH on GDN, Kimi Delta Attention (KDA) (Kimi Team, 2025), and Gated DeltaNet-2 (GDN2) (Hatamizadeh et al., 2026) under the shared pretraining protocol in Section 4.1. As shown in Table 3, MARCH improves NIAH in 11 of 12 backbone–length settings, with average gains of 14.85, 9.74, and 4.28 points for GDN, KDA, and GDN2, respectively. SWDE and FDA improve across all three backbones, while results on SQuAD vary by backbone. Short-context performance and training perplexity remain broadly similar.

Table 3: Comparison of GDN, KDA, and GDN2 with and without MARCH. NIAH scores are averaged over six needle-in-a-haystack tasks. The best result in each column is shown in bold.

	Train	Short-Context↑		In-Context Retrieval↑			NIAH (6-Task Avg.)↑				
Model	**PPL↓**	**LAMB.**	**Hella.**	**SQuAD**	**SWDE**	**FDA**	**4K**	**8K**	**16K**	**32K**	**Avg.**
GDN	6.796	48.52	33.10	34.85	45.00	31.40	53.80	35.67	21.73	15.13	31.58
w/ *MARCH*	6.751	49.52	34.84	37.70	51.85	44.56	59.58	54.96	39.46	**31.71**	**46.43**
KDA	6.691	52.11	35.41	38.64	51.94	37.93	51.17	34.75	22.54	18.08	31.64
w/ *MARCH*	6.703	**52.57**	34.95	33.58	**54.10**	42.83	58.96	52.25	32.13	22.17	41.38
GDN2	6.663	51.00	**35.58**	**39.11**	51.76	38.20	**61.83**	46.88	34.38	19.00	40.52
w/ *MARCH*	**6.610**	51.41	35.36	38.71	53.65	**48.46**	57.58	**56.00**	**40.46**	25.17	44.80

Positional encoding and length extrapolation. We evaluate RoPE in hybrid models with a 3:1 recurrent-to-attention layer ratio, trained with a maximum context length of 16K. Table 4 reports NIAH scores from 4K to 64K, with 32K and 64K testing extrapolation beyond training. Removing RoPE improves MARCH at all five lengths, most notably at 32K (13.79 to 68.79) and 64K (4.13 to 30.25). GDN also improves beyond 16K without RoPE, but loses accuracy at shorter lengths. Without RoPE, MARCH outperforms GDN at every evaluated length, with a five-length NIAH average of 68.26 versus 50.89 for GDN.

Effect of chunk size. Among training chunk sizes from 256 to 2048, $C = 512$ leads on four of the six reported metrics and uses half as many anchors as $C = 256$, motivating our default. For the checkpoint trained with $C = 512$, using $C = 256$ at inference improves all three in-context retrieval scores, including SQuAD from 37.70 to 40.35, while doubling the anchor count. Larger intervals and Fenwick-tree retention reduce memory at a cost to retrieval accuracy. Full results and settings are provided in Appendix D (Table 9).

Table 4: Effect of RoPE in hybrid models trained up to 16K context. Results at 32K and 64K evaluate length extrapolation. The Transformer baseline is a parameter-matched 24-layer model. NIAH Avg. is averaged across five context lengths. The best result in each column is shown in bold.

		In-Context Retrieval↑			NIAH (6-Task Avg.)↑					
Model	**RoPE**	**SQuAD**	**SWDE**	**FDA**	**4K**	**8K**	**16K**	**32K**	**64K**	**Avg.**
MARCH	✓	40.82	58.42	78.58	76.88	72.29	66.88	13.79	4.13	46.79
	×	41.12	**65.44**	80.76	**87.79**	**79.92**	**74.54**	**68.79**	**30.25**	**68.26**
GDN	✓	**43.40**	58.42	76.77	79.96	73.58	65.79	13.29	9.08	48.34
	×	40.11	65.17	75.23	68.00	61.04	55.63	48.46	21.33	50.89
Transformer	✓	40.48	64.99	**83.76**	78.04	65.38	66.88	0.00	0.00	42.06

Routing design. Top-4 can be applied at inference without retraining. On a fixed dense-trained checkpoint, average NIAH changes from 51.21 to 51.92; on a fixed read-query-adapter checkpoint, from 55.67 to 52.76 (Table 8). Thus, it reduces historical-read work with modest changes in average NIAH accuracy. Other routing ablations appear in Appendix C.

6 Related Work

Efficient sequence models reduce attention costs through kernelization, sparsity, or recurrent compression. Linear attention and selective or gated recurrences enable efficient decoding with fixed-size memory (Katharopoulos et al., 2020; Dao & Gu, 2024; Yang et al., 2025), but compressing the entire history into one state limits long-range retrieval (Arora et al., 2024b).

Recent work expands memory through sparse state banks and mixtures (Pan et al., 2025; Cabannes et al., 2026; Du et al., 2025), or combines recurrence with structured historical access (Guo et al., 2026; Wang et al., 2026; Wei et al., 2025; Wei & Gulcehre, 2026). Concurrent Memory Caching (Behrouz et al., 2026) also stores recurrent-state checkpoints; MARCH instead derives routing keys from anchor tokens conditioned on aligned checkpoints across layers. Sparse attention, including MoBA and Native Sparse Attention, routes to token- or block-level KV memories (Lu et al., 2025; Yuan et al., 2025), whereas MARCH retrieves readouts from compressed recurrent states. Appendix F provides a detailed review.

7 Conclusion and Future Work

We introduced MARCH, a recurrent memory architecture that retains cumulative state checkpoints and uses anchor-derived keys to route historical reads. The historical readouts augment the current-state output without changing the underlying recurrent update. Across commonsense reasoning, LongBench, and in-context retrieval, MARCH improves on the evaluated recurrent baselines, with gains also observed across multiple recurrent backbones. Ablations show that checkpoint density and sparse routing provide controllable trade-offs among retrieval quality, storage, and computation.

Limitations and Future Work. MARCH adopts periodic checkpointing and organizes all historical states in a single homogeneous anchor bank. Although this design is simple and efficient, fixed-interval anchoring does not account for the non-uniform evolution of recurrent memory, resulting in redundant anchors in stable regions while providing insufficient resolution when the context changes rapidly. A natural extension is to develop adaptive anchoring mechanisms and to consolidate or evict redundant anchors given a memory budget. In addition, fixed-interval checkpoints with dense retrieval add quadratic computation at fixed spacing. MARCH also has the potential to support external memory modules for optimized performance over specific downstream tasks, knowledge consolidation from experience to parametric information, and other memory manipulation mechanisms, opening up new scaling directions for test-time training and continual learning.

AI USE STATEMENT

During the preparation of this work, generative AI tools provided assistance with manuscript editing and programming. The authors reviewed the resulting text and code, making revisions and performing additional checks where appropriate. All claims, results, and conclusions presented in this paper remain the responsibility of the authors.

REFERENCES

Arshia Afzal, Aviv Bick, Eric P Xing, Volkan Cevher, and Albert Gu. Raven: High-recall sequence modeling with sparse memory routing. *arXiv preprint arXiv:2607.25357*, 2026. URL https://arxiv.org/abs/2607.25357.

Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes. *Proceedings of the VLDB Endowment*, 17(2):92–105, 2023. doi: 10.14778/3626292.3626294. URL https://www.vldb.org/pvldb/vol17/p92-arora.pdf.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. In *Proceedings of 12th International Conference on Learning Representations (ICLR)*. ICLR, 2024a. URL https://openreview.net/forum?id=LY3ukUANko.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 1763–1840. PMLR, 2024b. URL https://proceedings.mlr.press/v235/arora24a.html.

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. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, et al. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664, 2025. doi: 10.18653/v1/2025.acl-long.183. URL https://aclanthology.org/2025.acl-long.183/.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3786. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4ca07aa108036f80cbb5b82285fd4b1-Abstract-Conference.html.

Ali Behrouz, Zeman Li, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Memory caching: Rnns with growing memory. In *Forty-third International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.24281.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020. URL https://arxiv.org/abs/2004.05150.

Vincent-Pierre Berges, Barlas Oğuz, Daniel Haziza, Wen tau Yih, Luke Zettlemoyer, and Gargi Ghosh. Memory layers at scale, 2024. URL https://arxiv.org/abs/2412.09764.

Yonatan Bisk, Rowan Zellers, Ronan LeBras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 7432–7439. AAAI Press, 2020.

doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

Loïc Cabannes, Pierre-Emmanuel Mazaré, Gergely Szilvasy, Matthijs Douze, Maria Lomeli, Ilze Amanda Auzina, Justin Carpentier, Gabriel Synnaeve, and Hervé Jégou. Sparse delta memory: Scaling the state of linear rnns through sparsity. *arXiv preprint arXiv:2607.07386*, 2026. URL https://arxiv.org/abs/2607.07386.

Loïc Cabannes, Maximilian Beck, Gergely Szilvasy, Matthijs Douze, Maria Lomeli, Jade Copet, Pierre-Emmanuel Mazaré, Gabriel Synnaeve, and Hervé Jégou. Short window attention enables long-term memorization, 2025. URL https://arxiv.org/abs/2509.24552.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3829–3846, 2023. doi: 10.18653/v1/2023.emnlp-main.232. URL https://aclanthology.org/2023.emnlp-main.232/.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. URL https://arxiv.org/abs/1904.10509.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

Yuhong Chou, Man Yao, Kexin Wang, Yuqi Pan, Ruijie Zhu, Yiran Zhong, Yu Qiao, Jibin Wu, Bo Xu, and Guoqi Li. MetaLA: Unified optimal linear approximation to softmax attention map. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-2270. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/8329a45669017898bb0cc09d27f8d2bb-Abstract-Conference.html.

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*, abs/1803.05457, 2018. URL https://arxiv.org/abs/1803.05457.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mZn2Xyh9Ec.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10041–10071. PMLR, 2024. URL https://proceedings.mlr.press/v235/dao24a.html.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16344–16359, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract-Conference.html.

Chenlong Deng, Zhisong Zhang, Kelong Mao, Shuaiyi Li, Tianqing Fang, Hongming Zhang, Haitao Mi, Dong Yu, and Zhicheng Dou. Unigist: Towards general and hardware-aligned sequence-level long context compression. *arXiv preprint arXiv:2509.15763*, 2025. URL https://arxiv.org/abs/2509.15763.

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

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *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. 2368–2378, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1246. URL https://aclanthology.org/N19-1246.

Peter M. Fenwick. A new data structure for cumulative frequency tables. *Software: Practice and Experience*, 24(3):327–336, 1994. doi: 10.1002/spe.4380240306. URL https://doi.org/10.1002/spe.4380240306.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 2021. URL https://zenodo.org/records/5371629.

Riccardo Grazzi, Julien Siems, Arber Zela, Jörg K. H. Franke, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear rnns through negative eigenvalues. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5a0ce3abb720b740419e193c87afd080-Abstract-Conference.html.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. URL https://arxiv.org/abs/2312.00752.

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

Han Guo, Songlin Yang, Tarushii Goel, Eric P. Xing, Tri Dao, and Yoon Kim. Log-linear attention. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=mOJgZWkXKW.

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.

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

Xiang Hu, Xinyu Wei, Hao Gu, Minshen Zhang, Tian Liang, Huayang Li, Lei Zhu, Yan Wang, Sirui Han, Yushi Bai, et al. Hierarchical sparse attention done right: Toward infinite context modeling. *arXiv preprint arXiv:2607.02980*, 2026. URL https://arxiv.org/abs/2607.02980.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165. PMLR, 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

Kimi Team. Kimi Linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025. URL https://arxiv.org/abs/2510.26692.

Wai-Chung Kwan, Xingshan Zeng, Yuxin Jiang, Yufei Wang, Liangyou Li, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. Mt-eval: A multi-turn capabilities evaluation benchmark for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 20153–20177, 2024. doi: 10.18653/v1/2024.emnlp-main.1124. URL https://aclanthology.org/2024.emnlp-main.1124/.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Guillaume Lample, Alexandre Sablayrolles, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Large memory layers with product keys, 2019. URL https://arxiv.org/abs/1907.05242.

Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. Longhorn: State space models are amortized online learners. *arXiv preprint arXiv:2407.14207*, 2024a. URL https://arxiv.org/abs/2407.14207.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024b. URL https://aclanthology.org/2024.tacl-1.9/.

Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. OpenCeres: When open information extraction meets the semi-structured web. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *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. 3047–3056, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1309. URL https://aclanthology.org/N19-1309.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, et al. MoBA: Mixture of block attention for long-context LLMs. *arXiv preprint arXiv:2502.13189*, 2025. URL https://arxiv.org/abs/2502.13189.

Yuzhen Mao, Michael Y. Li, and Emily B. Fox. Simplified sparse attention via gist tokens, 2026. URL https://arxiv.org/abs/2604.20920.

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. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Jesse Mu, Xiang Li, and Noah Goodman. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36:19327–19352, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/3d77c6dcc7f143aa2154e7f4d5e22d68-Abstract-Conference.html.

Matanel Oren, Michael Hassid, Yossi Adi, and Roy Schwartz. Transformers are multi-state rnns. *ArXiv preprint*, abs/2401.06104, 2024. URL https://arxiv.org/abs/2401.06104.

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. URL https://proceedings.mlr.press/v202/orvieto23a.html.

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

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 Katrin Erk and Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, et al. Rwkv: Reinventing rnns for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 14048–14077, 2023. doi: 10.18653/v1/2023.findings-emnlp.936. URL https://aclanthology.org/2023.findings-emnlp.936/.

Aleksandar Petrov, Mark Sandler, Andrey Zhmoginov, Nolan Miller, and Max Vladymyrov. Long context in-context compression by getting to the gist of gisting. *arXiv preprint arXiv:2504.08934*, 2025. URL https://arxiv.org/abs/2504.08934.

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

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In Iryna Gurevych and Yusuke Miyao (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 784–789, Melbourne, Australia, 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2124. URL https://aclanthology.org/P18-2124.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021. doi: 10.1162/tacl_a_00353. URL https://aclanthology.org/2021.tacl-1.4.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 8732–8740. AAAI Press, 2020. doi: 10.1609/aaai.v34i05.6399. URL https://ojs.aaai.org/index.php/AAAI/article/view/6399.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers, 2021. URL https://arxiv.org/abs/2102.11174.

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. DeltaProduct: Improving state-tracking in linear RNNs via householder products. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-5141. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/e1ea2520fbf9cd1600b287dde67e0a3c-Abstract-Conference.html.

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

Weigao Sun, Zhen Qin, Dong Li, Xuyang Shen, Yu Qiao, and Yiran Zhong. Linear attention sequence parallelism. *ArXiv preprint*, abs/2404.02882, 2024. URL https://arxiv.org/abs/2404.02882.

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

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. 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. 4149–4158. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421/.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context LLM inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 47901–47911. PMLR, 2024. URL https://proceedings.mlr.press/v235/tang24l.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Bailin Wang, Chang Lan, Chong Wang, and Ruoming Pang. Rattention: Towards the minimal sliding window size in local-global attention models, 2025. URL https://arxiv.org/abs/2506.15545.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. *Advances in Neural Information Processing Systems*, 36:74530–74543, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/ebd82705f44793b6f9ade5a669d0f0bf-Abstract-Conference.html.

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

Xiuying Wei and Caglar Gulcehre. RAT+: Train dense, infer sparse – recurrence augmented attention for dilated inference. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.18196.

Xiuying Wei, Anunay Yadav, Razvan Pascanu, and Caglar Gulcehre. RAT: Bridging RNN efficiency and attention accuracy via chunk-based sequence modeling. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2507.04416.

Kaiyue Wen, Xingyu Dang, and Kaifeng Lyu. RNNs are not Transformers (Yet): The Key Bottleneck on In-context Retrieval. *ArXiv preprint*, abs/2402.18510, 2024. URL https://arxiv.org/abs/2402.18510.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14138–14148, 2021. doi: 10.1609/aaai.v35i16.17664. URL https://ojs.aaai.org/index.php/AAAI/article/view/17664.

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*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d13a3eae72366e61dfdc7eea82eeb685-Abstract-Conference.html.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=r8H7xhYPwz.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097, 2025. doi: 10.18653/v1/2025.acl-long.1126. URL https://aclanthology.org/2025.acl-long.1126/.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. Long context compression with activation beacon. *arXiv preprint arXiv:2401.03462*, 2024. URL https://arxiv.org/abs/2401.03462.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2O: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/6ceefa7b15572587b78ecfcebb2827f8-Abstract-Conference.html.

Tianyu Zhao and Llion Jones. Fast-weight product key memory, 2026. URL https://arxiv.org/abs/2601.00671.

Kaijian Zou, Muhammad Khalifa, and Lu Wang. On many-shot in-context learning for long-context evaluation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25605–25639, 2025. doi: 10.18653/v1/2025.acl-long.1245. URL https://aclanthology.org/2025.acl-long.1245/.

A Training Details

Table 5 lists the optimization settings shared by all models in Section 4.1.

Table 5: Shared pretraining optimization settings.

Setting	Value
Optimizer	Fused AdamW
Global batch size	Approximately 4.2M tokens
Adam β_1, β_2	0.9, 0.95
Adam ϵ	10^{-8}
Weight decay	0.1
Peak learning rate	4×10^{-4}
Learning-rate schedule	Warmup-stable-decay

B In-Context Retrieval Results

Table 6 reports the per-task results summarized in Section 4.2.

Table 6: In-context retrieval accuracy (↑). The best result among Gated DeltaNet and its variants is bold; relative gains over the stronger recurrent baseline are shown in green below the MARCH scores.

Model	SQuAD↑	SWDE↑	FDA↑	TriviaQA↑	DROP↑	NQ↑	Avg.↑
Transformer	41.3	59.3	80.4	2.2	2.9	1.7	31.3
w/ *24 Layers*	40.4	64.9	83.7	4.0	3.4	2.5	33.2
Gated DeltaNet	34.8	45.0	31.4	1.1	2.2	0.8	19.2
w/ *Log-Linear*	33.7	46.1	38.2	1.3	2.6	1.0	20.5
w/ *MARCH*	**37.7**	**51.9**	**44.6**	**1.6**	**2.9**	**1.2**	**23.3**
	(↑ 8%)	(↑ 13%)	(↑ 17%)	(↑ 23%)	(↑ 12%)	(↑ 20%)	(↑ 14%)

C Routing and Historical Reading Design

Table 7 varies routing dimension d_r, sparsity, the null option, and the historical read query around the dense $d_r = 64$ default. Top-4 retains four anchors per token and head. Larger d_r improves retrieval but lowers other averages; trained Top-4 lowers NIAH, and removing the null option lowers every aggregate.

Table 7: Routing and historical-read ablations. Top-4 is independently trained. CS, LongBench, Retrieval, and NIAH average 8, 12, 6, and 6 tasks; column-wise best results are bold.

Configuration			Benchmark Averages			NIAH			
Variant	d_r	Null	CS	LongBench	Retrieval	4K	8K	16K	Avg.
Dense (default)	64	Yes	41.48	**14.87**	23.31	59.58	54.96	39.46	51.33
Dense	192	Yes	40.94	13.88	**24.52**	56.71	51.33	37.38	48.47
Top-4	64	Yes	41.38	13.79	23.17	57.29	46.04	31.21	44.85
Dense	64	No	41.04	14.11	22.86	54.88	47.92	35.13	45.98
Read-query adapter	64	Yes	**42.67**	13.57	21.74	**60.62**	**57.50**	**48.88**	**55.67**

In the final row, current reads retain $\mathrm{Norm}(\mathbf{q}_{t,h})$, while historical reads use $\mathrm{Norm}(\mathbf{q}_{t,h} + \mathbf{B}_h\mathbf{A}_h\mathbf{q}_{t,h})$. The head-specific rank-32 residual has zero-initialized $\mathbf{B}_h$ and is trained with the model (about 1.47M extra parameters); routing is unchanged. CS and NIAH improve, whereas LongBench and in-context retrieval decline.

Inference-time ablation of history aggregation. We freeze two checkpoints: the dense-trained model (*Standard*) and the read-query-adapter model described above (*Adapter*, checkpoint-11921). Within each checkpoint, only inference-time aggregation changes; state construction, evaluation examples, and decoding settings are fixed. For Adapter, changes apply to all historical-reading layers during both prefill and decoding, with 100 examples per task and context length.

Dense uses learned weights over all visible anchors and the null route. Top-4 retains the four highest-scoring anchors (or all anchors if fewer are visible) and renormalizes their weights with the null route. Uniform, evaluated on Adapter, bypasses both learned weighting and the null route and averages the historical readouts:

$$\mathbf{o}_t = \mathbf{o}_t^{\text{current}} + \frac{1}{M_t} \sum_{j=1}^{M_t} \mathbf{r}_{t,j}, \qquad M_t > 0, \tag{13}$$

where M_t is the number of visible anchors and $\mathbf{r}_{t,j}$ is the readout from anchor j. When $M_t = 0$, only the current-state output is used.

Table 8: Inference-time history aggregation on NIAH (%). Scores at each length average six tasks; Overall averages all 18 task–length settings. Standard and Adapter are the dense-trained and read-query-adapter checkpoints, respectively. Model weights are fixed within each group.

Checkpoint	Aggregation	4K	8K	16K	Overall
Standard	Dense	59.67	54.67	39.29	51.21
	Top-4	60.58	54.62	40.54	51.92
Adapter	Dense	60.62	57.50	48.88	55.67
	Top-4	60.79	54.17	43.33	52.76
	Uniform	43.38	34.62	15.50	31.17

Table 8 reports six-task averages at each length. On Standard, inference-time Top-4 changes Overall from 51.21 to 51.92, with slightly higher averages at 4K and 16K and little change at 8K. On Adapter, it changes Overall from 55.67 to 52.76; the 4K averages are similar, but Top-4 trails at 8K and 16K. Uniform averages 31.17 on Adapter and falls to 15.50 at 16K, versus 48.88 for dense aggregation. Its 24.50-point overall deficit measures the joint removal of learned weighting and the null route.

D Chunk Size and State-Bank Organization

The chunk size C controls the interval between recurrent-state checkpoints. Table 9 reports three in-context retrieval benchmarks and NIAH scores averaged over six tasks at each context length. Panel (a) varies C from 256 to 2048 during both training and inference, with all other model and training settings fixed. Panel (b) uses the checkpoint trained with $C = 512$ and varies only the inference-time interval. Anchor counts are reported for a 16K context.

We also evaluate inference-time retention using the Fenwick tree scheme of Log-Linear Attention (Guo et al., 2026; Fenwick, 1994), which hierarchically organizes the state bank and retains $\mathcal{O}(\log T)$ anchors. This reduces the retained bank to six anchors in the reported setting, but lowers SWDE from 51.85 to 36.79 and FDA from 44.56 to 22.35 relative to the default interval of 512. Denser retention is not uniformly better either: inference with $C = 256$ outperforms $C = 64$ and $C = 128$ on all three in-context retrieval benchmarks, despite using fewer anchors.

Table 9: Effect of chunk size on in-context retrieval and NIAH. Panel (a) evaluates checkpoints using the same chunk size during training and inference. Panel (b) varies the inference-time chunk size for the checkpoint trained with chunk size 512. NIAH scores are averaged over six tasks at each context length. Bold indicates the selected setting in each panel and the best result in each column.

Chunk Size	# Anchors	In-context Retrieval SQuAD	SWDE	FDA	NIAH 4K	8K	16K
(a) Matched training and inference chunk sizes							
256	64	36.76	49.23	**44.83**	58.17	49.25	**44.83**
512	32	**37.70**	**51.85**	44.56	**59.58**	**54.96**	39.46
1024	16	36.49	48.51	43.28	53.21	43.21	33.54
2048	8	34.15	44.19	30.76	59.54	39.83	27.96
(b) Inference-time chunk size							
64	256	39.41	53.11	46.01	**63.33**	55.00	39.67
128	128	39.08	52.84	46.91	**63.33**	**57.33**	39.17
256	64	**40.35**	**53.38**	**47.46**	62.83	56.17	**40.83**
512	32	37.70	51.85	44.56	59.58	54.96	39.46
1024	16	37.23	43.74	34.57	55.33	48.33	32.67
2048	8	37.23	33.93	23.23	51.00	40.33	34.00
Fenwick Tree	6	36.16	36.79	22.35	52.17	39.56	33.14

E HISTORICAL-STATE RETRIEVAL: AN ORACLE DIAGNOSTIC

Oracle state read in GDN. We first intervene on a pretrained Gated DeltaNet (GDN) at inference time, without changing its parameters. On S-NIAH-1/2/3 at 4K, 8K, and 16K, let p be the needle's starting position. In layers 2–21, we save the state $\mathbf{S}^{(\ell)}_{p+64}$ and, from the first generated token onward, add its query-dependent readout with coefficient one to the current-state readout. The oracle uses the known needle position to select this fixed snapshot.

Oracle selection in trained MARCH. We next use the standard, fully recurrent MARCH model from Section 4.1, with $C = 512$ and routing dimension $d_r = 64$. Without retraining, the oracle selects the first 512-token snapshot after the complete target value, using its true location. In layers 2–21, it replaces the learned dense historical mixture (softmax over visible snapshots and the null route) with a unit-weight read from this snapshot; the current-state branch remains unchanged:

$$\mathbf{o}_t^{\text{dense}} = \mathbf{S}_t\mathbf{q}_t + \sum_{j\in\mathcal{V}_t} \pi_{t,j}\mathbf{A}^{(j)}\mathbf{q}_t, \qquad \mathbf{o}_t^{\text{oracle}} = \mathbf{S}_t\mathbf{q}_t + \mathbf{A}^{(j^*)}\mathbf{q}_t \quad (j^* \in \mathcal{V}_t), \tag{14}$$

Here j^* is the selected snapshot; the null route is omitted from the dense sum because its readout is zero.

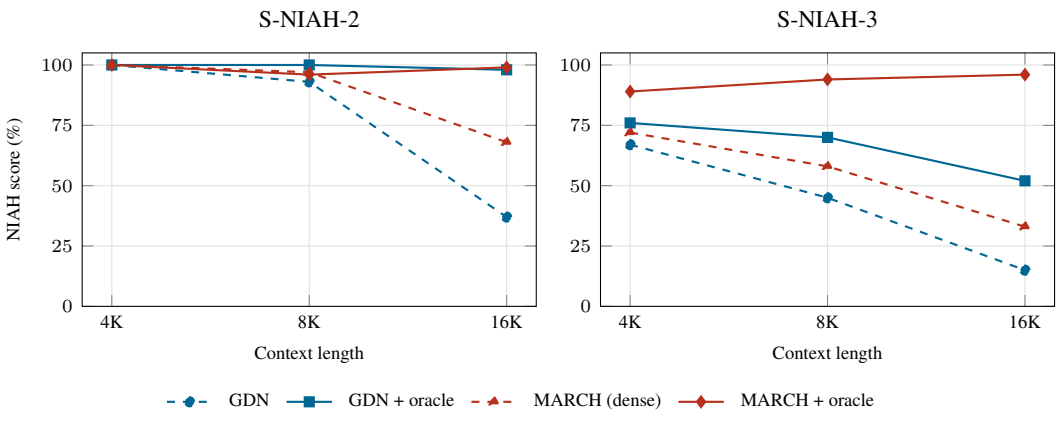

Figure 4: Oracle historical-state reads on S-NIAH-2 and S-NIAH-3. Dashed curves use the original GDN or dense MARCH read; solid curves use the corresponding unit-weight oracle read. The GDN snapshot is taken 64 tokens after the needle starts; MARCH selects the first checkpoint after the complete target value.

The change applies to the last prompt query and all decode queries; until j^* is causally visible, the dense read is retained, and these examples remain in the scores. Both settings use the same inputs and checkpoint, with 100 examples per task and length, zero-shot greedy decoding (up to 128 tokens), bf16, batch size one, and caching enabled. The oracle changes both selection and readout weight, so this is neither a null-gate ablation nor an upper bound for learned retrieval.

In GDN, S-NIAH-1 stays at 100 across all three lengths in both settings. At 16K, its oracle read raises S-NIAH-2 from 37 to 98 and S-NIAH-3 from 15 to 52. For trained MARCH, the oracle raises the three-length averages from 88.33 to 98.33 on S-NIAH-2 and from 54.33 to 93.00 on S-NIAH-3, gains of 10.00 and 38.67 points (Figure 4).

F Extended Related Work

Efficient Attention Mechanisms. Efficient attention addresses the cost of full self-attention by changing the interaction pattern, approximating attention, or improving its execution. Local sliding-window attention limits each query to a bounded neighborhood (Wang et al., 2025; Cabannes et al., 2025). Performer and Linear Attention use feature-map formulations and associativity to avoid forming the full attention matrix (Choromanski et al., 2021; Katharopoulos et al., 2020). Nyströmformer instead uses a Nyström approximation of the softmax attention matrix (Xiong et al., 2021). FlashAttention-2 accelerates exact dense attention without changing its attention pattern (Dao, 2024). For linear attention and recurrent-style operators, sequence parallelism and chunkwise kernels target communication and hardware efficiency (Sun et al., 2024; Yang et al., 2025).

Sparse Attention. Sparse attention retains content-based softmax retrieval but restricts each query to a small subset of token-level key–value pairs. Early methods rely on predefined connectivity: Sparse Transformer factorizes the attention pattern, while Longformer and BigBird combine local windows with global or random links (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020). Later methods make the sparse pattern input dependent: Routing Transformer clusters tokens by content, H_2O evicts low-utility cache entries, and Quest selects KV-cache pages conditioned on the current query (Roy et al., 2021; Zhang et al., 2023; Tang et al., 2024). More recent trainable designs route queries to relevant blocks, as in MoBA, or combine compressed, selectively retrieved, and local branches with hardware-aligned kernels, as in Native Sparse Attention (Lu et al., 2025; Yuan et al., 2025). These approaches reduce attention-score computation or memory traffic, but their accuracy hinges on token or block selection and they still store or manipulate token-level KV memories. In contrast, MARCH routes over compact keys associated with historical recurrent-state snapshots and combines readouts from compressed prefix states rather than selecting token-level KV pairs.

State Space Models and Gated Linear Recurrences. State space models (SSMs) and linear recurrent networks compress the prefix into a recurrent state. Linear Attention and related models share this view through outer-product state updates and query-based reads (Katharopoulos et al., 2020; Chou et al., 2024; Dao & Gu, 2024).

S4 uses structured linear dynamics, while Mamba and Mamba-2 use selective transitions; RetNet, RWKV, HGRN, and LRU explore other structured recurrences (Gu et al., 2022; Gu & Dao, 2023; Dao & Gu, 2024; Sun et al., 2023; Peng et al., 2023; Qin et al., 2024; Orvieto et al., 2023; Liu et al., 2024a). DeltaNet uses delta-rule updates (Yang et al., 2024), while GDN combines the delta rule with gating (Yang et al., 2025). GDN-2 decouples erase and write through channel-wise gates (Hatamizadeh et al., 2026). Separately, negative eigenvalues and Householder products improve state tracking in linear RNNs (Grazzi et al., 2025; Siems et al., 2025).

Despite these advances, most models retain one fixed-capacity state, whose dimension directly controls update and read cost; this bottleneck contributes to the retrieval gap with Transformers (Arora et al., 2024b; Wen et al., 2024). MARCH preserves efficient recurrence while expanding accessible memory through cumulative state checkpoints and content-routed historical readouts.

State Expansion and Associative Memory. Long-context studies identify recurrent state capacity as a central limitation of Linear Attention models (Arora et al., 2024a;b). Multi-State RNNs, HGRN2, and Log-Linear Attention expand or hierarchically organize recurrent states (Oren et al., 2024; Qin et al., 2024; Guo et al., 2026). Mixture-of-Memories, Sparse State Expansion, Product Key Memory, and Fast-weight Product Key Memory use memory experts or sparse banks, while Sparse Delta Memory sparsifies GDN reads and writes (Du et al., 2025; Pan et al., 2025; Lample et al., 2019; Berges et al., 2024; Zhao & Jones, 2026; Cabannes et al., 2026; Afzal et al., 2026). Context-compression methods instead condense long inputs into learned summary vectors, activation beacons, or gist tokens (Chevalier et al., 2023; Mu et al., 2023; Zhang et al., 2024; Deng et al., 2025; Petrov et al., 2025). Gist tokens can also guide sparse attention (Mao et al., 2026).

RAT (Wei et al., 2025) applies cross-chunk softmax attention to keys and values summarized by intra-chunk recurrence. RAT+ (Wei & Gulcehre, 2026) uses full-sequence recurrence and active recurrence learning to support flexible attention dilation at inference after a short adaptation stage. Unlike RAT's cross-chunk summaries or RAT+'s dilated token attention, MARCH routes over cumulative matrix-valued state checkpoints and combines their query-dependent readouts. This expands accessible memory without enlarging the state updated at every text token; restricting retrieval to a subset of checkpoints is an optional sparse variant.

Concurrent Memory Caching. Memory Caching (Behrouz et al., 2026) stores either checkpoints of one continuously updated memory or independent segment memories. Its gated and sparse variants use pooled segment representations to weight or select cached memories according to the current input. MARCH likewise reads cached states with the current query, but associates each checkpoint with an anchor token whose hidden representation reads that checkpoint. Through this readout, the anchor's routing key becomes conditioned on the checkpoint's contents in subsequent layers. MARCH also includes an explicit null route that lets its residual historical-reading branch contribute no historical readout.

G Design and Evaluation Clarifications

This section clarifies the computation in Section 3 and the scope of the reported comparisons.

Q1. Does each checkpoint summarize only its preceding chunk? No. By Equation (8), $\mathbf{A}^{(m,\ell)}$ is the recurrent state after the entire prefix through b_m; the recurrence does not reset at checkpoint boundaries. Checkpoints thus preserve overlapping versions of one evolving memory, not summaries of disjoint chunks. They retain the state before later updates can modify it, without guaranteeing perfect preservation of every association. The interval C determines checkpoint frequency.

Q2. How can a shared anchor embedding produce content-dependent keys? The anchor's input embedding is shared, but its representation can change across layers. In Equation (9), its checkpoint readout enters the output projection and residual pathway. Because the key and readout at layer ℓ both use that layer's input, the readout affects the routing key at layer $\ell + 1$, not at layer ℓ. Checkpoint-dependent information therefore first enters keys through this pathway from the second layer onward.

Q3. Why separate routing from state reading? The routing query $\boldsymbol{\rho}_t \in \mathbb{R}^{d_r}$ scores compact anchor keys to weight visible checkpoints. The state-read query $\mathbf{q}_t \in \mathbb{R}^{d_k}$ reads each matrix $\mathbf{A}^{(m)} \in \mathbb{R}^{d_v \times d_k}$, yielding a token-dependent vector $\mathbf{A}^{(m)}\mathbf{q}_t$. The key is not itself the retrieved value. Current and historical states share the state-read query, while routing has a separate projection; reducing d_r lowers key storage and scoring cost without shrinking the matrices.

Q4. Does the null route choose between the current state and history? No. The current-state readout lies outside the routing softmax and has coefficient one in Equation (12); historical weights sum to $1 - \pi_{t,\varnothing}$. With no visible checkpoint, the null option receives all probability mass and the readout reduces to $\mathbf{S}_t\mathbf{q}_t$. This equation-level identity does not imply that a trained MARCH model equals a separately trained recurrent baseline.

Q5. Why is producing checkpoints before reading them still causal? The execution schedule does not determine visibility. For text position t, the reader uses only checkpoints with $b_m < t$, as specified by $\mathcal{V}_t$ in Section 3.2. An anchor reads its aligned prefix checkpoint, so its later-layer representation does not depend on later text. The producer can therefore run before the historical reader without exposing future checkpoints to earlier tokens.

Q6. What do the routing ablations establish? Table 7 compares routing dimension, Top-4 aggregation, and the null option within the full MARCH architecture. Removing the null option lowers every reported benchmark aggregate, supporting its contribution in this setting. They do not isolate anchor-derived keys from the benefit of retaining historical states. Uniform aggregation in Table 8 removes both learned weights and the null route. Isolating key construction would require alternative keys with the same checkpoint bank; that comparison is not reported.

Q7. What does the shared training setup control? Section 4.1 specifies a shared corpus, 50B-token budget, 16K sequence length, and optimization settings. Recurrent comparisons also share depth and hidden size. These controls support quality comparisons under common training conditions, but do not equalize total FLOPs, cache capacity, peak memory, or latency; historical retrieval adds computation and storage.

Q8. Can dense accuracy and Top-4 speed be treated as one operating point? No. The independently trained Top-4 row in Table 7 scores 44.85 on average NIAH, versus 51.33 for the dense default. Inference-only Top-4 on fixed checkpoints instead changes NIAH from 51.21 to 51.92 or from 55.67 to 52.76, depending on the checkpoint (Table 8). Figure 3 separately measures training throughput. These results do not form one measured quality–speed operating point.