TRANSMEM: TRANSFORMING HIDDEN STATES INTO MEMORY FOR LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents increasingly operate over long interaction histories, where effective reasoning requires identifying and exploiting task-relevant evidence distributed across past observations and actions. However, useful information encoded in previously computed representations is often underutilized during subsequent generation. We propose **TransMem**, a lightweight inference-time parametric memory module that transforms sparse historical hidden states from a frozen LLM backbone into reusable memory representations. TransMem uses a lightweight gating network to dynamically apply the latent intervention to the current hidden states, without repeatedly encoding the preceding context. To learn transferable memory utilization rather than task-specific knowledge, we introduce evidence-conditioned self-distillation. A memory-augmented student processes the full context and matches the predictive distribution of an evidence-only teacher that shares the same frozen backbone. Experiments on LoCoMo, HotpotQA, and MemoryAgentBench demonstrate consistent improvements across different model architectures and scales. TransMem yields gains of 11.58–16.88 F_1 on LoCoMo and 10.20–13.03 F_1 on HotpotQA, while improving the average MemoryAgentBench score from 28.95% to 40.00%. These results establish sparse historical hidden states as an effective and efficient memory substrate for long-context LLM agents.

1 INTRODUCTION

Large language models (LLMs) are increasingly expected to support sustained interactions over long contexts, particularly in applications such as coding agents (Yang et al., 2024), long-horizon agent systems (Shi et al., 2026; Yu et al., 2026a), and complex reasoning (Agashe et al., 2025). The crucial information needed to answer a query may be distributed across a long interaction history or document (Wu et al., 2025). The challenge is therefore not merely to accept longer inputs, but to identify and exploit the evidence that matters for the current prediction (Du et al., 2025).

A straightforward response is to enlarge the context window or scale the backbone model (Ding et al., 2024; Zhao et al., 2025). However, model predictions can be sensitive to the position of relevant evidence, and long-context reasoning often degrades when useful information is distant from the generation point (Liu et al., 2024; Hsieh et al., 2024). External memory addresses this limitation by summarizing, retrieving, or storing information outside the current computation (Lei et al., 2026a; Yang et al., 2026). Parametric memory encodes information into auxiliary latent spaces, which helps mitigate limitations imposed by the native computational structure of Transformers (Wei et al., 2026; Lei et al., 2026b). These mechanisms are useful when information must persist across tasks or sessions. However, they do not directly address a distinct test-time problem, which is how an LLM can better exploit the representations already computed for the current long context.

Based on this question, we observe two properties of causal transformers. **(1) Hidden states naturally aggregate information** from their preceding tokens (Li et al., 2024; DeepSeek-AI et al., 2025). As shown in Figure 1(a), a hidden state near the end of a context segment can therefore serve as a compact representation of that segment and its prefix. **(2) These representations are position-sensitive.** Because attention tends to favor nearby tokens (Liu et al., 2024), representations formed at different positions preserve different parts of the context with different fidelity. This effect becomes more pronounced as the context length increases. As illustrated in Figure 1(b), even when

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.

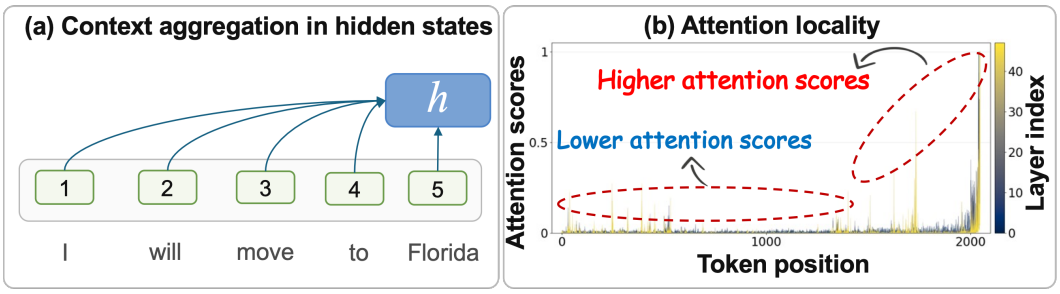


Figure 1: **Two properties of contextual representations in LLMs.** (a) *Hidden states aggregate preceding context.* Owing to the autoregressive computation of causal Transformers, each hidden state is computed from its preceding tokens. Consequently, later hidden states can serve as contextual representations of the preceding sequence. (b) *Transformer representations are region-sensitive.* Excluding the initial sink position, tokens appearing later in the context generally receives higher attention, reflecting a recency bias.

the information most relevant to the query appears earlier in the context, the Transformer may still assign higher attention scores to more recent tokens. Consequently, information from a long context is distributed across hidden states at different positions, rather than being fully retained by representations near the current generation position. These properties suggest that a small set of strategically selected hidden states can provide complementary views of a long input, without recomputing the entire long context as another memory. Consequently, substantial task-relevant information may remain latent in previously computed hidden states but underutilized during subsequent generation.

Based on these observations, we propose TransMem, an inference-time parametric memory module that transforms sparse historical hidden states into task-specific memory representations to enhance long-context reasoning. Rather than learning to store particular knowledge representations, TransMem learns the memory capability itself through an evidence-conditioned self-distillation training method. Specifically, it learns how to extract useful information from existing hidden states to answer the current query. As a result, TransMem does not memorize knowledge tied to a specific long context, but instead acquires a general capability for processing and reasoning over long contexts. It offers a scalable route toward improving memory capacity independently of the backbone model size. Our contributions are summarized as follows:

- We propose **TransMem, a lightweight inference-time parametric memory module** that transforms previously computed sparse hidden states from a frozen LLM backbone into reusable memory representations.
- We introduce **an evidence-conditioned self-distillation** training method. By exposing the teacher and student to different input contexts, the parameterized memory module learns to recover evidence-conditioned predictions from full long contexts.
- Extensive experiments on LoCoMo (Maharana et al., 2024), HotpotQA (Yang et al., 2018), and MemoryAgentBench (Hu et al., 2026) have shown consistent improvement on long-context reasoning across backbone architectures and scales.

2 Related Work

External Memory for LLMs. The integration of long-term memory mechanisms (Zhang et al., 2025c; Srivastava et al., 2026) represents an emerging direction in the evolution of LLM agents (Zhang et al., 2025c). Early frameworks like MemGPT (Packer et al., 2024) pioneered operating-system-inspired paging and segmentation to manage extended contexts. Building upon this, scalable architectures like Mem0 (Chhikara et al., 2025) dynamically consolidate memory states to mitigate the severe limitations of fixed context windows. Retrieval-augmented memory systems frequently suffer from critical information omissions and semantic noise (Latimer et al., 2025). To address this issue, recent studies have explored iterative retrieval paradigms, including MemCoT (Lei et al., 2026a), MemEvolve (Zhang et al., 2025b), SimpleMem (Liu et al., 2026), and

ReasoningBank (Ouyang et al., 2026). These approaches substantially improve retrieval accuracy by enabling progressive evidence refinement and adaptive memory exploration. However, they often suffer from limited robustness and increased computational overhead due to the repeated retrieval process (Srivastava et al., 2025). Moreover, these methods fundamentally do not eliminate the information loss introduced during memory extraction. Instead, they remain vulnerable to premature termination within the retrieval loop, which can prevent the model from locating the necessary evidence and ultimately lead to retrieval failure.

Parametric Memory for LLMs. Compared with memory mechanisms that store information in external knowledge repositories, parameterized memory provides a more expressive representation space by directly encoding knowledge into model parameters. Therefore, it provides an alternative mechanism for enhancing the memory capacity of LLMs. Early attempts along this direction typically employ auxiliary models to explicitly or implicitly encode additional information, including MeMo (Quek et al., 2026), MemVerse (Liu et al., 2025), Memory Decoder (Cao et al., 2025), MemSifter (Tan et al., 2026), MEMTS (Yu et al., 2026b), and Mem-π (Wang et al., 2026). More sophisticated approaches instead design specialized architectures that introduce dedicated representations to enhance long-context reasoning capabilities (Zhang et al., 2025a; Ji et al., 2026). For example, DRIFT (Xie & Wang, 2026) reveals that the last-layer feature may already contain highly compressed information from long contexts, suggesting that intermediate representations can serve as effective memory carriers. Meanwhile, the decoupling of memory capability from reasoning capability has emerged as an increasingly important research direction. MLP Memory (Wei et al., 2026) maps last-layer features into a latent distribution to inject long-context reasoning ability, while δ-Mem (Lei et al., 2026b) adopts a heterogeneous delta-network architecture to incorporate long-context reasoning capabilities into layer-wise representations.

Distillation for Long-Context Reasoning. On-policy distillation improves language models by providing token-level supervision on student-generated trajectories, alleviating the distribution mismatch in traditional knowledge distillation (Agarwal et al., 2024). Recent self-distillation methods further leverage privileged information to improve reasoning, where teacher and student share the same model but access different contexts (Zhao et al., 2026). For long-context scenarios, OPSDL (Zhang et al., 2026) uses a short-context self-teacher to supervise long-context responses, while other works combine on-policy optimization with distillation for long-context reasoning (Ramos et al., 2026).

3 METHOD

In this section, we present the TransMem framework, including its inference and training procedures. Figure 2 provides an overview of the framework. During inference, the frozen backbone performs its standard forward computation, while TransMem abstracts a small set of historical hidden states into a memory shift. This shift influences current output hidden states to better preserve key information from long contexts. We train TransMem through self-distillation to recover the predictive distribution of a teacher model conditioned on gold evidence, enabling the memory module to preserve crucial information while suppressing contextual noise.

3.1 PRELIMINARIES

During the prefill stage, the input long context is denoted as C. Let $Q \subseteq C$ denote the question and $E \subseteq C$ denote the evidence that supports the answer. At position t, the tokenized input is $x_{1:t} = [C; A_{\leq t}]$, where $C = x_{1:L}$ is the long context and $A_{\leq t} = (y_{L+1}, \ldots, y_t)$ is the generated answer prefix. The next token is y_{t+1}. The hidden states produced by the K-th layer for $x_{1:t}$ are:

$$h^K_{1:t} = [H_C; H_A] = [h^K_{1:L}; h^K_{L+1:t}], \tag{1}$$

where $h^K_t \in \mathbb{R}^{1\times d}$, with d denoting the hidden dimension of the frozen backbone model. The context features H_C are computed during prefill, during which TransMem does not intervene in generation.

With the KV cache, each layer only needs to receive the final hidden state at each decoding step. In the LLM forward pass, transformer layer $K+1$ produces the hidden state of the next layer as

$$h^{K+1}_t = f^{K+1}_\theta(h^K_t), \tag{2}$$

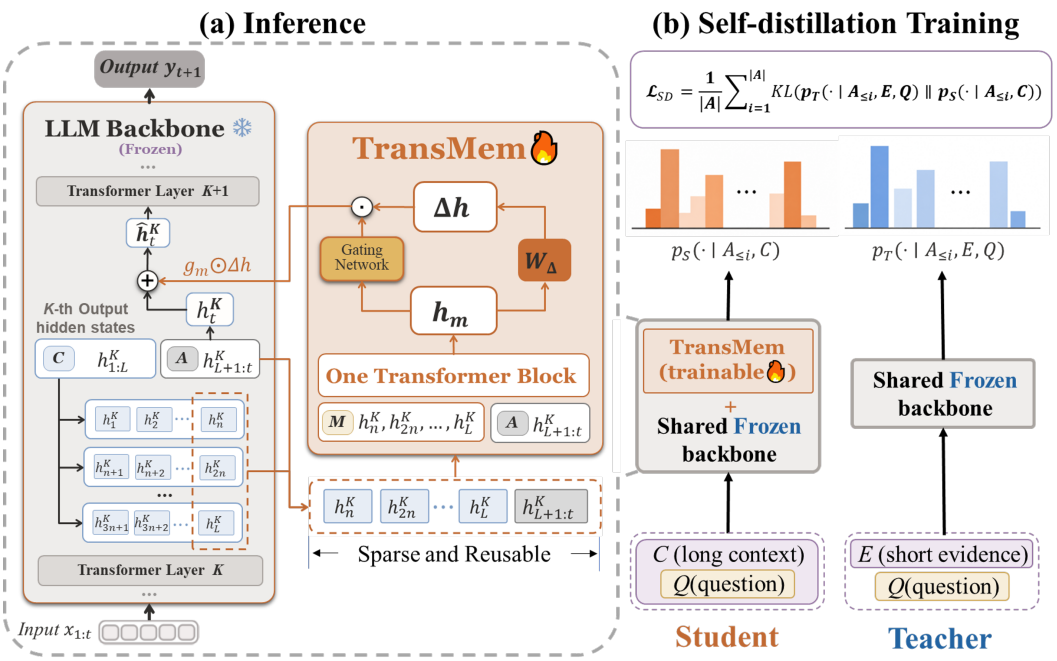


Figure 2: The overall architecture of the TransMem framework. (a) Inference with TransMem. TransMem is applied after the frozen LLM backbone produces the hidden states at the K-th layer. The selected historical hidden states are transformed into memory representations by the TransMem module. (b) Evidence-conditioned self-distillation. The student model consists of the frozen LLM backbone augmented with TransMem and receives the full long context C. The teacher model shares the same frozen backbone parameters but receives only the evidence E and question Q, providing an evidence-conditioned supervision signal. The student is trained to recover the teacher prediction distribution from the full long context.

where f_θ^{K+1} represents the $(K+1)$-th Transformer block.

3.2 Evidence-Conditioned Self-Distillation Training

We train TransMem with self-distillation. Our goal is to improve the ability of LLMs to identify and exploit query-relevant evidence from long contexts. Therefore, instead of training the model to memorize the entire context, we aim to encourage the model to suppress irrelevant information and focus on evidence that supports the answer. To achieve this goal, we construct the teacher by providing only the evidence E and query Q. Since the teacher has access to the evidence without irrelevant context noise, its output distribution provides an evidence-conditioned supervision signal. In contrast, the student is equipped with learnable TransMem. It receives the full context C and learns to recover the teacher's evidence-conditioned predictions through distillation.

The teacher runs the frozen backbone on (E, Q), while the student runs the TransMem-augmented backbone on C. The teacher generates the answer trajectory A, and the student follows its answer prefix during teacher forcing. Both paths share the same frozen LLM backbone, so no separate teacher model is introduced. Let $p_T(\cdot \mid A_{<i}, E, Q)$ denote the teacher distribution along its answer trajectory. A student model augments the frozen backbone with TransMem and produces $p_S(\cdot \mid A_{<i}, C)$. As shown in Figure 2(b), the training objective is the forward KL divergence at the language model output. We optimize the TransMem module by minimizing the forward KL divergence between the teacher and student distributions:

$$\mathcal{L}_{\text{SD}} = \frac{1}{|A|} \sum_{i=1}^{|A|} \text{KL}(p_T(\cdot \mid A_{<i}, E, Q) \,\|\, p_S(\cdot \mid A_{<i}, C)) \,. \tag{3}$$

Since the teacher provides soft supervision, we use the teacher distribution as the target distribution. The student distribution is read through the frozen language model head. Gradients from Eq. (3) pass through the frozen upper layers into every lower TransMem module. The backbone receives no parameter updates, while each correction is optimized for its final token-level effect.

A correction at layer K changes the input distribution of every higher layer. Training modules against independently cached layer targets therefore creates a mismatch once lower-layer corrections become active. In-loop training keeps the frozen backbone in the computation graph and optimizes all memory modules jointly. The resulting upper-layer states include the corrections from every active lower module (Zhang et al., 2026).

Notably, our training framework is designed to teach TransMem to extract query-relevant information from existing hidden states, rather than to memorize dataset-specific knowledge. The cross-domain results in Sec. 4.2 support the transferability of this capability without requiring a separate training set for each target domain.

3.3 Inference with TransMem

The TransMem module at the K-th layer consists of a Transformer block f_m^K, a learnable projection W_Δ^K, and a learnable gating network f_g^K.

TransMem applies its correction only while generating $A_{\le t}$. Once h_t^K is obtained, we divide H_C into segments of size n and use the final hidden state of each segment as a memory feature. This gives $H_M = \{h_n^K, h_{2n}^K, \ldots, h_L^K\}$. The size of H_M remains small, with a default of four memory states, ensuring a sparse memory representation.

As illustrated in Figure 2(a), we feed H_M and H_A into TransMem. They first undergo standard autoregressive attention to produce the memory feature $h_m \in \mathbb{R}^{1\times d}$ from the different segments. A learnable projection $W_\Delta^K \in \mathbb{R}^{d\times d}$ then maps h_m to the memory shift Δh by right multiplication. The complete process is

$$h_m = f_m^K(H_M, H_A), \tag{4}$$

$$\Delta h = h_m W_\Delta^K. \tag{5}$$

Not every memory shift Δh is useful, and noisy shifts may adversely affect the output. Therefore, we use the learnable gating network f_g^K to scale the memory shift. Specifically,

$$g_m = f_g^K(h_m) = \alpha \cdot \sigma\left(\frac{h_m W_g}{\tau}\right), \tag{6}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$, $\alpha = 2$, τ is the gate temperature, and $W_g \in \mathbb{R}^{d\times d}$ is a learnable gate weight matrix. The core correction formula is

$$\hat{h}_t^K = h_t^K + g_m \odot \Delta h, \tag{7}$$

where $t \ge L+1$, indicating that the memory correction is applied only during answer generation. Inference with the TransMem memory correction then becomes:

$$h_t^{K+1} = f_\theta^{K+1}(\hat{h}_t^K). \tag{8}$$

Therefore, our method enhances long-context reasoning through a lightweight network without extensively recomputing long text segments. Our method does not store additional textual information from the preceding context. Instead, its contextual awareness relies on the already computed vectors in H_M. The features in H_M can serve as memory because of the computational properties of a transformer. In particular, h_L^K encodes transformed information from $x_{1:L}$ and therefore acts as a feature representation of that prefix. A similar operation can also be found in EAGLE (Li et al., 2024) and DeepSeek MTP (DeepSeek-AI et al., 2025).

In this sense, the method parameterizes part of the transformer's memory capability in an independent network. It therefore takes a further step toward separating the reasoning and memory capabilities of an LLM.

TransMem is not applied to all layers. Empirically, we find that inserting memory modules into only the last four layers of the LLM is sufficient to achieve strong performance. It further makes

our method more lightweight. Furthermore, unlike existing latent memory methods, TransMem is invoked only when the model begins generating the answer sequence A. This sparse inference strategy further reduces the additional computation.

To construct sparse memory representations, we adopt a dynamic segmentation strategy. Specifically, we partition the long context into B segments and set the segment length to $n = L/B$. We note that this choice is not necessarily optimal, and the segment length can be adapted to different tasks or application scenarios.

4 Experiments

4.1 Experimental Setup

Evaluation and Benchmarks. We evaluate both general reasoning ability and memory effectiveness. We evaluate multi-hop question answering performance on the HotpotQA test set (Yang et al., 2018), which assesses multi-hop reasoning. Memory effectiveness is evaluated on LoCoMo (Maharana et al., 2024) and MemoryAgentBench (Hu et al., 2026), which measure the retention, retrieval, and use of information over extended interaction histories. Following the official evaluation protocol of MemoryAgentBench, the benchmark consists of four categories: Accurate Retrieval (AR), Test-Time Learning (TTL), Long-Range Understanding (LRU), and Selective Forgetting (SF). In total, MemoryAgentBench contains 3,671 evaluation questions from 146 context records, including 2,000 questions for AR, 700 questions for TTL, 171 questions for LRU, and 800 questions for SF. The final performance is reported as the question-count-weighted average score across all categories. Following mem0 (Chhikara et al., 2025), we exclude the adversarial question category from LoCoMo. LoCoMo comprises 841 single-hop (SH), 282 multi-hop (MH), 96 open-domain (OD), and 321 temporal (TP) reasoning challenges.

Baselines. We compare TransMem with representative memory approaches based on different memory mechanisms, including external memory and parametric memory methods. For external memory methods, we include BM25 RAG (Lewis et al., 2020), which retrieves relevant passages from historical contexts and appends them to the input, LLMLingua-2 (Pan et al., 2024), which compresses long contexts into shorter textual representations, MemoryBank (Zhong et al., 2024), Memory-R1-GRPO (Yan et al., 2026) and TriMem (Sun et al., 2026), which maintain historical information as retrievable textual memories. For parametric memory methods, we compare with Context2LoRA (Back et al., 2026), MemGen (Zhang et al., 2025a), MLP Memory (Wei et al., 2026) and δ-Mem (Lei et al., 2026b), which introduce additional trainable components to encode or transform contextual information into memory representations.

Implementation Details. To evaluate the generality of TransMem, we conduct experiments on LLM backbones with different architectures and parameter scales, including Qwen3-4B-Instruct (Qwen Team, 2025b), Qwen2.5-14B-Instruct (Qwen Team, 2025a), and Llama3.1-8B-Instruct (Grattafiori et al., 2024). TransMem is a learnable memory component, which is trained only on the HotpotQA training set. We follow the same data processing strategy as AgeMem (Yu et al., 2026c) to ensure the effective utilization of the HotpotQA training set. Notably, we ensure that the HotpotQA training set contains no content overlapping with the test set, thereby avoiding unfair performance gains due to test-set leakage. TransMem is inserted into the final four transformer layers of the student LLM model. Its transformer block consists of a single layer that follows the Qwen3 transformer architecture. Training uses one NVIDIA A100 Tensor Core GPU

4.2 Main Results

We evaluate TransMem on models with different scales and architectures. As shown in Table 1, TransMem consistently improves long-context reasoning performance across all evaluated models. On LoCoMo, it achieves relative improvements of at least 27% over the corresponding backbone models. On HotpotQA, TransMem yields relative EM improvements of approximately 14%. We further observe that the effectiveness of TransMem may depend on the underlying capability of the backbone model. For example, although Qwen3-4B-Instruct has fewer parameters than Qwen2.5-14B-Instruct, its newer architecture achieves comparable performance on LoCoMo. Moreover,

Table 1: **Main benchmark results comparing different memory mechanisms.**

Model	LoCoMo					HotpotQA	
	Avg.	Multi-Hop	Temporal	OpenDomain	Single-Hop	EM	F1
Qwen3-4B-Instruct	40.79	38.39	32.89	10.77	48.05	42.35	56.00
External Memory							
+ BM25 RAG	36.68$_{-4.11}$	38.12$_{-0.27}$	20.34$_{-12.55}$	9.99$_{-0.78}$	45.47$_{-2.58}$	40.35$_{-2.00}$	52.83$_{-3.17}$
+ LLMLingua-2	40.98$_{+0.19}$	39.07$_{+0.68}$	30.13$_{-2.76}$	10.98$_{+0.21}$	49.19$_{+1.14}$	36.93$_{-5.42}$	50.03$_{-5.97}$
+ MemoryBank	38.14$_{-2.65}$	37.88$_{-0.51}$	21.76$_{-11.13}$	13.35$_{+2.58}$	47.31$_{-0.74}$	39.16$_{-3.19}$	51.25$_{-4.75}$
Parametric Memory							
+ Context2LoRA	48.11$_{+7.32}$	37.95$_{-0.44}$	34.99$_{+2.10}$	16.75$_{+5.98}$	60.11$_{+12.06}$	37.85$_{-4.50}$	50.88$_{-5.12}$
+ MemGen	40.05$_{-0.74}$	32.93$_{-5.46}$	33.30$_{+0.41}$	12.67$_{+1.90}$	48.13$_{+0.08}$	35.43$_{-6.92}$	48.26$_{-7.74}$
+ MLP Memory	40.21$_{-0.58}$	32.87$_{-5.52}$	21.72$_{-11.17}$	13.81$_{+3.04}$	52.75$_{+4.70}$	44.92$_{+2.57}$	59.23$_{+3.23}$
+ δ-Mem	46.53$_{+5.74}$	42.14$_{+3.75}$	37.20$_{+4.31}$	13.35$_{+2.58}$	55.36$_{+7.31}$	49.41$_{+7.06}$	63.66$_{+7.66}$
+ **TransMem (Ours)**	**54.11**$_{+13.32}$	**52.04**$_{+13.65}$	**45.62**$_{+12.73}$	**20.64**$_{+9.87}$	**61.87**$_{+13.82}$	**50.15**$_{+7.80}$	**66.20**$_{+10.20}$
Qwen2.5-14B-Instruct	42.74	32.59	27.36	13.50	55.35	50.75	60.20
External Memory							
+ BM25 RAG	42.25$_{-0.49}$	32.17$_{-0.42}$	30.77$_{+3.41}$	23.21$_{+9.71}$	52.19$_{-3.16}$	50.46$_{-0.29}$	61.2$_{+1.00}$
+ LLMLingua-2	43.43$_{+0.69}$	34.12$_{+1.53}$	31.15$_{+3.79}$	19.48$_{+5.98}$	53.98$_{-1.37}$	45.45$_{-5.30}$	59.10$_{-1.10}$
+ MemoryBank	44.64$_{+1.90}$	34.50$_{+1.91}$	**44.70**$_{+17.34}$	23.86$_{+10.36}$	50.39$_{-4.96}$	46.66$_{-4.09}$	58.51$_{-1.69}$
Parametric Memory							
+ Context2LoRA	48.28$_{+5.54}$	35.16$_{+2.57}$	28.34$_{+0.98}$	17.23$_{+3.73}$	63.85$_{+8.50}$	48.15$_{-2.60}$	59.98$_{-0.22}$
+ MemGen	43.64$_{+0.90}$	29.36$_{-3.23}$	29.47$_{+2.11}$	17.54$_{+4.04}$	56.83$_{+1.48}$	46.89$_{-3.86}$	55.34$_{-4.86}$
+ MLP Memory	45.33$_{+2.59}$	35.65$_{+3.06}$	25.46$_{-1.90}$	21.39$_{+7.89}$	58.90$_{+3.55}$	47.35$_{-3.40}$	58.84$_{-1.36}$
+ δ-Mem	47.83$_{+5.09}$	34.99$_{+2.40}$	27.47$_{+0.11}$	27.37$_{+13.87}$	62.25$_{+6.90}$	51.38$_{+0.63}$	65.36$_{+5.16}$
+ **TransMem (Ours)**	**54.32**$_{+11.58}$	**41.20**$_{+8.61}$	41.44$_{+14.08}$	**30.38**$_{+16.88}$	**66.38**$_{+11.03}$	**58.01**$_{+7.26}$	**72.34**$_{+12.14}$
Llama3.1-8B-Instruct	34.76	39.15	19.42	25.86	40.15	49.77	58.60
External Memory							
+ BM25 RAG	34.55$_{-0.21}$	36.25$_{-2.90}$	11.58$_{-7.84}$	31.02$_{+5.16}$	43.15$_{+3.00}$	44.95$_{-4.82}$	55.12$_{-3.48}$
+ MemoryBank	33.96$_{-0.80}$	34.25$_{-4.90}$	13.01$_{-6.41}$	31.05$_{+5.19}$	42.19$_{+2.04}$	50.15$_{+0.38}$	65.97$_{+7.37}$
+ SimpleMem	36.33$_{+1.57}$	30.44$_{-8.71}$	18.77$_{-0.65}$	31.23$_{+5.37}$	45.59$_{+5.44}$	52.06$_{+2.29}$	66.15$_{+7.55}$
+ Memory-R1-GRPO	36.67$_{+1.91}$	32.87$_{-6.28}$	16.72$_{-2.70}$	8.81$_{-17.05}$	48.73$_{+8.58}$	48.54$_{-1.23}$	62.15$_{+3.55}$
+ TriMem	38.70$_{+3.94}$	34.56$_{-4.59}$	32.36$_{+12.94}$	**32.71**$_{+6.85}$	43.20$_{+3.05}$	55.36$_{+5.59}$	68.11$_{+9.51}$
Parametric Memory							
+ δ-Mem	46.21$_{+11.45}$	36.64$_{-2.51}$	31.28$_{+11.86}$	24.87$_{-0.99}$	57.56$_{+17.41}$	56.27$_{+6.50}$	69.47$_{+10.87}$
+ **TransMem (Ours)**	**51.64**$_{+16.88}$	**47.27**$_{+8.12}$	**37.29**$_{+17.87}$	28.69$_{+2.83}$	**61.21**$_{+21.06}$	**56.57**$_{+6.80}$	**71.63**$_{+13.03}$

TransMem demonstrates robustness to substantial changes in the backbone architecture. Although Llama3.1-8B-Instruct does not achieve performance comparable to the other two backbones, the transformer block within TransMem adopts the more advanced Qwen3 architecture, allowing it to achieve an F_1 score of 51.64 on LoCoMo and 71.63 on HotpotQA. These results are comparable to those of Qwen2.5-14B-Instruct with a larger parameter scale. As shown in Table 2, we further evaluate TransMem on MemoryAgentBench, which contains substantially longer contexts than LoCoMo, with many examples exceeding 256K tokens. As a result, it achieves overall performance improvements across different memory settings, demonstrating the effectiveness of the proposed memory mechanism under ultra-long-context reasoning.

Notably, although TransMem is trained only on the HotpotQA training set, it generalizes effectively to both LoCoMo and MemoryAgentBench. This phenomenon demonstrates that the learned memory capability transfers well across diverse long-context reasoning tasks, rather than memorizing knowledge specific to the HotpotQA dataset.

4.3 Ablations and Analysis

Ablation Studies. To investigate the contribution of each component in TransMem, as shown in Table 3, we conduct an ablation study by removing individual modules. Although the Gating Network provides a relatively modest improvement in the overall score, it yields a noticeably larger gain on multi-hop reasoning. This observation suggests that the gating mechanism facilitates the retrieval and utilization of relevant memory representations, thereby improving the model's ability to exploit stored information during complex reasoning. The learnable projection W_{Δ}^{K} also helps

Table 2: Results on MemoryAgentBench comparing different memory mechanisms.

Model	Avg.	AR	TTL	LRU	SF
Qwen3-4B-Instruct	28.95	35.30	26.14	**34.45**	14.37
External Memory					
+ BM25 RAG	24.43	32.20	9.74	37.86	15.00
+ LLMLingua-2	15.63	21.45	1.43	38.45	8.62
+ MemoryBank	17.65	22.65	7.67	36.36	9.88
Parametric Memory					
+ Context2LoRA	32.53	40.00	29.86	25.15	17.75
+ MemGen	29.61	34.85	28.45	**44.30**	14.38
+ MLP Memory	28.80	35.35	26.00	31.19	14.38
+ δ-Mem	38.85	**44.40**	47.29	41.55	17.00
+ **TransMem (Ours)**	**40.00**	41.60	**54.64**	36.19	**24.00**

Table 3: Module ablation on LoCoMo with Qwen3-4B-Instruct.

Model	Avg.	MH	TP	OD	SH
TransMem	**54.11**	**52.04**	**45.62**	20.64	**61.87**
w/o Gating-Net	53.09	48.04	45.09	**21.57**	61.45
w/o W_Δ^K	51.57	51.15	40.54	20.48	59.48

Table 4: Training objectives on LoCoMo F_1 with Qwen3-4B-Instruct.

Method	Avg.	MH	TP	OD	SH
SFT	48.82	47.06	40.66	19.45	55.89
GRPO	52.42	50.44	43.92	17.42	60.34
OPD	52.80	51.17	45.01	18.96	60.19
ECSD	**54.11**	**52.04**	**45.62**	**20.64**	**61.87**

stabilize overall performance by decoupling the generation of the memory shift from its modulation by the gating network

Efficiency. To evaluate the efficiency of TransMem, we measure its additional computational cost and inference latency under different context lengths and memory configurations. We exclude the frozen backbone prefill computation and report only the overhead introduced by the memory module. As shown in Figure 3(a), δ-Mem incurs a nearly linear increase in computation as context length grows, with approximately 100 GFLOPs overhead per additional 10K tokens. In contrast, TransMem has context-independent computational cost, which only depends on the number of selected memory slots B (i.e., $|H_M|$). By injecting memory modules into only a subset of layers, TransMem maintains lower overhead than δ-Mem even under long contexts. Figure 3(b) further shows that TransMem introduces nearly constant inference latency across different context lengths. Unlike methods requiring repeated encoding of historical representations, TransMem performs sparse memory computation only during generation, enabling scalable long-context reasoning.

Training Method. In addition to evidence-conditioned self-distillation (ECSD), we explored supervised fine-tuning (SFT) (Ouyang et al., 2022) and two subsequent policy-optimization objectives. As illustrated in Table 4, direct SFT on gold answers performs substantially worse than ECSD. Starting from the SFT checkpoint, both on-policy distillation (OPD) (Li et al., 2026) and group relative policy optimization (GRPO) (Shao et al., 2024) improve cross-domain LoCoMo performance, reaching 52.80 and 52.42 F1, respectively. However, neither matches ECSD, which achieves 54.11 F1. GRPO reaches 52.42 F_1, compared with 54.11 for ECSD. This result suggests that answer-level reinforcement learning does not compensate for a weaker memory-representation objective: ECSD's token-level supervision from a privileged-evidence teacher remains important for learning the TransMem module.

Gate Parameterization. We compare fixed residual scaling with two learned gates, $g = 2\sigma(z)$ and $g = 3\sigma(z) - 1$, on Qwen3-4B with the Last-4 configuration using HotpotQA for training, where $z = h_m W_g/\tau$. Both learned gates improve over the fixed scale, while $2\sigma(z)$ achieves the best overall performance (+1.02 F1 points). The signed variant provides no additional benefit and is therefore not adopted.

Effect of the Training Set We compare different training corpora with Qwen3-4B and dynamically gated TransMem modules inserted into the last four layers. All models are evaluated on the same 1,540 LoCoMo questions using token-level F_1 (Table 6). HQA, LME, and LoCoMo-T denote HotpotQA, LongMemEval, and LoCoMo-train, respectively. The HQA+LME mixture achieves 53.95 F_1, close to the main HotpotQA setting (53.95 vs. 54.11). In contrast, reducing HotpotQA samples and adding LoCoMo-T lowers performance to 51.80, mainly due to temporal degradation. Removing overlapping LoCoMo questions from evaluation yields a similar score (51.87), indicating that the drop is not caused by evaluation leakage.

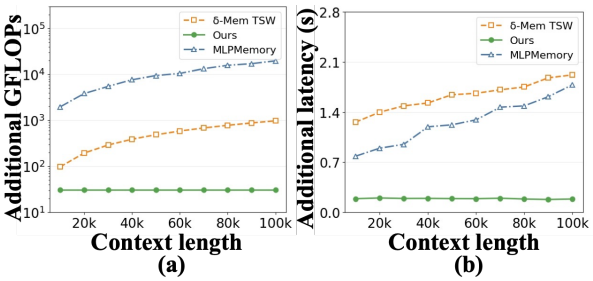


Figure 3: Efficiency comparison in terms of computation cost and inference latency. (a) Additional computational cost introduced by the memory module, shown on a logarithmic scale. (b) Average additional inference latency introduced by the memory module.

Table 5: LoCoMo F_1 under different gate parameterizations.

Gate g	Avg.	MH	TP	OD	SH
1	53.09	48.04	45.09	**21.57**	61.45
$2\sigma(z)$	**54.11**	**52.04**	45.62	20.64	**61.87**
$3\sigma(z) - 1$	53.83	50.51	**45.87**	20.87	61.76

Table 6: LoCoMo F_1 with different training corpora for Qwen3-4B TransMem.

Training set	Avg.	MH	TP	OD	SH
HQA	**54.11**	**52.04**	**45.62**	20.64	61.87
HQA + LME	53.95	50.40	45.54	**21.85**	**62.02**
Mixed dataset	51.80	50.53	37.20	19.13	61.53

Table 7: **LoCoMo F_1 under different TransMem injection depths and layer positions.** Experiments are conducted on Qwen3-4B-Instruct.

Model	**Avg.**	**MH**	**TP**	**OD**	**SH**
Last-2	53.14	49.91	44.70	18.35	61.41
Last-4	**54.11**	**52.04**	**45.62**	20.64	**61.87**
Last-8	53.27	50.29	44.31	20.44	61.43
Middle 28-31	52.85	50.18	43.74	20.07	60.96
Middle 22-25	52.32	47.24	43.26	**20.67**	61.09
Middle 18-21	51.54	45.18	44.48	20.17	59.96
Middle 12-15	51.11	44.00	42.92	20.57	60.11

Table 8: HotpotQA results (%) with Qwen3-4B-Instruct. L_{TM} is the number of Transformer blocks inside its single memory module.

Method	Memory layout	Training set	HQA score
Final-hidden	$L_{\mathrm{TM}} = 4$	HQA+LME	39.8/50.8
	$L_{\mathrm{TM}} = 2$	HQA+LME	39.1/48.4
	$L_{\mathrm{TM}} = 2$	HQA	39.1/50.8
	$L_{\mathrm{TM}} = 4$	LME	36.7/50.0
	$L_{\mathrm{TM}} = 2$	LME	38.3/52.3
TransMem	Last-4	HQA	**50.15/66.20**

Injection Method. We investigate the effect of inserting TransMem at different layers. As shown in Table 7, increasing the number of inserted layers does not always improve performance. Moreover, inserting TransMem into intermediate layers yields weaker improvements than inserting it into the last layers. These results suggest that deeper layers provide more effective representations for memory enhancement.

We compare layer-wise injection with a final-hidden variant, where a single TransMem module is inserted after the final normalization. As shown in Table 8, increasing the internal Transformer depth does not consistently improve performance. These results suggest that simply increasing memory capacity at a single location is insufficient, motivating our layer-wise design that distributes memory corrections across multiple backbone layers.

5 Conclusion

We propose TransMem, a lightweight parametric memory module that improves test-time long-context reasoning without explicit memory construction or backbone scaling. TransMem transforms sparse, position-sensitive historical hidden states into reusable memory representations and injects them into current reasoning. Evidence-conditioned self-distillation trains the module to recover an evidence-only teacher's predictions from full contexts. Experiments on LoCoMo, MemoryAgentBench, and HotpotQA show consistent gains across backbones with sparse memory representations, supporting efficient decoupling of memory capability from backbone reasoning capacity. Future work will explore adaptive MoE memory extensions for TransMem and applications to long-horizon agent scenarios.

Limitations: Our method is sensitive to training data quality. Because TransMem is not trained on ultra-long datasets, its effectiveness on such tasks may be limited. Future work will explore further improvements for better long-context generalization.

AI USE STATEMENT

In this work, we used generative AI tools for language editing and assistance with code development. All AI-assisted content, mathematical derivations, experimental implementations, and generated text were reviewed and verified by the authors. We take responsibility for the final content of this work, including text, claims, code, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not introduce new training data or collect human-subject data. The proposed method inherits potential risks and biases from the underlying pretrained generative models.

REPRODUCIBILITY STATEMENT

We provide the complete mathematical formulation, optimization procedure, and implementation details required to reproduce the proposed method. Hyperparameter settings, evaluation protocols, and additional experimental results are provided in the main paper and appendix. Code will be released upon publication.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024.

Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s2: A compositional generalist-specialist framework for computer use agents. In *Second Conference on Language Modeling*, 2025.

Seungju Back, Dongwoo Lee, Naun Kang, Taehee Lee, S. K. Hong, Youngjune Gwon, and Sungjin Ahn. Understanding lora as knowledge memory: An empirical analysis, 2026.

Jiaqi Cao, Jiarui Wang, Rubin Wei, Qipeng Guo, Kai Chen, Bowen Zhou, and Zhouhan Lin. Memory decoder: A pretrained, plug-and-play memory for large language models. In *Advances in Neural Information Processing Systems*, 2025.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory, 2025.

DeepSeek-AI, Aixin Liu, Bei Feng, et al. Deepseek-v3 technical report, 2025.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. LongRoPE: Extending LLM context window beyond 2 million tokens. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11091–11104. PMLR, 21–27 Jul 2024.

Yiming Du, Wenyu Huang, Danna Zheng, Zhaowei Wang, Sebastien Montella, Mirella Lapata, Kam-Fai Wong, and Jeff Z. Pan. Rethinking memory in llm based agents: Representations, operations, and emerging topics, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, et al. The llama 3 herd of models, 2024.

Cheng-Yu Hsieh, Yung-Sung Chuang, Chun-Liang Li, Zifeng Wang, Long Le, Abhishek Kumar, James Glass, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, and Tomas Pfister. Found in the middle: Calibrating positional attention bias improves long context utilization. In *Findings of the Association for Computational Linguistics*, pp. 14982–14995. Association for Computational Linguistics, August 2024.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. In *The Fourteenth International Conference on Learning Representations*, 2026.

Fengxian Ji, Zhuohan Xie, Jingpu Yang, Fan Zhang, Zirui Song, and Xiuying Chen. Parametric memory decoding for zero-shot routing in lora-based external parametric memory, 2026.

Chris Latimer, Nicoló Boschi, Andrew Neeser, Chris Bartholomew, Gaurav Srivastava, Xuan Wang, and Naren Ramakrishnan. Hindsight is 20/20: Building agent memory that retains, recalls, and reflects, 2025.

Haodong Lei, Junming Liu, Yirong Chen, Ding Wang, and Hongsong Wang. Memcot: Test-time scaling through memory-driven chain-of-thought, 2026a.

Jingdi Lei, Di Zhang, Junxian Li, Weida Wang, Kaixuan Fan, Xiang Liu, Qihan Liu, Xiaoteng Ma, Baian Chen, and Soujanya Poria. δ-mem: Efficient online memory for large language models, 2026b.

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.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huan ang Gao, Wenkai Yang, Zhiyuan Liu, and Ning Ding. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe, 2026.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning*, 2024.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for llm agents, 2026.

Junming Liu, Yifei Sun, Weihua Cheng, Haodong Lei, Yirong Chen, Licheng Wen, Xuemeng Yang, Daocheng Fu, Pinlong Cai, Nianchen Deng, Yi Yu, Shuyue Hu, Botian Shi, and Ding Wang. Memverse: Multimodal memory for lifelong learning agents, 2025.

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

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 13851–13870. Association for Computational Linguistics, August 2024.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22. Curran Associates Inc., 2022. ISBN 9781713871088.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, 2026.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Ruhle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics ACL*, pp. 963–981. Association for Computational Linguistics, 2024.

Ryan Wei Heng Quek, Sanghyuk Lee, Alfred Wei Lun Leong, Arun Verma, Alok Prakash, Nancy F. Chen, Bryan Kian Hsiang Low, Daniela Rus, and Armando Solar-Lezama. Memo: Memory as a model, 2026.

Qwen Team. Qwen2.5 technical report, 2025a.

Qwen Team. Qwen3 technical report, 2025b.

Miguel Moura Ramos, Duarte M. Alves, and André F. T. Martins. A recipe for long-context reasoning in large language models via on-policy optimization and distillation, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.

Yaorui Shi, Yuxin Chen, Siyuan Wang, Sihang Li, Hengxing Cai, Qi GU, Xiang Wang, and An Zhang. Look back to reason forward: Revisitable memory for long-context LLM agents. In *International Conference on Learning Representations*, 2026.

Gaurav Srivastava, Shuxiang Cao, and Xuan Wang. ThinkSLM: Towards reasoning in small language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 32612–32662. Association for Computational Linguistics, November 2025.

Gaurav Srivastava, Aafiya Hussain, Chi Wang, Yingyan Celine Lin, and Xuan Wang. effgen: Enabling small language models as capable autonomous agents, 2026.

Jingwei Sun, Jianing Zhu, Jiangchao Yao, Tongliang Liu, and Bo Han. Rethinking how to remember: Beyond atomic facts in lifelong llm agent memory. *arXiv preprint arXiv:2605.19952*, 2026.

Jiejun Tan, Zhicheng Dou, Liancheng Zhang, Yuyang Hu, Yiruo Cheng, and Ji-Rong Wen. Memsifter: Offloading llm memory retrieval via outcome-driven proxy reasoning, 2026.

Xiaoqiang Wang, Chao Wang, Hadi Nekoei, Christopher Pal, Alexandre Lacoste, Spandana Gella, Bang Liu, and Perouz Taslakian. Mem-π: Adaptive memory through learning when and what to generate, 2026.

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. In *International Conference on Learning Representations*, 2026.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*, 2025.

Wenxuan Xie and Xuhong Wang. Decoupled reasoning with implicit fact tokens (DRIFT): A dual-model framework for efficient long-context inference. In *Findings of the Association for Computational Linguistics: ACL*, pp. 38047–38067. Association for Computational Linguistics, 2026.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schuetze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In *Annual Meeting of the Association for Computational Linguistics (Long Papers)*, pp. 12805–12825. Association for Computational Linguistics, July 2026.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652, 2024.

Wei Yang, Jinwei Xiao, Hongming Zhang, Qingyang Zhang, Yanna Wang, and Bo Xu. Coarse-to-fine grounded memory for llm agent planning, 2026.

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*, 2018.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. Memagent: Reshaping long-context LLM with multi-conv RL-based memory agent. In *International Conference on Learning Representations*, 2026a.

Xiaoyun Yu, Li fan, Xiangfei Qiu, Nanqing Dong, Yonggui Huang, Honggang Qi, Geguang Pu, Wanli Ouyang, Xi Chen, and Jilin Hu. Memts: Internalizing domain knowledge via parameterized memory for retrieval-free domain adaptation of time series foundation models, 2026b.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. In *Annual Meeting of the Association for Computational Linguistics (Long Papers)*, pp. 21457–21483. Association for Computational Linguistics, July 2026c.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. Memgen: Weaving generative latent memory for self-evolving agents, 2025a.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. Memevolve: Meta-evolution of agent memory systems, 2025b.

Xinsen Zhang, Zhenkai Ding, Tianjun Pan, Run Yang, Chun Kang, Xue Xiong, and Jingnan Gu. Opsdl: On-policy self-distillation for long-context language models, 2026.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model-based agents. *ACM Trans. Inf. Syst.*, 43(6), September 2025c. ISSN 1046-8188.

Huaqin Zhao, Jiaxi Li, Yi Pan, Shizhe Liang, Xiaofeng Yang, Fei Dou, Tianming Liu, and Jin Lu. HELENE: Hessian layer-wise clipping and gradient annealing for accelerating fine-tuning LLM with zeroth-order optimization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 26044–26067. Association for Computational Linguistics, 2025.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. In *Forty-third International Conference on Machine Learning*, 2026.

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, pp. 19724–19731, 2024.

CONTENTS

A More Experiments **14**

A.1 Additional Hidden-State Supervision 14

A.2 TransMem Feature Source 14

A.3 Gate Statistics on Qwen3-4B 15

A.4 Layer-Wise Gate Behavior 15

A.5 More Models 15

A.6 Sensitivity to the Number of Memory Segments 16

A.7 Comparison of Training Data Scale 16

A MORE EXPERIMENTS

A.1 ADDITIONAL HIDDEN-STATE SUPERVISION

We also tested whether directly matching the teacher's final-layer feature improves knowledge transfer. In addition to the self-distillation objective, the Last-4 model was trained with

$$\mathcal{L} = \mathcal{L}_{\mathrm{SD}} + \beta \left\| H'_{\mathrm{last}} - H^{\mathrm{tea}}_{\mathrm{last}} \right\|_2^2 .$$

Here, H'_{last} is the answer-position feature from the student LLM's final layer after the TransMem update, and $H^{\mathrm{tea}}_{\mathrm{last}}$ is the corresponding final-layer feature produced by the privileged-evidence teacher. The prime therefore denotes the TransMem-updated student representation. All variants use Qwen3-4B and the same QASPER–HotpotQA multi-domain training set.

Table 9: LoCoMo F_1 when adding hidden-state regression to the self-distillation objective. All rows use the Qwen3-4B Last-4 architecture.

β	Overall	MH	Temp.	Open	SH
0	**54.11**	**52.04**	45.62	20.64	**61.87**
0.3	51.91	46.94	44.12	19.16	60.28
1.0	52.18	46.38	44.26	19.29	60.90
3.0	51.79	45.29	43.62	**20.89**	60.62

Hidden-state supervision provides no benefit in this ablation: every $\beta > 0$ setting performs below the no-regression variant ($\beta = 0$). One likely explanation is that the output-distribution objective already supplies the task-relevant behavioral signal, whereas imposing additional Euclidean alignment over-constrains the student's hidden space. In other words, supervising hidden states too strongly may restrict functionally equivalent representations without improving cross-domain memory use.

A.2 TRANSMEM FEATURE SOURCE

Our default Last-4 model computes the memory update for LLM layer K from the output H^K of that layer. We compare it with *transmem-before*, which computes the same update from the layer input H^{K-1} while still adding the resulting residual to H^K. All other architectural and training choices remain unchanged.

Table 10: Feature-source ablation for Qwen3-4B Last-4 TransMem.

Variant	Source state	Added to	LoCoMo F_1
Default	H^K	H^K	**54.11**
Before	H^{K-1}	H^K	52.18

Table 11: Gate statistics and LoCoMo F_1 for Qwen3-4B. HQA, LME, and LoCoMo-F denote HotpotQA, LongMemEval, and the full LoCoMo training set.

Setting	Training set	Gate	μ/σ	F_1
Last-4	HQA	$2\sigma(z)$	1.137/0.384	**54.11**
Last-8	HQA	$2\sigma(z)$	0.605/0.319	53.27
Last-4	HQA	$3\sigma(z) - 1$	1.200/0.398	53.83
Last-4 (before)	HQA	$2\sigma(z)$	1.237/0.313	53.92
Last-4	LME+ LoCoMo-F	$2\sigma(z)$	1.561/0.143	50.68

Table 12: LoCoMo F_1 with additional student backbones.

Student backbone	Injection	Training set	F_1
Qwen2.5-7B	Last-4	HQA	49.98
Qwen3-8B	Last-4	HQA	50.76

Reading the post-block state is 0.19 F_1 points better, although the small gap indicates that TransMem is not highly sensitive to this choice. We retain H^K because it exposes each memory module to the representation already transformed by its corresponding LLM layer.

A.3 Gate Statistics on Qwen3-4B

Table 11 compares the learned gate distributions under architectural, parameterization, and training-set changes. Every experiment in this table uses Qwen3-4B. The reported mean and standard deviation aggregate validation answer positions across all injected layers; because the gate is token-dependent, they are descriptive statistics rather than fixed model parameters.

The Last-4 and Last-8 models learn qualitatively different global scales: Last-4 mildly amplifies memory updates on average, whereas Last-8 suppresses them. The signed gate and transmem-before variants remain close to the main model. Training on the much smaller LME–LoCoMo mixture produces a high-mean, low-variance gate and a lower F_1 despite containing target-domain examples, consistent with overfitting rather than improved memory use.

A.4 Layer-Wise Gate Behavior

Aggregate statistics hide a consistent division of labor among the four injected layers. Figure 4 reports the token-level gate distribution for the main Qwen3-4B Last-4 checkpoint on 128 in-domain HotpotQA development questions and a 386-question cross-domain LoCoMo diagnostic subset.

The gate forms a monotonic amplification-to-suppression profile: Layer 32 strongly amplifies the TransMem update, Layers 33–34 progressively reduce it, and Layer 35 suppresses the direct intervention. More importantly, the same profile is preserved on LoCoMo. The corresponding means from Layers 32 to 35 are $1.618, 1.332, 0.975, 0.610$ on HotpotQA and $1.472, 1.298, 0.996, 0.599$ on LoCoMo. This cross-domain stability suggests that the gate behavior is not limited to a dataset-specific binary switch.

A.5 More Models

We additionally train Last-4 TransMem with HotpotQA on two other student backbones. Table 12 reports their absolute LoCoMo scores together with Qwen3-4B as a reference.

The method can be trained without changing the Last-4 design across these model sizes and families. Absolute performance does not scale monotonically with parameter count under this protocol, so these results support architectural portability rather than a claim that a larger frozen student necessarily yields a larger gain.

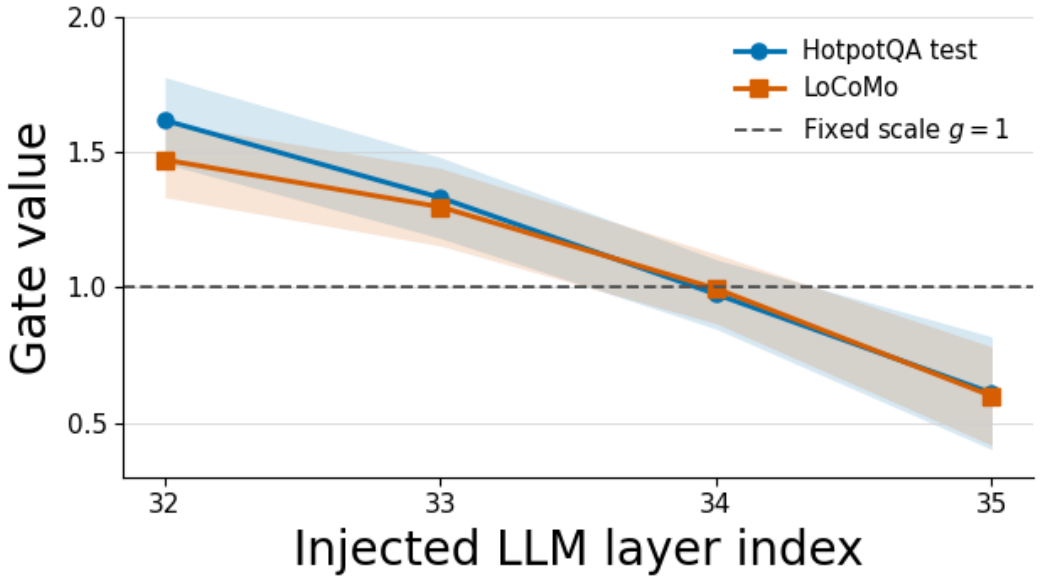


Figure 4: Layer-wise gate values for Qwen3-4B Last-4 TransMem. Curves show means and shaded regions show one standard deviation across answer tokens. The dashed line is the fixed residual scale $g = 1$.

A.6 Sensitivity to the Number of Memory Segments

We first study the memory-segmentation granularity, denoted by B in the paper. We reuse the Qwen3-4B TransMem checkpoint trained with $B = 4$ and vary only B at inference time. Thus, this experiment measures inference-time sensitivity rather than the effect of retraining a separate model for every value of B. Table 13 reports the official task metrics (percentage points) on the deterministic subsets of MemoryAgentBench.

The results are nearly invariant over $B \in \{4, 8, 16, 32\}$: increasing B from 4 to 32 changes EventQA by only 0.2 points and improves RULER-1 by 1.0 point, while the other three scores remain unchanged. This suggests that TransMem is not brittle to the precise segmentation granularity. We therefore retain $B = 4$ as the default because it uses fewer memory states.

Table 13: Inference-time sensitivity to the number of memory segments B. FC-SH and FC-MH denote the single-hop and multi-hop FactConsolidation subsets, respectively.

B	RULER-1	RULER-2	EventQA	FC-SH	FC-MH
4	40.00	**30.00**	47.40	**43.00**	**5.00**
8	40.00	**30.00**	47.20	**43.00**	**5.00**
16	40.00	**30.00**	47.20	**43.00**	**5.00**
32	**41.00**	**30.00**	**47.60**	**43.00**	**5.00**

A.7 Comparison of Training Data Scale

We compare the amount of task-specific training data used by TransMem with the training data reported for three related memory methods. The counting units differ across methods, so Table 14 reports each method's original unit rather than treating all values as comparable question counts.

The official HotpotQA distractor training split contains 90,447 examples (Yang et al., 2018); our HotpotQA-derived training set contains 32,768 questions. The δ-mem paper specifies one epoch over the shortest 2,219-example QASPER split. For the 1B-parameter MLP Memory, 33M is the reported *maximum* number of token-level training samples, while its Wikipedia-2021 source datastore

Table 14: Reported training-data scale for memory methods. Our HotpotQA count is the subset used to train TransMem, not the full HotpotQA training split.

Method	Training corpus	Size	Counting unit
TransMem (ours)	HotpotQA subset	32,768	QA examples
δ-mem (Lei et al., 2026b)	QASPER (shortest split)	2,219	examples
MLP Memory (Wei et al., 2026)	Wikipedia-2021	$\leq$33M	token-level samples (cap)
MemGen (Zhang et al., 2025a)	task-specific splits	448–8,000	examples per task

contains approximately 5B tokens; the paper does not establish 33M as the realized unique-sample count. MemGen trains separately on each task: 3.32K ALFWorld, 4.13K TriviaQA, 8K KodCode, 912 BigCodeBench, 448 GPQA, 7.47K GSM8K, and 1.6K MATH examples. PopQA has no dedicated training split and uses the TriviaQA-trained model.