# STAMP: LEARNING CROSS-SESSION MEMORY IN NATIVE RECURRENT STATES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Hybrid attention-recurrent language models process each session as a token sequence, using attention caches for local context while compressing part of the history into fixed-size native recurrent states. These states provide bounded memory within a sequence, but long-term memory requires information from a completed session to initialize later independent calls after the source tokens and attention key–value cache are discarded. We introduce STAMP, a learned cross-session interface that makes the backbone's existing recurrent states persistent and reusable across independent calls. A learned write mechanism encodes future-useful information into terminal recurrent states, and a lightweight state adapter maps them into initial states for future sessions. STAMP is trained end-to-end over multi-session trajectories from a general-purpose context–question–answer corpus, but at inference it requires no per-session optimization. Experiments show that STAMP transfers without benchmark-specific retraining to LoCoMo, NextMem, PersonalMem, and LongMemEval, improving state-only recall over backbone-matched Metis while keeping the memory footprint fixed as sessions accumulate.

## 1 INTRODUCTION

Long-term memory in an interactive language model is not just a question of longer context (Maharana et al., 2024; Wu et al., 2025; Jiang et al., 2025; Zhang et al., 2026d). A model that serves a user across repeated interactions must preserve selected information after the source dialogue has ended and the next call begins as an independent sequence. This setting turns memory into a state-transition problem: what bounded model-internal state should survive one session and initialize the next?

Current architectures already contain a natural substrate for such a transition. Within a single inference sequence, Transformers retrieve earlier tokens through self-attention (Vaswani et al., 2017), with key–value (KV) caches giving each new token access to the preceding context (Pope et al., 2023). This mechanism is precise but grows linearly with context length. Linear attention instead summarizes token history through recurrent updates (Katharopoulos et al., 2020), where each KV pair modifies a fixed-size state that later tokens query. Recent hybrid attention-recurrent backbones such as Kimi Linear (Kimi Team et al., 2025) and Qwen3.5 (Qwen Team, 2026) combine such bounded recurrent state with full attention at scale. These backbones therefore suggest a compact route to *memory-native foundation models*: long-term memory carried by bounded native recurrent state rather than by an external or token-level store that grows with interaction history.

The difficulty is that recurrent states in these models are trained mainly for within-sequence computation. A bounded state can summarize a continuous token stream, but repeated updates cause new associations to compete with old ones, producing interference and eventual overwrite. Prior work improves this within-sequence memory through adaptive decay and selective gating (Sun et al., 2023; Yang et al., 2025; Hatamizadeh et al., 2026), structured state updates (Dao & Gu, 2024), hybrid attention (De et al., 2024; Ren et al., 2025; Du et al., 2026a), and expanded or routed state capacity (Du et al., 2026b; Zhang et al., 2026b; Bayat et al., 2026). Cross-session memory asks for a different capability. A session is a self-contained dialogue processed as its own sequence; without persistence, the next session starts with an empty attention cache and a zero recurrent state, $\mathbf{S}_0 = \mathbf{0}$. Thus, even relevant information from the previous dialogue is unavailable to the next session unless some state is carried across the session boundary.



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.

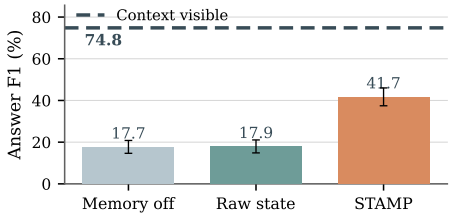



Figure 1: Motivating cross-session Answer F1: raw terminal-state carry is nearly identical to memory off, while our proposed STAMP learns a persistent state transition.

The most direct idea is to reuse the normally zero-initialized recurrent state as persistent memory. Session $\tau$ would begin from the previous persistent state $\mathbf{P}_{\tau-1}$, process its current sequence, and convert the terminal recurrent state into $\mathbf{P}_\tau$ for later sessions. Repeating this transition keeps storage fixed as conversations accumulate. However, the raw terminal state is not trained for this role: pretraining prepares it for same-sequence continuation, with the KV cache still present, rather than for initializing a future independent dialogue after the source tokens and attention cache have been discarded. Although $\mathbf{S}_0$-Tuning (Young, 2026) shows that initial states can be expressive, persistent memory requires amortizing how such states are written and updated from completed sessions. Figure 1 illustrates this *session-boundary gap*: raw terminal-state carry is akin to memory off, while a learned state-transition interface performs much better with the same state size.

To address this gap, we introduce State Transition Adaptation for Memory Persistence (STAMP), a learned cross-session interface that converts a session's terminal recurrent state into the persistent initializer for future sessions. STAMP keeps the Qwen3.5 hybrid attention-recurrent backbone and recurrent kernels frozen, and trains only lightweight components around the native state: learned write tokens along with write-only low-rank adaptation (LoRA) (Hu et al., 2022) concentrate future-useful information during the session and low-rank state transformations adapt the terminal state for future initialization. This lifts native token-level recurrence to session-level persistence without introducing a separate deep memory module.

We train STAMP end-to-end over multi-session trajectories from a small general-purpose context–question–answer corpus, then evaluate it on frozen Qwen3.5-4B and 27B hybrid attention-recurrent backbones. The same learned interface is used at inference without per-session optimization or benchmark-specific retraining, and transfers to LoCoMo (Maharana et al., 2024), LongMemEval (Wu et al., 2025), PersonalMem (Zhang et al., 2025; Du et al., 2024), and NextMem (Zhang et al., 2026d). Scaling and repeated-write experiments additionally show that native recurrent states can support compact long-term recall, and also identify repeated-write interference and long-horizon retention as the main remaining limitations. To sum up, this paper makes three contributions:

- We formulate persistent memory as the problem of carrying information after the source context and KV cache are removed, exposing a session-boundary gap.
- We present STAMP, a lightweight learned transition that makes the native recurrent states persistent across sessions without introducing a separate deep memory module.
- We evaluate STAMP across multiple long-term memory benchmarks, model scales, repeated-write regimes, and storage settings, showing strong recall with a fixed-size persistent state.

## 2 Related Work

We compare memory methods along three axes: what representation persists, how it is written, and whether it remains available across a session boundary. These axes distinguish memory within an uninterrupted sequence from STAMP's learned transition between independent sessions.

### 2.1 Native Recurrent-State Memory

Recurrent models retain history in evolving representations. Long short-term memory (LSTM) carries hidden states through a sequence (Hochreiter & Schmidhuber, 1997), while Transformer-XL (Dai et al., 2019), Compressive Transformers (Rae et al., 2020), and Recurrent Memory Transformers (Bulatov et al., 2022) carry or compress representations across contiguous segments. These methods primarily study within-sequence memory, treating each segment as a continuation of the same stream.

Within bounded-state architectures, one line of work studies how recurrent states summarize a growing history. HiPPO (Gu et al., 2020) and S4 (Gu et al., 2022) develop principled history compression and scalable state-space representations. Subsequent recurrent architectures pursue bounded-state language modeling through time mixing in RWKV (Peng et al., 2023), parallelizable retention in

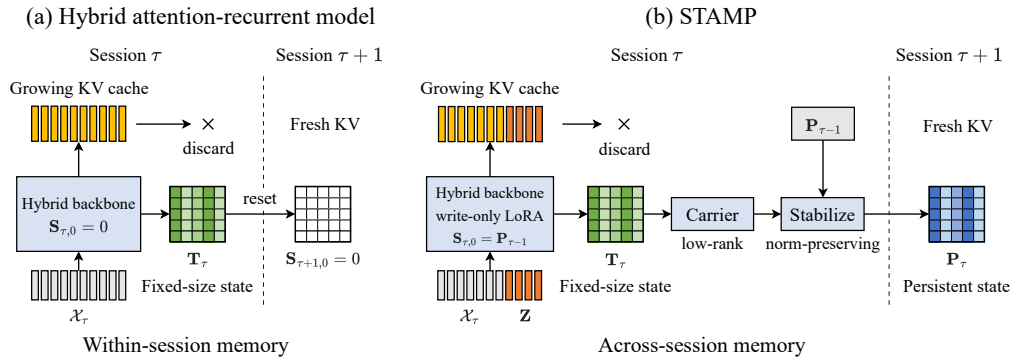



Figure 2: **Overview of STAMP across sessions.** (a) In the standard hybrid baseline shown here, recurrent states reset between independent sessions. (b) Starting from the persistent state $\mathbf{P}_{\tau-1}$, STAMP processes the current context with learned write tokens and write-only LoRA to produce the terminal recurrent state $\mathbf{T}_\tau$. A low-rank carrier and norm-controlled update transform $\mathbf{T}_\tau$ into $\mathbf{P}_\tau$. At the session boundary, the source context and KV caches are discarded, and the next session receives fresh input and fresh KV caches, with its recurrent state initialized from $\mathbf{P}_\tau$.

RetNet (Sun et al., 2023), and selective state-space updates in Mamba (Gu & Dao, 2024). Linear attention makes this recurrence explicit: kernelized attention can be evaluated as a recurrent linear state update (Katharopoulos et al., 2020), and the resulting update admits an associative fast-weight interpretation (Schlag et al., 2021). Later linear attention variants refine this update through expanded gated state (Qin et al., 2024), attention–state-space duality and adaptive decay (Dao & Gu, 2024), and gated delta-rule updates (Yang et al., 2024; 2025; Hatamizadeh et al., 2026).

A complementary line addresses the limits of compression for exact recall and fine-grained selection (Arora et al., 2024). Hybrid architectures combine recurrent states with local or full attention, allowing attention to complement compressed history (De et al., 2024; Ren et al., 2025; Lieber et al., 2024; Glorioso et al., 2024; Kimi Team et al., 2025; Qwen Team, 2026). Mixture-of-memories routing (Du et al., 2026b), native hybrid-attention designs (Du et al., 2026a), content-routed state anchors (Zhang et al., 2026b), and incremental or sliding memory activation (Bayat et al., 2026; Liu & Liu, 2026) further enlarge effective memory. Together, these methods shape what a model can represent or recall as a token stream proceeds. STAMP studies the subsequent boundary: how the native recurrent matrices of a pretrained hybrid backbone can retain useful information when one session ends and initialize a later independent call.

### 2.2 Alternative Memory Mechanisms

Beyond native recurrent states, alternative memory mechanisms store information as token-level context, auxiliary latent or state representations, or parameter updates. For *token-level context*, KV cache retains selected activations for attention to history. StreamingLLM (Xiao et al., 2024) keeps attention sink and recent tokens; SnapKV (Li et al., 2024) selects important KV positions per head, while PyramidKV (Cai et al., 2025) varies cache budgets across layers. Such retention reduces inference costs without itself specifying persistence across independent sessions.

*Auxiliary latent and state representations* store information outside the backbone's ordinary recurrent state. Early antecedents include Memory Networks (Weston et al., 2015), which learn access to explicit memory slots; End-to-End Memory Networks (Sukhbaatar et al., 2015), which make memory access end-to-end trainable; and Neural Turing Machines (Graves et al., 2014), which learn differentiable reads and writes over external storage. For LLMs, the In-context Autoencoder (Ge et al., 2024) compresses context into memory vectors, MemGen (Zhang et al., 2026a) generates latent tokens from an agent's reasoning state, and GradMem (Kuratov et al., 2026) optimizes prefix memory tokens at test time with model weights frozen. Compact online representations include $\delta$-mem (Lei et al., 2026), which couples an associative state to a frozen backbone, and Metis (Zhang et al., 2026c), which introduces persistent dynamic states through a dedicated memory pathway. These two methods resemble STAMP in compact-state form but introduce an added state or pathway.

*Parameter update methods* store information in weights or adapters. A related parametric precedent is persistent attention (Sukhbaatar et al., 2019), whose learned vectors encode input-independent knowledge rather than new context. Context-conditioned methods instead write information into

weights or adapters. Doc-to-LoRA (Charakorn et al., 2026) generates context-specific adapters, while SHINE (Liu et al., 2026) maps context to LoRA updates in one forward pass. Optimization provides another writing mechanism: test-time training (Sun et al., 2020) adapts model parameters through self-supervision, Temp-LoRA (Wang et al., 2024) progressively trains temporary adapters on generated text, and Titans (Behrouz et al., 2025) updates the parameters of a learned memory module at test time. STAMP learns a terminal-to-initial transition for the backbone's native recurrent matrices, carrying session-specific information across independent sessions after source tokens and KV caches are discarded, without per-session optimization.

# 3 PRELIMINARIES

## 3.1 NATIVE RECURRENT STATES IN HYBRID ATTENTION-RECURRENT LANGUAGE MODELS

Hybrid attention-recurrent language models combine full-attention layers that retain token-wise key–value (KV) caches with recurrent layers that summarize token history in fixed-size native states. We write the recurrence for one head and omit its index. Linear attention (Katharopoulos et al., 2020) provides a simple illustration. Let $\mathbf{q}_t, \mathbf{k}_t \in \mathbb{R}^{d_k}$ denote the feature-mapped query and key, and $\mathbf{v}_t \in \mathbb{R}^{d_v}$ the value. After token $t$, the recurrence maintains a matrix-valued state $\mathbf{S}_t \in \mathbb{R}^{d_k \times d_v}$. We omit specific query–key normalization and output scaling for clarity. The recurrent update is

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

Each key–value pair writes an association into this state for subsequent queries, and its shape is independent of sequence length. Qwen3.5 backbones use Gated DeltaNet (Yang et al., 2025), which introduces data-dependent decay and delta-rule updates while retaining this fixed-size matrix state. We keep the native recurrence unchanged and learn a cross-session interface around it. Let $\mathcal{I}_{\text{rec}}$ denote the set of layers. We collect their native states at token position $t$ in session $\tau$ as

$$\mathbf{S}_{\tau,t} = \{\mathbf{S}_{\tau,t;\,j}\}_{j \in \mathcal{I}_{\text{rec}}} \in \mathcal{R}, \tag{2}$$

where $\mathcal{R}$ denotes the product space of all native recurrent matrices. For $\mathbf{X}, \mathbf{Y} \in \mathcal{R}$, we define

$$\langle \mathbf{X}, \mathbf{Y} \rangle_{\mathcal{R}} = \sum_{j \in \mathcal{I}_{\text{rec}}} \langle \mathbf{X}_j, \mathbf{Y}_j \rangle_F, \qquad \|\mathbf{X}\|_{\mathcal{R}} = \sqrt{\langle \mathbf{X}, \mathbf{X} \rangle_{\mathcal{R}}}. \tag{3}$$

For multi-head layers, the collections and Frobenius inner products include all heads.

## 3.2 CROSS-SESSION PERSISTENCE

We use $\tau = 1, 2, \ldots$ to index chronological sessions and $t = 1, \ldots, n_\tau$ to index tokens within session $\tau$. Let $\mathcal{X}_\tau = (x_{\tau,1}, \ldots, x_{\tau,n_\tau})$ denote session $\tau$, and let $\mathbf{P}_\tau \in \mathcal{R}$ denote the persistent recurrent state retained after that session. Before the first session, we set $\mathbf{P}_0 = \mathbf{0}$. To carry memory across sessions, session $\tau$ begins from

$$\mathbf{S}_{\tau,0} = \mathbf{P}_{\tau-1}, \qquad \mathcal{K}^{\text{attn}}_{\tau,0} = \varnothing, \tag{4}$$

where $\mathcal{K}^{\text{attn}}$ denotes the ordinary KV caches of full-attention layers. After the session is written, its source tokens, transient activations, and attention KV caches are discarded. A later write or query inherits only $\mathbf{P}_\tau$ from past sessions and starts with fresh attention caches. Native recurrence supports within-sequence continuation with existing states and attention caches. Cross-session persistence instead requires a completed session's state to initialize an independent call after its source context and attention cache are removed. We refer to this change in role as the *session-boundary gap*.

**Low-rank parameterization.** We use LoRA (Hu et al., 2022) for lightweight adaptation, parameterizing a frozen pretrained weight as $\mathbf{W}' = \mathbf{W} + \mathbf{BA}$ with rank $r \ll \min(d_{\text{in}}, d_{\text{out}})$. We use this low-rank parameterization for both the write branch and the state carrier, as detailed in Section 4.

# 4 METHODOLOGY

## 4.1 OVERVIEW

To bridge the session-boundary gap, STAMP learns how a completed session's native recurrent state should initialize a future independent call. Given the previous persistent state $\mathbf{P}_{\tau-1}$ and current

session $\mathcal{X}_\tau$, the interface updates memory as

$$\mathbf{P}_\tau = \mathcal{T}_\Theta\left(\mathbf{P}_{\tau-1}, \mathcal{X}_\tau\right). \tag{5}$$

This transition has two steps: an *extractor* writes information for future use into the terminal state $\mathbf{T}_\tau$, and a *state carrier* converts it into the persistent initializer $\mathbf{P}_\tau$:

$$(\mathbf{P}_{\tau-1}, \mathcal{X}_\tau) \xrightarrow{\text{extractor}} \mathbf{T}_\tau \xrightarrow{\text{state carrier}} \mathbf{P}_\tau. \tag{6}$$

After writing, the source tokens and attention KV cache are discarded. The next call starts from $\mathbf{P}_\tau$ and fresh attention caches, so only a fixed-size native state carries memory across sessions. We keep the backbone and recurrent kernels frozen and learn the interface over multi-session trajectories; inference requires no per-session optimization.

### 4.2 Session Information Extractor

The extractor prepares session information for later recall after the source context is removed. It appends $m$ learned write tokens $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_m)$ and activates a write-only LoRA on selected frozen projections,

$$\mathbf{W}^{\mathrm{w}} = \mathbf{W} + \mathbf{B}^{\mathrm{w}}\mathbf{A}^{\mathrm{w}}, \tag{7}$$

where $\mathbf{W}$ denotes the frozen pretrained backbone weights and $\mathbf{B}^{\mathrm{w}}\mathbf{A}^{\mathrm{w}}$ is a rank-$r_{\mathrm{w}}$ update. The LoRA is enabled for writing and disabled for answering; pretrained weights remain frozen. Session $\tau$ starts from the previous persistent state $\mathbf{P}_{\tau-1}$ and a fresh attention KV cache, as defined in Equation 4. A single forward pass over the session and appended write tokens gives

$$\mathbf{T}_\tau = \Pi_{\mathcal{R}}\, \mathcal{F}_{\phi,\omega}\left([\mathcal{X}_\tau; \mathbf{Z}] \,\middle|\, \mathbf{S}_{\tau,0} = \mathbf{P}_{\tau-1}, \mathcal{K}^{\mathrm{attn}}_{\tau,0} = \varnothing\right), \tag{8}$$

where $\phi$ denotes the frozen backbone, $\omega$ the write-LoRA parameters, and $\Pi_{\mathcal{R}}$ extracts the terminal recurrent states from all recurrent layers. We denote this collection by $\mathbf{T}_\tau$. The write-only LoRA shapes this pass toward information useful beyond the current session. We then discard the source tokens, transient activations, and attention KV cache, and pass $\mathbf{T}_\tau$ to the state carrier.

### 4.3 Low-rank terminal-to-initial carrier

The extractor writes information into a terminal state, but later sessions need an initializer that works without the original context or attention cache. The state carrier adapts $\mathbf{T}_\tau$ for this role using low-rank transformations of each recurrent matrix. For layer $j$, the carrier proposal is

$$\mathbf{C}_{\tau;j} = \mathcal{G}_{\psi,j}\left(\mathbf{T}_{\tau;j}\right) = g_j^0 \mathbf{T}_{\tau;j} + g_j^{\mathrm{k}} \mathbf{B}_j^{\mathrm{k}} \mathbf{A}_j^{\mathrm{k}} \mathbf{T}_{\tau;j} + g_j^{\mathrm{v}} \mathbf{T}_{\tau;j} \mathbf{B}_j^{\mathrm{v}} \mathbf{A}_j^{\mathrm{v}}. \tag{9}$$

Here, $\mathbf{B}_j^{\mathrm{k}}\mathbf{A}_j^{\mathrm{k}}$ and $\mathbf{B}_j^{\mathrm{v}}\mathbf{A}_j^{\mathrm{v}}$ are rank-$r$ transformations, and $g_j^a$ for $a \in \{0, \mathrm{k}, \mathrm{v}\}$ are learned scalar gains. The three terms retain a scaled copy of the terminal state and transform its key and value axes. To control state magnitude across repeated writes, we set $\mathbf{P}_1 = \mathbf{C}_1$ and use norm-preserving spherical linear interpolation (SLERP) between $\mathbf{P}_{\tau-1}$ and $\mathbf{C}_\tau$ for later writes. The resulting $\mathbf{P}_\tau$ initializes the next call. The interpolation details are specified in Section 5.

### 4.4 Training Objective

We train the extractor and state carrier jointly on multi-session trajectories from a general-purpose context–QA corpus. Each session $\tau$ contains a context $\mathcal{X}_\tau$ and its associated QA set $\mathcal{D}_\tau$. Starting from $\mathbf{P}_0 = \mathbf{0}$, the persistent state is updated recursively as in Equation 5. At depth $\tau$, $\mathcal{B}_\tau$ contains QA pairs from the current and sampled previous sessions, with $\mathcal{B}_1 = \mathcal{D}_1$. Only $\Theta = \{\mathbf{Z}, \omega, \psi\}$, corresponding to the write tokens, write-LoRA, and carrier parameters, is optimized, while the pretrained backbone $\phi$ remains frozen. The three objectives encourage recall after context removal, compatibility with new context, and preservation of the backbone's general capabilities.

For QA supervision, let $\ell_{\mathrm{NLL}}$ denote the per-example negative log-likelihood over target answer tokens. The *post-transition* path evaluates state-only recall after context and KV removal, while the *pre-transition* path applies the same supervision while the current context remains visible:

$$\ell^{(\tau)}_{\mathrm{post}} = \mathbb{E}_{(q,a)\sim\mathcal{B}_\tau} \ell_{\mathrm{NLL}}\left(a \mid q, \mathbf{P}_\tau\right), \; \ell^{(\tau)}_{\mathrm{pre}} = \mathbb{E}_{(q,a)\sim\mathcal{B}_\tau} \ell_{\mathrm{NLL}}\left(a \mid \mathcal{X}_\tau, q, \mathbf{P}_{\tau-1}\right). \tag{10}$$

The post loss therefore optimizes information recoverable from the persistent state alone, while the pre loss encourages existing memory to remain compatible with new evidence.

We further encourage preservation of the backbone's general capabilities by distilling memory-conditioned predictions on unrelated prompts $q \sim \mathcal{Q}_{\text{unrel}}$ from the frozen memory-off model:

$$\ell_{\text{pres}}^{(\tau)} = \mathbb{E}_{q \sim \mathcal{Q}_{\text{unrel}}} D_{\text{KL}} \left( \text{sg} \left[ p_\phi^{\text{off}}(\cdot \mid q) \right] \,\|\, p_\phi(\cdot \mid q, \mathbf{P}_\tau) \right). \tag{11}$$

Because the two predictions differ only in the persistent state, $\ell_{\text{pres}}$ isolates memory-induced behavioral drift. The objective at depth $\tau$ is therefore

$$\ell^{(\tau)} = \ell_{\text{post}}^{(\tau)} + \eta_{\text{pre}} \ell_{\text{pre}}^{(\tau)} + \eta_{\text{pres}} \ell_{\text{pres}}^{(\tau)}. \tag{12}$$

We train with a curriculum over cross-session trajectory length. At optimization step $s$, we sample the number of sessions $U \sim \pi_s$, with the support of $\pi_s$ gradually expanding from one to ten sessions. The primary training objective is

$$\ell_{\text{train}}(s) = \mathbb{E}_{U \sim \pi_s} \left[ \ell^{(U)} \right]. \tag{13}$$

All $U$ session transitions are executed sequentially, with supervision applied only after the final session. We use truncated backpropagation through time, retaining the full forward state trajectory while propagating gradients through only the two most recent session transitions.

## 5 Experimental Setup

This section describes the model configurations and training procedure, benchmark settings and baselines, and evaluation metrics used throughout our experiments.

**Models and training details.** We instantiate STAMP on Qwen3.5-4B and Qwen3.5-27B hybrid backbones. We keep the pretrained backbone frozen and use a memory interface consisting of eight learned write tokens, a rank-32 state carrier (Equation 9), and a write-only LoRA (Equation 7), with rank 8 for 4B and rank 16 for 27B. By default, we use SLERP with $\lambda = 0.75$ to preserve the recurrent-state norm after the first write. We report ablations in Section 6.3 and details in Appendix C.2. Interface training uses a general-purpose context– question–answer corpus derived from MS MARCO MQA (Bajaj et al., 2016), following the data formulation of SHINE (Liu et al., 2026). The training data are separate from the downstream benchmarks, so evaluation tests transfer to unseen conversations and question distributions. We use the resulting memory interface unchanged across evaluation benchmarks, and we provide details in Appendix A.

**Benchmarks and evaluation settings.** We evaluate persistent memory by discarding the source context and its attention KV cache before answering, leaving only the persistent recurrent state. Following Metis (Zhang et al., 2026c), our main evaluation uses all 1,527 LoCoMo-Gold questions from ten conversations: each question starts from a reset state, writes the relevant evidence sessions in chronological order, and is answered from the resulting persistent state. These evaluation settings isolate memory construction and state-only retrieval: the evidence content is available during writing but is no longer accessible when the question is answered. Thus, the static results test memory construction from supplied evidence. We apply the same state-only evaluation setting to NextMem (1,697 questions), PersonalMem (644 questions), and LongMemEval (500 questions). Benchmark-specific preprocessing and evaluation details are provided in Appendix B.

**Baselines and controlled comparisons.** We compare against the memory-off backbone and representative approaches that differ in what is retained after the source context is removed. Metis (Zhang et al., 2026c) maintains a dedicated dynamic memory pathway on the same Qwen3.5 backbones, while $\delta$-mem (Lei et al., 2026) uses an online associative memory. Our latent-memory baseline compresses each context into 16 soft tokens. Temp-LoRA (Wang et al., 2024), $S_0$-Tuning (Young, 2026), and SHINE (Liu et al., 2026) instead retain information through temporary parameters, optimized initial states, or meta-learned context-to-parameter mappings. Together, these baselines cover state, latent, and parametric mechanisms while keeping the source context unavailable during evaluation. We use released checkpoints or official implementations where available, and align evidence order, questions, decoding, and scoring across methods.

Table 1: **Static memory construction across four benchmarks.** Answer F1 and GPT-4.1-mini judge scores are reported in % (higher is better) on LoCoMo (LCM), LongMemEval (LME), NextMem (NM), and PersonalMem (PM); Overall is the macro-average across the four benchmarks. Trainable Params is reported in millions (M), and Aux. Memory denotes persistent storage beyond the backbone's native recurrent state, in MiB. STAMP retains 48 MiB (4B) or 144 MiB (27B) of native state per persistent instance, and these bytes are excluded from the Aux. column. Bold and underline indicate the best and second-best results, respectively.

| | | | | **Answer Prediction (F1)** | | | | | **LLM Judge (GPT-4.1-mini)** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | **Size** | **Params** | **Aux.** | LCM | LME | NM | PM | **Overall** | LCM | LME | NM | PM | **Overall** |
| Memory off | 4B | 0 | 0 | 8.63 | 5.78 | 18.76 | 9.55 | 10.68 | 8.71 | 6.90 | 21.82 | 7.34 | 11.19 |
| $S_0$-Tuning | 4B | 13 | 0 | 12.93 | 8.51 | 24.09 | 14.79 | 15.08 | 13.88 | 7.75 | 27.87 | 10.79 | 15.07 |
| Temp-LoRA | 4B | 130 | 495 | 19.22 | 8.09 | 35.85 | 13.01 | 19.04 | 21.94 | 9.30 | 42.30 | 15.26 | 22.20 |
| $\delta$-Mem | 4B | 5 | 0.004 | 8.72 | 9.52 | 18.42 | 10.22 | 11.72 | 10.61 | 8.05 | 20.04 | 5.82 | 11.13 |
| Metis | 4B | 126 | 16 | 15.51 | 19.97 | 33.75 | 14.08 | 20.83 | 16.37 | 14.82 | 41.37 | 14.36 | 21.73 |
| Latent memory | 4B | 16 | 0.078 | 17.95 | 16.41 | 40.04 | 16.28 | 22.67 | 17.86 | 14.70 | 43.46 | 13.39 | 22.35 |
| SHINE | 8B | 888 | 1124 | 22.53 | 19.95 | **54.86** | 14.29 | <u>27.91</u> | 25.38 | 20.60 | <u>59.09</u> | <u>19.18</u> | <u>31.06</u> |
| Metis | 27B | 671 | 32 | 19.55 | 20.50 | 36.86 | <u>17.19</u> | 23.53 | <u>26.96</u> | 19.15 | 51.40 | 18.09 | 28.90 |
| STAMP | 4B | 17 | 0 | <u>25.40</u> | <u>25.03</u> | 40.49 | 17.01 | 26.98 | 26.06 | <u>21.95</u> | 44.65 | 13.59 | 26.56 |
| STAMP | 27B | 118 | 0 | **30.44** | **32.93** | <u>53.90</u> | **25.62** | **35.72** | **33.27** | **30.85** | **60.42** | **28.84** | **38.35** |

**Evaluation metrics.** We report answer-prediction F1 together with an LLM-based evaluation using GPT-4.1-mini. F1 measures lexical agreement with the reference answer, while the LLM judge provides a complementary measure that is less sensitive to surface-form variation. The same evaluation pipeline is applied to all methods within each benchmark.

## 6 Experiments

In this section, we evaluate our method under question-conditioned static memory construction, where memory is reset for each question, and continual writing, where memory persists across successive writes. Static QA may involve multiple evidence-session writes, as in LoCoMo. We further ablate the model components, update geometry, and multi-state reading strategies.

### 6.1 Static Memory Construction

We first evaluate whether STAMP can convert a completed context into a persistent state that remains useful after the source text and attention KV cache are removed. As shown in Table 1, STAMP-4B achieves an average F1 of 26.98 and an average Judge score of 26.56, outperforming backbone-matched Metis by 6.15 and 4.83 points, respectively, and exceeding the other 4B baselines overall. Scaling to Qwen3.5-27B further improves performance to 35.72 F1 and 38.35 Judge, surpassing Metis-27B by 12.19 and 9.45 points and giving the strongest overall results among the compared methods. Importantly, the same memory interface is used without benchmark-specific retraining. F1 gains extend across all four suites, although 4B PersonalMem Judge is below Metis.

Table 2: **LoCoMo results by question category.** Single, Temp., Multi, and Open denote single-hop, temporal, multi-hop, and open-domain questions; parentheses indicate the number of questions.

| **Method** | **Single** (840) | **Temp.** (320) | **Multi** (278) | **Open** (89) | **Overall** (1,527) |
|---|---|---|---|---|---|
| Memory off | 9.77 | 2.65 | 9.17 | 17.65 | 8.63 |
| Temp-LoRA | 24.62 | 6.60 | 16.45 | 22.25 | 19.22 |
| $S_0$-Tuning | 14.75 | 5.27 | 12.68 | <u>24.12</u> | 12.93 |
| $\delta$-Mem-4B | 10.37 | 2.89 | 9.27 | 12.48 | 8.72 |
| Metis-4B | 15.74 | 14.86 | 14.24 | 19.72 | 15.51 |
| Latent memory | 20.93 | 8.46 | <u>20.13</u> | 17.09 | 17.95 |
| SHINE | 25.36 | 17.82 | 19.72 | 21.55 | 22.53 |
| Metis-27B | 21.38 | 18.93 | 16.21 | 14.97 | 19.55 |
| STAMP-4B | <u>27.94</u> | **25.37** | 18.75 | 22.37 | <u>25.40</u> |
| STAMP-27B | **35.96** | <u>20.25</u> | **26.14** | **28.39** | **30.44** |

These results are obtained with only 17M and 118M trainable parameters at 4B and 27B, respectively, compared with 126M and 671M for backbone-matched Metis. Moreover, STAMP adds no auxiliary memory module, but each persistent instance must retain its native recurrent state: 48 MiB at 4B or 144 MiB at 27B. This footprint does not grow with the number of writes.

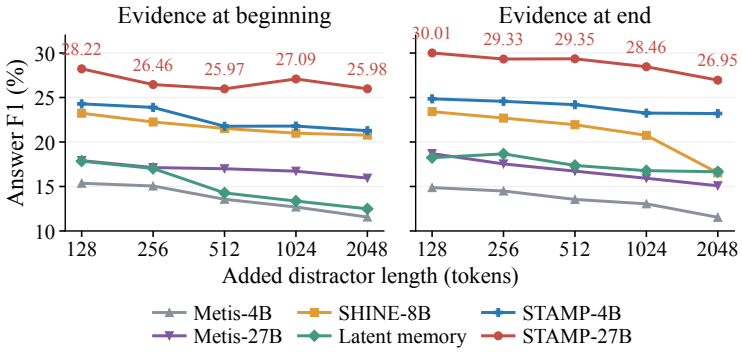



Figure 3: Answer F1 on LoCoMo as the added distractor length increases. Evidence is placed at the beginning (left) or end (right). Numbers annotate STAMP-27B.

The LoCoMo category breakdown in Table 2 further shows that the improvement is not limited to single-hop recall. STAMP-27B performs best on single-hop, multi-hop, and open-domain questions, while STAMP-4B obtains the strongest temporal result, indicating that the persistent state remains useful for questions requiring temporal interpretation or integration of multiple pieces of information.

**Scaling with context length.** In Figure 3, we test robustness to irrelevant context on all 1,527 LoCoMo questions by adding 128–2,048 distractor tokens per question across the original evidence-session writes, with evidence placed before or after the distractors. As distractor length increases sixteenfold, STAMP-27B drops from 28.22 to 25.98 F1 with evidence first. With evidence last, STAMP-4B and STAMP-27B drop from 24.85 to 23.20 and from 30.01 to 26.95, respectively. These results show moderate degradation under increasing distraction.

### 6.2 Continual Memory Writing

The preceding experiments evaluate whether a fixed-size recurrent state can store a completed context. We next examine whether that state remains useful after repeated updates, when newly written evidence must coexist with previously stored information.

We evaluate 2-, 4-, and 8-write trajectories, with each write processing one complete evidence record. Memory resets between trajectories. After each write, the source text and attention KV cache are discarded, leaving only the recurrent state. We report *final all-seen F1* over questions about all records at trajectory endpoints, counting each question once per condition. All methods use the same evidence sequences and questions. Final QA sets are fixed across write depths for LongMemEval, NextMem, and PersonalMem, while LoCoMo uses growing chronological prefixes (Appendix B).

Table 3: **Continual memory writing.** Final all-seen Answer F1 (%) after 2, 4, and 8 writes.

| | | LCM | | | LME | | | NM | | | PM | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | **Size** | 2 | 4 | 8 | 2 | 4 | 8 | 2 | 4 | 8 | 2 | 4 | 8 |
| Memory off | 4B | 7.45 | 7.41 | 8.47 | 9.82 | 9.82 | 9.82 | 18.27 | 18.27 | 18.27 | 9.37 | 9.37 | 9.37 |
| Temp-LoRA | 4B | 8.02 | 7.90 | 9.15 | 10.43 | 10.22 | 11.02 | 20.54 | 21.50 | 21.99 | 9.28 | 9.06 | 10.17 |
| $\delta$-Mem | 4B | 7.29 | 7.88 | 8.42 | 9.95 | 10.42 | 10.01 | 17.61 | 17.56 | 17.58 | 6.06 | 6.25 | 5.90 |
| Metis | 4B | 8.21 | 9.49 | 10.44 | 15.16 | 15.82 | 17.24 | 21.26 | 14.72 | 12.79 | 7.46 | 6.47 | 5.40 |
| SHINE | 8B | 24.85 | 19.33 | 18.87 | 14.53 | 12.59 | 12.84 | 36.64 | 27.00 | 22.60 | 8.40 | 8.14 | 9.34 |
| Metis | 27B | 16.79 | 13.93 | 12.99 | 17.00 | 16.94 | **17.33** | 25.40 | 20.48 | 17.98 | 11.83 | 10.19 | 9.43 |
| STAMP | 4B | 21.26 | 17.53 | 19.46 | 19.50 | 16.83 | 13.29 | 25.32 | 21.98 | 18.95 | 11.51 | 10.69 | 9.56 |
| STAMP | 27B | **26.76** | **19.60** | **20.40** | **29.55** | **20.80** | 15.82 | **46.17** | **37.64** | **29.99** | **17.85** | **15.07** | **15.16** |

Table 3 shows that STAMP retains an overall advantage under continual writing, outperforming backbone-matched Metis on three of four benchmarks after eight writes at both model scales. These results support the effectiveness of the learned cross-session interface and stronger state-only recall after repeated updates, with fixed-size native recurrent memory and no per-session optimization. However, interference under repeated writes remains substantial.

### 6.3 Ablation Study

This section examines the extractor and carrier, carry geometry, and whether earlier state snapshots help recover information. In all three continual diagnostics below, each question's context is divided into ten ordered segments and written sequentially before that question is answered.

**Component ablation.** Table 4 isolates the extractor and carrier on LoCoMo. We train the extractor-only, carrier-only, and full variants separately under the same settings, optimizing only the included memory components while keeping the backbone frozen. The carrier-only variant outperforms the extractor-only variant in both static (20.78 vs. 12.24 F1) and ten-write continual (12.79 vs. 10.31 F1) settings, consistent with the terminal-to-initial mismatch contributing to raw-carry failure. Training both components gives the strongest results (25.40 and 15.87 F1). Results on the other datasets are provided in Appendix C.1.

Table 4: Component ablation on LoCoMo.

| **Variant** | Static | Continual |
|---|---|---|
| Raw state | 7.55 | 8.87 |
| Extractor only | 12.24 | 10.31 |
| Carrier only | <u>20.78</u> | <u>12.79</u> |
| STAMP (full) | **25.40** | **15.87** |

**Carry geometry.** Repeated state updates may accumulate scale or directional errors, so we study how the carry geometry affects memory retention. We compare default SLERP with alternative interpolation weights, radial clipping, polar clipping, and unconstrained replacement (Appendix C.2). As shown in Figure 4, default SLERP achieves 25.40 F1 in static evaluation and 15.87 F1 in continual evaluation on LoCoMo. Radial clipping gives the highest static F1 (26.11), while SLERP with $\lambda = 1.0$ gives the highest continual F1 (16.10). These results favor norm control or constrained movement over unconstrained replacement.

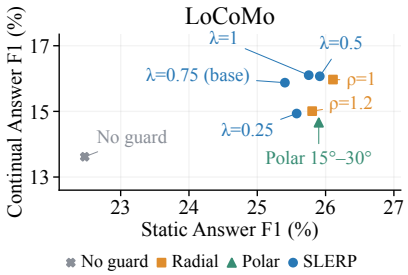



Figure 4: Write geometry on LoCoMo.

**Multi-state memory.** Repeated writes may make the latest state poor for earlier facts, so we inspect snapshots from each ten-write trajectory. Recent- State Bank (RSB) answers from the latest $K$ states and selects by generation confidence, and Cascade evaluates the latest state and at most one router-selected older state. As shown in Figure 5, with $K = 8$, RSB raises F1 from 15.87 to 18.33 on LoCoMo and from 22.01 to 24.57 on LongMemEval; Cascade reaches 18.01 and 23.66. These gains require extra storage, and future work could combine multi-state memory with the state compression explored in Appendix C.3 to reduce storage costs.

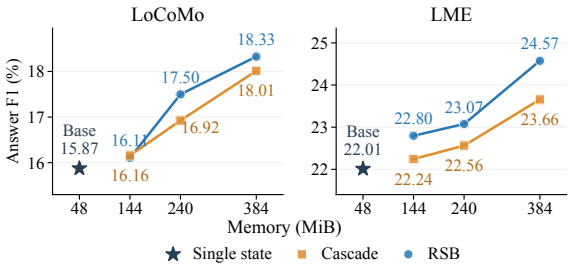



Figure 5: Multi-state diagnostic on LCM and LME.

## 7 Conclusion

Long-term interaction calls for memory beyond a single context window. Hybrid-attention models provide compact native states, but reusing them across independent sessions remains underexplored. In this paper, we propose STAMP to turn a session-terminal recurrent state into persistent memory. Its extractor uses learned write tokens and a write-only LoRA; a low-rank carrier prepares the state for reinitialization, and norm-preserving carry supports repeated writes. In static evaluation with evidence selected per question, STAMP improves state-only recall over backbone-matched Metis on four benchmarks at both model scales, without benchmark-specific retraining.

This is an initial exploration of model-native long-term memory. Repeated writes show persistence after the source context and attention KV cache are discarded, but also reveal interference. Our held-out diagnostic still trails full-context answering. A single fixed-size linear-attention state cannot retain an unbounded history. Future work should explore bounded growth of persistent states and selective consolidation into model parameters. By making transient session states reusable across calls, STAMP lays a foundation for more durable memory.

## AI USE STATEMENT

The authors used generative AI tools for several auxiliary tasks, including improving the manuscript's language, organizing LaTeX content, checking numerical calculations, and formatting tables. Any material produced with AI assistance was examined and, where necessary, revised by the authors before inclusion. The authors conducted the research and are responsible for the methods, results, citations, and manuscript.

## ETHICS STATEMENT

This work is an exploratory study of model-native persistent memory using open-source backbone models and publicly available datasets, focusing on methodological development and empirical evaluation. Within the scope of the methodology and experiments presented here, we have not identified additional ethical concerns specific to this study.

## REPRODUCIBILITY STATEMENT

Section 4 describes the memory interface and primary training objectives, with additional training details in Appendix A. Appendix B documents the evaluated subsets, evidence construction, memory resets, decoding, and scoring. We distinguish per-question static evaluation, continual writing of complete evidence records, and the ten-segment diagnostics. Component ablations, carry rules, and post-write state compression are described in Appendices C.1, C.2, and C.3, respectively.

## REFERENCES

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

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. MS MARCO: A human generated MAchine reading COmprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Reza Bayat, Ali Behrouz, Vahab Mirrokni, and Aaron Courville. Proteus: Incremental memory activation for long-context sequence modeling. *arXiv preprint arXiv:2608.16844*, 2026.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *Advances in Neural Information Processing Systems*, volume 38, pp. 113506–113543, 2025.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. In *Advances in Neural Information Processing Systems*, volume 35, pp. 11079–11091. Curran Associates, Inc., 2022.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, et al. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *Second Conference on Language Modeling*, 2025.

Rujikorn Charakorn, Edoardo Cetin, Shinnosuke Uesaka, and Robert Tjarko Lange. Doc-to-LoRA: Learning to instantly internalize contexts. *arXiv preprint arXiv:2602.15902*, 2026.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988. Association for Computational Linguistics, 2019.

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

Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

Jusen Du, Jiaxi Hu, Zhang Tao, Weigao Sun, and Yu Cheng. Native Hybrid Attention for efficient sequence modeling. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3826–3842, 2026a.

Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, Zhang Tao, and Yu Cheng. MoM: Linear sequence modeling with mixture-of-memories. In *International Conference on Learning Representations*, 2026b.

Yiming Du, Hongru Wang, Zhengyi Zhao, Bin Liang, Baojun Wang, Wanjun Zhong, Zezhong Wang, and Kam-Fai Wong. Perltqa: A personal long-term memory dataset for memory classification, retrieval, and fusion in question answering. In *Proceedings of the 10th SIGHAN Workshop on Chinese Language Processing (SIGHAN-10)*, pp. 152–164, 2024.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *International Conference on Learning Representations*, 2024.

Paolo Glorioso, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. Zamba: A compact 7B SSM hybrid model. *arXiv preprint arXiv:2405.16712*, 2024.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. HiPPO: Recurrent memory with optimal polynomial projections. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1474–1487, 2020.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022.

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

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Bowen Jiang, Zhuoqun Hao, Young-Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo J Taylor, and Dan Roth. Know me, respond to me: Benchmarking LLMs for dynamic user profiling and personalized responses at scale. In *Second Conference on Language Modeling*, 2025.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165. PMLR, 2020.

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

Yuri Kuratov, Matvey Kairov, Aydar Bulatov, Ivan Rodkin, and Mikhail Burtsev. GradMem: Learning to write context into memory with test-time gradient descent. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Jingdi Lei, Di Zhang, Junxian Li, Weida Wang, Kaixuan Fan, Xiang Liu, Qihan Liu, Xiaoteng Ma, Baian Chen, and Soujanya Poria. $\delta$-mem: Efficient online memory for large language models. *arXiv preprint arXiv:2605.12357*, 2026.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 22947–22970, 2024.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid Transformer-Mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

Bo Liu and Qiang Liu. Maglev: Sliding recurrent memory. *arXiv preprint arXiv:2608.02870*, 2026.

Yewei Liu, Xiyuan Wang, Yansheng Mao, Yoav Gelbery, Haggai Maron, and Muhan Zhang. SHINE: A scalable in-context hypernetwork for mapping context to LoRA in a single pass. *arXiv preprint arXiv:2602.06358*, 2026.

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 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, 2024.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Xingjian Du, et al. RWKV: Reinventing RNNs for the Transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 14048–14077, 2023.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In *Proceedings of Machine Learning and Systems*, volume 5, pp. 606–624, 2023.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. HGRN2: Gated linear RNNs with state expansion. In *First Conference on Language Modeling*, 2024.

Qwen Team. Qwen3.5-Omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

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

Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. In *International Conference on Learning Representations*, 2025.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9355–9366. PMLR, 2021.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, volume 28, pp. 2440–2448, 2015.

Sainbayar Sukhbaatar, Edouard Grave, Guillaume Lample, Hervé Jégou, and Armand Joulin. Augmenting self-attention with persistent memory. *arXiv preprint arXiv:1907.01470*, 2019.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9229–9248. PMLR, 2020.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Yan Wang, Dongyang Ma, and Deng Cai. With greater text comes greater necessity: Inference-time training helps long text generation. In *First Conference on Language Modeling*, 2024.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *International Conference on Learning Representations*, 2015.

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.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 56501–56523. PMLR, 2024.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, 2025.

Jack Young. S0 tuning: Zero-overhead adaptation of hybrid recurrent-attention models. *arXiv preprint arXiv:2604.01168*, 2026.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. MemGen: Weaving generative latent memory for self-evolving agents. In *International Conference on Learning Representations*, 2026a.

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

Zeyu Zhang, Yang Zhang, Haoran Tan, Rui Li, and Xu Chen. Explicit vs implicit memory: Exploring multi-hop complex reasoning over personalized information. *arXiv preprint arXiv:2508.13250*, 2025.

Zeyu Zhang, Ziliang Guo, Yihang Sun, Xichong Zhang, Xixuan Hao, Zehao Lin, Yang Zhang, Xiaoyan Zhao, Tong Shen, Bo Tang, et al. Metis: Memory foundation model. *arXiv preprint arXiv:2607.26760*, 2026c.

Zeyu Zhang, Rui Li, Xiaoyan Zhao, Yang Zhang, Wenjie Wang, Xu Chen, and Tat-Seng Chua. NextMem: Towards latent factual memory for LLM-based agents. *arXiv preprint arXiv:2603.15634*, 2026d.

# A Training Details

**Data.** We form session trajectories from a general context–question–answer corpus derived from MS MARCO MQA, following SHINE. Each session contains a context and its questions. At the end of a trajectory, training samples questions about both recent and earlier sessions. Validation uses a fixed, separate sample from the main training corpus.

**Optimization.** The Qwen3.5 backbone is frozen. The interface has eight learned write tokens, a rank-32 carrier, and write-only LoRA of rank 8 for 4B or rank 16 for 27B. We sample trajectories of up to ten sessions and backpropagate through the two most recent writes. The final interface is trained for 50,000 steps. Both learning-rate stages use AdamW, weight decay $10^{-4}$, and gradient norm clipping at 1. For Equation 12, the post-transition QA loss has unit weight and $\eta_{\text{pre}} = 0.25$. We set $\eta_{\text{pres}}$ to 0.10 for 4B and 0.20 for 27B.

Table 5: Learning-rate schedule for the interface.

| Stage | Learning rate | Schedule | Warmup updates |
|---|---|---|---|
| First 5,000 updates | $3 \times 10^{-6}$ to $3 \times 10^{-7}$ | Cosine | 200 |
| Following 45,000 updates | $10^{-6}$ | Constant | 0 |

**State regularization.** Training adds two auxiliary penalties with equal weights of 0.01. Using the state-space norm $\|\cdot\|_{\mathcal{R}}$ defined in the main text, we penalize relative updates after the first write and terminal norm growth:

$$\ell_{\text{upd}} = \frac{1}{U-1}\sum_{\tau=2}^{U}\frac{\|\mathbf{P}_\tau - \mathbf{P}_{\tau-1}\|_{\mathcal{R}}^2}{\max(\|\mathbf{P}_{\tau-1}\|_{\mathcal{R}}, \epsilon)^2}, \quad \ell_{\text{norm}} = \left[\log\frac{\|\mathbf{P}_U\|_{\mathcal{R}}}{\text{sg}[\|\mathbf{P}_1\|_{\mathcal{R}}]} - \log(1.25)\right]_+^2. \tag{14}$$

Here $[x]_+ = \max(x, 0)$ and $\epsilon = 10^{-6}$; the log argument is also clamped below by $\epsilon$ in implementation. We set $\ell_{\text{upd}} = 0$ for $U = 1$. These penalties are separate from the inference-time carry rule.

# B Evaluation Details

**Dataset statistics.** Table 6 summarizes our evaluation subsets, rather than the full benchmark releases. QA counts denote scored question–answer pairs. Static writes count evidence-packet presentations across per-question memory resets, not unique source contexts. Source text and its attention KV cache are removed before answering.

**LoCoMo (LCM).** We evaluate 1,527 non-adversarial questions from ten conversations in LoCoMo (Maharana et al., 2024), covering single-hop, multi-hop, temporal, and open-domain QA. For each question, sessions containing annotated evidence are written chronologically, yielding 4,011 writes in total. Memory resets between questions; thus, static evaluation may involve multiple evidence-session writes for one question.

**LongMemEval (LME).** We use the 500-question LongMemEval-S set (Wu et al., 2025). In static evaluation, each question's gold evidence is merged into one packet and written once. This differs from processing the full native session history, whose number of writes varies across questions.

**PersonalMem (PM).** Following the data split used in NextMem (Zhang et al., 2026d), we group 360 MPR examples (Zhang et al., 2025) and 284 PerLTQA examples (Du et al., 2024) into PersonalMem, a dedicated category for reasoning over personal facts and memories. Each example's supplied references form one static write.

**NextMem (NM).** We use NextMem's processed contextual-generation test subsets (Zhang et al., 2026d), comprising 1,697 QA pairs: 692 from SQuAD, 446 from HotpotQA, 350 from LongMemEval, and 209 from LoCoMo. Each example's references form one static write. These processed subsets differ from our standalone LCM and LME settings; the evaluation suites are therefore not disjoint in their source datasets.

Table 6: **Statistics of the evaluated subsets.** Continual QA counts refer to the questions scored at each endpoint.

| | Static evaluation | | Continual QA pairs | | |
|---|---|---|---|---|---|
| Dataset | QA pairs | Writes | 2 writes | 4 writes | 8 writes |
| LoCoMo | 1,527 | 4,011 | 134 | 268 | 536 |
| LongMemEval | 500 | 500 | 496 | 496 | 496 |
| NextMem | 1,697 | 1,697 | 1,664 | 1,664 | 1,664 |
| PersonalMem | 644 | 644 | 416 | 416 | 416 |

**Continual evaluation.** For LongMemEval, PersonalMem, and NextMem, we deduplicate records by exact evidence text, retain a fixed order, and discard trailing records that cannot form a complete group of eight. Each eight-record group is partitioned into four two-write trajectories, two four-write trajectories, or one eight-write trajectory. Memory resets between trajectories, and each write presents one complete evidence packet. We score all questions associated with the records seen at each trajectory's final endpoint. Thus, each retained QA pair is scored once per write-depth condition, keeping the evaluated records and their order fixed while varying the reset interval.

LoCoMo uses 67 chronological eight-write trajectories from ten conversations, evaluated at their 2-, 4-, and 8-write prefixes, with 134, 268, and 536 QA pairs, respectively. Unlike the other suites, its evaluated question set grows with write depth; differences in endpoint F1 therefore do not isolate forgetting on a fixed question set.

**Scoring and baselines.** We generate answers using greedy decoding with thinking disabled. For answer F1, we extract the first non-empty answer line after removing answer prefixes and compute token-level F1 against the reference answer. Predictions and references are normalized by lowercasing, removing punctuation and the words *a*, *an*, *the*, and *and*, and standardizing whitespace. Overall denotes the unweighted mean of the scores across the four benchmarks. For LLM-based evaluation, each answer is scored three times by GPT-4.1-mini at temperature zero, and we retain the median score. In static memory construction, $S_0$-Tuning and Temp-LoRA each perform 20 gradient-based optimization steps per memory write using next-token prediction on the provided context. $S_0$-Tuning optimizes the initial states of the backbone's recurrent layers, whereas Temp-LoRA updates the low-rank adapter in its linear projections. The pretrained backbone weights remain frozen in both.

# C Supplementary Experimental Results

## C.1 Component ablations

Table 7 reports component ablations on all four benchmarks under both static and continual memory writing settings. Each ablated variant is retrained under its corresponding configuration and evaluated using the same protocol as the full model. The transformation ablation removes both the learned state carrier and the associated deployment-time stabilization.

Table 7: Complete 4B component ablations. Answer F1 (%).

| Variant | LoCoMo | LongMemEval | NextMem | PersonalMem |
|---|---|---|---|---|
| *Static* | | | | |
| Raw state | 7.55 | 9.82 | 22.36 | 12.18 |
| Extractor only | 12.24 | 12.62 | 21.56 | 12.64 |
| Carrier only | 20.78 | 17.01 | 34.88 | **18.61** |
| STAMP | **25.40** | **25.03** | **40.49** | 17.01 |
| *Continual* | | | | |
| Raw state | 8.87 | 7.42 | 17.09 | 13.01 |
| Extractor only | 10.31 | 9.76 | 18.51 | 11.84 |
| Carrier only | 12.79 | 14.81 | 14.93 | 8.41 |
| STAMP | **15.87** | **22.01** | **21.15** | **14.07** |

### C.2 NORM-PRESERVING STATE CARRY

**Default inference setting.** We use spherical linear interpolation (SLERP) as the default inference-time carry rule, initializing the persistent state with the first carrier output, $\mathbf{P}_1 = \mathbf{C}_1$. For subsequent writes, let $\mathbf{u}$ and $\mathbf{v}$ denote the unit-normalized directions of the previous state $\mathbf{P}_{\tau-1}$ and the carrier state $\mathbf{C}_\tau$, respectively. Let $r = \|\mathbf{P}_{\tau-1}\|_{\mathcal{R}}$ be the previous-state norm, and let $\theta = \arccos\big(\mathrm{clip}\big(\langle \mathbf{u}, \mathbf{v}\rangle_{\mathcal{R}}, -1, 1\big)\big)$ be the angle between the two directions. The update is

$$\mathbf{P}_\tau = r\left[\frac{\sin((1-\lambda)\theta)}{\sin\theta}\mathbf{u} + \frac{\sin(\lambda\theta)}{\sin\theta}\mathbf{v}\right], \qquad \lambda = 0.75. \tag{15}$$

All norms and inner products are computed jointly over the recurrent-state tensors. This update moves toward the carrier direction while preserving the previous-state norm and limiting cumulative scale drift. For nearly aligned directions, we use normalized linear interpolation; if normalization or interpolation is undefined, we retain the previous state.

All variants use the same first-write initialization and denote the carrier state at write $\tau$ by $\mathbf{C}_\tau$.

**SLERP** varies $\lambda \in \{0.25, 0.5, 0.75, 1\}$ while keeping the norm-preserving update fixed.

**Radial clipping** retains the carrier direction and clips its norm relative to the previous state:

$$\mathbf{P}_\tau = \min\left(1, \frac{\rho r}{\|\mathbf{C}_\tau\|_{\mathcal{R}}}\right)\mathbf{C}_\tau, \qquad \rho \in \{1.0, 1.1, 1.2\}. \tag{16}$$

Unlike SLERP, radial clipping permits norm contraction.

**Polar clipping** clips both the carrier norm and its angular movement:

$$r' = \min\big(\|\mathbf{C}_\tau\|_{\mathcal{R}}, \rho r\big), \qquad \lambda' = \min\left(1, \frac{\theta_{\max}}{\theta}\right), \qquad \mathbf{P}_\tau = r'\,\mathrm{slerp}(\mathbf{u}, \mathbf{v}; \lambda'). \tag{17}$$

We set $\rho = 1.1$ and evaluate $\theta_{\max} \in \{15^\circ, 20^\circ, 25^\circ, 30^\circ\}$.

**Unconstrained replacement** directly sets $\mathbf{P}_\tau = \mathbf{C}_\tau$ without clipping or geometric constraints.

### C.3 COMPRESSIBILITY OF THE PERSISTENT STATE

We test whether the final persistent state can be stored more compactly without retraining. After the last static write, we apply truncated SVD independently to each layer's and head's $128 \times 128$ state matrix, storing two FP32 factors with the singular values absorbed into the first factor. The compression rank is distinct from the learned carrier rank. Every condition uses the same original write trajectory, and compressed states are reconstructed only for answering.

Table 8: Post-write state compression on all 1,527 static LoCoMo questions. Storage counts FP32 state tensors or their two factors. All conditions share the same uncompressed write trajectory.

| Compression rank | Storage (MiB) | Reduction | F1 (%) | Δ F1 |
|---|---|---|---|---|
| Dense | 48 | 1× | 25.40 | 0.00 |
| 4 | 3 | 16× | 25.48 | +0.07 |
| 8 | 6 | 8× | 25.24 | −0.17 |
| 16 | 12 | 4× | 25.47 | +0.06 |
| 32 | 24 | 2× | 25.58 | +0.17 |

As shown in Table 8, on all 1,527 LoCoMo questions, rank 4 reduces persistent tensor storage from 48 to 3 MiB, a 16× reduction, while observed F1 changes from 25.40 to 25.48. All four ranks remain within 0.2 points of the dense reference. These results support substantial redundancy in the stored state, rather than an accuracy improvement from compression. Storage figures exclude model weights and serialization headers. The measured LoCoMo overhead is 2.79 seconds for a full SVD per trajectory and 0.20 milliseconds for rank-4 reconstruction.

## D SPECTRAL CONCENTRATION IN CARRIER STATES

The following figures are qualitative diagnostics. We examine recurrent-state heads in 4B and 27B checkpoints. For each head matrix with descending singular values $\sigma_i$, let $p_i = \sigma_i^2 / \sum_j \sigma_j^2$. We report energy effective rank $r_{\text{eff}} = \exp(-\sum_i p_i \log p_i)$ and top-mode energy $100p_1$ (%). These describe activation spectra, while the carrier's parameter rank is a separate quantity.




Figure 6: Carrier-output spectra before SLERP on ten matched LoCoMo contexts from earlier 4B and 27B checkpoints. Bars show means and dots show individual contexts.

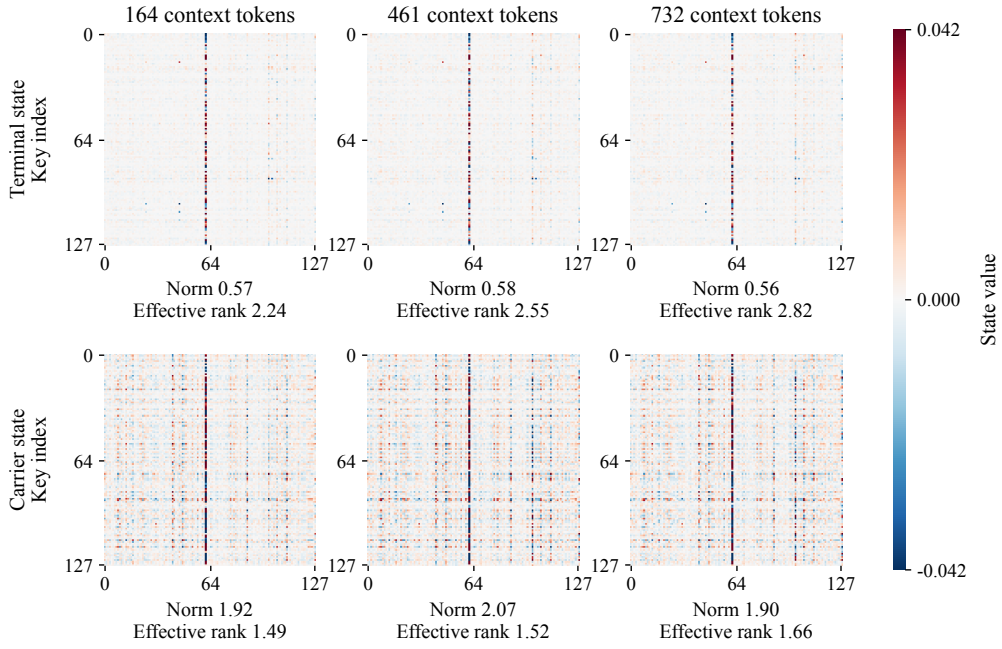



Figure 7: Terminal states and carrier outputs for three prefixes of one LoCoMo session, each encoded from zero (earlier 4B, layer 16, head 9). All panels share a color scale.

Figure 6 shows higher effective rank and lower top-mode energy for the earlier 27B checkpoint than 4B. In Figure 7, the carrier increases norm and reduces effective rank at each prefix. Effective rank grows with prefix length.