δ-mem: Efficient Online Memory For Large Language Models

Jingdi Lei[1†*] **Di Zhang**[2†*], **Junxian Li**[3], **Weida Wang**[2], **Kaixuan Fan**[4], **Xiang Liu**[5], **Qihan liu**[6], **Xiaoteng Ma**[6], **Baian Chen**[6], **Soujanya Poria**[1†]
[1]Nanyang Technological University, [2]Fudan University
[3]Shanghai Jiao Tong University [4]The Chinese University of Hong Kong
[5]The Hong Kong University of Science and Technology (Guangzhou) [6]Mind Lab
jingdi001@e.ntu.edu.sg, di.zhang@ustc.edu,
soujanya.poria@ntu.edu.sg

Abstract

Large language models increasingly need to accumulate and reuse historical information in long-term assistants and agent systems. Simply expanding the context window is costly and often fails to ensure effective context utilization. We propose δ-mem, a lightweight memory mechanism that augments a frozen full-attention backbone with a compact online state of associative memory. δ-mem compresses past information into a fixed-size state matrix updated by delta-rule learning, and uses its readout to generate low-rank corrections to the backbone's attention computation during generation. With only an 8×8 online memory state, δ-mem improves the average score to $1.10\times$ that of the frozen backbone and $1.15\times$ that of the strongest non-δ-mem memory baseline. It achieves larger gains on memory-heavy benchmarks, reaching $1.31\times$ on MEMORYAGENTBENCH and $1.20\times$ on LOCOMO, while largely preserving general capabilities. These results show that effective memory can be realized through a compact online state directly coupled with attention computation, without full fine-tuning, backbone replacement, or explicit context extension. Code is available at: https://github.com/declare-lab/delta-Mem.

1 Introduction

As large language models (LLMs) are increasingly deployed in memory-heavy scenarios requiring continuous interaction, such as long-term personalized assistants [27, 16] and long-horizon agent systems [41, 26, 1], their life-cycle must go beyond responding to isolated prompts and instead accumulate, update, and reuse historical information over extended memory-heavy tasks [41, 32, 27, 35, 44]. In these settings, model performance depends not only on understanding the current input, but also on effectively leveraging relevant past context during test-time [17, 45]. An intuitive way is to simply expand the input context and retain more interaction history. However, this strategy only reduces the memory problem to a long-context processing problem, which is both computationally expensive and increasingly difficult to harness. On the one hand, standard attention incurs quadratic cost with respect to context length [42, 18, 33]. On the other hand, simply increasing the context window does not guarantee effective use of the additional information, as models often suffer from context degradation or context rot when the context becomes very long [13, 11], which suggests that even million-token context windows [25, 12] do not fundamentally solve the memory problem. These limitations call for more advanced memory mechanisms (MMs) that can represent historical information more compactly within a given context window, maintain it dynamically across interactions, and make it effectively usable by the backbone model during test-time [45, 27, 34, 35, 4, 36, 44].

*Equal Contribution
†Corresponding author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

From a unified perspective, existing memory mechanisms can be characterized along two dimensions under a given context window: memory state, which defines how historical information is stored, and memory steering, which determines how stored information influences backbone reasoning. Under this framework, prior methods fall into three paradigms. Textual memory mechanisms (TMMs) [27, 45, 28, 5, 6] store memory as text and inject it through the input context, offering flexibility without architectural changes but suffering from context-window limits, retrieval noise, and inevitable compaction loss. Outside-channel memory mechanisms (OMMs) [38, 34, 37] keep memory in external modules and interact with the backbone via retrieval or encoding on outside pathways, enabling modularity but introducing overhead, integration complexity, and potential misalignments with the backbone. Parametric memory mechanisms (PMMs) [14, 20, 22, 23] encode memory into parameters of prefixes or adapters, making them efficient and compatible with frozen backbones, but their static nature limits adaptation to dynamically evolving information. Taken together, these limitations point to a need for a memory mechanism that can maintain a compact and dynamically evolving memory state while steering the backbone through a pathway tightly aligned with its internal attention computation.

Following this motivation, we propose δ-mem, a memory mechanism that keeps a compact and dynamically updated memory alongside a frozen full-attention backbone. Instead of storing all historical tokens in the input context, δ-mem compresses past information into an online state of associative memory (OSAM). This state is continuously updated via delta-rule learning as new tokens arrive, allowing the model to maintain useful historical information in a fixed-size matrix representation of associative memories. During generation, δ-mem does not simply retrieve text from memory. Instead, the current input queries the online state to extract context-relevant associative memory signals, which are then transformed into a low-rank correction to the backbone's attention components. In this way, associative memory directly participates in the backbone's forward computation while leaving the backbone frozen. The online state is further updated after each interaction, enabling δ-mem to evolve its associative memory over time.

Finally, we evaluate δ-mem on memory-heavy benchmarks, including HOTPOTQA [40], LOCOMO [21], and MEMORYAGENTBENCH [15], together with general capability benchmarks IFEVAL [46] and GPQA-DIAMOND [31]. With only a fixed 8×8 online state of associative memory, δ-mem improves the final average score by $1.10\times$ over the frozen backbone and outperforms the strongest non-δ-mem memory baseline by $1.15\times$. On memory-heavy tasks, the improvement is larger: MemoryAgentBench increases over $1.31\times$, LoCoMo over $1.20\times$, and the TTL subtask nearly doubles from 26.14 to 50.50. These results show that a compact online state, when directly coupled with attention computation, can provide effective associative memory without relying on extending explicit context or heavy external retrieval modules.

Our contributions can be summarized as follows:

- We propose **δ-mem**, a memory mechanism that augments a frozen full-attention backbone with a compact online state of associative memory, enabling historical information to be dynamically maintained and directly coupled with the backbone's attention computation.
- We show that an extremely small memory state, implemented as an 8×8 matrix, can retain useful historical signals through OSAM and help the model recover context-relevant information even after explicit history is removed.
- We evaluate δ-mem on multiple memory-heavy and general capability benchmarks with significant gains on memory-heavy tasks such as MEMORYAGENTBENCH and LOCOMO, without full fine-tuning or replacing the backbone architecture.

2 Preliminaries

In terms of a Transformer for sequence modeling, let $\mathbf{x} \in \mathbb{R}^{N \times d}$ denote the input hidden sequence of a selected Transformer layer, where N is the sequence length and d is the hidden dimension. The hidden state at a single position is denoted as $\mathbf{x}_t \in \mathbb{R}^d$. For concise notation, each single-position vector is treated as a column vector. The sequence form can be understood as stacking these vectors along the position dimension. We use $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ to denote the query, key, and value in attention, and use $\mathbf{S}_t$ to denote the online state after processing position t. Unless otherwise specified, we omit the layer index in the following.

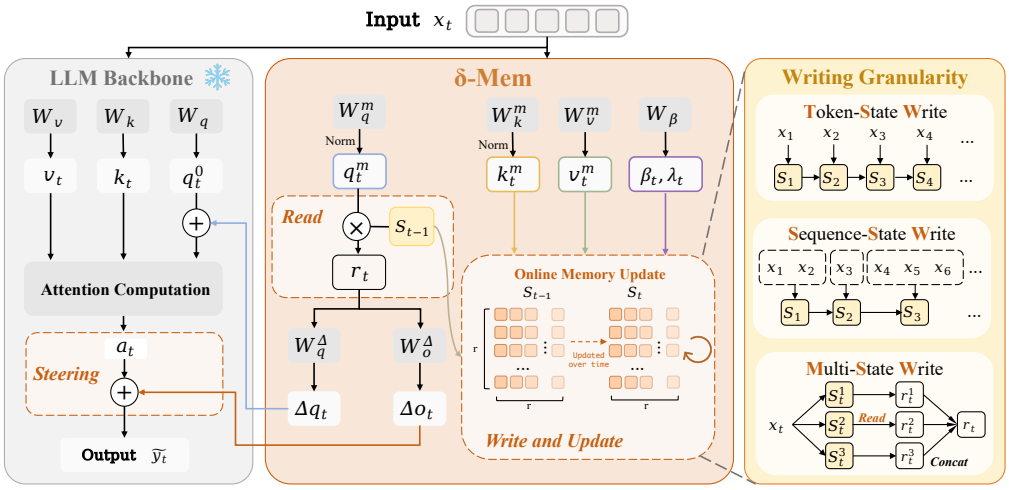


Figure 1: The overview of δ-mem. Given the hidden state from a frozen Transformer backbone, δ-mem first projects it into a low-dimensional memory space, and reads context-relevant associative memory signals from the previous online state, and uses the signal to generate low-rank corrections to the attention computation. After the computation, the memory state is updated with the current key-value information via delta-rule learning.

Concretely, δ-mem maintains a matrix $\mathbf{S}$ as the online state of associative memory. As tokens are processed, this state is updated sequentially to compactly encode key–value associations from the historical context. Given a memory key $\mathbf{k}_t \in \mathbb{R}^r$ and value $\mathbf{v}_t \in \mathbb{R}^r$ at position t, the state is expected to store the association $\mathbf{k}_t \mapsto \mathbf{v}_t$. The prediction made by the previous state is

$$\hat{\mathbf{v}}_t = \mathbf{S}_{t-1}\mathbf{k}_t. \tag{1}$$

This memory update can then be regarded as optimizing an online regression loss using SGD:

$$\mathcal{L}_t(\mathbf{S}) = \frac{1}{2}\left\|\mathbf{S}\mathbf{k}_t - \mathbf{v}_t\right\|^2, \quad \mathbf{S}_t = \mathbf{S}_{t-1} - \beta_t \nabla_{\mathbf{S}_{t-1}} \mathcal{L}_t(\mathbf{S}_{t-1}) = \mathbf{S}_{t-1} + \beta_t \left(\mathbf{v}_t - \mathbf{S}_{t-1}\mathbf{k}_t\right)\mathbf{k}_t^\top, \tag{2}$$

This formulation writes only the residual information along the key direction. Consequently, well-learned associations induce negligible updates, whereas predictive discrepancies dynamically correct the memory state. Inspired by gated retention design in Qwen-Next [39], we further introduce a forget gate to control long-range state evolution:

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

Here λ_t controls how much previous memory is retained, while β_t controls the strength of the residual write. This gated delta update forms the basis of the stable online memory dynamics in δ-mem.

3 δ-mem

At each position, δ-mem follows the same computation order: read associative memory signals from the old state, use the signals to steer attention, and then write the current information into the state. In this way, the model can compress history into a state that evolves with the sequence and use it in later reasoning, without updating the backbone parameters. Figure 1 provides an overview of this design. The frozen backbone performs the standard attention computation, while δ-mem reads from the previous state, generates query-side and output-side attention corrections, and updates the online state with the current memory key-value information. The figure also summarizes the three writing strategies studied in this work, corresponding to token-level updates, segment-level updates, and multi-state memory organization.

3.1 Memory Projections

To form the online state of associative memory, given a hidden state $\mathbf{x}_t \in \mathbb{R}^d$ at the current position, δ-mem projects it into a low-dimensional associative memory space:

$$\mathbf{q}_t^m = L_2\,\mathrm{norm}\left(\tanh(\mathbf{W}_q^m \mathbf{x}_t)\right), \quad \mathbf{k}_t^m = L_2\,\mathrm{norm}\left(\tanh(\mathbf{W}_k^m \mathbf{x}_t)\right), \quad \mathbf{v}_t^m = \mathbf{W}_v^m \mathbf{x}_t, \tag{4}$$

where $\mathbf{q}_t^m, \mathbf{k}_t^m, \mathbf{v}_t^m \in \mathbb{R}^r$. These three vectors correspond to reading and writing in memory. $\mathbf{q}_t^m$ queries the old state, while $\mathbf{k}_t^m$ and $\mathbf{v}_t^m$ describe how the current information should be written into the state. Normalizing the query and key can reduce state instability caused by scale drift during long-sequence recurrence.

The write gate and retention gate are also determined by the current hidden state:

$$\boldsymbol{\beta}_t = \sigma(\mathbf{W}_\beta \mathbf{x}_t + \mathbf{b}), \quad \boldsymbol{\lambda}_t = \mathbf{1} - \boldsymbol{\beta}_t. \tag{5}$$

where $\boldsymbol{\beta}_t, \boldsymbol{\lambda}_t \in \mathbb{R}^r$, $\mathbf{b}$ is the bias, and σ is the sigmoid function. This allows the state update to be adjusted dimension by dimension: some dimensions retain old memory, while others write the current information more actively.

3.2 Reading from Online State of Associative Memory

Before writing the current information, δ-mem first reads from the old state:

$$\mathbf{r}_t = \mathbf{S}_{t-1}\mathbf{q}_t^m. \tag{6}$$

The read vector $\mathbf{r}_t \in \mathbb{R}^r$ is the result of querying the online memory state with the current input. Since the size of $\mathbf{S}_{t-1}$ is fixed, the cost of this step is independent of the history length.

This reading form is complementary to standard attention. Attention compares the query with all keys within the explicit context, while δ-mem directly obtains continuous associative memory signals from the compressed state. It does not return text segments or add context tokens. Instead, it provides history-dependent steering signals before the attention computation.

3.3 Steering Attention through Low-Rank Corrections

The associative memory signals steer the attention computation through two lightweight linear mappings. First, the read signal $\mathbf{r}_t$ is projected into a query-side correction and an output-side correction:

$$\Delta\mathbf{q}_t = \mathbf{W}_q^\Delta \mathbf{r}_t, \qquad \Delta\mathbf{o}_t = \mathbf{W}_o^\Delta \mathbf{r}_t. \tag{7}$$

The query-side correction is then added to the original query of the frozen backbone:

$$\mathbf{q}_t^0 = \mathbf{W}_Q \mathbf{x}_t, \qquad \tilde{\mathbf{q}}_t = \mathbf{q}_t^0 + \frac{\alpha}{r}\Delta\mathbf{q}_t. \tag{8}$$

The attention output $\mathbf{a}_t$ is then computed using the corrected query and the frozen backbone keys and values, while the output-side correction is added after attention:

$$\mathbf{a}_t = \mathrm{Attn}(\tilde{\mathbf{q}}_t, \mathbf{K}_{\le t}, \mathbf{V}_{\le t}), \qquad \tilde{\mathbf{y}}_t = \mathbf{a}_t + \frac{\alpha}{r}\Delta\mathbf{o}_t. \tag{9}$$

The main implementation only uses the two correction terms on the query and output sides, and we detail these choices in Section 5. The low-rank correction here is different from a static adapter. Although $\mathbf{W}_q^\Delta$ and $\mathbf{W}_o^\Delta$ are fixed after training, their input $\mathbf{r}_t$ comes from the dynamic state $\mathbf{S}_{t-1}$. Therefore, the same set of parameters can produce different steering effects under different histories.

3.4 Writing into Online State of Associative Memory

After the current attention computation is completed, δ-mem writes the information at the current position into the online state. Given the current memory key-value pair $(\mathbf{k}_t^m, \mathbf{v}_t^m)$, the previous state first predicts the value associated with the current key as $\mathbf{S}_{t-1}\mathbf{k}_t^m$. The difference between the target value and this prediction defines the residual information to be written. As described in Section 2 δ-mem updates the state with a dimension-wise gated delta-rule:

$$\mathbf{S}_t = \mathrm{Diag}(\boldsymbol{\lambda}_t)\mathbf{S}_{t-1} + \mathrm{Diag}(\boldsymbol{\beta}_t)\left(\mathbf{v}_t^m - \mathbf{S}_{t-1}\mathbf{k}_t^m\right)(\mathbf{k}_t^m)^\top. \tag{10}$$

Expanding the update gives:

$$\mathbf{S}_t = \mathrm{Diag}(\boldsymbol{\lambda}_t)\mathbf{S}_{t-1} - \mathrm{Diag}(\boldsymbol{\beta}_t)\mathbf{S}_{t-1}\mathbf{k}_t^m(\mathbf{k}_t^m)^\top + \mathrm{Diag}(\boldsymbol{\beta}_t)\mathbf{v}_t^m(\mathbf{k}_t^m)^\top. \tag{11}$$

The three terms have clear roles: the first term retains the previous state, the second term removes the old prediction component along the current key direction, and the third term writes the new value into the same direction. Thus, the memory state is updated by error correction with controlled forgetting, rather than by unselectively accumulating new outer products.

The dimension-wise nature of the gates can be seen by expanding the update row by row. Let $\mathbf{s}_t^{(i)}$ denote the i-th row of $\mathbf{S}_t$. Then,

$$\mathbf{s}_t^{(i)} = \lambda_{t,i}\mathbf{s}_{t-1}^{(i)} + \beta_{t,i}\left(v_{t,i}^m - \mathbf{s}_{t-1}^{(i)}\mathbf{k}_t^m\right)(\mathbf{k}_t^m)^\top. \tag{12}$$

This shows that each memory dimension can independently control how much old information is retained and how strongly the current residual is written. Such dimension-wise gating is useful for continuous interactions, where the state must preserve stable historical information while still adapting to new inputs.

3.5 Writing Granularity of Online State

The above formulas explain how a single write operation is performed, but the memory mechanism also depends on the definition of writing granularity. A token is the finest granularity, but it is not always the most suitable one. In conversations and agent trajectories, messages, semantic segments, or stage-level events are often more stable. We therefore examine three writing strategies. As illustrated in Figure 1, TSW writes at every token, SSW averages the hidden states within each segment and writes per segment, and MSW writes into multiple parallel sub-states and then aggregates their readouts.

Token-State Write (TSW). Token-State Write updates the online state at each token position:

$$\mathbf{S}_t = \mathrm{Update}(\mathbf{S}_{t-1}, \mathbf{x}_t). \tag{13}$$

It preserves the finest-grained information and is suitable for scenarios that need to capture local changes. However, since every token triggers a write operation, the state is also more easily affected by format symbols, repeated expressions, and short-term noise.

Sequence-State Write (SSW). Sequence-State Write raises the writing granularity from unit tokens to a message segment. Let $\mathcal{M}^{(j)}$ denote the set of token indices in the j-th message. We first obtain the segment representation by averaging the hidden states of all tokens within this message:

$$\bar{\mathbf{x}}^{(j)} = \frac{1}{|\mathcal{M}^{(j)}|}\sum_{t\in\mathcal{M}^{(j)}} \mathbf{x}_t. \tag{14}$$

Then, each message updates the online state once. Let $\mathbf{S}_{(j)}$ denote the state after incorporating the j-th message:

$$\mathbf{S}_{(j)} = \mathrm{Update}(\mathbf{S}_{(j-1)}, \bar{\mathbf{x}}^{(j)}). \tag{15}$$

SSW reduces redundant writes and smooths the state evolution. The cost is that some fine-grained token-level details are absorbed by the averaged segment representation.

Multi-State Write (MSW). The first two strategies adjust the writing granularity, while MSW adjusts the state organization. A single state needs to contain facts, preferences, task progress, and local events at the same time, which may easily lead to overwriting and interference. MSW decomposes memory into multiple parallel sub-states:

$$\mathcal{S}_t = \{\mathbf{S}_t^{(1)}, \ldots, \mathbf{S}_t^{(N)}\}, \qquad \mathbf{S}_t^{(i)} = \mathrm{Update}^{(i)}(\mathbf{S}_{t-1}^{(i)}, \mathbf{x}_t), \qquad \mathbf{r}_t = Concat(\mathbf{r}_t^{(1)}, \ldots, \mathbf{r}_t^{(N)}). \tag{16}$$

where N is the number of state, $\mathbf{S}_t^{(i)} \in \mathbb{R}^{r\times r}$, and $\mathbf{r}_t^{(i)} = \mathbf{S}_{t-1}^{(i)}\mathbf{q}_t^{m,(i)}$ for $i = 1, \ldots, N$. This organization allows different sub-states to accumulate different types of information, thereby reducing mutual interference within a single state.

3.6 Training Objective

δ-mem is trained with the standard SFT loss. For each example, the context tokens are first written into the online state, producing $\mathbf{S}_C$, while they are not replayed as explicit backbone input during prediction. The frozen backbone only receives the query Q and response Y, and the stored state steers attention through δ-mem. The loss is the autoregressive cross-entropy over response tokens:

$$\mathcal{L}_{\mathrm{SFT}} = -\sum_{j=1}^{|Y|} \log p_{\phi,\theta}(y_j \mid Q, y_{<j}, \mathbf{S}_C), \tag{17}$$

where θ denotes the trainable δ-mem parameters and ϕ denotes the frozen backbone parameters.

4 Experiments

4.1 Experimental Setup

Evaluation and Benchmarks. To independently measure general reasoning and memory effectiveness, we evaluate our method on general tasks and memory-heavy benchmarks. General multi-hop reasoning, knowledge-intensive QA, and instruction-following are assessed using HotpotQA [40], GPQA-Diamond [31], and IFEval [46]. For the memory-heavy side, we utilize LoCoMo [21] (following [6], the adversarial question category is excluded), alongside MemoryAgentBench [15] to evaluate the retention, retrieval, and utilization of memory information across extended interaction histories.

Baselines. We compare δ-mem against representative memory mechanisms. All methods are built on the same Qwen3-4B-Instruct backbone. For textual memory mechanisms, we consider BM25 RAG [19], which retrieves relevant historical text and prepends it to the context; LLMLingua-2 [28], which compresses long histories into a shorter textual context; and MemoryBank [45], which maintains continuous interaction history through textual memory entries. For parametric memory mechanisms, we compare with Context2LoRA [14, 2] and MemGen [43], which encode memory or context-dependent adaptation into additional trainable parameters. For outside-channel memory, we include an MLP Memory [37] baseline that retrieves information in a separate module and then fuses back into the model. We additionally report trainable parameter counts for rank-8 configurations to compare memory effectiveness under similar or smaller adaptation budgets in Appendix E.

Implementation Details We select LLM backbones of varying sizes, including Qwen3-8B [39], Qwen3-4B-Instruct [39], and SmolLM3-3B [3]. More training setup and evaluation configurations are listed in Appendix D.

4.2 Main Results across Memory Mechanisms

Table 1: Main benchmark results comparing different memory mechanisms on Qwen3-4B-Instruct. All values represent the performance metric for each task is detailed in Appendix D. For the final average score, HotpotQA is counted using Exact Match (EM).

Model	IFEval	HotpotQA		GPQA-D	Memory Agent Bench					LoCoMo					Avg.
		EM	F1		Avg.	AR	TTL	LRU	SF	Avg.	Multi	Temp	Open	Single	
Qwen3-4B-Instruct	81.89	42.35	56.00	39.39	29.54	35.30	26.14	**47.08**	14.37	40.79	38.39	32.89	10.77	48.05	46.79
Textual Memory															
+ BM25 RAG	-	40.35	52.83	-	24.49	32.20	9.74	37.86	15.00	36.68	38.12	20.34	9.99	45.47	44.56
+ LLMLingua-2	-	36.93	50.03	-	15.63	21.45	1.43	38.45	8.62	40.98	39.07	30.13	10.98	49.19	42.96
+ MemoryBank	-	-	-	-	17.65	22.65	7.67	36.36	9.88	38.14	37.88	21.76	13.35	47.31	43.88
Parametric Memory															
+ Context2LoRA	76.71	37.85	50.88	29.29	32.53	40.00	29.86	25.15	**17.75**	48.11	37.95	34.99	16.75	**60.11**	44.90
+ MemGen	39.37	5.36	16.27	38.89	29.61	34.85	28.45	44.30	14.38	40.05	32.93	33.30	12.67	48.13	30.66
Outside-channel Memory															
+ MLP Memory	24.95	10.94	25.83	22.73	28.80	35.35	26.00	31.19	14.38	26.85	32.87	16.72	8.81	30.75	22.85
δ-Mem															
+ δ-Mem (SSW)	81.70	49.22	63.43	**41.41**	37.84	41.50	**50.50**	43.02	16.50	47.05	41.00	36.48	14.08	56.88	51.44
+ δ-Mem (TSW)	**82.99**	**49.41**	**63.66**	40.40	36.48	42.45	40.64	46.08	15.88	46.53	42.14	37.20	13.35	55.36	**51.66**
+ δ-Mem (MSW)	81.52	46.86	60.47	37.37	**38.85**	**44.40**	47.29	41.55	17.00	**49.12**	**42.57**	**39.31**	**18.12**	58.59	50.74

Table 1 compares δ-mem with representative memory-augmented baselines on general reasoning, instruction following, and memory-heavy benchmarks. δ-mem achieves the strongest performance across all methods. The TSW variant reaches the best average score of 51.66%, improving over the

Qwen3-4B-Instruct backbone (46.79%) by +4.87 points and over Context2LoRA (44.90%) by +6.76 points. SSW and MSW also perform strongly, achieving 51.44% and 50.74%, respectively. The gains are most pronounced on memory-heavy benchmarks. On MemoryAgentBench, δ-mem improves the average score from 29.54% to 38.85%, with MSW performing best. On LoCoMo, MSW achieves the highest average of 49.12% and performs best on Multi, Temporal, and Open subsets. On HotpotQA, TSW improves EM/F1 from 42.35%/56.00% to 49.41%/63.66%.

Across baselines, different memory mechanisms exhibit distinct limitations. Textual memory methods show inconsistent gains, likely due to retrieval noise and information loss introduced by compressing memory into token space. Parametric memory methods such as Context2LoRA tend to generalize less robustly across tasks, as their memory is statically encoded in parameters and can overfit to training distributions. The MLP Memory baseline performs relatively limited, indicating it lacks sequential state accumulation and cannot explicitly model long-range dependencies, while also introducing information loss by approximating instance-level retrieval. In contrast, δ-mem consistently improves performance across both general and memory-heavy evaluations, suggesting that maintaining memory as an online state provides a more robust memory mechanism.

4.3 Main Results Across Different Backbone Models

Table 2: General benchmark and long-context evaluation results across backbone models. All values represent the performance metric for each task as detailed in Appendix D. For the final average score, HotpotQA is counted using Exact Match (EM).

Model	IFEval	HotpotQA		GPQA-D	Memory Agent Bench					LoCoMo					Avg.
		EM	F1		Avg.	AR	TTL	LRU	SF	Avg.	Multi	Temp	Open	Single	
Qwen3-4B-Instruct	81.89	42.35	56.00	39.39	29.54	35.30	26.14	**47.08**	14.37	40.79	38.39	32.89	10.77	48.05	46.79
+ δ-Mem (SSW)	81.70	49.22	63.43	**41.41**	37.84	41.50	**50.50**	43.02	16.50	47.05	41.00	36.48	14.08	56.88	51.44
+ δ-Mem (TSW)	**82.99**	**49.41**	**63.66**	40.40	36.48	42.45	40.64	46.08	15.88	46.53	42.14	37.20	13.35	55.36	**51.66**
+ δ-Mem (MSW)	81.52	46.86	60.47	37.37	**38.85**	**44.40**	47.29	41.55	17.00	**49.12**	**42.57**	**39.31**	**18.12**	58.59	50.74
+ Context2LoRA	76.71	37.85	50.88	29.29	32.53	40.00	29.86	25.15	**17.75**	48.11	37.95	34.99	16.75	**60.11**	44.90
Qwen3-8B	79.67	32.48	41.42	44.95	31.87	45.10	12.79	**48.22**	12.00	47.02	41.28	30.43	14.20	59.03	47.20
+ δ-Mem (SSW)	80.41	**43.81**	**56.61**	45.45	**33.63**	45.55	**19.52**	46.11	13.50	**51.01**	**46.46**	**35.36**	20.05	62.05	**50.86**
+ δ-Mem (TSW)	**82.81**	41.97	53.61	44.95	32.97	**45.65**	16.43	46.41	12.88	50.70	44.35	34.76	17.90	**62.66**	50.68
+ δ-Mem (MSW)	80.96	40.15	51.34	**49.49**	32.66	45.55	14.95	44.60	13.38	50.92	45.52	34.82	**21.21**	62.27	50.84
+ Context2LoRA	77.26	36.22	49.19	38.38	30.52	43.15	10.05	34.04	**16.13**	47.20	35.18	35.05	20.49	58.92	45.92
SmolLM3-3B	67.10	1.67	14.40	23.23	14.21	12.57	5.53	30.72	8.00	24.18	22.41	16.49	10.87	29.22	26.08
+ δ-Mem (SSW)	**70.61**	27.35	43.26	**26.77**	19.22	16.80	6.50	37.84	15.75	39.39	29.12	27.09	**22.57**	49.45	36.67
+ δ-Mem (TSW)	66.36	24.90	41.28	26.26	**20.74**	17.98	**8.71**	**41.63**	15.75	35.46	27.48	25.34	17.04	44.10	34.74
+ δ-Mem (MSW)	67.47	**31.61**	**46.77**	25.76	20.54	**18.10**	8.32	39.63	**16.12**	**39.41**	**29.78**	**27.15**	19.59	**49.59**	**36.96**
+ Context2LoRA	62.29	30.28	44.39	**26.77**	17.62	16.08	2.86	36.77	14.75	37.74	26.35	26.05	15.41	48.58	34.94

Table 2 evaluates δ-mem across three backbone models, demonstrating consistent improvements in average scores across the board. δ-mem improves the average score on all backbones. Specifically, it boosts Qwen3-4B-Instruct from 46.79% to 51.66%, Qwen3-8B from 47.20% to 50.86%, and SmolLM3-3B from 26.08% to 36.96%. Notably, the effectiveness of the writing strategies varies by model capacity. On the more capable Qwen3-8B, the improvements are more modest but steady, with SSW securing the top average score of 50.86%. This suggests that for backbones with stronger inherent reasoning, segment-level writing (SSW) smooths state updates and effectively mitigates token-level noise. In contrast, the smaller SmolLM3-3B exhibits a substantial performance leap (from 26.08% to 36.96%) driven by MSW, indicating that smaller backbones benefit significantly from separating memory into multiple states to minimize interference.

5 Ablative Study

5.1 Context Recovery

To examine whether the online state of associative memory can preserve useful historical information without explicit context replay, we evaluate δ-mem under a no-context setting, where the original historical context is removed and only the compressed memory state is injected. As shown in Figure 2, δ-mem consistently improves over the no-context baseline on both HotpotQA and LoCoMo. On HotpotQA, the overall EM increases from 0.08% to 6.48%, and the overall F1 improves from 8.27% to 15.20%. The gains are especially large on the Bridge subset, where EM rises from 0.08% to 3.97% and F1 increases from 6.25% to 11.05%, indicating that the

online state can recover part of the missing multi-hop evidence. On LoCoMo, δ-mem also improves the overall average from 3.49% to 8.05%, with clear gains across multi-hop, temporal, open-domain, and single-hop questions. These results suggest that the online state of associative memory stores context-relevant historical signals that can be reused when explicit context is unavailable.

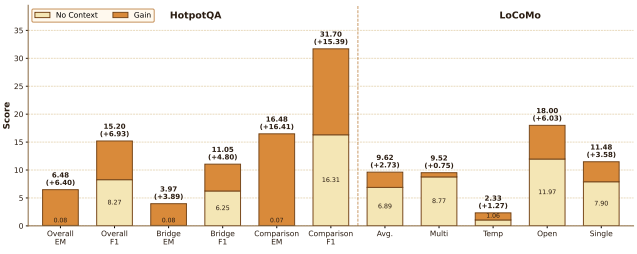


Figure 2: Context recovery performance on HotpotQA and LoCoMo with Qwen3-4B-Instruct as the backbone.

5.2 Heads Ablation

Table 3: Head Ablation Results on HotpotQA and LoCoMo with Qwen3-4B-Instruct as the backbone.

Head	HotpotQA						LoCoMo					Avg.
	Overall		Bridge		Comparison		Avg.	Multi	Temp	Open	Single	
	EM	F1	EM	F1	EM	F1						
q	45.87	60.59	44.96	60.34	49.50	61.58	43.15	**42.43**	33.60	10.03	50.82	44.51
k	43.39	57.28	42.35	56.82	47.55	59.09	40.98	38.44	33.12	10.93	48.26	42.19
v	46.12	60.95	45.07	60.54	50.30	62.57	42.35	39.54	34.24	11.11	49.95	44.24
o	48.94	63.69	47.67	63.58	54.00	64.15	45.15	39.68	36.31	12.77	54.06	47.05
qk	45.46	59.62	44.19	59.12	50.50	61.59	42.94	40.64	33.65	10.09	51.01	44.20
qv	47.02	61.95	45.86	61.64	51.65	63.20	43.24	40.38	35.57	10.76	50.84	45.13
qo	49.41	63.66	47.65	63.22	**56.42**	65.42	**46.53**	42.14	37.20	**13.35**	**55.36**	47.97
kv	45.67	60.43	44.90	60.25	48.76	61.14	42.02	39.45	33.91	11.00	49.52	43.85
qko	48.24	62.42	46.79	62.11	54.00	63.66	46.01	40.89	36.82	12.61	55.05	47.13
qkv	47.47	62.56	46.40	62.25	51.71	63.81	42.42	39.40	34.70	10.04	50.08	44.95
qkvo	**49.94**	**65.01**	**48.39**	**64.63**	56.09	**66.56**	46.15	41.08	**37.25**	13.14	55.02	**48.05**

We first study where the memory-induced correction should be injected within the attention block. As shown in Table 3, applying δ-mem to both query and output branches already yields strong performance, suggesting that query-side and output-side corrections provide an effective interface for memory injection. Among single-branch variants, the output branch performs best, achieving an average score of 47.05%, while the key branch is less effective. Combining multiple branches further improves performance. The full qkvo configuration achieves the best average score of 48.05%. These results suggest that associative memory signals are most effective when they can jointly influence query formation, key-value interaction, and output representation. While qkvo yields the highest average score, its marginal gain over qo does not justify the extra parameter overhead. Thus, we default to qo for an optimal performance-efficiency trade-off.

5.3 Insertion Depth Ablation

Table 4: Insertion depth results on HotpotQA and LoCoMo with Qwen3-4B-Instruct as the backbone.

Layer	HotpotQA						LoCoMo					Avg.
	Overall		Bridge		Comparison		Avg.	Multi	Temp	Open	Single	
	EM	F1	EM	F1	EM	F1						
Front 12	45.52	61.01	45.32	61.08	46.33	60.77	43.26	39.06	33.65	10.19	52.10	44.39
Middle 12	47.44	60.59	45.56	60.06	54.94	62.71	45.87	44.00	35.75	13.09	54.10	46.66
Back 12	44.58	59.04	43.56	58.65	48.62	60.58	43.53	40.60	36.31	11.11	50.97	44.06
All Layers	49.41	63.66	47.65	63.22	56.42	65.42	46.53	42.14	37.20	13.35	55.36	47.97

Table 4 studies the insertion depth of δ-mem across model's layers. Applying memory correction to all layers achieves the best overall performance, with an average score of 47.97%. It also obtains the strongest HotpotQA result, improving the overall EM/F1 to 49.41%/63.66%, and reaches the best LoCoMo average of 46.53%. These results suggest that associative memory signals are most effective when they can influence the representation hierarchy across the full depth of the backbone.

Among partial-layer variants, the middle-layer configuration performs best, reaching an average score of 46.66%. It clearly outperforms both the front-layer and back-layer configurations on the final average score. This indicates that intermediate layers provide a particularly effective interface for memory injection, balancing semantic abstraction and task-specific computation. In contrast, front-layer injections act on overly local representations, while back-layer injections leave insufficient depth for associative memory signals to propagate through subsequent computations.

6 Related Work

Textual Memory Mechanisms. Textual memory mechanisms externalize memory as text entries, summaries, or retrievable documents, and re-inject selected evidence into the input context or retrieval-augmented generation process. Early retrieval-augmented systems [19, 5] demonstrate the effectiveness of scalable textual stores for knowledge-intensive generation, while later agent-oriented methods [27, 29, 45, 6] extend this paradigm to continuous interaction by organizing past history and experience through logging, summarization, and reflection. Despite their flexibility, textual memory remains constrained by its tokenized form: memory use is sensitive to compression fidelity, retrieval noise, and context budget [17, 13]. δ-mem does not route compressed history back through token space. Instead, it maintains a compact online state and uses its readout to steer the frozen Transformer through low-rank attention corrections, separating memory maintenance from prompt-level reinsertion.

Outside-Channel Memory Mechanisms. A related line of work stores memory outside the backbone while preserving it in latent rather than textual form [38, 34, 37]. Memorizing Transformers [38] store past internal representations as non-differentiable key-value memories and retrieve them with approximate kNN, while LongMem [34] uses a frozen backbone as a memory encoder and an adaptive residual side network to read from an external memory bank. Compared with textual memory, latent memory can avoid part of the information loss introduced by natural-language summarization and preserve richer internal representations. However, memory still interacts with the backbone through a separate retrieval or reader pathway, introducing retrieval overhead, fusion complexity, and possible mismatch between stored and current representations. δ-mem differs in that its memory is not retrieved as an auxiliary external source; instead, its compact online state directly produces low-rank corrections to the attention computation, allowing memory to participate in the current forward pass.

Parametric Memory Mechanisms. Parametric memory mechanisms encode memory into additional parameters or localized weight edits. Prefix-Tuning [20] learns continuous virtual tokens for a frozen model, while LoRA [14] injects low-rank trainable updates into selected layers, showing that small parameter additions can effectively steer model behavior. Model-editing methods such as ROME [22] and MEMIT [23] further treat parameters as a writable memory substrate by inserting factual associations through localized or low-rank weight updates. However, these methods are less suited to online memory: their memory is usually fixed after training or updated through discrete editing steps, rather than evolving continuously with the sequence. Their write granularity is also less aligned with interaction history, which often unfolds at token-, message-, or segment-level resolution. As a result, parametric memory often acts as a persistent modification to model behavior, rather than a state-conditioned memory mechanism whose influence changes with the current history. δ-mem is close to LoRA in its low-rank interface, but differs fundamentally in that LoRA's low-rank update is static, whereas δ-mem generates low-rank attention corrections from a compact online state at runtime.

Hybrid Attention and Recurrent Memory. The closest prior work can be divided into three groups. Hybrid sequence architectures, including NAtS-L [8], NHA [10], Hymba [9], combine softmax attention with linear attention, recurrent slots, or SSM/RNN branches to balance modeling quality and computational efficiency; these components constitute the model's token-mixing backbone and maintain sequence-local rather than interaction-persistent states in their standard formulations. Infini-attention [24] instead propagate compressed matrices or memory tokens across segments of a long input stream. Although Infini-attention also considers a delta-rule update, it fuses the retrieved memory with local attention. Titans [4] and LongMem [34] are more explicitly memory-oriented, but respectively update neural-memory parameters at test time or retrieve representations from

an external memory bank through a residual side network. In contrast, δ-Mem retains a frozen pretrained full-attention backbone, persists a fixed-size associative state across interactions, and uses the previous-state readout only to generate low-rank query- and output-side corrections. Only the lightweight δ-Mem modules are trained with SFT, while inference updates the state without replaying the stored history. Thus, δ-Mem is not merely another hybrid token mixer: its contribution lies in combining online persistence, gated delta writing, and state-conditioned attention steering. It provides an efficient pipeline to combine with current LLMs for local deployment and enterprise scenarios.

7 Conclusion

In this work, we introduced δ-mem, a lightweight memory mechanism that equips a frozen full-attention backbone with a compact and dynamically updated online state of associative memory. δ-mem compresses past information into a fixed-size online state and uses its readout to generate low-rank corrections to the backbone's attention components. This design allows memory to be maintained online and to directly participate in forward computation without full fine-tuning or replacing the backbone architecture. Empirically, δ-mem improves performance on memory-heavy benchmarks while largely preserving the general capabilities of the frozen backbone. Notably, even with an extremely small 8×8 online state, the model can recover useful historical information after explicit context is removed, showing that effective memory does not require extending explicit context or heavy external retrieval modules. These results suggest that compact online states can serve as a scalable and efficient interface for test-time memory in frozen Transformer backbones.

Acknowledgment

This research/project is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG3-GV2023-010). This work is also supported by the National Research Foundation, Singapore, under its National Large Language Models Funding Initiative (AISG Award No: AISG-NMLP-2024-005), and NTU SUG project #025628-00001: Posttraining to Improve Embodied AI Agents. This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

References

[1] Anthropic. Claude code overview. `https://code.claude.com/docs/en/overview`, 2026. Claude Code Docs, accessed April 14, 2026.

[2] Seungju Back, Dongwoo Lee, Naun Kang, Taehee Lee, SK Hong, Youngjune Gwon, and Sungjin Ahn. Understanding lora as knowledge memory: An empirical analysis. *arXiv preprint arXiv:2603.01097*, 2026.

[3] Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. `https://huggingface.co/blog/smollm3`, 2025.

[4] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

[5] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR, 2022.

[6] 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.

[7] Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610, 2021.

[8] Difan Deng, Andreas Bentzen Winje, Lukas Fehring, and Marius Lindauer. Neural attention search linear: Towards adaptive token-level hybrid attention models. *CoRR*, abs/2602.03681, 2026.

[9] Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, Yingyan Celine Lin, Jan Kautz, and Pavlo Molchanov. Hymba: A hybrid-head architecture for small language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[10] Jusen Du, Jiaxi Hu, Zhang Tao, Weigao Sun, and Yu Cheng. Native hybrid attention for efficient sequence modeling. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens, editors, *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pages 3826–3842. Association for Computational Linguistics, 2026.

[11] Yufeng Du, Minyang Tian, Srikanth Ronanki, Subendhu Rongali, Sravan Bodapati, Aram Galstyan, Azton Wells, Roy Schwartz, Eliu A Huerta, and Hao Peng. Context length alone hurts llm performance despite perfect retrieval. *arXiv preprint arXiv:2510.05381*, 2025.

[12] Google. A new era of intelligence with gemini 3. `https://blog.google/products-and-platforms/products/gemini/gemini-3/`.

[13] Kelly Hong, Anton Troynikov, and Jeff Huber. Context rot: How increasing input tokens impacts llm performance. Technical report, Chroma, July 2025.

[14] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

[15] Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in llm agents via incremental multi-turn interactions. *arXiv preprint arXiv:2507.05257*, 2025.

[16] Bowen Jiang, Yuan Yuan, Maohao Shen, Zhuoqun Hao, Zhangchen Xu, Zichen Chen, Ziyi Liu, Anvesh Rao Vijjini, Jiashu He, Hanchao Yu, et al. Personamem-v2: Towards personalized intelligence via learning implicit user personas and agentic memory. *arXiv preprint arXiv:2512.06688*, 2025.

[17] Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. Llms get lost in multi-turn conversation. *arXiv preprint arXiv:2505.06120*, 2025.

[18] Jingdi Lei, Di Zhang, and Soujanya Poria. Error-free linear attention is a free lunch: Exact solution from continuous-time dynamics. *arXiv preprint arXiv:2512.12602*, 2025.

[19] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[20] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, 2021.

[21] Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.

[22] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

[23] Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*, 2022.

[24] Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention. *CoRR*, abs/2404.07143, 2024.

[25] OpenAI. Introducing gpt-5.5. `https://openai.com/index/introducing-gpt-5-5/`.

[26] OpenAI. Codex. `https://developers.openai.com/codex`, 2026. OpenAI Developers documentation, accessed April 14, 2026.

[27] 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.

[28] Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, et al. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 963–981, 2024.

[29] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22, 2023.

[30] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3505–3506, 2020.

[31] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

[32] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

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

[34] 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.

[35] Yu Wang and Xi Chen. Mirix: Multi-agent memory system for llm-based agents. *arXiv preprint arXiv:2507.07957*, 2025.

[36] Yu Wang, Dmitry Krotov, Yuanzhe Hu, Yifan Gao, Wangchunshu Zhou, Julian McAuley, Dan Gutfreund, Rogerio Feris, and Zexue He. M+: Extending memoryllm with scalable long-term memory. *arXiv preprint arXiv:2502.00592*, 2025.

[37] Rubin Wei, Jiaqi Cao, Jiarui Wang, Jushi Kai, Qipeng Guo, Bowen Zhou, and Zhouhan Lin. Mlp memory: A retriever-pretrained memory for large language models, 2026.

[38] Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. *arXiv preprint arXiv:2203.08913*, 2022.

[39] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[40] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[41] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

[42] 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)*, pages 23078–23097, 2025.

[43] Guibin Zhang, Muxin Fu, and Shuicheng Yan. Memgen: Weaving generative latent memory for self-evolving agents. *arXiv preprint arXiv:2509.24704*, 2025.

[44] Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, et al. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*, 2025.

[45] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 19724–19731, 2024.

[46] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023.

A Limitations

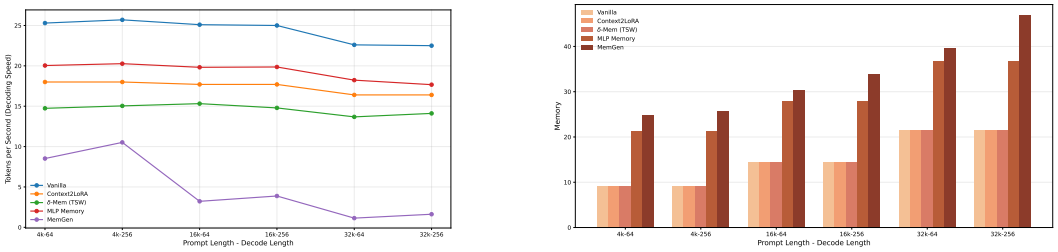
Although δ-mem provides online memory mechanism, we acknowledge two main areas for future refinement. First, as with most of memory mechanisms, the compact state inherently represents an approximation of the explicit context. While it significantly outperforms zero-history baselines, further aligning its representational capacity with exact full-context inference remains an important direction for future research. Second, the dynamic state update operations introduce marginal computational steps during decoding. However, since the GPU memory overhead is negligible, this theoretical impact on inference speed can be readily mitigated through system-level optimizations, which we defer to future work.

B Broader Impacts

δ-Mem aims to improve the efficiency and effectiveness of memory-augmented language models by enabling historical information to be maintained in a compact online state rather than relying solely on explicit context expansion. This may benefit long-term assistants, interactive agents, and other applications that require persistent context under limited computational budgets. At the same time, stronger memory mechanisms may introduce risks related to privacy, unintended retention of sensitive information, accumulation of stale or incorrect memories, and potential user profiling. These risks are particularly relevant when memory is maintained across long interactions or deployed in personalized systems. Therefore, δ-Mem should be used with appropriate safeguards, including explicit user consent, transparent memory controls, deletion mechanisms, and filtering or auditing procedures for sensitive information. We view δ-Mem as a technical memory mechanism, and its deployment should be accompanied by application-specific safety and privacy protections.

C Inference Efficiency and Memory Use

We further compare the inference efficiency of different memory-augmented methods under varying prompt and decoding lengths, as shown in Figures 3b and 3a. δ-mem achieves nearly the same GPU memory usage as Vanilla and Context2LoRA, indicating that its compact recurrent state introduces negligible memory overhead even when the prompt length increases to 32K. By contrast, MLP Memory and MemGen require substantially more memory, reflecting the cost of maintaining or generating larger auxiliary memory representations. In terms of decoding throughput, δ-mem is slower than Vanilla and Context2LoRA because each step involves reading from and updating the online state, but it remains considerably faster and more stable than MemGen across all tested settings. These results demonstrate that δ-mem improves long-context memory with a lightweight computational footprint, offering a practical balance between memory capability and inference efficiency.



(a) Decoding throughput.

(b) Memory usage.

Figure 3: Inference efficiency under different prompt and decode lengths.

D Implementation Details

Training Setup. All models are trained for one epoch on the shortest 2,219-sample split of QASPER [7], whose maximum sequence length is 8,269 tokens. The maximum backbone training sequence length is set to 512, while the memory write budget is set to 8,192 tokens. Unless otherwise

specified, δ-mem uses $r = 8$ and $\alpha = 16$, and is applied only to the query and output branches. The number of states in MSW is set to 4. Training is conducted on $8 \times$ A800 GPUs with bfloat16 precision, DeepSpeed ZeRO-2 [30], and fused AdamW. We use a peak learning rate of 2×10^{-4} with cosine decay and a warmup ratio of 0.1. The per-device batch size is 1, with 4 gradient accumulation steps, resulting in an effective global batch size of 32. The random seed is fixed to 42.

Evaluation Details. We follow the official evaluation prompts and decoding settings for all benchmarks. Specifically, we report prompt-level strict accuracy for IFEval, Exact Match (EM) and F1 for HotpotQA, accuracy for GPQA, and F1 for LoCoMo. For MemoryAgentBench, Table 5 summarizes the evaluation categories, datasets, and metrics. Each dataset is evaluated using its corresponding metric, and the final MemoryAgentBench score is computed as the sample-weighted average.

Table 5: Evaluation categories, datasets, and metrics in MemoryAgentBench.

Category	Dataset	Metric
Accurate Retrieval	SH-Doc QA MH-Doc QA LongMemEval (S*) EventQA	Accuracy
Test-time Learning	BANKING77 CLINC150 NLU TREC Coarse TREC Fine Movie Recommendation	Accuracy Recall@5
Long Range Understanding	∞Bench-Sum Detective QA	F1-Score Accuracy
Selective Forgetting	FactConsolidation-SH FactConsolidation-MH	Accuracy

E Parameter Overhead

We compare the trainable parameter overhead of δ-mem with representative memory-augmented baselines, as shown in Figure 4. δ-mem introduces only 4.87M trainable parameters for both SSW and TSW variants, accounting for merely 0.12% of the backbone parameters. Even the MSW variant, which maintains multiple memory states, requires only 19.47M trainable parameters, corresponding to 0.48% of the backbone. In contrast, MemGen uses 46.20M trainable parameters, while MLP Memory requires 3078.00M parameters, reaching 76.40% of the backbone scale. These results show that δ-mem achieves online memory augmentation with substantially lower parameter overhead, making it a lightweight alternative to larger auxiliary-memory modules.

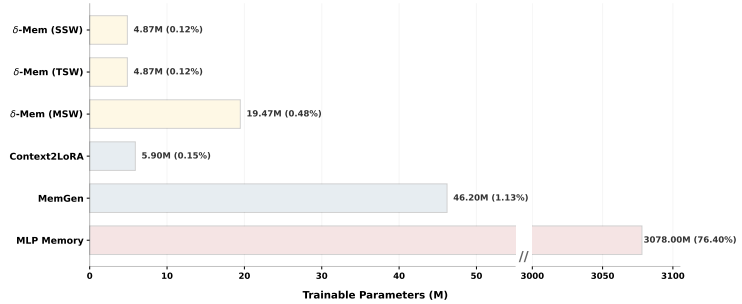


Figure 4: Trainable parameter comparison across memory-augmented methods.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main contribution of δ-mem as a compact online memory mechanism for frozen LLMs, and the claims are supported by the method description in Section 2 and Section 3 and the empirical results in Section 4 and Section 5.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated Limitations section in Appendix A, discussing that the compact state is an approximation of explicit context and that dynamic state updates introduce additional decoding-time computation.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present formal theoretical theorems or proof-based claims. The mathematical derivations in Section 2 and Section 3 are used to define the online memory update and model architecture rather than to establish formal theoretical guarantees.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper describes the model architecture and training objective in Section 3, reports the evaluated benchmarks and baselines in Section 4, and provides training hyperparameters, evaluation metrics, and decoding settings in Appendix D.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The paper uses publicly available models and benchmarks and provides the main experimental settings in Appendix D, and the current submission does include an open-source code release and reproduction scripts in abstract.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 specifies the evaluated benchmarks, baselines, and model backbones, while Appendix D reports the training dataset, sequence length, memory budget, rank, scaling factor, optimizer, learning rate schedule, batch size, precision, hardware, seed, and evaluation metrics.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The paper reports deterministic benchmark scores and fixes the random seed, but it does not include multiple-run error bars, confidence intervals, or statistical significance tests due to the computational cost of repeated LLM training and evaluation.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix D reports the GPU type, number of GPUs, precision, and distributed training setup, and Appendix C reports inference memory and throughput comparisons.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work is a machine learning research on language models memory mechanism and does not involve human subjects, private data, or safety-sensitive deployment. The experiments use standard public datasets and benchmarks, and we are not aware of any aspect of the research that conflicts with the NeurIPS Code of Ethics.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix B discusses the potential benefits of δ-Mem for efficient long-term assistants and interactive agents, as well as possible risks related to privacy, unintended retention of sensitive information, stale or incorrect memories, and user profiling. It also highlights the need for user consent, transparent memory controls, deletion mechanisms, and safeguards for sensitive information.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a new pretrained language model, image generator, scraped dataset, or other high-risk asset. The work proposes and evaluates a memory mechanism using existing benchmarks and backbones.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper properly credits the creators and original owners of the datasets, benchmarks, pretrained backbones, and baseline methods used in the experiments through corresponding citations. We use these existing assets in accordance with their intended research purposes and respect their stated licenses and terms of use.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce or release a new dataset, benchmark, or public model asset. The contribution is a memory mechanism and its empirical evaluation.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing, user studies, human-subject experiments, or participant compensation. All experiments are conducted on standard machine learning benchmarks.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human-subject research, crowdsourcing, user studies, or collection of participant data. Therefore, IRB approval or an equivalent human-subjects review is not applicable.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core methodology does not use LLMs as an important, original, or non-standard component. Any use of LLMs, if applicable, is limited to writing, editing, or formatting assistance and does not affect the method, experiments, scientific claims, or originality of the research.