TAYRA: HIERARCHICAL ATTENTION OVER EVOLVING RECURRENT STATES

Anonymous authors
Paper under double-blind review

ABSTRACT

Recurrent state space models (SSMs) enable efficient context processing by compressing past inputs into compact states, but continued updates can make earlier information harder to recover. Larger state spaces and multiple-state architectures extend recurrent memory, yet its usefulness also depends on how queries locate and read the information retained within these states. We introduce **Tayra**, a recurrent memory architecture in which the keys used to read a state also provide the basis for addressing it. Tayra combines a sparsely updated online memory with archived key–value states. Compact indices derived from stored keys guide historical access without replacing the states' internal contents, which remain available for query-dependent reading. A joint attention interface combines online and historical readouts. After pretraining on 50B tokens, Tayra achieves a 43.4% relative improvement over Raven in mean accuracy on six needle-in-a-haystack tasks across 4K–16K contexts. It also improves the average LongBench score by approximately 23% over a Raven-based MARCH adaptation. Recall gains extend to twice the training context length, while general language capabilities are preserved.

1 INTRODUCTION

Long-running language-model agents must retain information across interactions and revisit it as their tasks evolve. Many agent-memory systems manage external records and expose selected memories through the model's context (Packer et al., 2023; Chhikara et al., 2025), while learned context-management policies can consolidate interaction histories into compact textual states (Zhou et al., 2025). These approaches motivate complementary support within the backbone: memory that can be updated and queried directly during computation. We study how recurrent states can serve this role, retaining their internal structure for later access as the online memory evolves.

Recurrent state space models (SSMs) process sequences through a compact, continuously updated memory (Gu & Dao, 2023; Dao & Gu, 2024). A conventional, uncompressed KV cache grows by appending token-associated keys and values, while existing entries remain unchanged. A recurrent state instead integrates new information by updating a fixed-size working memory, moving part of information consolidation into the write operation itself. This consolidation is lossy: later updates can alter or attenuate earlier associations, making them harder to recover from the current state. Recent work refines this online compression through finer control over erasure and writing (Hatamizadeh et al., 2026) or memory updates formulated as online learning (Sun et al., 2025; Behrouz et al., 2025).

A complementary line expands the capacity and organization of recurrent memory. Sparse Delta Memory sparsely accesses a larger state (Cabannes et al., 2026), while Log-Linear Attention and Dynamic Linear Attention organize multiple states through temporal hierarchies or adaptive merging (Guo et al., 2026; Wang et al., 2026). Memory Caching and MARCH retain earlier recurrent states for later queries (Behrouz et al., 2026a; Zhang et al., 2026). Such archives keep earlier versions available as the online memory continues to evolve, making compressed states the units of historical storage and retrieval. How memory is read also matters: Elastic Memory reconstructs summaries from a compressed recurrent state, while HOLA adds a bounded exact KV cache for associations that the recurrent state may lose (Song et al., 2026; Cui, 2026).

Recurrent compression need not discard the key–value organization of attention. Bounded-memory attention models retain compressed keys and values that remain accessible to subsequent queries (Peng et al., 2022; Zhang et al., 2024; Karami et al., 2025). Sparse attention, in turn, separates

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.

coarse-grained memory selection from fine-grained content reading, using summaries or key statistics to locate relevant regions (Mohtashami & Jaggi, 2023; Tang et al., 2024; Lu et al., 2025; Hu et al., 2026). Archived recurrent states contain associations compressed by recurrent updates rather than raw token blocks, and cumulative snapshots may retain overlapping information. Their stored keys suggest a way to connect these two lines of work: derive state-level indices from the same keys used for query-dependent reading within each state.

We introduce **Tayra**, a recurrent memory architecture that makes archived states accessible through their own stored keys. Tayra maintains a sparsely updated key–value state as its online memory and periodically retains cumulative snapshots (Zhang et al., 2024; Afzal et al., 2026; Zhang et al., 2026). Anchor tokens derive compact indices from the archived keys (Hu et al., 2026), while subsequent queries read the states' full key–value contents through attention. The indices determine each state's contribution without replacing its internal contents. A joint attention readout combines historical reads with the online output. Stored keys thus support both state-level addressing and query-dependent content reading, without explicit positional encodings in either operation. Temporal order remains part of the recurrent computation and causal visibility, but archived states need not be assigned new token positions when queried.

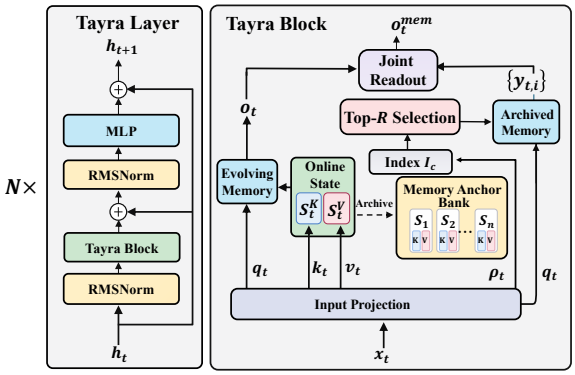

Figure 1: **Tayra architecture.** An evolving online memory and archived key–value states are accessed through a unified attention interface.

We pretrain Tayra on 50B tokens and evaluate it against recurrent baselines and a full-attention Transformer across general language, long-context understanding, and retrieval benchmarks. We highlight three findings. (1) Tayra improves the average LongBench score by approximately 56%, 23%, and 16.7% over Raven, MARCH, and Transformer, respectively, while also exceeding Transformer's average general language score by approximately 4.1%. (2) On RULER recall tasks, it narrows the gap with full attention and outperforms Transformer on multi-query and multi-value retrieval across all tested lengths, including 32K, twice its training context. (3) Ablations show that historical-read placement affects recall even with the same number of reading layers. Reading history in five consecutive middle layers approaches the mean NIAH accuracy of all-layer reading.

2 Preliminaries

Recurrent memory formulation. For each layer input $\mathbf{x}_t \in \mathbb{R}^d$, where d is the model's hidden dimension, learned projections produce a query $\mathbf{q}_t \in \mathbb{R}^{d_k}$ and a key–value pair $(\mathbf{k}_t, \mathbf{v}_t) \in \mathbb{R}^{d_k} \times \mathbb{R}^{d_v}$. The dimensions d_k and d_v refer to the query/key and value vectors of one head; layer and head indices are omitted.

We formulate recurrent memory through separate write and read operators (Zhang et al., 2026). Starting from the preceding state S_{t-1}, the current input produces an updated state S_t and an output $\mathbf{o}_t$:

$$S_t = \mathrm{Write}(S_{t-1}; \mathbf{x}_t), \qquad \mathbf{o}_t = \mathrm{Read}(S_t; \mathbf{q}_t). \tag{1}$$

The write operator includes input-dependent retention and incorporation of $(\mathbf{k}_t, \mathbf{v}_t)$; the read operator produces $\mathbf{o}_t \in \mathbb{R}^{d_v}$ without altering the state. Writing precedes reading, and neither operation accesses future inputs.

Sparse memory writing. Raven maintains two coupled states $S_t = (S_t^K, S_t^V)$, with $S_t^K \in \mathbb{R}^{M \times d_k}$ and $S_t^V \in \mathbb{R}^{M \times d_v}$, where M is the number of memory slots per head (Zhang et al., 2024; Afzal et al., 2026). Both states are initialized to zero: $S_0^K = \mathbf{0}$ and $S_0^V = \mathbf{0}$. The m-th rows form a key–value slot that accumulates content routed to it over time. We use Raven's tied sparse update: a learned router selects K_w slots per position, with $1 \le K_w \le M$, and the same routing weights control both forgetting and writing. Let $\mathcal{F}_t$ denote the selected slots and $\bar{\mathbf{r}}_t \in \mathbb{R}^M_{\ge 0}$ their router scores, with all

unselected entries set to zero. The normalized routing-weight vector $\boldsymbol{\omega}_t \in \mathbb{R}^M_{\geq 0}$ is

$$\boldsymbol{\omega}_t = \frac{\bar{\mathbf{r}}_t}{\sum_{m=1}^{M} \bar{r}_{t,m} + \epsilon}. \tag{2}$$

Here $\epsilon > 0$ stabilizes normalization; the router parameterization is given in Appendix A.1. These weights determine the slot-wise retention and write gates:

$$\lambda_{t,m} = \exp(\delta_t \omega_{t,m}), \qquad s_{t,m} = 1 - \lambda_{t,m}, \qquad \delta_t < 0. \tag{3}$$

The input-dependent scalar δ_t controls the decay rate for each head, with any fixed router normalization scale absorbed into it. The two states are updated with the same retention and write weights:

$$S^K_t = \mathrm{Diag}(\boldsymbol{\lambda}_t) S^K_{t-1} + \mathbf{s}_t \mathbf{k}_t^\top, \qquad S^V_t = \mathrm{Diag}(\boldsymbol{\lambda}_t) S^V_{t-1} + \mathbf{s}_t \mathbf{v}_t^\top. \tag{4}$$

Each selected row blends its previous contents with the incoming key or value, without mixing with other rows. For $m \notin \mathcal{F}_t$, $\omega_{t,m} = 0$ implies $\lambda_{t,m} = 1$ and $s_{t,m} = 0$: both rows remain exactly unchanged until that slot is selected again. Sparsity therefore restricts forgetting as well as writing, rather than merely suppressing the insertion of new content.

Within-state attention. The query attends to the keys currently retained in the recurrent state, rather than to the original token keys. Unlike the sparse write, this read considers all M slots; unwritten slots remain at their zero initialization and are not masked. For a state $S = (S^K, S^V)$ and native query $\mathbf{q} \in \mathbb{R}^{d_k}$, with the backbone's query transformations applied and attention scale absorbed into $\mathbf{q}$,

$$\boldsymbol{\ell}(S; \mathbf{q}) = S^K \mathbf{q}, \qquad \mathrm{Read}(S; \mathbf{q}) = (S^V)^\top \mathrm{softmax}(\boldsymbol{\ell}(S; \mathbf{q})). \tag{5}$$

Here $\boldsymbol{\ell}(S; \mathbf{q}) \in \mathbb{R}^M$ contains the slot-attention logits, with $\ell_m(S; \mathbf{q})$ denoting the query–key score of slot m. Softmax normalizes these scores over slots, and the resulting weights combine the corresponding rows of S^V. These read weights are distinct from $\boldsymbol{\omega}_t$, which determines where the current input is written. The read operator leaves S unchanged and can therefore be applied to an archived snapshot using the same within-state attention.

We also define the log-partition of these slot-attention logits:

$$L(S; \mathbf{q}) = \log \sum_{m=1}^{M} \exp(\ell_m(S; \mathbf{q})), \qquad L_t = L(S_t; \mathbf{q}_t). \tag{6}$$

The statistic L_t is the log-normalizer of the same logits used for the online readout. It enters the joint weighting of online and historical reads in Section 3.3.

3 Method

Tayra organizes access to online and archived recurrent states through a shared attention interface (Figure 2). State-level indices are derived from stored keys, native queries read each state's contents, and a joint normalization combines the resulting readouts. We use the write and read operators defined in Section 2 and omit layer and head indices throughout.

3.1 Online Memory and State Archiving

The online memory follows Raven's sparse recurrence, with $S_t = \mathrm{Write}(S_{t-1}; \mathbf{x}_t)$ and readout $\mathbf{o}_t = \mathrm{Read}(S_t; \mathbf{q}_t)$. At archival text positions $0 < \tau_1 < \tau_2 < \cdots$, we retain snapshots of the evolving state:

$$A_c = S_{\tau_c}, \qquad \mathcal{V}_t = \{c \geq 1 : \tau_c < t\}, \qquad N_t = |\mathcal{V}_t|. \tag{7}$$

Each snapshot $A_c = (A^K_c, A^V_c)$ preserves the key–value structure of the online memory at τ_c. Archiving neither resets nor otherwise modifies the online state: snapshots retain cumulative memories of their preceding prefixes, rather than independent summaries of the intervals between archival positions. At position t, retrieval is restricted to the N_t snapshots in $\mathcal{V}_t$, ensuring causal access.

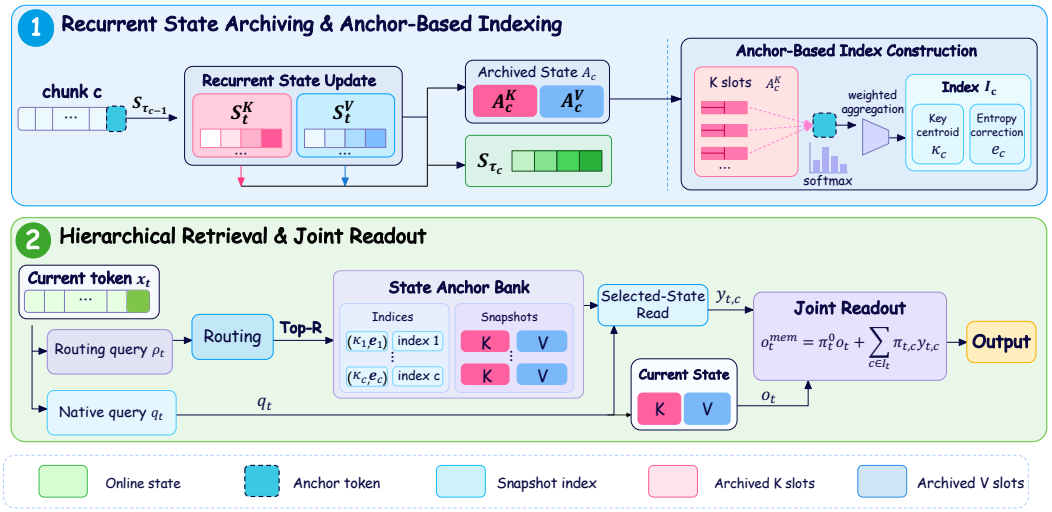

Figure 2: **State archiving and hierarchical readout in Tayra.** (1) Anchor routing queries build compact indices for cumulative state snapshots. (2) Text routing queries score snapshots, native queries read their contents, and a joint readout combines online and historical information. The illustrated Top-R variant reads up to R snapshots; dense reading includes all causally visible snapshots.

3.2 Anchor-Based State Indexing

We derive each index from the keys retained in an archived state, tying state-level scoring to the contents available for later reading. An anchor token is inserted immediately after each archival position τ_c, with all anchors sharing a learned initial embedding. At each layer, the incoming anchor representation produces an anchor routing query $\boldsymbol{\rho}_c^{\text{anc}} \in \mathbb{R}^{d_k}$, using the same routing transformation as text tokens (Section 3.3). We absorb the standard attention scale into $\boldsymbol{\rho}_c^{\text{anc}}$. Following the summary-and-entropy construction of HiLS (Hu et al., 2026), we attend to the archived keys:

$$\mathbf{p}_c = \text{softmax}\big(A_c^K \boldsymbol{\rho}_c^{\text{anc}}\big), \qquad \boldsymbol{\kappa}_c = (A_c^K)^\top \mathbf{p}_c, \qquad e_c = -\sum_{m=1}^{M} p_{c,m} \log p_{c,m}. \tag{8}$$

Here $\mathbf{p}_c$ distributes attention over the M slots. Geometrically, $\boldsymbol{\kappa}_c$ is an attention-weighted centroid of the stored keys, with the anchor routing query determining their contributions. Together with the entropy correction e_c, it forms the index $I_c = (\boldsymbol{\kappa}_c, e_c)$. Constructed within the same layer, I_c is cached and reused across subsequent text queries. It supplies state-level scores without replacing the full key–value state, which remains available for query-dependent reading.

Each anchor also reads its associated state using its native Raven query, computed from the same incoming representation. This readout incorporates a summary of the state's contents into the anchor representation, providing context for index construction in subsequent layers without modifying the recurrent state. The current-layer index does not depend on this readout. Anchor positions neither advance the text-position index nor contribute language-model prediction targets.

3.3 Hierarchical Retrieval and Readout

State scoring and selection. For each text token, we form a routing query $\boldsymbol{\rho}_t = W_r \mathbf{x}_t$, leaving activation, normalization, and the standard attention scale implicit. We score each visible state through its cached index:

$$a_{t,c} = \boldsymbol{\rho}_t^\top \boldsymbol{\kappa}_c + e_c, \qquad c \in \mathcal{V}_t. \tag{9}$$

We parameterize $W_r = W_q + W_{\text{up}} W_{\text{down}}$, where W_q is the backbone's query projection, $W_{\text{down}} \in \mathbb{R}^{r \times d}$, and $W_{\text{up}} \in \mathbb{R}^{d_k \times r}$, with adapter rank r. This residual parameterization reuses the backbone's query projection while allowing an addressing-specific adjustment. The routing query scores states through their cached indices, whereas the native query $\mathbf{q}_t$ resolves content within each state. Parameter sharing and query normalization are detailed in Appendix B.1.

The index-based scores weight historical readouts in both dense and sparse modes. We denote the states participating in historical reading by $\mathcal{I}_t$ and their number by n_t. Dense reading uses $\mathcal{I}_t = \mathcal{V}_t$ and $n_t = N_t$. The Top-R variant additionally uses the scores to retain only the $n_t = \min(R, N_t)$ highest-scoring states, independently for each position and head. The online state is always included in the readout, outside this retrieval budget.

Joint readout. For the current query $\mathbf{q}_t$, the native read operator maps each participating state $c \in \mathcal{I}_t$ to a content vector $\mathbf{y}_{t,c} = \mathrm{Read}(A_c; \mathbf{q}_t)$. A shared normalization lets each query allocate weight between these historical readouts and the online output $\mathbf{o}_t$. The online state is scored by its slot log-partition L_t, while historical states use their index-based scores $a_{t,c}$. Without a count correction, adding historical candidates increases their aggregate softmax weight at fixed scores. For $n_t > 0$, we therefore use

$$\big[\pi_t^0, \{\pi_{t,c}\}_{c\in\mathcal{I}_t}\big] = \mathrm{softmax}\big[L_t + \log n_t, \{a_{t,c}\}_{c\in\mathcal{I}_t}\big]. \tag{10}$$

The count correction makes the online exponential score compete with the mean, rather than the sum, of the participating historical exponential scores (Appendix A.3). It adjusts for candidate count, not overlap in their contents. The resulting output is a query-dependent barycenter of their readouts:

$$\mathbf{o}_t^{\mathrm{mem}} = \pi_t^0 \mathbf{o}_t + \sum_{c\in\mathcal{I}_t} \pi_{t,c}\mathbf{y}_{t,c}. \tag{11}$$

This combines information read from different states without interpolating the recurrent states themselves. With no participating history, we return the online output. Reading remains exact within each participating state, whereas state weighting uses compact indices; the resulting aggregation is therefore not an exact factorization of flat attention over all archived slots. Tayra is trained end-to-end with the next-token prediction loss.

Interpretation of state scores. The index-based score has an exact relation to the slot log-partition defined in Section 2. For any scaled query $\mathbf{u} \in \mathbb{R}^{d_k}$, expanding the KL divergence gives

$$L(A_c; \mathbf{u}) - \big(\mathbf{u}^\top \boldsymbol{\kappa}_c + e_c\big) = D_{\mathrm{KL}}\big(\mathbf{p}_c \,\|\, \mathrm{softmax}(A_c^K \mathbf{u})\big) \geq 0. \tag{12}$$

Thus, $a_{t,c}$ is a global lower-bound surrogate for $L(A_c; \boldsymbol{\rho}_t)$, exact when the anchor and text routing queries induce the same slot distribution. Geometrically, the affine score is the tangent plane to $L(A_c; \cdot)$ at $\boldsymbol{\rho}_c^{\mathrm{anc}}$, and the KL divergence measures the gap above that plane. The proof is given in Appendix A.2. This identity concerns the text routing query; it does not generally bound the partition $L(A_c; \mathbf{q}_t)$ under the different native query used for content reading.

3.4 Computational Cost

Tayra separates compact index scoring from full-state reading. Per layer and head, each text query scores N_t cached indices and reads n_t states, with historical memory-access cost $O(N_t d_k + n_t M(d_k + d_v))$, excluding Top-$R$ selection and in addition to the online recurrence and read. Dense reading uses $n_t = N_t$, whereas Top-R limits full-state reads to $n_t = \min(R, N_t)$ while retaining archive-wide index scoring. With an archival interval of C text tokens, $N_t \approx t/C$: dense historical reading therefore accesses approximately $(M/C)t$ slots, compared with the t token-level KV entries accessed by full attention. At matched key and value dimensions, $M < C$ reduces this slot-reading workload; Top-R further bounds it by RM slots per query. Each snapshot stores $M(d_k + d_v)$ state elements and $d_k + 1$ index elements, giving an archive size of $O(\frac{T}{C}[M(d_k + d_v) + d_k])$ after T tokens. Index construction and the anchor's slot read require $O(M(d_k + d_v))$ work per archival event. Together, M and C determine archive size and granularity, while R controls the historical-read budget.

4 Experiments

4.1 Experimental Setup

Baselines. We compare Tayra (845.4M parameters) with Raven (841.8M) (Afzal et al., 2026) and a Raven-based adaptation of MARCH (855.7M) (Zhang et al., 2026). These comparisons evaluate the benefit of historical memory and the choice of access architecture on a shared recurrent backbone.

We additionally include Gated DeltaNet (GDN, 792.8M) (Yang et al., 2025), Gated Slot Attention (GSA, 841.6M) (Zhang et al., 2024), and a 21-layer full-attention Transformer (692.9M) as broader architectural references. GDN stores key–value associations in a single matrix per head, whereas GSA and Raven maintain separate key and value slot states. We group the Raven variants under the backbone name in the tables and refer to them as MARCH and Tayra in the text.

Datasets. We evaluate Tayra across three aspects: (1) general language capabilities, using eight zero-shot tasks (LAMBADA, PIQA, HellaSwag, WinoGrande, ARC-Easy, ARC-Challenge, OpenBookQA, and CommonsenseQA); (2) in-context retrieval on six datasets (SQuAD, SWDE, FDA, TriviaQA, DROP, and Natural Questions), following the protocol of MARCH (Zhang et al., 2026); and (3) long-context understanding and recall, using twelve LongBench tasks (Bai et al., 2024) and six RULER needle-in-a-haystack tasks (Hsieh et al., 2024). RULER is evaluated at 4K, 8K, 16K, and 32K, the last being twice Tayra's training context length. We use the LM Evaluation Harness (Gao et al., 2021) for the zero-shot tasks. Dataset details, evaluation protocols, and metric definitions are provided in Appendix B.3.

Training Configuration. Following the training protocol of MARCH and Log-Linear Attention (Zhang et al., 2026; Guo et al., 2026), we pretrain all models from scratch on 50B text tokens from Long-Data-Collections with a sequence length of 16K. All models share the same optimization settings. Raven and Tayra use 21 layers, a hidden size of 1536, six heads, and $M = 256$ slots per head, with $K_w = 32$ slots selected for writing at each token. Tayra archives a cumulative state every $C = 512$ text tokens and uses a shared rank-64 adapter for text and anchor routing queries. Main results use dense historical reading at every layer; Top-4 retrieval is evaluated separately in Section 4.3. Inserted anchors do not count toward the text-token budget. Optimization and implementation details are provided in Appendix B.1.

4.2 Main Results

Language modeling and commonsense reasoning. Table 1 shows that Tayra achieves the highest average accuracy of 43.0%, exceeding Raven, MARCH, and Transformer by 1.4, 1.0, and 1.7 percentage points, respectively. Notably, it improves on MARCH across all eight accuracy benchmarks, so its average advantage is not driven by a single task. Both models retain historical states on the same recurrent backbone; this comparison suggests that how those states are accessed matters in addition to whether they are retained. Tayra also obtains the lowest LAMBADA perplexity (10.69) and ranks first on five of the eight accuracy benchmarks. Its gains thus extend beyond dedicated long-context retrieval to standard zero-shot evaluations, although they do not imply uniformly stronger performance: Raven remains better on ARC-Easy and ARC-Challenge, while GSA leads on HellaSwag.

Long-context understanding. Tayra achieves the highest average LongBench score of 16.1 (Table 2), improving over Raven, MARCH, and Transformer by approximately 56%, 23%, and 17%, respectively. Compared with MARCH, it improves on ten of twelve tasks and achieves higher average scores in all four task categories. The gains are more pronounced in summarization and few-shot learning than in document QA, with category-level improvements of 3.5 and 6.6 points, respectively.

Table 1: Zero-shot performance on language modeling and commonsense reasoning. Scores are accuracies (%, ↑), except for LAMBADA (LMB.) perplexity (ppl, ↓). Avg. is the unweighted mean of the eight accuracy scores. Red and blue indicate the best and second-best scores in each column, respectively, including ties.

Model	**LMB.** ppl ↓	**LMB.** acc ↑	**PIQA** acc ↑	**Hella.** acc ↑	**Wino.** acc ↑	**ARC-e** acc_n ↑	**ARC-c** acc_n ↑	**OBQA** acc_n ↑	**CSQA** acc ↑	**Avg.** ↑
Transformer	11.14	49.4	66.5	33.9	52.1	47.8	26.4	32.0	22.1	41.3
GDN	11.37	48.5	66.1	33.1	50.8	45.9	25.3	30.0	21.1	40.1
GSA	12.13	46.9	67.1	35.2	51.1	48.9	27.1	31.4	26.1	41.7
Raven	13.48	45.4	67.0	34.0	52.7	49.7	29.0	30.0	24.9	41.6
w/ MARCH	11.12	49.4	65.7	33.8	53.3	49.1	26.5	31.6	26.3	42.0
w/ Tayra	10.69	50.2	67.8	34.7	53.8	49.4	28.7	32.4	27.0	43.0

This pattern shows that the benefits extend beyond factual question answering to other uses of context, including document summarization and learning from demonstrations. However, Transformer retains a slightly higher average on multi-document QA, indicating that the overall advantage of Tayra does not translate into uniformly stronger performance across task types.

Table 2: Long-context understanding on twelve LongBench tasks, grouped into four categories. Each task uses its standard evaluation metric (higher is better); Avg. is the unweighted mean across tasks. Red and blue indicate the best and second-best scores in each column, respectively, including ties.

	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
GDN	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
GSA	2.5	5.2	10.9	6.6	8.2	3.5	6.0	12.3	9.4	20.0	25.9	25.2	11.3
Raven	2.2	4.5	10.8	4.6	8.9	2.2	1.7	13.7	8.5	8.0	31.8	26.7	10.3
w/ MARCH	5.1	5.6	13.2	5.0	8.3	4.1	8.3	11.8	12.4	16.0	39.7	27.6	13.1
w/ Tayra	5.3	4.9	16.2	6.7	10.1	3.9	16.8	12.8	13.4	20.5	51.9	30.6	16.1

In-context retrieval. Tayra achieves the highest average accuracy among recurrent models at 26.1%, compared with 17.5% for Raven and 25.4% for MARCH (Table 3). Both historical-memory models substantially improve over Raven, while the additional gain of Tayra over MARCH is smaller and task-dependent. Tayra improves on Raven across all six datasets and on MARCH across four, with a 5.0-point gain on SQuAD offset in part by a 4.9-point deficit on SWDE. These differences show that the two historical-access architectures have distinct strengths rather than a uniform performance ordering. Unlike LongBench, Transformer remains stronger in aggregate (31.3%), with its advantage concentrated in SWDE and FDA. Tayra therefore narrows the retrieval gap between Raven and full attention, but substantial differences remain on these extraction tasks.

Table 3: In-context retrieval accuracy (%) across six datasets. Avg. is the unweighted mean; higher is better. Red and blue mark the best and second-best scores in each column, including ties.

Model	SQuAD	SWDE	FDA	TriviaQA	DROP	NQ	**Avg.↑**
Transformer	41.3	59.3	80.4	2.2	2.9	1.7	31.3
GDN	34.8	45.0	31.4	1.1	2.2	0.8	19.2
GSA	34.5	35.6	20.9	2.1	2.9	2.8	16.5
Raven	32.4	42.5	21.8	3.3	2.7	2.5	17.5
w/ MARCH	35.4	53.3	56.7	2.0	3.1	2.0	25.4
w/ Tayra	40.4	48.4	58.4	3.8	3.0	2.8	26.1

Associative recall across context lengths. We use six RULER tasks to evaluate whether specific associations remain accessible as context grows. Across 4K–32K, Tayra outperforms MARCH in both Single NIAH and Complex NIAH, with a 21.8% relative improvement on the latter (32.25% versus 26.48%; Table 4). The per-task results further show that Tayra leads on multi-query and multi-value retrieval at every evaluated length (Figure 3). These gains extend beyond single-needle lookup to recovering multiple associations from compressed memory, and are already evident within the training context. At 32K, twice the training length, Tayra achieves the best or tied-best accuracy on all six tasks and retains perfect accuracy on S-NIAH-1. Its length generalization may partly reflect content-based memory access without explicit positional encodings, consistent with prior observations in NoPE models (Kazemnejad et al., 2023; Wang et al., 2024). Nevertheless, accuracy on the more demanding tasks still declines with context length, indicating that improved historical access mitigates, but does not eliminate, the difficulty of long-range recall.

Table 4: RULER NIAH accuracy (%), averaged over three tasks per group and four context lengths (4K–32K).

Group	Trans.	GDN	GSA	Raven	*w/ MARCH*	*w/ Tayra*
Single NIAH↑	70.50	54.42	54.83	58.92	64.92	70.42
Complex NIAH↑	31.25	14.33	19.25	13.29	26.48	32.25

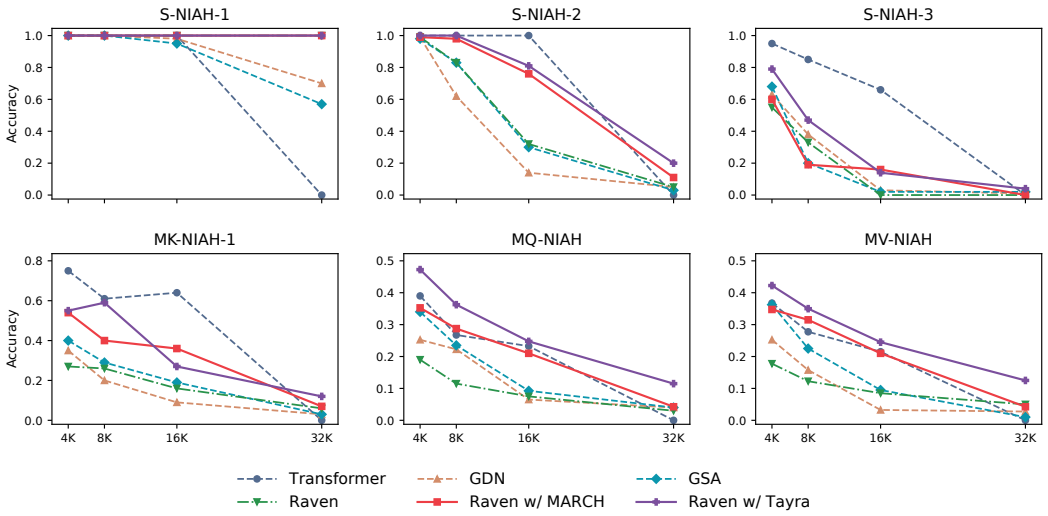

Figure 3: **Needle-in-a-haystack retrieval.** Accuracy on six RULER tasks: single-needle variants (top) and multi-key, multi-query, and multi-value retrieval (bottom). Context lengths range from 4K to 32K, twice the 16K training length. Y-axis ranges vary across panels.

4.3 Ablation Studies

We examine how historical-read placement, query parameterization, and retrieval budget affect Tayra (Table 5). All variants are trained separately on 50B text tokens. The default configuration reads all visible historical states at all 21 layers with a shared rank-64 residual adapter for text and anchor routing queries. Detailed configurations are provided in Appendix B.4.

Table 5: Ablation results for Tayra. PPL denotes LAMBADA perplexity. NIAH averages six tasks; Avg. averages across context lengths.

	Short-Context			In-Context Retrieval			NIAH (6-Task Avg.)			
Configuration	PPL↓	LAMB.↑	Hella.↑	SQuAD↑	SWDE↑	FDA↑	4K↑	8K↑	16K↑	**Avg.↑**
Tayra (default)	10.69	50.2	34.7	40.4	48.4	58.4	70.6	62.9	45.21	59.57
w/ interleaved layers	11.35	48.9	34.0	35.7	46.0	28.8	53.1	41.1	42.79	45.66
w/ middle layers	10.61	50.9	34.2	40.1	51.1	50.2	61.9	58.7	52.79	57.80
w/ independent query	10.59	50.2	34.5	37.3	50.2	48.5	61.5	56.1	43.25	53.62
w/ Top-4 retrieval	11.39	47.9	34.5	36.8	50.1	43.6	60.0	53.4	40.46	51.29

Historical-read placement. Restricting historical reads to five consecutive middle layers (indices 8–12, counting from zero) achieves a mean NIAH accuracy of 57.80, only 1.77 percentage points below reading at all 21 layers. With the same five-layer budget, this configuration outperforms interleaved reading at indices 4, 8, 12, 16, and 20 by 12.14 points. It also exceeds all-layer reading at 16K, despite lower scores at shorter lengths. These results suggest that the middle layers provide an effective stage for integrating long-range information from archived states, allowing near-all-layer recall with historical access confined to just five layers.

Query parameterization. We compare the default low-rank adaptation of the native query projection with an independent full-rank projection for state addressing. Text and anchor routing queries share the addressing projection in both configurations, while native slot reading is unchanged. Despite slightly lower LAMBADA perplexity (10.59 versus 10.69), the independent projection reduces mean NIAH accuracy by 5.95 percentage points. SQuAD and FDA accuracy also decline, although SWDE improves. These results suggest that low-rank residual adaptation of the native query provides a useful inductive bias for learning to address archived states, building on the query structure already used for content reading.

Retrieval budget. Top-4 retains all-layer reading and the rank-64 adapter but restricts each query and head to at most four historical states. With the same archiving policy, it achieves a mean NIAH accuracy of 51.29, compared with 59.57 for dense reading. Dense reading performs better on SQuAD and FDA, while Top-4 slightly improves SWDE. These results highlight the distinction between memory retention and access: the same archive capacity can support different recall levels depending on the read budget. Top-4 bounds full-state reading work without reducing archive storage, offering a trade-off between recall and the number of historical reads.

5 Related Work

5.1 Recurrent State Memory

Recurrent models compress context through segment-level memories (Dai et al., 2019; Rae et al., 2020; Bulatov et al., 2022) or compact states in linear attention and selective SSMs (Katharopoulos et al., 2020; Gu & Dao, 2023; Dao & Gu, 2024). Delta-rule and gated architectures refine state updates (Schlag et al., 2021; Yang et al., 2025; Kimi Team et al., 2025), while TTT, Titans, and ATLAS learn memory through test-time optimization (Sun et al., 2025; Behrouz et al., 2025; 2026b). Other approaches expand or organize memory through larger states and memory experts (Qin et al., 2024; Du et al., 2025), temporal hierarchies in Log-Linear Attention, adaptive merging in Dynamic Linear Attention (Guo et al., 2026; Wang et al., 2026), or sparse access in SSE and Sparse Delta Memory (Pan et al., 2025; Cabannes et al., 2026). Tayra extends Raven's sparse key–value memory (Afzal et al., 2026) with access to archived states.

5.2 Retrieval-Augmented Memory

Memorizing Transformers, LongMem, and InfLLM retrieve cached internal representations (Wu et al., 2022; Wang et al., 2023; Xiao et al., 2024), whereas RETRO retrieves chunks from an external text corpus (Borgeaud et al., 2022). HMT combines segment-level recurrence with hierarchical memory recall (He et al., 2025). Memory Caching retains segment-level recurrent states, while MARCH archives cumulative checkpoints and addresses them through content-conditioned anchor keys (Behrouz et al., 2026a; Zhang et al., 2026). Tayra derives same-layer indices directly from archived keys while retaining native slot attention.

5.3 Hierarchical Sparse Access

Landmark Attention and Quest locate relevant blocks or pages using learned landmarks or compact key statistics (Mohtashami & Jaggi, 2023; Tang et al., 2024). MoBA and MiniMax Sparse Attention perform query-dependent block selection before reading token-level contents (Lu et al., 2025; Lai et al., 2026). Native Sparse Attention combines compressed, selected, and local context (Yuan et al., 2025), whereas InfLLM-V2 reuses dense-attention parameters for switchable dense and sparse execution (Zhao et al., 2025). DeepSeek Sparse Attention uses a lightweight indexer for token-level selection (DeepSeek-AI et al., 2025). To reduce indexing overhead, HISA applies coarse block filtering before token-level refinement (Xu et al., 2026), while LongCat Sparse Attention combines hierarchical indexing with cross-layer reuse (Zan et al., 2026). HiLS couples chunk-level routing with within-chunk attention through landmark-derived summaries and entropy corrections (Hu et al., 2026). Tayra applies this construction to recurrent key-state slots rather than token blocks.

6 Conclusion

In this paper, we presented Tayra, a recurrent memory architecture that connects online compression with content-based access to archived states. Stored keys support both anchor-based indexing and query-dependent reading, while a joint attention readout combines online and historical information. After 50B-token pretraining, Tayra improves long-context understanding and retrieval over the evaluated recurrent baselines and surpasses the Transformer baseline in average LongBench and general language scores. Recall gains extend to twice the training context length. These results highlight the role of state structure and access organization in recurrent memory.

AI USE STATEMENT

We used generative AI tools to assist with writing and coding. All AI-assisted content was reviewed by the authors and verified where applicable. The authors take full responsibility for the accuracy and integrity of the final manuscript and accompanying research artifacts.

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. doi: 10.48550/arXiv.2607.25357. 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. *arXiv preprint arXiv:2304.09433*, 2023. doi: 10.48550/arXiv.2304.09433.

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, 2024. 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, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, 2024. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2025. doi: 10.48550/arXiv.2501.00663. URL https://arxiv.org/abs/2501.00663.

Ali Behrouz, Zeman Li, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Memory caching: RNNs with growing memory. *arXiv preprint arXiv:2602.24281*, 2026a. doi: 10.48550/arXiv.2602.24281. URL https://arxiv.org/abs/2602.24281.

Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. ATLAS: Learning to optimally memorize the context at test time. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026b. URL https://arxiv.org/abs/2505.23735.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239.

Sebastian Borgeaud et al. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2206–2240, 2022. URL https://proceedings.mlr.press/v162/borgeaud22a.html.

Aydar Bulatov, Yuri Kuratov, and Mikhail Burtsev. Recurrent memory transformer. *arXiv preprint arXiv:2207.06881*, 2022.

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

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025. doi: 10.48550/arXiv.2504.19413. URL https://arxiv.org/abs/2504.19413.

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. doi: 10.48550/arXiv.1803.05457.

Wanyun Cui. A hippocampus for linear attention: An exact memory for what the recurrent state forgets. *arXiv preprint arXiv:2607.02303*, 2026. doi: 10.48550/arXiv.2607.02303. URL https://arxiv.org/abs/2607.02303.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of ACL*, 2019.

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*, 2024.

DeepSeek-AI et al. DeepSeek-V3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025. doi: 10.48550/arXiv.2512.02556. URL https://arxiv.org/abs/2512.02556.

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 *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pp. 2368–2378, 2019. doi: 10.18653/v1/N19-1246. URL https://aclanthology.org/N19-1246/.

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.

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.

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

Ali Hatamizadeh, Yejin Choi, and Jan Kautz. Gated DeltaNet-2: Decoupling erase and write in linear attention. *arXiv preprint arXiv:2605.22791*, 2026. doi: 10.48550/arXiv.2605.22791. URL https://arxiv.org/abs/2605.22791.

Zifan He, Yingqi Cao, Zongyue Qin, Neha Prakriya, Yizhou Sun, and Jason Cong. HMT: Hierarchical memory transformer for efficient long context language processing. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8068–8089, 2025. doi: 10.18653/v1/2025.naacl-long.410. URL https://aclanthology.org/2025.naacl-long.410/.

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, Kewei Tu, Haitao Mi, and Leo Liang. Hierarchical sparse attention done right: Toward infinite context modeling. *arXiv preprint arXiv:2607.02980*, 2026. doi: 10.48550/arXiv.2607.02980. 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 *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Mahdi Karami, Ali Behrouz, Praneeth Kacham, and Vahab Mirrokni. Trellis: Learning to compress key-value memory in attention models. In *Second Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2512.23852.

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*, 2020.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in Transformers. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2305.19466.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025. doi: 10.48550/arXiv.2510.26692.

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/.

Xunhao Lai, Weiqi Xu, Yufeng Yang, Qiaorui Chen, Yang Xu, Lunbin Zeng, Xiaolong Li, Haohai Sun, Haichao Zhu, Vito Zhang, and Pengyu Zhao. MiniMax sparse attention. *arXiv preprint arXiv:2606.13392*, 2026. doi: 10.48550/arXiv.2606.13392. URL https://arxiv.org/abs/2606.13392.

Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. OpenCeres: When open information extraction meets the semi-structured web. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pp. 3047–3056, 2019. 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, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Neo Y. Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. MoBA: Mixture of block attention for long-context LLMs. *arXiv preprint arXiv:2502.13189*, 2025. doi: 10.48550/arXiv.2502.13189. URL https://arxiv.org/abs/2502.13189.

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/.

Amirkeivan Mohtashami and Martin Jaggi. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*, 2023. doi: 10.48550/arXiv.2305.16300. URL https://arxiv.org/abs/2305.16300.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. doi: 10.48550/arXiv.2310.08560. URL https://arxiv.org/abs/2310.08560.

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. doi: 10.48550/arXiv.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 *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, 2016. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

Hao Peng, Jungo Kasai, Nikolaos Pappas, Dani Yogatama, Zhaofeng Wu, Lingpeng Kong, Roy Schwartz, and Noah A. Smith. ABC: Attention with bounded-memory control. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7469–7483. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.acl-long.515. URL https://aclanthology.org/2022.acl-long.515/.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. In *First Conference on Language Modeling*, 2024.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020.

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

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020. doi: 10.1609/aaai.v34i05.6399.

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

Yunchong Song, Jushi Kai, Liming Lu, Kaixi Qiu, and Zhouhan Lin. Towards compressive and scalable recurrent memory. *arXiv preprint arXiv:2602.11212*, 2026. doi: 10.48550/arXiv.2602.11212. URL https://arxiv.org/abs/2602.11212.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (Learn at test time): RNNs with expressive hidden states. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 57503–57522, 2025. URL https://proceedings.mlr.press/v267/sun25h.html.

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*, pp. 4149–4158, 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, 2024. URL https://proceedings.mlr.press/v235/tang24l.html.

Jie Wang, Tao Ji, Yuanbin Wu, Hang Yan, Tao Gui, Qi Zhang, Xuanjing Huang, and Xiaoling Wang. Length generalization of causal Transformers without position encoding. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 14024–14040, 2024. doi: 10.18653/v1/2024.findings-acl.834. URL https://aclanthology.org/2024.findings-acl.834/.

Weizhi Wang et al. Augmenting language models with long-term memory. *arXiv preprint*, 2023.

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. doi: 10.48550/arXiv.2606.10650.

Yuhuai Wu et al. Memorizing transformers. In *International Conference on Learning Representations*, 2022.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. InfLLM: Training-free long-context extrapolation for LLMs with an efficient context memory. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2402.04617.

Yufei Xu, Fanxu Meng, Fan Jiang, Yuxuan Wang, Ruijie Zhou, Zhaohui Wang, Jiexi Wu, Zhixin Pan, Xiaojuan Tang, Wenjie Pei, Tongxuan Liu, Di Yin, Xing Sun, and Muhan Zhang. HISA: Efficient hierarchical indexing for fine-grained sparse attention. *arXiv preprint arXiv:2603.28458*, 2026. doi: 10.48550/arXiv.2603.28458. URL https://arxiv.org/abs/2603.28458.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. *arXiv preprint arXiv:2412.06464*, 2025. doi: 10.48550/arXiv.2412.06464.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*, 2025. doi: 10.48550/arXiv.2502.11089. URL https://arxiv.org/abs/2502.11089.

Wen Zan, Jiaqi Zhang, Jianchao Tan, Hong Liu, Cunguang Wang, Xiang Li, Duyue Ma, Guanyu Wu, Yifan Lu, Fengcun Li, Yerui Sun, Peng Pei, Yuchen Xie, and Xunliang Cai. LongCat sparse attention: Taming the lightning via streaming-aware hierarchical cross-layer indexing. *arXiv preprint arXiv:2608.01662*, 2026. doi: 10.48550/arXiv.2608.01662. URL https://arxiv.org/abs/2608.01662.

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. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

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

Yu Zhang, Songlin Yang, Ruijie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, Peng Zhou, and Guohong Fu. Gated slot attention for efficient linear-time sequence modeling. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://papers.nips.cc/paper_files/paper/2024/hash/d3f39e51f5f634fb16cc3e658f8512b9-Abstract-Conference.html.

Weilin Zhao, Zihan Zhou, Zhou Su, Chaojun Xiao, Yuxuan Li, Yanghao Li, Yudi Zhang, Weilun Zhao, Zhen Li, Yuxiang Huang, Ao Sun, Xu Han, and Zhiyuan Liu. InfLLM-V2: Dense-sparse switchable attention for seamless short-to-long adaptation. *arXiv preprint arXiv:2509.24663*, 2025. doi: 10.48550/arXiv.2509.24663. URL https://arxiv.org/abs/2509.24663.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*, 2025. doi: 10.48550/arXiv.2506.15841. URL https://arxiv.org/abs/2506.15841.

A Additional Method Details

A.1 Raven Gate Parameterization

Raven's tied sparse update uses the same routing weights for forgetting and writing. A projection $W_s \in \mathbb{R}^{M\times d}$ produces slot logits, followed by optional perturbation and a sigmoid activation σ. Let $\mathcal{F}_t$ contain the K_w highest-scoring slots. The masked scores and normalized routing weights are

$$\bar{r}_{t,m} = \sigma((W_s \mathbf{x}_t)_m + \xi_{t,m})\,\mathbf{1}[m \in \mathcal{F}_t], \qquad \omega_{t,m} = \frac{\bar{r}_{t,m}}{\sum_j \bar{r}_{t,j} + \epsilon}. \tag{13}$$

Here $\epsilon > 0$ stabilizes normalization. When training-time exploration is enabled, $\xi_{t,m} \sim \mathrm{Gumbel}(0,1)$; otherwise, including at inference, $\xi_{t,m} = 0$.

The head-specific scalar decay is

$$\delta_t = -\exp(A_{\log})\ \mathrm{softplus}(\mathbf{w}_a^\top \mathbf{x}_t + \Delta), \tag{14}$$

where $A_{\log}$, $\mathbf{w}_a$, and Δ are learned, ensuring $\delta_t < 0$. The retention and write gates are then

$$\lambda_{t,m} = \exp(\delta_t \omega_{t,m}), \qquad s_{t,m} = 1 - \lambda_{t,m}. \tag{15}$$

For unselected slots, $\omega_{t,m} = 0$, so $\lambda_{t,m} = 1$ and $s_{t,m} = 0$. Any fixed positive router normalization factor is absorbed into a constant offset in $A_{\log}$ without changing these gates. For numerical accuracy near zero, we compute $s_{t,m} = -\operatorname{expm1}(\delta_t \omega_{t,m})$. The write budget K_w controls online slot updates and is separate from the historical-state read budget R.

A.2 Interpretation of the Anchor Surrogate

We derive the exact gap between the anchor score and the state log-partition in Equation (12). Fix a snapshot A_c and its anchor distribution $\mathbf{p}_c$ from Equation (8). For a scaled query $\mathbf{u} \in \mathbb{R}^{d_k}$, define

$$L(A_c; \mathbf{u}) = \log \sum_{m=1}^{M} \exp((A_c^K \mathbf{u})_m), \qquad \mathbf{p}_c(\mathbf{u}) = \mathrm{softmax}(A_c^K \mathbf{u}). \tag{16}$$

Using $\boldsymbol{\kappa}_c = (A_c^K)^\top \mathbf{p}_c$, $e_c = -\sum_m p_{c,m} \log p_{c,m}$, and $\sum_m p_{c,m} = 1$, we obtain

$$\begin{aligned} D_{\mathrm{KL}}(\mathbf{p}_c \,\|\, \mathbf{p}_c(\mathbf{u})) &= \sum_m p_{c,m} \log p_{c,m} - \sum_m p_{c,m} \big((A_c^K \mathbf{u})_m - L(A_c; \mathbf{u})\big) \\ &= L(A_c; \mathbf{u}) - \mathbf{u}^\top \boldsymbol{\kappa}_c - e_c. \end{aligned} \tag{17}$$

Rearranging gives

$$L(A_c; \mathbf{u}) = \mathbf{u}^\top \boldsymbol{\kappa}_c + e_c + D_{\mathrm{KL}}(\mathbf{p}_c \,\|\, \mathbf{p}_c(\mathbf{u}))\,. \tag{18}$$

The divergence is nonnegative, so the affine score is a global lower bound on $L(A_c; \mathbf{u})$. Equality holds whenever $\mathbf{p}_c(\mathbf{u}) = \mathbf{p}_c$, including at the anchor query $\mathbf{u} = \boldsymbol{\rho}_c^{\mathrm{anc}}$. At this query, the gradient of the log-partition with respect to $\mathbf{u}$ is $\boldsymbol{\kappa}_c$, establishing the tangent-plane interpretation. The summary and entropy terms follow HiLS (Hu et al., 2026); here they are computed over recurrent-state slots rather than token keys.

For text routing, setting $\mathbf{u} = \boldsymbol{\rho}_t$ gives the score $a_{t,c}$. The bound therefore concerns the routing-query partition, not the partition under the different native query $\mathbf{q}_t$ used for slot reading. Together with the candidate-count correction in the joint readout, this distinction means that Tayra is not an exact factorization of flat attention over online and archived slots.

A.3 Candidate-Count Correction

The $\log n_t$ offset in Equation (10) adjusts the relative weight of online and historical memory for the number of participating snapshots. For $n_t > 0$, dividing the numerator and denominator of the online weight by n_t gives

$$\pi_t^0 = \frac{\exp(L_t)}{\exp(L_t) + \frac{1}{n_t}\sum_{c\in\mathcal{I}_t} \exp(a_{t,c})}. \tag{19}$$

Thus, the online exponential score competes with the mean, rather than the sum, of the historical exponential scores. For a fixed participating set, the weights are equivalently proportional to $\frac{1}{2}\exp(L_t)$ for the online state and $\frac{1}{2n_t}\exp(a_{t,c})$ for each historical state. This can be interpreted as assigning equal total prior mass to the online and historical branches, without requiring equal final weights. Dense reading uses the same normalization with $\mathcal{I}_t = \mathcal{V}_t$ and $n_t = N_t$.

To illustrate the role of this correction, consider a fixed participating set and the more general online logit $L_t + \gamma \log n_t$. The online weight becomes

$$\pi_t^0 = \frac{\exp(L_t)}{\exp(L_t) + n_t^{-\gamma} \sum_{c \in \mathcal{I}_t} \exp(a_{t,c})}. \tag{20}$$

Tayra uses $\gamma = 1$; setting $\gamma = 0$ removes the correction. If $L_t = a_{t,c}$ for every participating snapshot, these choices give $\pi_t^0 = 1/2$ and $\pi_t^0 = 1/(n_t + 1)$, respectively. This comparison holds at fixed scores: longer contexts can change both the participating set and its scores, and the correction does not account for overlap in stored contents.

All online and archived states contain M slots. Under the slot-log-mean convention, subtracting $\log M$ from every logit in Equation (10) changes neither the joint weights nor the historical ranking. We therefore omit this common offset from the equations.

B Additional Experimental Details

B.1 Training and Model Configuration

Table 6 summarizes training and model settings. Archiving leaves the online state size and writer unchanged. States and archives reset between documents; anchors are excluded from the text-token budget and next-token prediction targets.

Table 6: Training and model settings: common training configuration, shared Raven–Tayra backbone, and Tayra's default historical memory.

Setting	**Value**
Training	
Corpus	Long-Data-Collections
Training budget	50B text tokens
Sequence length	16K text tokens
Global batch size	Approximately 4.2M text tokens
Optimizer	Fused AdamW
AdamW $(\beta_1, \beta_2, \epsilon)$	$(0.9, 0.95, 10^{-8})$
Weight decay	0.1
Peak learning rate	4×10^{-4}
Learning-rate schedule	Warmup–stable–decay
Recurrent backbone	
Layers / hidden size	21 / 1536
Heads per layer	6
Key / value dimension per head	$d_k = 192$ / $d_v = 384$
Memory slots per head	$M = 256$
Selected write slots	$K_w = 32$
Initial key and value states	Zero
Short convolution	Disabled
Tayra memory	
Snapshot mode	Cumulative
Archival interval	$C = 512$ text tokens
Historical reading layers	All 21 layers
Historical retrieval budget	Dense (all visible snapshots)
Text / anchor routing adapter	Shared, rank 64
Index source	Stored state keys
Candidate-count correction	Online logit offset $\log n_t$ for $n_t > 0$

The Raven-based models use the checkpoint-provided `LlamaTokenizer` with a 32,000-token BPE vocabulary. Their WSD schedule uses 256 linear warmup steps, 10,473 constant-rate steps, and 1,192 cosine-decay steps, ending at a learning rate of 4×10^{-5}.

Query parameterization. Text and anchor routing queries share $W_r = W_q + W_{\text{up}} W_{\text{down}}$, followed by SiLU, head-wise RMS normalization, and scaling by $d_k^{-1/2}$. The normalization is also shared between text and anchors. The adapter rank is 64, distinct from $d_k = 192$. Within each layer, W_{down} is shared across heads and W_{up} denotes the current head's output block. Native slot-reading queries omit the adapter and retain the backbone's normalization and attention scale.

Implementation and precision. We use Triton kernels for recurrent updates and historical reads. Memory states, snapshots, and reader reductions are maintained in FP32; the resulting readouts are cast to BF16. Raven, MARCH, and Tayra use Gumbel exploration for slot-write routing during training only; historical-state routing receives no such noise.

B.2 Baseline Configurations

Table 7 reports exact parameter counts for Transformer, Gated DeltaNet (GDN) (Yang et al., 2025), Gated Slot Attention (GSA) (Zhang et al., 2024), and the Raven variants. MARCH adapts the historical-memory mechanism of Zhang et al. (2026) to Raven, replacing the original Gated DeltaNet backbone. Raven, MARCH, and Tayra therefore share a recurrent backbone but differ in historical access. The Transformer has 21 layers; its zero-shot configuration in Table 1 uses 16 heads and a RoPE base of 500K.

Our MARCH baseline archives cumulative states every 512 text tokens and reads all causally visible snapshots at layers 1–20 (zero-based), excluding layer 0. It uses 64-dimensional routing keys projected from anchor representations, with one key head shared by six query heads. Routing retains a learned null option, and historical readouts are added to the online output without reweighting the online branch.

Table 7: Parameter counts of the evaluated models. M denotes 10^6 parameters.

Model	Parameters	Parameters (M)
Transformer (21 layers)	692,912,640	692.9
GDN	792,785,532	792.8
GSA	841,564,416	841.6
Raven	841,758,204	841.8
w/ MARCH	855,710,772	855.7
w/ Tayra	845,376,444	845.4

B.3 Evaluation Tasks and Metrics

General capabilities. We evaluate eight zero-shot tasks using the LM Evaluation Harness (Gao et al., 2021): LAMBADA, PIQA, HellaSwag, WinoGrande, ARC-Easy, ARC-Challenge, OpenBookQA, and CommonsenseQA (Paperno et al., 2016; Bisk et al., 2020; Zellers et al., 2019; Sakaguchi et al., 2020; Clark et al., 2018; Mihaylov et al., 2018; Talmor et al., 2019). Table 1 uses normalized accuracy for ARC-Easy, ARC-Challenge, and OpenBookQA, and accuracy otherwise. Avg. weights the eight accuracy scores equally, excluding the separately reported LAMBADA perplexity.

LongBench. The twelve LongBench tasks span four families (Bai et al., 2024): NarrativeQA (NQA), Qasper (QQA), and MultiFieldQA-en (MFQ) for single-document QA; HotpotQA (HQA), 2WikiMultihopQA (2WM), and MuSiQue (Mus) for multi-document QA; GovReport (GvR), QMSum (QMS), and MultiNews (MNs) for summarization; and TREC (TRC), TriviaQA (TQA), and SAMSum (SSM) for few-shot learning. We use standard task metrics, averaged equally over all twelve tasks or the three tasks within each family.

In-context retrieval. We evaluate SQuAD, SWDE, FDA, TriviaQA, DROP, and Natural Questions (Rajpurkar et al., 2018; Lockard et al., 2019; Arora et al., 2023; Joshi et al., 2017; Dua et al., 2019; Kwiatkowski et al., 2019) using the protocol of MARCH and DLA (Zhang et al., 2026; Wang et al., 2026). The SQuAD, SWDE, and FDA recall formulations come from the Based evaluation suite (Arora et al., 2024). Table 3 reports accuracies (%) and their unweighted mean. The TriviaQA protocol differs from the few-shot evaluation in LongBench.

Controlled recall. Following MARCH, we evaluate S-NIAH-1/2/3 (single-needle), MK-NIAH-1 (multi-key), MQ-NIAH (multi-query), and MV-NIAH (multi-value) at 4K, 8K, 16K, and 32K (Zhang et al., 2026; Hsieh et al., 2024). The longest context is twice Tayra's training length. Figure 3 uses task-specific vertical scales. Table 4 groups the single-needle variants as Single NIAH and the remaining tasks as Complex NIAH, averaging twelve task–length accuracies equally per group. Ablations in Table 5 instead average all six tasks at each of 4K, 8K, and 16K; Avg. weights these three lengths equally.

At 32K, Raven, MARCH, and Tayra use their final 16K-trained checkpoints without additional long-context training or test-time changes to memory or routing. These evaluations use 100 samples per RULER task.

B.4 Ablation Configurations

Each variant in Table 5 is trained separately on 50B text tokens, rather than obtained by an inference-only change to the default checkpoint.

Historical-read placement. Using zero-based layer indices, the interleaved and middle-layer variants read history at $\{4, 8, 12, 16, 20\}$ and $\{8, 9, 10, 11, 12\}$, respectively. Both use five reading layers, dense retrieval, and the shared rank-64 adapter. Other layers retain online updates and readouts, anchor processing, and snapshot creation; only text-token historical reading is disabled.

Query parameterization. The independent-query variant replaces $W_q + W_{\mathrm{up}}W_{\mathrm{down}}$ with an independent full-rank projection shared by text and anchor routing queries. It retains all-layer dense reading and leaves native slot reading unchanged.

Retrieval budget. Top-4 retains all-layer reading, the rank-64 adapter, and the default archiving policy, but reads at most the four highest-scoring visible snapshots per query and head. The count correction uses $n_t = \min(4, N_t)$, compared with $n_t = N_t$ for dense reading.