KEEP IT IN MIND: IMPLICIT MEMORY VIA LATENT-SPACE KNOWLEDGE AUGMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Successful trajectories provide language-model agents with reusable experience, but retaining and applying this experience across tasks remains challenging. External memory requires repeated retrieval and interpretation, while trajectory-based training embeds experience in model parameters, tying the incorporation of new experience to further model training. We introduce *Knowledge-Augmenting Implicit Memory* (KIM), which uses *latent-space knowledge augmentation* to retain learned behavioral adjustments in a memory separate from the backbone. Building on a fixed shared adapter, KIM learns local adjustments that favor successful continuations and jointly fits a memory field over their associated interaction states. This field integrates adjustments across trajectories to guide behavior in new states. During a new task, KIM reads a state-conditioned adjustment from the field and applies it through a low-rank intervention in the frozen backbone. Tool observations refresh the readout, guiding successive decisions without replaying source trajectories. With a memory field built from 3.6k successful trajectories selected by similarity to evaluation queries, KIM achieves 41.6% average accuracy across seven computational and knowledge-intensive reasoning benchmarks, compared with 30.8% for trajectory RAG using a 54k-trajectory retrieval bank. Applied to Tool-Star, KIM further raises average accuracy from 46.6% to 50.7%, extending its benefits to a model trained for multi-tool reasoning. Our code is available here.

1 INTRODUCTION

Language-model agents accumulate interaction experience as they complete tasks over time. A successful task leaves more than an answer: it records how to find evidence, decompose a problem, use tools, and correct unproductive steps. The value of this experience extends beyond solving the original task again to helping the agent address problems it has not encountered before (Zhao et al., 2024; Wang et al., 2025b). How successful experience can yield lasting improvements on subsequent tasks is therefore a central question for long-term agent memory.

Existing methods primarily use external memory or trajectory-based training to exploit this experience. External memory stores past trajectories or distilled insights as text for retrieval on new tasks (Zhao et al., 2024; Wang et al., 2025b; Ouyang et al., 2026; Ferraz et al., 2026). Records are easy to revise, but each use requires the model to reread the selected text and determine how to apply it. Trajectory-based training uses supervised learning or reinforcement learning to absorb problem-solving and tool-use patterns into model parameters (Jin et al., 2025; Dong et al., 2026). These patterns become available without rereading source trajectories, but incorporating additional experience requires further model training. Recent latent memories offer alternatives to textual replay through generated memory tokens or retrievable per-problem soft prompts (Zhang et al., 2026; Dorovatas et al., 2026). We ask how a memory separate from the backbone can jointly organize experience from multiple trajectories and guide behavior as the interaction state changes.

Our insight is to retain what successful continuations teach about seeking evidence and choosing actions as state-dependent behavioral adjustments in a memory separate from the backbone (Figure 1).

We introduce *Knowledge-Augmenting Implicit Memory* (KIM). Building on a shared adapter trained on historical trajectories, KIM learns local adjustments and jointly fits a memory field over their associated states. The field supplies a low-rank intervention for the current state and refreshes it

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.

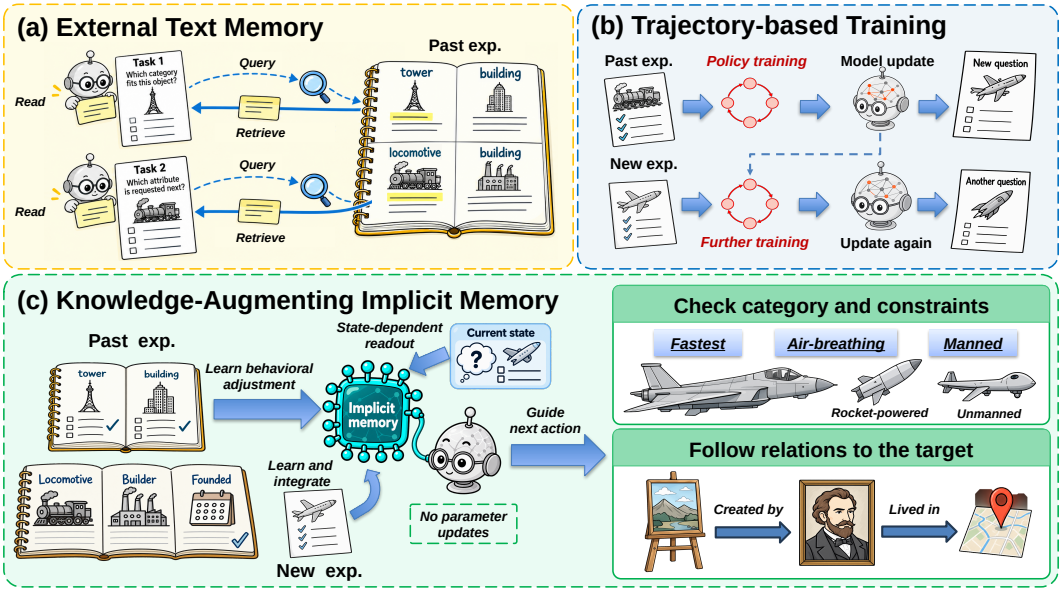

Figure 1: **Three ways to reuse experience.** External memory retrieves text; trajectory-based training updates model parameters. KIM reads behavioral adjustments from a separate memory according to the current state, keeping the backbone fixed.

after tool observations. Additional experience enters through local writes and field updates while the backbone and shared adapter remain fixed.

With a memory field built from 3.6k successful trajectories selected by similarity to evaluation queries, KIM achieves 41.6% average accuracy across seven computational and knowledge-intensive reasoning benchmarks, compared with 30.8% for trajectory RAG using a 54k-trajectory retrieval bank. Applied to Tool-Star, KIM raises average accuracy from 46.6% to 50.7% (Table 1).

Our main contributions are as follows:

- We formulate experience reuse as latent-space knowledge augmentation: a separate memory supplies behavioral adjustments without replaying source trajectories.
- We develop KIM's local-learning and joint-fitting mechanism to combine multiple trajectories into a memory field with state-dependent readouts.
- Across seven reasoning benchmarks, KIM outperforms trajectory RAG with a smaller query-selected bank and improves both instruction-tuned and tool-trained policies.

2 Related Work

External experience memory. Text-based external memory stores interaction experience as retrievable records for subsequent tasks. Early approaches use observations, reflections, and verbal feedback to support decisions (Park et al., 2023; Shinn et al., 2023). Later work organizes trajectories into reusable insights, workflows, or reasoning strategies, as in ExpeL, Agent Workflow Memory, and ReasoningBank (Zhao et al., 2024; Wang et al., 2025b; Ouyang et al., 2026). Other studies address when experience applies and how to continually refine it through feedback (Fu et al., 2024; Cao et al., 2026). Despite their different forms of experience organization, these methods primarily influence model decisions through text in the context. KIM instead converts experience into parameter-based behavioral adjustments without replaying source trajectories in the inference context.

Learning policies from trajectories. Trajectory-based training incorporates interaction experience into policy parameters through supervised or reinforcement learning. AgentTuning and Agent-FLAN use agent trajectories to improve interaction capabilities (Zeng et al., 2024; Chen et al., 2024), while Search-R1 and Tool-Star use reinforcement learning to develop search and multi-tool reasoning,

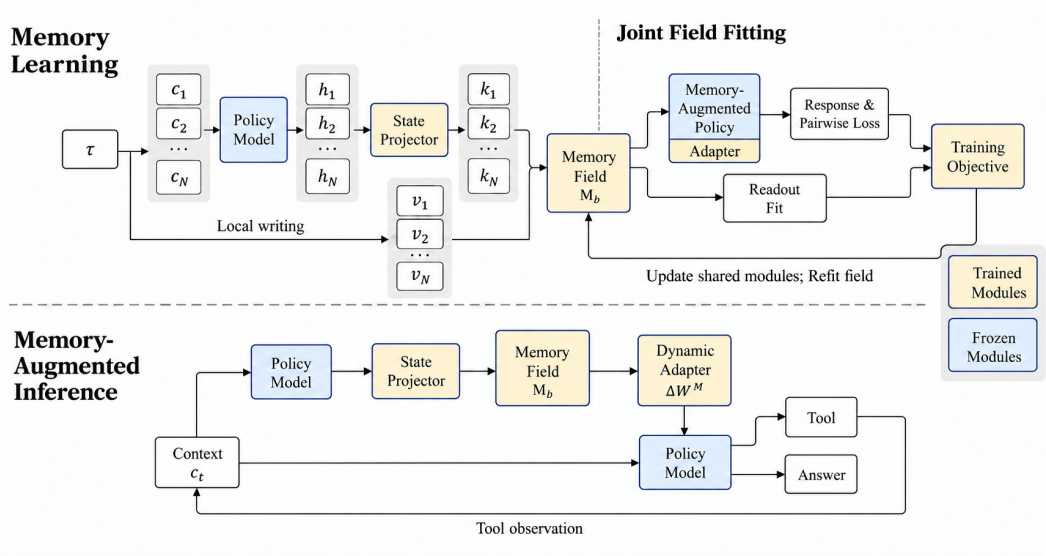

Figure 2: **KIM contextual experience memory.** Top: field learning from local writes and joint fitting. Bottom: state-dependent adjustment at inference. Policy Models denote the fixed reference policy. During contextual training, yellow modules are learned and blue modules frozen; all stored parameters remain fixed at inference.

respectively (Jin et al., 2025; Dong et al., 2026). These methods can learn transferable behavior, but incorporating additional experience requires further policy optimization. Building on a trained policy, KIM confines learning from new experience to a separate memory: local writes and memory-field updates incorporate experience while the backbone and shared adapter remain fixed. The distinction concerns what is optimized, not whether optimization is needed.

Latent memory. Latent memory stores and reuses information through continuous representations rather than text. MemoryLLM and M+ explore updatable latent memory and mechanisms for its long-term storage (Wang et al., 2024; 2025a). For experience reuse, MemGen generates continuous memory tokens conditioned on the current reasoning state and inserts them into the context to guide subsequent generation (Zhang et al., 2026). KIM likewise accesses experience according to the interaction state, but reads behavioral adjustments applied to the model rather than memory tokens added to the context. It jointly fits adjustments learned across trajectories into a memory field and influences current decisions through low-rank parameter interventions.

3 METHODOLOGY

3.1 PROBLEM FORMULATION AND OVERVIEW

At model turn t, an agent's context c_t contains the task query q, previous model outputs, and tool observations. Its output a_t contains reasoning followed by a tool call or final answer; a tool call returns an observation o_t. Given successful trajectories $\mathcal{M} = \{\tau_j\}_{j=1}^{n}$, Knowledge-Augmenting Implicit Memory (KIM) jointly compiles learned local behavioral adjustments into a state-conditioned memory field for reuse without replaying source trajectories (Figure 2).

In the adapter-based configuration, *general tool-use memory* is a shared adapter ΔW^{P}, while *contextual experience memory* supplies $\Delta W^{\mathrm{M}}(u_t)$. The readout $u_t \in \mathbb{R}^K$ combines K shared adjustment operators. For a selected linear projection ℓ, let W_ℓ be its frozen backbone weight and $W_{\ell,t}$ its effective weight at turn t:

$$W_{\ell,t} = W_\ell + \Delta W_\ell^{\mathrm{P}} + \Delta W_\ell^{\mathrm{M}}(u_t). \tag{1}$$

The backbone stays frozen throughout; inference changes only the readout, not stored parameters.

3.2 ESTABLISHING A REFERENCE POLICY

The reference policy π_0 incorporates general tool-use memory. For instruction-tuned backbones, we train a shared low-rank adapter using response-token cross-entropy on complete historical turns, including reasoning, tool calls, and final answers, then freeze it. KIM+Tool-Star instead uses the existing tool-trained policy as π_0, with no new adapter. In both cases, $u_t = 0$ recovers π_0.

3.3 WRITING CONTEXTUAL EXPERIENCE

From trajectories selected for contextual writing, we extract samples (c_i, y_i, b_i) at two decision boundaries: before the first model turn and after a tool observation. Here i indexes states, c_i is the context, y_i the demonstrated next turn, and b_i the boundary type (initial or post-observation). A trajectory can supply multiple samples. Each local vector $v_i \in \mathbb{R}^K$ adjusts π_0 to favor y_i; it represents a behavioral change, not the context. For field compilation, π_0 separately encodes c_i as its last-token hidden representation $h_i \in \mathbb{R}^{d_h}$, where d_h is the hidden width.

Shared low-rank factors (Hu et al., 2022) place local adjustments in a common coordinate system for field fitting. The combination vector u is a local write v_i during writing or a field readout u_t at inference. It adjusts projection ℓ as

$$\Delta W_\ell^{\mathrm{M}}(u) = \frac{\alpha}{r}\left(\sum_{a=1}^{K} \tanh(u_a) B_{\ell,a}\right) A_\ell, \tag{2}$$

where $A_\ell \in \mathbb{R}^{r \times d_{\mathrm{in},\ell}}$ and $B_{\ell,a} \in \mathbb{R}^{d_{\mathrm{out},\ell} \times r}$ are shared across source states; $d_{\mathrm{in},\ell}$ and $d_{\mathrm{out},\ell}$ are the projection's input and output dimensions; α scales the adjustment. The same vector supplies bounded, signed coefficients across projections. Sharing A_ℓ keeps both individual operators and their combination at rank at most r, not Kr. Trajectory count controls source experience, whereas K controls adjustment directions without fixed semantic labels. Our main Qwen model uses $K = 16$ and $r = 4$.

Let $\ell_i(u)$ be the mean response-token negative log likelihood of y_i given c_i under adjustment u; $\ell_i(0)$ is the fixed loss under π_0. Local vectors and shared factors minimize

$$\mathcal{L}_{\mathrm{write}}^{(i)} = \ell_i(v_i) + \lambda_{\mathrm{b}} \log(1 + \exp[m + \ell_i(v_i) - \ell_i(0)]) + \frac{\lambda_u}{K}\|v_i\|_2^2. \tag{3}$$

Here λ_{b} weights a soft margin m favoring improvement over $\ell_i(0)$; λ_u penalizes vector magnitude.

3.4 COMPILING A SHARED MEMORY FIELD

To supply adjustments at new states without demonstrated continuations, we fit a state-to-adjustment map to the learned pairs (h_i, v_i).

For each boundary type b, a learned layer normalization and linear projector transform h_i into an L_2-normalized key $k_i \in \mathbb{R}^{d_k}$, where d_k is the key dimension. Let $\mathcal{I}_b = \{i : b_i = b\}$ collect states of type b, and let $\tau(i)$ denote the trajectory containing state i, with $N_{\tau(i)}$ writable states. We set $w_i = 1/N_{\tau(i)}$, giving each trajectory unit total weight across its states. With ridge coefficient $\lambda > 0$, weighted ridge regression fits the field (Hoerl & Kennard, 1970):

$$M_b = \arg\min_{M \in \mathbb{R}^{K \times d_k}} \left\{ \sum_{i \in \mathcal{I}_b} w_i \|v_i - M k_i\|_2^2 + \lambda \|M\|_F^2 \right\}, \tag{4}$$

For fixed keys and local vectors, the closed-form solution is $M_b = C_b(G_b + \lambda I_{d_k})^{-1}$, where $G_b = \sum_{i \in \mathcal{I}_b} w_i k_i k_i^\top$ and $C_b = \sum_{i \in \mathcal{I}_b} w_i v_i k_i^\top$. Here I_{d_k} is the identity matrix; G_b and C_b summarize key second moments and key–adjustment relationships, respectively.

For a new key k_t of type b, the unshrunk readout is

$$\hat{u}_t = M_b k_t = \sum_{i \in \mathcal{I}_b} \underbrace{w_i k_i^\top (G_b + \lambda I_{d_k})^{-1} k_t}_{\beta_i(k_t)} v_i. \tag{5}$$

Unlike independent similarity weights, $\beta_i(k_t)$ depends on all stored state directions through G_b, can be signed, and need not sum to one. Deployment computes $M_b k_t$ without scanning trajectories.

To suppress small adjustments, we apply block soft-thresholding, the proximal operator of the Euclidean norm (Parikh & Boyd, 2014, Section 6.5.1), with a learned threshold $\tau_b \geq 0$:

$$u_t = \begin{cases} 0, & \|\hat{u}_t\|_2 \leq \tau_b, \\ \left(1 - \dfrac{\tau_b}{\|\hat{u}_t\|_2}\right) \hat{u}_t, & \|\hat{u}_t\|_2 > \tau_b. \end{cases} \tag{6}$$

Before $\tanh$ in Eq. 2, this either recovers π_0 or shrinks the readout. The threshold controls intervention magnitude; it does not predict answer correctness.

Joint fitting freezes v_i and π_0 while optimizing projectors, readout thresholds, and low-rank factors by gradient descent. For the thresholded readout u_i at training state i, it combines demonstrated-response likelihood and relative gain from Eq. 3, squared agreement between u_i and v_i, and an offline preference loss on successful and unsuccessful continuations from the same context. The latter adapts DPO (Rafailov et al., 2023) using response-length-normalized log-likelihood ratios relative to π_0 (Appendix B). We periodically refit M_b as keys change. Between refits, M_b stays constant: gradients flow through the readout, not the ridge solve.

3.5 Inference and contextual memory updates

At each turn, KIM encodes the context under π_0 and holds the field readout fixed during generation through Eq. 2. After a tool observation, KIM removes the adjustment before re-encoding and refreshing the readout, until a final answer or the interaction limit. Field multiplication alone costs $O(Kd_k)$, independent of trajectory count for fixed memory dimensions; context encoding and response generation still require policy computation.

Between tasks, KIM learns from new experience offline, in the spirit of sleep-time computation (Lin et al., 2025) and offline memory consolidation (Ye et al., 2026): we learn local vectors for accumulated successful trajectories, add their contributions to the retained sufficient statistics G_b and C_b, and recompute M_b. The backbone, general memory, projectors, thresholds, factors, prior local vectors, and write weights stay fixed. Consolidation updates the stored field, unlike online readout refreshes.

4 Experiments

4.1 Experimental setup

Tasks and datasets. We evaluate KIM on seven reasoning benchmarks: GSM8K (Cobbe et al., 2021), MATH500 (Hendrycks et al., 2021; Lightman et al., 2024), and AIME24 (Mathematical Association of America, 2024) for mathematical reasoning, and HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023) for knowledge-intensive reasoning. All methods use the same 1,155 evaluation questions. Appendix A details the evaluation splits and experience pools.

Baselines. We compare unmodified backbones, textual memory methods, and task-trained policies. Textual baselines include trajectory RAG, ExpeL (Zhao et al., 2024), AWM (Wang et al., 2025b), and ReasoningBank (Ouyang et al., 2026). Each uses Qwen2.5-3B-Instruct, a 54k-trajectory source bank, and a 3,072-token memory budget within a common tool-interaction framework. Task-trained comparisons include Search-R1 (Jin et al., 2025), Tool-Star (Dong et al., 2026), and MemGen (Zhang et al., 2026); their interfaces and training configurations appear in Appendix A.1. Except for the Llama rows and MemGen's Qwen2.5-1.5B GSM8K-SFT host, comparisons use the Qwen2.5-3B family. The unmodified backbones can call tools but load no additional adapter or memory.

Implementation and interaction. KIM (Qwen) uses frozen Qwen2.5-3B-Instruct (Yang et al., 2024), general tool-use memory trained on approximately 54k trajectories, and a contextual field with 16 rank-4 adjustment operators. For each of the 1,155 evaluation questions, we retrieve the top four semantically similar historical trajectories; deduplication and correctness filtering yield 3,609 trajectories for contextual writing (Appendix A). Contextual training also samples

Table 1: Accuracy (%) on seven benchmarks and their average. Parentheses identify KIM's host; MemGen uses 1.5B. Column bests are bold.

Method	Computational Reasoning			Knowledge-Intensive Reasoning				Avg.
	GSM8K	MATH500	AIME24	Hotpot	2Wiki	MuSiQ.	Bamb.	
Qwen2.5-3B-Instruct	39.5	42.0	3.3	17.0	13.0	6.5	13.6	19.3
Llama-3.2-3B-Instruct	37.5	27.0	0.0	20.0	12.5	7.0	18.4	17.5
Frozen-Backbone Methods								
RAG	72.5	49.5	6.7	32.5	20.0	8.5	25.6	30.8
ExpeL	58.5	48.0	6.7	19.5	9.5	6.0	21.6	24.3
AWM	29.5	28.5	0.0	16.0	14.5	5.0	6.4	14.3
ReasoningBank	33.5	33.0	3.3	11.5	3.5	2.0	14.4	14.5
Backbone-Trained Methods								
Search-R1	66.5	51.5	3.3	39.5	32.0	12.5	24.0	32.8
MemGen	58.5	31.0	0.0	17.0	20.5	4.0	18.4	21.3
Tool-Star	**81.5**	73.5	16.7	50.5	43.5	20.0	40.8	46.6
KIM Family (Ours)								
KIM (Qwen)	77.0	62.0	16.7	43.5	38.5	18.0	35.2	41.6
KIM (Llama)	76.0	56.5	6.7	38.5	28.5	12.0	38.4	36.7
KIM (Tool-Star)	80.0	**76.0**	**20.0**	**51.5**	**53.0**	**25.5**	**48.8**	**50.7**

response supervision from the larger historical pool (Appendix B). KIM (Llama) uses Llama-3.2-3B-Instruct (Meta, 2024) and trains both memories on 3,609 trajectories; KIM (Tool-Star) uses its existing policy as the reference. KIM and the textual memory baselines allow four tool calls, 16,384 new tokens per round, and a 32,768-token context. Mathematical tasks use Python; knowledge-intensive tasks also allow retrieval. Appendix A.1 gives prompts and interface details.

Evaluation metrics. Qwen3-30B-A3B (Yang et al., 2025) judges answers under a common reference-comparison protocol, with numeric-equivalence cross-checks for closed-form mathematical answers. We report dataset-level accuracy and its unweighted mean (macro-average). Avg. covers all seven datasets; Math and QA average the three mathematical and four knowledge-intensive datasets, respectively. Hyphens denote unreported results.

4.2 Main results

Comparison with textual memory. Under the evaluated tool-use setting, KIM (Qwen) achieves 41.6% average accuracy, compared with 19.3% for the unmodified backbone and 30.8% for trajectory RAG, the strongest textual memory baseline (Table 1). KIM improves over RAG on all seven benchmarks, with average gains of 9.0 percentage points in Math and 12.2 in QA. KIM's contextual field uses 3.6k query-selected trajectories, whereas RAG retrieves from a 54k-trajectory bank. This shows effective use of a smaller, evaluation-query-matched bank, not less total training data.

Complementing a tool-trained policy. With Tool-Star's original policy fixed and no additional shared adapter, KIM raises average accuracy from 46.6% to 50.7%. Six of seven benchmarks improve, with an average gain of 6.0 percentage points in QA and 1.4 in Math; GSM8K decreases by 1.5 points. This aggregate gain supports augmenting a tool-trained policy with contextual experience.

Application to another backbone. Independently training both memories on Llama-3.2-3B-Instruct yields 36.7% average accuracy. Its general-memory training pool differs in size from KIM (Qwen), so this cross-backbone application does not isolate a backbone effect.

4.3 Memory components

Table 2 compares the two memory components and alternative readouts with main-table Full KIM on the same 1,155 questions (Appendix C).

Table 2: Component comparison. Accuracy (%) is averaged over four knowledge, three math, or all seven datasets. Red parentheses show drops from Full KIM before rounding; paired 95% CIs measure Full KIM's overall gain.

Variant	Knowledge	Math	Overall	95% CI
Full KIM	**33.80**	**51.89**	**41.55**	-
w/o General Tool-use Memory	15.55 (-18.25)	14.67 (-37.22)	15.17 (-26.38)	[23.18, 29.78]
w/o Contextual Experience Memory	20.50 (-13.30)	45.89 (-6.00)	31.38 (-10.17)	[7.08, 13.36]
w/o Joint Memory Field	26.20 (-7.60)	45.94 (-5.94)	34.66 (-6.89)	[3.98, 9.90]
w/o State Refresh	23.83 (-9.98)	46.56 (-5.33)	33.57 (-7.99)	[4.85, 11.20]

Variants. Removing general memory relearns contextual memory under the bare backbone; removing contextual memory sets $u_t = 0$. The remaining variants replace the joint field with four-neighbor local retrieval or hold the initial readout fixed. Tool observations remain in the generation context.

Roles of the two memories. Neither memory component alone matches their combination. Removing general memory produces the largest overall drop; removing contextual memory hurts QA more than Math, indicating its particular value for knowledge-intensive reasoning.

Reading and refreshing experience. Local retrieval and a fixed initial readout outperform no contextual memory overall but trail Full KIM, especially in QA. Full KIM's gains over all four variants have positive paired 95% CIs.

4.4 Training Dynamics

General-memory response loss falls early and then fluctuates around a plateau (Figure 3). During field fitting, validation preference loss decreases from 0.721 to 0.405 over 800 steps despite intermediate fluctuations, while response loss stays near 0.41. The preference objective thus captures a change largely absent from response loss. The plotted component is distinct from the full training and checkpoint-selection objectives; its decrease alone establishes neither downstream accuracy gains nor convergence. Validation and checkpoint-selection details appear in Appendix B.

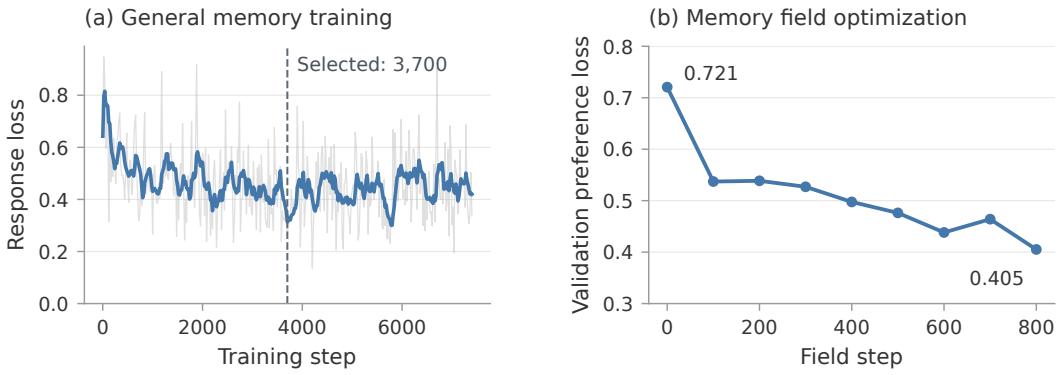

Figure 3: KIM optimization with eight operators. (a) General-memory response loss: raw batches (gray), seven-record trailing mean (blue), and step 3,700 selected by development-set generation performance (dashed). (b) Unsmoothed validation preference loss during field fitting.

4.5 Tool Interaction and Efficiency

We examine feedback use and interaction completion on all 1,155 questions under identical Qwen/KIM prompts and tool budgets (Appendix C.3). Feedback-use accuracy counts validly labeled continuations that apply relevant results, change strategy after inadequate evidence, or repair errors. Repetition or a correct final answer alone does not qualify. Completion counts questions with a nonempty, closed answer block and a completed tool loop, regardless of correctness.

KIM improves both measures over Qwen and RAG in Math and QA (Figure 4). Feedback-use gains over RAG are 20.1 and 18.3 percentage points, respectively. Completion improves over Qwen on all seven datasets, from 56.28% to 92.94% in Math and from 20.68% to 81.90% in QA. These metrics measure feedback use and interaction completion, not final-answer accuracy.

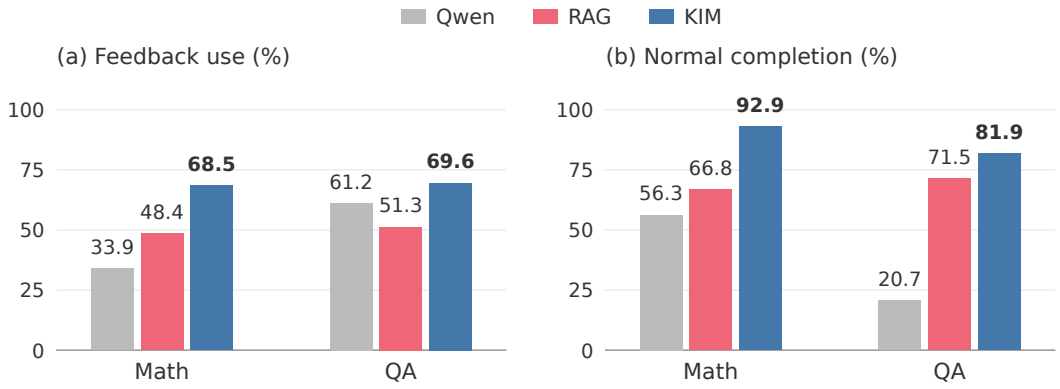

Figure 4: Tool interaction on 1,155 questions. Left: correct feedback use among validly labeled continuations. Right: normal completion over all questions. Both panels equally weight datasets within Math and QA; higher is better. Qwen denotes the unmodified Qwen2.5-3B-Instruct model.

Runtime and workload. KIM makes more tool calls but generates fewer tokens than either comparator, with runtime about 31% below RAG and near the backbone (Table 3). Timing excludes initial memory lookup and prompt assembly; encoding, readout, and decoding are timed together.

Table 3: Per-question generation-and-tool runtime and workload (128 questions, A100-80GB; Appendix D).

Method	Runtime (s) ↓	Generated tokens	Tool calls
Qwen2.5-3B-Instruct	17.16	615.3	1.04
RAG	23.71	855.7	0.99
KIM	16.36	525.9	1.72

Training and construction. Including general-adapter training, KIM (Qwen) costs 5.89 A100 GPU-hours, 3.2% of Tool-Star's 184 A100 GPU-hours. These totals compare the respective training pipelines, not data-matched or equal-accuracy costs (Appendix D).

4.6 Experience scale and memory capacity

Experience retention. Figure 5(a) retains nested subsets of the full 54k historical bank, not the 3.6k query-selected bank. We recompile the full-bank checkpoint's field with learned modules and local vectors fixed. At 0%, only general memory remains. This tests post-training retention, not training-data efficiency or deployment storage: inference uses the compiled field rather than source trajectories.

Average accuracy rises from 31.38% at 0% to 39.94% at 10%, with a larger gain in QA than Math. Across 10–100% retention, it varies by only 0.28 points, with little further gain in either task family.

Operator capacity. Figure 5(b) varies $K \in \{4, 8, 16, 32\}$ at fixed data, training budget, and combined rank at most $r = 4$ (Eq. 2). Math and QA both improve from four through sixteen operators, but diverge at thirty-two: QA improves further while Math declines. Sixteen operators achieve the highest seven-dataset average, 41.55%, and are used in the main table (dataset results in Appendix C.2).

Operator use. Readouts on all 1,155 questions show the sixteen-operator model's usage (Figure 5c). Magnitude-adjusted shares spread across operators, with different Math and QA mixtures. The mean effective operator count is 12.0 out of 16, measuring concentration of usage rather than the number of nonzero operators or their causal effects (Appendix C.2).

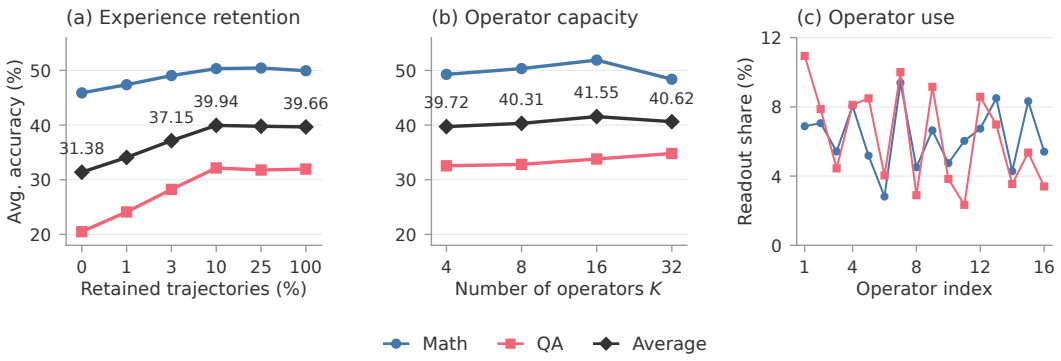

Figure 5: Experience scale and capacity. (a) Retention at eight operators; levels are equally spaced. (b) Operator count at rank $r = 4$. Labels show seven-dataset averages. (c) Relative operator magnitudes from sixteen-operator readouts; each task-group profile sums to 100% (Appendix C.2).

4.7 Periodic self-experience updates

We test periodic updates from the main-table KIM checkpoint using two independent chains with complementary, dataset-stratified experience and evaluation halves. Each chain processes three disjoint source batches: the current KIM generates trajectories, official correctness feedback selects successful ones, and two passes of local writing precede field recompilation. Shared modules and earlier writes remain fixed. Evaluation questions and feedback never enter the corresponding update chain. The held-out halves together recover all 1,155 evaluation questions each round (Appendix C.4).

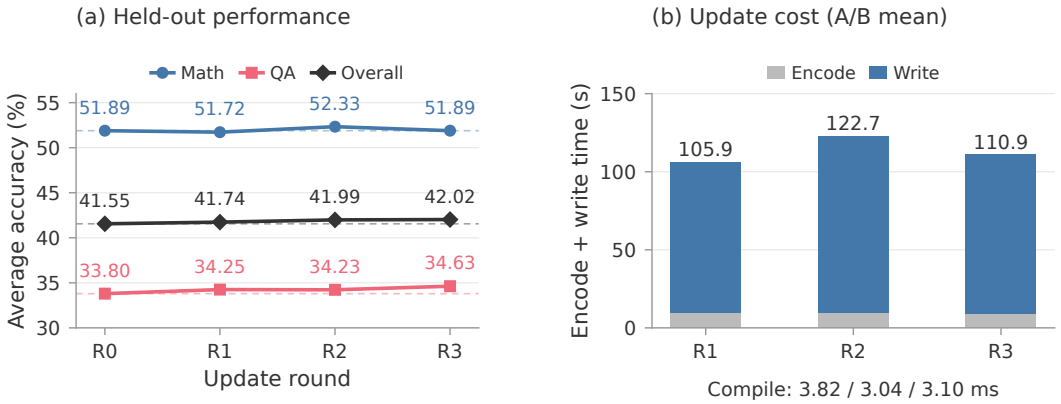

Figure 6: Periodic updates. (a) Absolute accuracy; dashed lines mark R0. (b) Mean A/B cost per chain on A100-80GB: encoding and writing in seconds; compilation below in milliseconds (R1–R3).

Average accuracy rises from 41.55% to 42.02%; QA gains 0.83 points, while Math returns to its initial average (Figure 6). Evaluation questions are excluded from updates but not necessarily from initial training. Averaged over A/B, each round takes 106–123 seconds per chain, mostly for local writing. Timing covers encoding, writing, and recompilation, excluding experience generation, judging, model loading, and checkpoint I/O (Appendix C.4).

5 Conclusion

KIM combines a learned tool-use policy with contextual experience compiled into a shared memory field. Interaction states read low-rank adjustments without replaying source trajectories. With Qwen frozen, KIM answers 527 of 1,155 questions correctly and reaches 41.6% average accuracy, exceeding trajectory RAG and supporting state-dependent experience reuse. Hard reasoning and fine-grained factual access remain challenging.

AI USE STATEMENT

We used generative AI tools to improve the clarity and readability of the manuscript and to assist with literature search and the discovery of related work. We take responsibility for the final content, including the accuracy of the claims and references.

REFERENCES

Zouying Cao, Jiaji Deng, Li Yu, Weikang Zhou, Zhaoyang Liu, Bolin Ding, and Hai Zhao. Remember me, refine me: A dynamic procedural memory framework for experience-driven agent evolution. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 16803–16822, 2026. doi: 10.18653/v1/2026.findings-acl.829. URL https://aclanthology.org/2026.findings-acl.829/.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. Agent-FLAN: Designing data and methods of effective agent tuning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 9354–9366, 2024. doi: 10.18653/v1/2024.findings-acl.557. URL https://aclanthology.org/2024.findings-acl.557/.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. Tool-Star: Empowering multi-tool collaborative web agent via reinforcement learning. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 301–312, 2026. doi: 10.1145/3805712.3809712. URL https://doi.org/10.1145/3805712.3809712.

Vaggelis Dorovatas, Nancy Kalaj, and Rahaf Aljundi. Continual self-improvement with lightweight experiential latent memories. *arXiv preprint arXiv:2606.17803*, 2026. URL https://arxiv.org/abs/2606.17803.

Thomas Palmeira Ferraz, Romain Deffayet, Vassilina Nikoulina, Hervé Déjean, and Stéphane Clinchant. Retrieval-augmented LLM agents: Learning to learn from experience. *arXiv preprint arXiv:2603.18272*, 2026. URL https://arxiv.org/abs/2603.18272.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. AutoGuide: Automated generation and selection of context-aware guidelines for large language model agents. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d8efbb5dd415974eb095c3f06bff1f48-Abstract-Conference.html.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. doi: 10.1080/00401706.1970.10488634. URL https://doi.org/10.1080/00401706.1970.10488634.

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. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025. URL https://arxiv.org/abs/2503.09516.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Kevin Lin, Charlie Snell, Yu Wang, Charles Packer, Sarah Wooders, Ion Stoica, and Joseph E. Gonzalez. Sleep-time compute: Beyond inference scaling at test-time. *arXiv preprint arXiv:2504.13171*, 2025. URL https://arxiv.org/abs/2504.13171.

Mathematical Association of America. 2024 american invitational mathematics examination (AIME I and II). Competition problem booklets, 2024. URL https://www.xuemath.org/aime-resources. MAA-authorized archive.

Meta. Llama-3.2-3B-Instruct model card. Hugging Face, 2024. URL https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. 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. URL https://openreview.net/forum?id=jL7fwchScm.

Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1 (3):123–231, 2014. URL https://web.stanford.edu/~boyd/papers/prox_algs.html.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian McAuley. MEMORYLLM: Towards self-updatable large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 50453–50466, 2024. URL https://proceedings.mlr.press/v235/wang24s.html.

Yu Wang, Dmitry Krotov, Yuanzhe Hu, Yifan Gao, Wangchunshu Zhou, Julian McAuley, Dan Gutfreund, Rogerio Feris, and Zexue He. M+: Extending MemoryLLM with scalable long-term memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63308–63323, 2025a. URL https://proceedings.mlr.press/v267/wang25au.html.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911, 2025b. URL https://proceedings.mlr.press/v267/wang25bx.html.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Chongrui Ye, Yuxiang Liu, Yu Wang, Haofei Yu, Yining Zhao, Ge Liu, Julian McAuley, and Jiaxuan You. Auto-Dreamer: Learning offline memory consolidation for language agents. *arXiv preprint arXiv:2605.20616*, 2026. URL https://arxiv.org/abs/2605.20616.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling generalized agent abilities for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3053–3077, 2024. doi: 10.18653/v1/2024.findings-acl.181. URL https://aclanthology.org/2024.findings-acl.181/.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. MemGen: Weaving generative latent memory for self-evolving agents. In *International Conference on Learning Representations*, 2026. URL https://iclr.cc/virtual/2026/poster/10006821.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38 (17):19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

A Data and Evaluation Settings

Evaluation set. All methods use the same 1,155 questions and evaluation order (Table 4). Math and QA average three and four datasets, respectively; the overall score weights all seven equally.

Table 4: Number of evaluation questions per dataset.

GSM8K	MATH500	AIME24	HotpotQA	2Wiki	MuSiQue	Bamboogle
200	200	30	200	200	200	125

Experience pools. For KIM (Qwen), the shared adapter implementing general tool-use memory is trained on approximately 54k trajectories. For contextual writing, each of the 1,155 evaluation questions retrieves the top four historical trajectories by semantic similarity. We pool the retrievals, remove duplicates, and retain correct trajectories, yielding 3,609 trajectories and 8,594 writable states. This bank is selected using the evaluation queries, rather than independently of them. Contextual learning also draws target-response supervision from the broader pool. The pre-filter source bank includes 347 trajectories whose source questions overlap the evaluation set; textual-memory baselines exclude these additional trajectories. Targets must be complete under the 32,768-token context and 4,096-token response budgets. The outcome objective uses 88 successful/unsuccessful continuation pairs with shared prefixes, split into 66 training and 22 validation pairs. Reference answers label outcomes but are not policy inputs.

A.1 Baseline interfaces

Qwen, textual-memory methods, and KIM use a tool loop with at most four calls, 16,384 new tokens per round, and a 32,768-token context. Mathematics uses Python; QA additionally retrieves three search results per call. Textual memory uses at most 3,072 tokens (settings in Table 5).

Table 5: Baseline interfaces. Qwen2.5-3B-Instruct constructs and uses textual memories.

Method	Configuration
RAG	Top four trajectories from the 54k source bank.
ExpeL	10 Math and 7 QA rules, plus four retrieved experiences.
AWM	Four retrieved workflows from 7,161 induced workflows.
ReasoningBank	Lessons associated with four retrieved source experiences, from 97,530 lessons.
Search-R1	Released Qwen2.5-3B v0.3 policy with its native search-only prompt: four actions, three documents per search, 500 tokens per action, temperature 1, and 4,096-token context.
Tool-Star	Released Tool-Star-Qwen-3B policy with its native generation loop and local tool endpoints.
MemGen	Released Qwen2.5-1.5B GSM8K-SFT checkpoint transferred to seven tasks without further training.

KIM (Llama) trains general and contextual memory independently on 3,609 trajectories with Llama-3.2-3B-Instruct. KIM (Tool-Star) uses the existing policy as π_0, with no new shared adapter.

Interaction and answer extraction. The host's chat template renders the system instruction and user question. The tool loop recognizes `<search>`, `<python>`, and `<answer>` blocks; observations are appended in `<result>` blocks. Final exact answers use `\boxed{}` inside `<answer>`.

A.2 Answer evaluation

All answers use the Qwen3-30B-A3B `qwen3_30b_a3b_reference_comparison_v3` judge, with deterministic numeric-equivalence cross-checks for closed-form answers. The user message supplies the question, JSON-serialized reference, and extracted candidate, followed by *Compare only the reference and candidate; do not re-solve the question.* Thinking is disabled. Invalid replies are retried with 128–1,024-token budgets without regenerating candidates. The parser uses the last verdict tag; missing candidates count as incorrect and malformed replies as unresolved.

Answer Judge: System Prompt

You are an impartial answer-correctness judge. The supplied reference answer is authoritative. Do not solve the original problem again. Only compare the candidate answer with the reference answer in the context of the question.

Accept harmless differences in notation, formatting, leading zeros, aliases, or wording when the mathematical value or factual meaning is the same. Reject a different numerical value, entity, relation, or conclusion. Extra explanation is acceptable unless it contradicts the answer.

Give one short comparison sentence, then end with exactly one tag on the final line:
`<verdict>correct</verdict>`
`<verdict>incorrect</verdict>`

Judge validation. We assess judge reliability on 420 KIM responses, comprising 180 mathematics and 240 knowledge QA responses. Two annotators independently label each response as correct or incorrect, then adjudicate disagreements to establish the human reference. Table 6 reports agreement between the independent annotations and agreement between Qwen3-30B-A3B and the adjudicated labels. Cohen's κ also compares the judge with the adjudicated labels.

Table 6: Judge validation on KIM responses. Overall pools all 420 responses.

		Agreement (%)			Error rate (%)	
Group	N	Human–Human	Judge–Human	Cohen's κ	FPR	FNR
Math	180	97.78	98.33	0.967	1.18	2.11
Knowledge QA	240	96.25	95.42	0.908	4.62	4.55
Overall	420	96.90	96.67	0.933	3.26	3.41

FPR is the fraction of human-labeled incorrect responses that the judge accepts; FNR is the fraction of human-labeled correct responses that it rejects. Both use the adjudicated human labels.

The judge agrees with the human reference on 96.67% of responses ($\kappa = 0.933$), with higher agreement on mathematics than knowledge QA. The similar overall false-positive and false-negative rates indicate balanced error directions in this sample. These results support the judge's reliability for assessing the sampled KIM answers.

B Training and Implementation

Optimization settings. Both the rank-4 general adapter and contextual adjustments modify `q/k/v/o` projections in the last four transformer layers. The main-table Qwen model uses 16 operators, latent dimension 64, scale $\alpha = 8$, ridge coefficient 10^{-3}, and initial shrinkage threshold 10^{-2}. Training-curve, tool-interaction, retention, runtime, and case analyses use eight operators.

Table 7: KIM (Qwen) optimization schedule.

Stage	Steps	Peak learning rate	Validation interval
General adapter	7,400	2×10^{-4}	1,850
Local writes	4,300	2×10^{-3}	-
Joint field	800	1×10^{-4}	100

During local writing, the listed learning rate applies to local vectors; low-rank operators use 10^{-4}. Joint fitting freezes local vectors and updates projectors, thresholds, and operators at 10^{-4}. Gradient accumulation is 16 for general memory and 4 for contextual memory. Both use bfloat16, seed 20260725, clipping at 1.0, and cosine decay to one tenth of the peak rate. Warmup lasts 200 and 50 steps, respectively. AdamW weight decay is 0 for local vectors and 0.01 otherwise. The benefit weight is 0.5, margin 0.05, local-code penalty 0.01, outcome weight 0.2, and field anchor weight 0.1.

Checkpoint selection and plotted curves. Development-set generation selects general-adapter step 3,700 from steps 1,850, 3,700, 5,550, and 7,400. The main-table contextual model selects step 5,100 using the composite validation objective: response loss, excess over the reference-policy loss, and preference loss weighted by 0.2. Figure 3 plots the preference component on 16 validation pairs (32 continuations), not this composite objective. Its general-memory curve uses a seven-record trailing mean alongside raw batches; validation points are unsmoothed.

B.1 Learning objectives

State encoding and response supervision. State encoding uses the fixed general-memory policy on the most recent 4,096 tokens, taking the last-token hidden representation. Contextual memory is disabled during encoding. Response supervision uses complete native turns: tool-call targets end at the closing tag, and final-answer targets include the true EOS symbol. For a target y_i of L_i tokens after context c_i, the response loss is

$$\ell_i(u) = -\frac{1}{L_i}\sum_{j=1}^{L_i}\log \pi_u(y_{i,j} \mid c_i, y_{i,<j}), \tag{7}$$

Here π_u includes the contextual adjustment; π_0 uses only general memory, with fixed loss $\ell_i(0)$.

Joint field objective. Each state in trajectory τ receives weight $w_i = 1/N_\tau$. With local targets v_i fixed, let u_i be the field readout after shrinkage, $\mathcal{D}$ the response examples, and $\mathcal{S}$ the memory states. Joint fitting minimizes

$$\mathcal{L}_{\text{joint}} = \mathcal{L}_{\text{target}} + \lambda_{\text{field}}\mathcal{L}_{\text{fit}} + \lambda_{\text{out}}\mathcal{L}_{\text{pair}}, \tag{8}$$

$$\mathcal{L}_{\text{target}} = \frac{1}{|\mathcal{D}|}\sum_{i\in\mathcal{D}}\left\{\ell_i(u_i) + \lambda_{\text{b}}\log(1+\exp[m+\ell_i(u_i)-\ell_i(0)])\right\}, \tag{9}$$

$$\mathcal{L}_{\text{fit}} = \frac{1}{K|\mathcal{S}|}\sum_{i\in\mathcal{S}}\|u_i - v_i\|_2^2. \tag{10}$$

The field is recomputed from all states every 20 steps and held constant between solves; gradients do not propagate through the ridge solution.

Pairwise outcome loss. We adapt DPO (Rafailov et al., 2023) with a length-normalized reference-policy log-likelihood ratio. For continuation y after context c,

$$g_u(y;c) = \frac{1}{|y|}\sum_{j=1}^{|y|}\log\frac{\pi_u(y_j \mid c, y_{<j})}{\pi_0(y_j \mid c, y_{<j})}. \tag{11}$$

For successful/unsuccessful pairs $(c_p, y_p^+, y_p^-) \in \mathcal{P}$, both continuations use the same readout u_p:

$$d_p = g_{u_p}(y_p^+; c_p) - g_{u_p}(y_p^-; c_p), \tag{12}$$

$$\mathcal{L}_{\text{pair}} = \frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}}\log(1+e^{-d_p}). \tag{13}$$

Incremental compilation. For each boundary type b, sufficient statistics $G_b = \sum_i w_i k_i k_i^\top$ and $C_b = \sum_i w_i v_i k_i^\top$ give

$$M_b = C_b(G_b + \lambda I)^{-1}. \tag{14}$$

With $\lambda > 0$, adding new outer-product sums equals refitting the union if old keys, local vectors, weights, and λ stay fixed. Changing these quantities requires new statistics.

C Supplementary Experiments

C.1 Ablations and uncertainty

The four variants in Table 2 are implemented as follows.

- **w/o General Tool-use Memory:** remove the shared adapter, re-encode source states, and retrain local vectors, projectors, fields, operators, and thresholds under the frozen backbone.
- **w/o Contextual Experience Memory:** set $u_t = 0$; retain general memory with no retraining.
- **w/o Joint Memory Field:** read the four nearest local vectors of the same boundary type, with cosine similarities normalized by temperature-1 softmax; all other modules stay fixed.
- **w/o State Refresh:** hold the initial memory readout fixed across turns; tool observations remain in the generation context.

The paired 95% intervals use 10,000 bootstrap draws with seed 20260725. Within each dataset, question indices are resampled with replacement and shared across the two systems; each draw recomputes the seven-task mean difference. The 2.5th and 97.5th percentiles give marginal intervals conditional on the trained models. All paired verdicts use comparison-v3.

C.2 Memory retention and operator capacity

Retention protocol. Using the full 54k-bank checkpoint, a seed-20260725 permutation of unique trajectory identifiers defines nested 1%, 3%, 10%, and 25% subsets; 100% retains the full bank and 0% sets $u_t = 0$. All learned modules and local vectors remain fixed. Each retained trajectory contributes all writable states with its original weights. Both boundary-specific fields are recompiled with the original ridge coefficient and backend, without further training or evaluation-based selection.

Capacity protocol. We independently train $K \in \{4, 8, 16, 32\}$ operators with rank $r = 4$, key dimension 64, and fixed general memory, data, and optimization budget. Each run relearns local vectors, operators, projectors, and thresholds and selects step 5,100 by the same validation rule. Table 8 gives the dataset-level results underlying Figure 5(b). Two additional correct AIME24 answers account for 0.95 of the 1.24-point overall gain from eight to sixteen operators.

Table 8: Accuracy (%) by operator count on all seven benchmarks.

K	GSM8K	MATH500	AIME24	Hotpot	2Wiki	MuSiQue	Bamb.
4	74.50	60.00	13.33	44.50	37.00	17.50	31.20
8	77.00	64.00	10.00	45.00	39.50	15.50	31.20
16	77.00	62.00	16.67	43.50	38.50	18.00	35.20
32	74.00	64.50	6.67	45.00	40.50	14.50	39.20

Operator use. The sixteen-operator evaluation records 3,456 readouts on 1,155 questions. Let D_k concatenate operator k's adjustment matrices across target modules, including adapter scale. Its normalized magnitude at state t is

$$p_{t,k} = \frac{|\tanh(u_{t,k})|\,\|D_k\|_F}{\sum_j |\tanh(u_{t,j})|\,\|D_j\|_F}. \tag{15}$$

Every recorded state has nonzero total magnitude. We average within each question, then each dataset, then equally across datasets in the task group. The effective count $1/\sum_k p_{t,k}^2$, averaged in the same order, is 12.12 for Math and 11.88 for QA. This describes operator-term magnitudes, not causal accuracy contributions or cancellation between terms.

C.3 Tool interaction

The 1,155-question behavior analysis uses matching Qwen/KIM prompts and a four-call loop, before the main-table Qwen mathematics `print()` revision. RAG adds its retrieved examples. Normal completion requires a nonempty, closed `<answer>` block and loop status `complete`. Its denominator includes every question, including protocol failures and budget exhaustion; recovered partial answers do not count as normal completions.

Qwen3-30B-A3B labels feedback use from the immediate continuation after each tool return. Correct use includes applying relevant evidence, changing strategy after inadequate evidence, or repairing

an error. The annotator sees the prompt, generated prefix, current call, observation, and immediate continuation, but no method name, reference answer, or later tool return. Feedback-use accuracy is the fraction of valid yes/no labels marked yes. Empty continuations and uncertain, inapplicable, or invalid labels are excluded. Both metrics equally weight datasets within Math and QA.

C.4 Periodic self-experience updates

The dataset-stratified split assigns 577 questions to one half and 578 to the other before success selection. Each chain partitions its source half into three fixed, disjoint batches and evaluates on the opposite half. The current KIM answers each source question once, without reference answers or retries. Selection requires a correct comparison-v3 verdict and a complete trajectory, with numeric-equivalence checks for Math. Each batch receives two passes on new local vectors and contributes its statistics once; shared modules and earlier writes remain fixed. The updated model generates the next source batch. Evaluation feedback is used only for reporting; round three is the prespecified endpoint.

Table 9 combines the two evaluation halves; each question contributes only its held-out prediction. Changes differ between chains: A moves from 40.65% to 39.30%, and B from 42.47% to 44.76%.

Update timing. Figure 6(b) reports per-round means over the two chains, each using one A100-SXM4-80GB, rather than their summed cost. Encoding takes 9.25, 9.99, and 8.92 seconds; two local-writing passes take 96.69, 112.75, and 101.94 seconds. Closed-form field compilation takes 3.82, 3.04, and 3.10 milliseconds, reported separately below the second-scale bars. Timing excludes source generation, judging, model loading, and checkpoint I/O.

Table 9: Periodic updates: correct counts by dataset and average accuracy (%) by task group.

Dataset (questions)	R0	R1	R2	R3
GSM8K (200)	154	154	152	155
MATH500 (200)	124	123	122	123
AIME24 (30)	5	5	6	5
HotpotQA (200)	87	89	88	89
2Wiki (200)	77	78	79	78
MuSiQue (200)	36	35	38	38
Bamboogle (125)	44	45	43	45
Math	51.89	51.72	52.33	51.89
QA	33.80	34.25	34.23	34.63
Overall	41.55	41.74	41.99	42.02

D Efficiency Measurements

Runtime and hardware. The 128-question panel uses the first 20 examples per dataset, except AIME24 (8), in the formal evaluation order. Runs use one A100-SXM4-80GB, bfloat16, batch size one, greedy decoding, repetition penalty 1.1, and the four-call/16,384-token protocol. Qwen receives 8 CPU cores; RAG and KIM receive 16. After one excluded warm-up question, timing covers generation, dynamic memory reads, and tool execution; model loading, initial retrieval, and prompt assembly are excluded. Peak VRAM is the maximum CUDA tensor allocation in the generation process, excluding the separate retrieval service. Token counts cover all model turns, not tool returns.

Training and storage. The main-table sixteen-operator KIM (Qwen) requires 5.89 A100 GPU-hours for general-adapter training, state encoding, local writing, and joint fitting. The A100 training measurement for Tool-Star is 184 GPU-hours, covering cold-start fine-tuning, reinforcement learning, and self-critic DPO. These totals describe the respective training pipelines, which differ in trainable parameters, data usage, and optimization budget; the ratio does not isolate an algorithmic speedup at matched data or accuracy. The 3,609-trajectory contextual bank is not KIM's total training set: both general-adapter training and contextual response supervision use the larger historical pool. The KIM cost is for the Qwen configuration, not the total cost of KIM (Tool-Star).

Table 10 reports storage without backbone weights: KIM's 4.89 MiB comprises 0.79 MiB of general and 4.10 MiB of contextual memory; RAG's 1.09 GiB covers its trajectory bank and retrieval index.

Table 10: Peak GPU memory (128 questions) and method storage without backbone weights.

Method	Peak VRAM (GiB)	Extra storage
Qwen2.5-3B-Instruct	6.40	0 MiB
RAG	6.47	1.09 GiB
KIM	6.39	4.89 MiB

E CASE STUDY

Four recorded trajectories compare Qwen2.5-3B-Instruct with KIM under the configuration in Appendix B. Queries and observations are condensed; answers retain their comparison-v3 verdicts. QA prompts match; mathematics uses the baseline's explicit `print()` instruction and KIM's standard prompt. The cases illustrate behavior, not its frequency or individual-module effects.

E.1 KNOWLEDGE-INTENSIVE REASONING

Case A: Grounding the Next Query

Question. In what year was the company that was founded as Sound of Music added to the S&P 500?

Bamboogle, index 2. Two searches per method; no execution failures.

Qwen2.5-3B-Instruct	KIM
Query 1. Sound of Music **Blackstone Group** addition to S&P 500	**Query 1.** What company used to be called "Sound of Music"?
Observation. Results discuss Blackstone but do not establish the assumed company identity.	**Observation.** A Best Buy entry states that Sound of Music was renamed **Best Buy** in 1983.
Query 2. Blackstone Group addition to S&P 500	**Query 2.** When was **Best Buy** added to the S&P 500?
Observation. Blackstone became public in 2007; the model uses this to guess its index-entry year.	**Observation.** The result explicitly dates Best Buy's addition to the index to **1999**.
Answer. Around 2007 *(incorrect)*	**Answer.** 1999 *(correct)*

Analysis. With the same search count, KIM resolves the company identity before querying its index-entry year; the baseline continues searching an unsupported entity.

Case B: Interpreting the Same Evidence

Question. The author of *The Thing of It Is...* is what Nationality?

HotpotQA, index 8. Two searches per method; no execution failures.

Both methods identify **William Goldman** in their first search. Their second query, *William Goldman nationality*, and its complete returned text are identical. The relevant passage describes Goldman as born in Chicago and raised in a Jewish family.

Qwen2.5-3B-Instruct	KIM
Treats family background as nationality.	Distinguishes nationality from family background.
Answer. Jewish nationality *(incorrect)*	**Answer.** American *(correct)*

Analysis. The difference lies in interpreting the same feedback, not retrieving more evidence. KIM answers the requested attribute, although birthplace alone does not establish nationality.

E.2 MATHEMATICAL REASONING

Case C: Checking Constraints and Repairing a Tool Call

Question. Let N be the greatest four-digit positive integer with the property that whenever one of its digits is changed to 1, the resulting number is divisible by 7. Let Q and R be the quotient and remainder, respectively, when N is divided by 1000. Find $Q + R$.

AIME24, index 27. Baseline: no tool calls. KIM: three calls, one failure.

Qwen2.5-3B-Instruct	KIM
Candidate. Proposes $N = 8960$ through manual reasoning.	**Call 1.** Enumerates four-digit numbers in descending order and rejects any candidate failing one of the four replacements.
Check. Notes that replacing the hundreds digit gives 8160, which is not divisible by 7, but nevertheless accepts 8960.	**Return.** `5694`, the first candidate passing every check.
Calculation. Computes $8 + 960 = 968$ from the invalid candidate.	**Call 2.** Executes `print(Q + R)`; the definitions of Q and R appeared only in reasoning text.
	Error. `NameError: name 'Q' is not defined`
	Call 3. Defines $N = 5694$, $Q = N \,//\, 1000$, and $R = N \bmod 1000$, then prints $Q + R$. **Return:** `699`.
Answer. 968 *(incorrect)*	**Answer.** 699 *(correct)*

Analysis. KIM checks every constraint and uses the execution error to repair its next call. The failed call remains part of the trace; the baseline's boxed answer is recoverable despite an invalid protocol status.

Case D: Verifying the Requested Unknown

Question. Solve for x: $2^{2x} = 256^{\frac{1}{2}}$.

MATH500, index 116. One Python call per method; no execution failures.

Qwen2.5-3B-Instruct	KIM
Computation. Rewrites $256 = 2^8$ and calculates the right-hand exponent as $8/2$.	**Derivation.** Obtains $2x = 4$, hence $x = 2$, and substitutes this value into the original equation.
Return. `4.0`	**Return.** `(True, 2)` from checking equality of both sides and returning x.
Answer. $x = 4$ *(incorrect)*	**Answer.** $x = 2$ *(correct)*

Analysis. The baseline's arithmetic is correct, but its return is the value of $2x$, not x. KIM distinguishes the intermediate quantity from the requested unknown and verifies its solution.