MEMAGENT-CRITIC: ADAPTIVE MEMORY-DRIVEN CRITIC TRAINING FOR LONG-CONTEXT MEMORY AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Training long-context memory agents with reinforcement learning faces sparse feedback: a single verifiable reward arrives only after the entire document is processed. A learned memory-trace critic supplies additional trajectory-level guidance, but a fixed critic can become misaligned with the evolving policy. To adapt this guidance to current-policy trajectories, we propose **MemAgent-Critic**, which updates a 1.5B critic online alongside a 7B memory policy. After initialization with GPT-5.4- and policy-generated traces, the critic is updated using within-question comparisons of combined answer and critic rewards. These updates reuse existing policy rollouts without additional teacher annotations. On RULER-HotpotQA, the resulting 7B policy achieves **93.75%** accuracy at 7K tokens and **80.47%** at 3.5M tokens. Across ten context lengths, the adaptive variant obtains an average accuracy of **88.75%**, compared with 87.27% for the frozen-critic variant, although the variants use different training durations and the frozen critic performs better at the two longest lengths. Additional evaluations report gains on several document QA, summarization, and code-completion tasks.

1 INTRODUCTION

Modern language models fail to reliably process extended documents despite large nominal context windows: models that succeed on simple retrieval tasks collapse when the answer requires combining evidence from distant positions (Hsieh et al., 2024; Bai et al., 2023). A promising alternative is to treat long-context reasoning as sequential memory management (Yu et al., 2025a): an agent reads the document in chunks, maintains a question-conditioned working memory at each step, and answers from the final memory state. Yu et al. (2025a) showed that training such an agent with RLVR on binary answer-match reward generalizes far beyond the training distribution, outperforming models with nominally larger context windows.

However, reinforcement learning with verifiable reward (RLVR) introduces a severe credit assignment problem (Williams, 1992; Sutton & Barto, 2018). A single binary reward arrives only after the entire document is processed, with no feedback on which memory updates were beneficial or harmful. Different memory traces can receive the same answer reward although they preserve different evidence, retain unsupported inferences, or discard facts needed for later reasoning. In a recurrent agent, an error introduced into one memory state can also be carried into subsequent updates. The terminal answer reward alone does not distinguish these intermediate behaviors among trajectories with the same outcome.

A learned memory-trace reward can supplement this sparse supervision by ranking complete sequences of memory updates. However, the usefulness of a fixed critic may change as the policy's trajectory distribution evolves. Memory strategies encountered during later training can differ from those represented in the critic's initialization data, and a fixed score need not remain informative about the current policy's alternatives. Reward-model overoptimization further motivates examining how learned rewards behave during policy training (Gao et al., 2023). These considerations motivate updating the critic on current-policy traces, alongside the agent that produces them.

We propose **MemAgent-Critic**, which adds a 1.5B memory-trace critic to a 7B memory policy. After initialization with teacher- and policy-generated traces, the critic provides an auxiliary terminal

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.

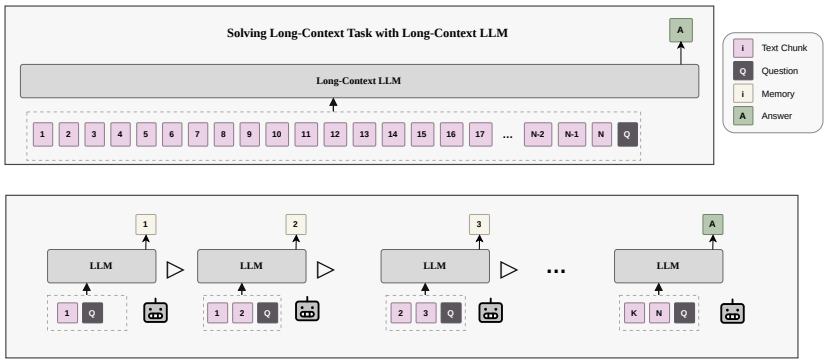


Figure 1: Sequential chunk-by-chunk memory updating for long-context question answering. A long document is first partitioned into ordered text chunks. Given the question, the memory agent reads one chunk at a time and updates a question-conditioned memory after each chunk. The updated memory is then carried forward to the next step, allowing the agent to accumulate task-relevant evidence without attending to the entire document at once. After the final chunk is processed, the agent generates the answer from the final memory state.

reward that is combined with the verifiable answer reward. During online training, trajectories are labeled `Good` or `Bad` according to whether their combined reward is above or below the within-question group mean. These labels update the critic using existing GRPO/DAPO-style rollouts (Yu et al., 2025b), without additional rollout collection or online teacher annotations. The resulting signal refines trajectory-level comparisons; it does not assign a separate quality label to each memory update. Because the labels include the critic's own scores, the online procedure is a form of coupled self-training, whose utility we evaluate empirically through frozen- and adaptive-critic variants.

Our contributions are:

1. We adapt online, combined-reward self-training to recurrent long-context memory agents, using complete memory traces and within-question GRPO comparisons.
2. We evaluate frozen and online-updated critics across ten context lengths and additional tasks. MemAgent-Critic achieves 93.75% accuracy at 7K and 80.47% at 3.5M tokens, with the highest ten-length average among the evaluated critic variants.

2 Related Work

Long-context models and memory agents. Existing long-context methods mainly extend the input length that a model can process in one pass or replace full-context attention with a compressed state. Positional-extension methods modify rotary or related position representations to support longer sequences (Su et al., 2021; Peng et al., 2023b; Ding et al., 2024). LongLoRA combines parameter-efficient fine-tuning with shifted sparse attention to extend context length (Chen et al., 2024). Sparse and efficient attention methods reduce the cost of long-sequence processing through local attention, block-sparse patterns, IO-aware kernels, or streaming KV-cache management (Beltagy et al., 2020; Zaheer et al., 2020; Dao et al., 2022; Xiao et al., 2023). Recurrent and memory-augmented models maintain state across segments instead of attending to the full document at once (Dai et al., 2019; Gu & Dao, 2023; Peng et al., 2023a; Wu et al., 2022; Munkhdalai et al., 2024; Packer et al., 2023; Ge et al., 2023). MemAgent-Critic is an extension of MemAgent (Yu et al., 2025a) that introduces an adaptive memory-trace critic to provide trajectory-level guidance alongside sparse final-answer reward. ReMemR1 (Shi et al., 2025) enables callbacks to historical memories and combines final-answer rewards with dense step-level signals. Our method retains the MemAgent architecture and learns an adaptive reward over complete memory traces.

Reinforcement learning for language models. RLHF commonly trains a reward model from human preferences and optimizes the policy with PPO (Schulman et al., 2017; Stiennon et al., 2020; Ouyang et al., 2022). For tasks with automatically checkable answers, RLVR replaces human

preference labels with verifiable task rewards. GRPO normalizes rewards within a group of sampled rollouts and avoids training a separate value network (Shao et al., 2024); this group-based formulation has been used in recent reasoning models such as DeepSeek-R1 (DeepSeek-AI, 2025). In long-context memory agents, however, the verifiable reward is observed only after many memory-update decisions, creating a deep credit assignment problem (Williams, 1992; Sutton & Barto, 2018). MemAgent-Critic uses the same group structure that GRPO already produces, but converts within-group rankings into critic supervision.

Process reward models and memory-trace reward. Process reward models provide feedback on intermediate reasoning steps rather than only final answers. Existing labels can come from human annotation (Lightman et al., 2023), automatic completion-based supervision (Wang et al., 2024), or tree search (Luo et al., 2024). Cobbe et al. (2021) instead train verifiers to assess complete solutions. GAR (Liu et al., 2025) jointly trains its reasoner and discriminator on-policy; its evaluator is not fixed. Reward shaping also provides a classical framework for adding intermediate rewards when a useful potential function is available (Ng et al., 1999). Offline reward models can inherit a fixed criterion from their training data, whereas online approaches such as GAR update their evaluators. Our critic uses complete memory traces and receives labels from current-policy within-group comparisons at every training step.

Reward overoptimization and online updates. Reward-model overoptimization studies how policies can exploit a fixed reward model (Gao et al., 2023). MemAgent-Critic updates its critic on current-policy trajectories through an alternating training schedule. Its labels depend on combined rewards, so online updating alone does not ensure that critic errors are corrected.

3 METHOD

MemAgent-Critic augments sparse answer rewards with a learned trajectory-level reward. We describe the recurrent memory agent, its group-based RL objective, and the critic's scoring and online update procedure.

Framework overview. Figure 2 contrasts MemAgent-Critic with standard GRPO. Both use group-relative advantages derived from complete rollouts. MemAgent-Critic additionally scores the sequence of memory states and combines this trajectory-level auxiliary reward with the final-answer reward.

3.1 PROBLEM FORMULATION: RECURRENT MEMORY AGENT

We consider long-context question answering with a question q, a long document D, and a ground-truth answer a^*. The document is partitioned into K chunks, $D = (c^1, c^2, \ldots, c^K)$. A recurrent memory agent processes the document sequentially rather than attending to the full document at once. Starting from an empty memory $\hat{m}_0 = \varnothing$, the policy π_θ reads one chunk at a time and updates its working memory:

$$\hat{m}_k \sim \pi_\theta\big(\cdot \;\big|\; q,\, c^k,\, \hat{m}_{k-1}\big)\,, \qquad k = 1, \ldots, K. \tag{1}$$

After all chunks have been processed, the policy generates the final answer from the accumulated memory:

$$a \sim \pi_\theta(\cdot \mid q,\, \hat{m}_K)\,. \tag{2}$$

The complete trajectory

$$\tau = (\hat{m}_1, \hat{m}_2, \ldots, \hat{m}_K, a) \tag{3}$$

is the *memory trace*. In this paper, each $\hat{m}_k$ is a natural-language memory state, so the trace can be inspected and scored by a language-model critic.

The final answer depends on many memory updates, while the verifier supplies only a terminal reward. The auxiliary critic described below also supplies a terminal score, based on the memory sequence.

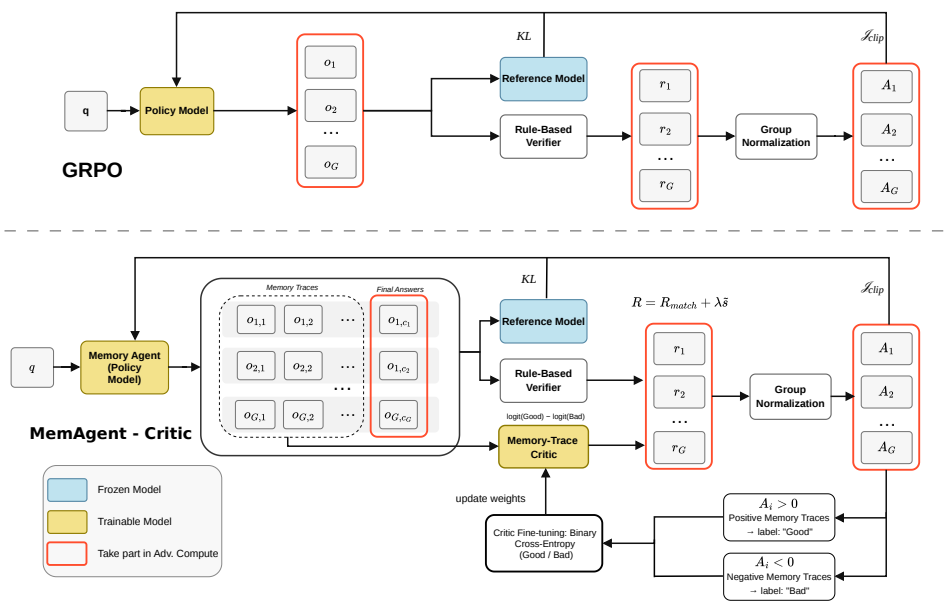


Figure 2: **Overview of MemAgent-Critic.** The top row shows standard GRPO: for each question, the policy samples a group of rollouts, a rule-based verifier assigns final-answer rewards, group normalization converts these rewards into relative advantages, and the clipped policy objective updates the actor under a KL penalty to the reference model. The bottom row shows our extension for recurrent memory agents. The policy processes a long document through multiple memory-update conversations, producing a group of memory traces and final answers. A lightweight critic scores each complete memory trace using the log-odds $\mathrm{logit}(\texttt{Good}) - \mathrm{logit}(\texttt{Bad})$; this score is added to the verifier reward as an auxiliary memory-trace signal for the policy update. In parallel, the sign of the combined-reward within-group advantage is used to label traces as `Good` or `Bad`, and these labels online fine-tune the critic with binary cross-entropy. Blue modules are frozen, yellow modules are trainable, and red outlines indicate components that participate in advantage computation.

3.2 Sparse Group-Based RL Objective

The policy is trained with a group-based RL objective using verifiable answer rewards. For each training example (D_i, q_i, a_i^*), the current policy samples a group of rollouts $\{\tau_{i,1}, \ldots, \tau_{i,G}\}$. Each rollout receives a binary answer-match reward:

$$R_{\text{match}}(\tau_{i,g}) = \mathbb{1}[f(a_{i,g}) = f(a_i^*)], \tag{4}$$

where $f(\cdot)$ is an answer-normalization function.

The reward is normalized within each group to obtain a sparse-reward relative advantage:

$$\hat{A}_{i,g}^{\text{match}} = \frac{R_{\text{match}}(\tau_{i,g}) - \mu_{R,i}}{\sigma_{R,i} + \epsilon}, \tag{5}$$

where $\mu_{R,i}$ and $\sigma_{R,i}$ are the mean and standard deviation of the rewards for question i. Groups in which all rollouts succeed or all fail have $\sigma_{R,i} = 0$, producing zero advantage for every trajectory. With group filtering enabled, these groups are excluded from actor updates using the sparse reward, as in DAPO (Yu et al., 2025b). The implementation retains the original trajectory batch for critic relabeling: filtered groups can still supply critic labels when their combined rewards differ. Actor filtering and critic labeling therefore have different scopes.

Let $\pi_{\theta_{\text{old}}}$ denote the policy that generated the rollouts. For a generated token $o_{i,g,t}$ with history $h_{i,g,t}$, define the importance ratio

$$r_{i,g,t}(\theta) = \frac{\pi_\theta(o_{i,g,t} \mid h_{i,g,t})}{\pi_{\theta_{\text{old}}}(o_{i,g,t} \mid h_{i,g,t})}. \tag{6}$$

Following DAPO, we optimize the token-level clipped objective

$$\mathcal{L}_{\mathrm{DAPO}}(\theta) = -\mathbb{E}_{i,g,t}\left[\min\left(r_{i,g,t}(\theta)\hat{A}^{\mathrm{match}}_{i,g},\ \mathrm{clip}(r_{i,g,t}(\theta), 1-\varepsilon_{\mathrm{low}}, 1+\varepsilon_{\mathrm{high}})\,\hat{A}^{\mathrm{match}}_{i,g}\right)\right] + \beta\,\mathcal{D}_{\mathrm{KL}}[\pi_\theta \,\|\, \pi_{\mathrm{ref}}]\,. \tag{7}$$

Equations 5 and 7 describe the verifier-only baseline. With the critic enabled, the actor instead uses combined-reward advantages in the same clipped objective, and their signs also provide critic labels, as defined below.

3.3 Memory-Trace Critic

The memory-trace critic D_ϕ receives the question and the sequence of memory states, serialized as $x_{i,g}$ for rollout g of example i. It scores the output tokens `Good` and `Bad`. During online training, these labels indicate whether the trajectory's combined reward is above or below its within-question group mean (Section 3.4).

We convert the binary judgment into a continuous reward by taking the log-odds between the two output tokens:

$$s_{i,g} = \mathrm{logit}_\phi\big(\texttt{Good} \mid x_{i,g}\big) - \mathrm{logit}_\phi\big(\texttt{Bad} \mid x_{i,g}\big), \tag{8}$$

where $\mathrm{logit}_\phi(\cdot \mid x)$ denotes the pre-softmax logit under the critic. A larger $s_{i,g}$ indicates a stronger critic preference for `Good` over `Bad`.

Before combining, the raw critic scores $\{s_{i,g}\}$ are z-score normalized within the rollout batch:

$$\tilde{s}_{i,g} = \frac{s_{i,g} - \mu_s}{\sigma_s + \epsilon}, \tag{9}$$

where μ_s and σ_s are the mean and standard deviation over all scored trajectories in the batch. This standardizes the batch-level scale of critic scores before they are combined with the answer reward.

The policy receives a combined reward:

$$R_{i,g} = R_{\mathrm{match},i,g} + \lambda \tilde{s}_{i,g}, \tag{10}$$

where λ=0.1 controls the contribution of the critic. Importantly, the critic reward is applied as a *terminal bonus*: it is added only at the final token position of each trajectory rather than distributed across all memory-update steps.

For policy optimization, this combined reward is normalized within the rollout group to obtain policy advantages.

Before policy RL, the critic is initialized with binary cross-entropy, labeling GPT-5.4-generated traces as positive and policy-generated traces as negative. Appendix B describes the training pipeline and initialization data.

3.4 Adaptive Critic via Within-Group Relabeling

The central idea of MemAgent-Critic is to obtain critic supervision from the same rollout groups already generated for policy RL. For each question, group-based RL samples multiple rollouts and computes within-group advantages. Let $\mu_{C,i}$ and $\sigma_{C,i}$ be the mean and standard deviation of the combined rewards $R_{i,g}$ within question i. We compute

$$\hat{A}^{\mathrm{combined}}_{i,g} = \frac{R_{i,g} - \mu_{C,i}}{\sigma_{C,i} + \epsilon}. \tag{11}$$

Rollouts with positive combined-reward advantage perform better than the current group average under the combined reward, while rollouts with negative combined-reward advantage perform worse. We therefore use the sign of this combined advantage as an online training label, excluding zero-advantage trajectories:

$$\mathcal{P}_{\mathrm{pos}} = \{\tau_{i,g} : \hat{A}^{\mathrm{combined}}_{i,g} > 0\}, \qquad \mathcal{P}_{\mathrm{neg}} = \{\tau_{i,g} : \hat{A}^{\mathrm{combined}}_{i,g} < 0\}. \tag{12}$$

Trajectories with $\hat{A}_{i,g}^{\text{combined}} = 0$ are excluded. Labels are computed over the original collected trajectories, including uniform-answer-reward groups excluded from actor updates. These labels measure relative combined reward, not independent answer correctness.

The critic is updated on balanced samples from these two sets using

$$\mathcal{L}_{\text{critic}}(\phi) = -\frac{1}{|\mathcal{B}|} \sum_{(\tau,y)\in\mathcal{B}} \left[y \log \sigma(s_\phi(\tau)) + (1-y) \log(1 - \sigma(s_\phi(\tau))) \right]. \tag{13}$$

Online relabeling reuses current-policy rollouts without additional human process annotation or rollout collection. The critic is updated on balanced current-batch examples before the actor update, which uses the already computed advantages. Because labels depend on the combined reward, they also depend on the critic's current scores; detaching them does not remove this self-training dependency. Appendix A details the update procedure.

Prompt templates are provided in Appendix D.

4 Experiments

4.1 Setup

We evaluate MemAgent-Critic along three dimensions: in-distribution training progress, long-context length generalization, and document QA transfer, with additional non-QA evaluations.

Benchmarks. We use **HotpotQA dev** Yu et al. (2025a) as the in-distribution validation set, sampling 200 development examples and reporting exact-match accuracy after answer normalization. For long-context generalization, we evaluate on **RULER-HotpotQA** Hsieh et al. (2024); Yang et al. (2018) at ten context lengths: 7K, 14K, 28K, 56K, 112K, 224K, 448K, 896K, 1.75M, and 3.5M tokens. For transfer evaluation, we use **LongBench-QA** Bai et al. (2023), including 2WikiMultiHopQA, HotpotQA, MuSiQue, and Qasper. Each RULER context length contains 128 questions, evaluated using one selected checkpoint and one trajectory/answer per question, without best-of-G selection or voting. The 16-rollout group is training-only. Additional LongBench evaluations use all 200 local Multi-News and 500 LCC test examples (Section 4.4).

Compared methods. We compare against three groups of baselines. First, proprietary long-context models are evaluated within their supported context windows, with longer contexts marked N/A. GPT-5.4 denotes a strong proprietary teacher model used for baseline comparison and critic initialization. Second, we compare against open-source long-context and reasoning models, including Qwen2.5-Instruct-1M, QwenLong-L1, DeepSeek-R1-Distill-Qwen, and standard Qwen2.5-Instruct models. Third, we compare against RL-MemAgent-7B and RL-MemAgent-14B, which use the same recurrent memory-agent architecture but are trained only with sparse final-answer reward.

Implementation details. All policy variants use the same recurrent memory-agent format and the same group-based RL setup. The critic reward is added to the sparse answer reward during policy optimization, while MemAgent-Critic updates the critic online from combined-reward within-group rollout labels. Shared hyperparameters are summarized in Table 6 in Appendix B.

4.2 Main Results

RULER-HotpotQA. Table 1 reports accuracy across context lengths from 7K to 3.5M tokens. MemAgent-Critic achieves strong performance across the evaluated context lengths. Compared with RL-MemAgent-7B, it improves by 12.50 points at 7K tokens and by 8.59 points at 3.5M tokens. The same MemAgent-Critic checkpoint yields 120/128 correct answers at 7K and 103/128 at 3.5M. Critic ablations are analyzed separately in Section 4.3. These end-to-end scores do not isolate memory writing from final-answer generation.

Training dynamics. Figure 3 shows validation accuracy during the two main training stages. Frozen-Critic raises HotpotQA dev accuracy from 84.4% to 88.5%. MemAgent-Critic then adapts

Table 1: RULER-HotpotQA accuracy across context lengths. All values are accuracy (%). **Bold** marks the best accuracy after rounding to one decimal place for cross-model comparison; displayed values retain their reported precision. N/A indicates the model's context window does not cover that length.

Model	Context Length									
	7K	**14K**	**28K**	**56K**	**112K**	**224K**	**448K**	**896K**	**1.75M**	**3.5M**
Proprietary models										
GPT-4o	85.2	85.2	85.9	81.2	82.8	76.6	N/A	N/A	N/A	N/A
GPT-5.4	89.8	89.8	**92.2**	88.3	89.1	**87.5**	81.2	75.0	N/A	N/A
Gemini-2.5-Pro	77.3	74.2	68.8	76.6	74.2	68.8	74.2	68.0	N/A	N/A
DeepSeek-R1	50.8	47.7	48.4	48.4	47.7	N/A	N/A	N/A	N/A	N/A
Open-source long-context models										
QwenLong-L1-32B	72.66	75.00	72.66	60.94	31.25	17.19	13.28	11.72	N/A	N/A
Qwen2.5-Instruct-14B-1M	60.16	60.94	50.00	57.03	50.00	37.50	8.59	0.00	N/A	N/A
Qwen2.5-Instruct-7B-1M	61.72	56.25	53.91	55.47	51.56	33.59	12.50	0.00	N/A	N/A
DS-Distill-Qwen-32B	70.31	66.41	65.62	46.88	23.44	13.28	7.81	7.03	N/A	N/A
DS-Distill-Qwen-14B	64.06	64.84	57.03	40.62	14.84	8.59	3.12	6.25	N/A	N/A
DS-Distill-Qwen-7B	30.47	12.50	3.12	0.00	0.00	0.78	0.00	0.00	N/A	N/A
Qwen2.5-Instruct-32B	69.53	64.84	60.16	51.56	44.53	21.88	14.06	7.03	N/A	N/A
Qwen2.5-Instruct-14B	75.00	67.19	57.03	54.69	44.53	21.88	10.94	2.34	N/A	N/A
Qwen2.5-Instruct-7B	52.34	57.03	51.56	44.53	32.81	13.28	6.25	1.56	N/A	N/A
Memory-agent baselines										
RL-MemAgent-14B	80.47	82.03	82.03	83.59	81.25	77.34	79.69	75.78	78.91	71.09
RL-MemAgent-7B	81.25	81.25	82.03	80.47	79.69	75.78	76.56	74.22	77.34	71.88
MemAgent-Critic	**93.75**	**92.97**	**92.19**	**91.41**	**91.41**	86.72	**87.50**	**87.50**	**83.59**	**80.47**

the critic online from this checkpoint and reaches 92.0%, giving a total gain of 7.6 points over the RL-MemAgent initialization. These values describe successive training stages rather than matched-duration comparisons.

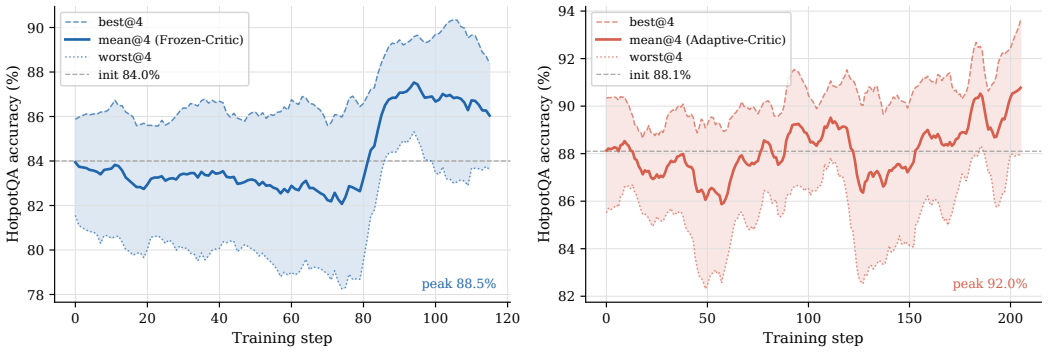


Figure 3: HotpotQA dev accuracy across training stages. Left: Stage 2 (Frozen-Critic) reaches 88.5% accuracy. Right: Stage 3 (MemAgent-Critic with online adaptation) continues from the frozen checkpoint and reaches 92.0%. Shaded bands span worst@4 to best@4; solid lines show mean@4. Dashed horizontal lines mark the initialization accuracy of each stage.

LongBench-QA. Table 2 reports results on four LongBench-QA datasets. MemAgent-Critic obtains the best average score, improving over MemAgent-7B by 4.1 points and over MemAgent-14B by 1.5 points. The largest gain is on Qasper, where MemAgent-Critic reaches 39.5%, compared with 29.0% for MemAgent-7B and 26.0% for MemAgent-14B. The method does not dominate every subtask: DS-Distill-Qwen models are stronger on 2Wiki, and MemAgent-14B remains strongest on MuSiQue.

Table 2: LongBench-QA accuracy.

Method	2Wiki	HQA	MuSiQue	Qasper	AVG
QwenLong-L1-32B	83.0	69.5	51.0	24.0	56.9
Qwen2.5-Instruct-14B-1M	70.5	65.5	35.0	22.0	48.3
Qwen2.5-Instruct-7B-1M	67.5	58.5	26.0	24.0	44.0
DS-Distill-Qwen-32B	**83.5**	69.0	47.5	21.0	55.3
DS-Distill-Qwen-14B	**83.5**	67.0	42.0	22.0	53.6
DS-Distill-Qwen-7B	47.0	31.5	7.5	17.5	25.9
Qwen2.5-Instruct-32B	68.5	66.0	37.0	22.5	48.5
Qwen2.5-Instruct-14B	67.0	63.5	39.0	20.5	47.5
Qwen2.5-Instruct-7B	52.5	59.0	24.0	19.0	38.6
MemAgent-14B	79.0	73.0	**52.0**	26.0	57.5
MemAgent-7B	74.0	69.5	47.0	29.0	54.9
MemAgent-Critic	73.0	**75.5**	48.0	**39.5**	**59.0**

Table 3: Ablation on RULER-HotpotQA. No-Critic uses only sparse final-answer reward; Frozen-Critic adds a fixed memory-trace critic reward; Adaptive-Critic is the full MemAgent-Critic model with online critic relabeling. All values are accuracy (%).

Variant	Context length										AVG
	7K	14K	28K	56K	112K	224K	448K	896K	1.75M	3.5M	
No-Critic-7B	90.62	86.72	88.28	84.38	85.94	82.81	81.25	82.81	83.59	75.78	84.22
Frozen-Critic-7B	92.97	89.06	89.06	89.84	89.06	84.38	82.81	83.59	**87.50**	**84.38**	87.27
Adaptive-Critic-7B	**93.75**	**92.97**	**92.19**	**91.41**	**91.41**	**86.72**	**87.50**	**87.50**	83.59	80.47	**88.75**

4.3 Ablations

Ablation design. We compare three training variants: **No-Critic-7B**, **Frozen-Critic-7B**, and **Adaptive-Critic-7B** (MemAgent-Critic). No-Critic uses only sparse final-answer reward. Frozen-Critic adds a fixed memory-trace reward. Adaptive-Critic is the full method, which further updates the critic online using within-group relabeling.

Frozen and adaptive critics. Table 3 reports average accuracies of 84.22% for No-Critic, 87.27% for Frozen-Critic, and 88.75% for Adaptive-Critic. The adaptive variant achieves the highest score at eight of ten context lengths. Frozen-Critic performs better at 1.75M (87.50% versus 83.59%) and 3.5M (84.38% versus 80.47%). The variants use different training durations, so these comparisons do not isolate the effect of online critic updates. Training diagnostics are provided in Appendix C.

4.4 Additional Evaluations

Inference memory budget. We examine whether the observed gains persist under different memory limits. The RULER-HotpotQA evaluation contains 128 examples, each with 100 documents and approximately 14K tokens on average. We retain 5,000-token chunks and vary the memory budget under deterministic decoding (temperature 0, top-p 1), keeping other evaluation settings fixed. As shown in Table 4, MemAgent-Critic scores higher than both RL-MemAgent baselines at all three evaluated budgets. MemAgent-Critic improves from 0.90 to 0.92 between 512 and 1,024 tokens and remains at 0.92 with 2,048 tokens at this precision. Increasing the maximum memory size therefore does not yield a further improvement beyond 1,024 tokens in this evaluation. This tests inference memory budgets at one context length, not separately retrained policies or robustness across every context length.

Summarization and code completion. We also evaluate two tasks beyond question answering. We evaluate the complete local test sets of Multi-News (200 examples) and LCC (500 examples), using deterministic decoding (temperature 0, top-p 1) on one NVIDIA A100 GPU per model. Multi-News measures multi-document summarization; LCC measures long-context code completion. Table 5

Table 4: Memory-budget evaluation at approximately 14K context. Values are accuracies on a 0–1 scale.

Memory tokens	MemAgent-Critic	RL-MemAgent-7B	RL-MemAgent-14B
512	**0.90**	0.78	0.78
1,024	**0.92**	0.81	0.82
2,048	**0.92**	0.81	0.83

reports higher scores for MemAgent-Critic than RL-MemAgent-7B on both datasets. On Multi-News, ROUGE-1/2/L scores are 0.37/0.12/0.19 for MemAgent-Critic and 0.35/0.10/0.17 for RL-MemAgent-7B. On LCC, CodeSim is 0.26 versus 0.22 and token F1 is 0.19 versus 0.12. These results extend the evaluation to two additional task families; they do not isolate the contribution of online updating, and no statistical significance is established.

Table 5: Non-QA LongBench results. Scores are reported on the original 0–1 scale, not as percentages.

Dataset	Metric	MemAgent-Critic	RL-MemAgent-7B
Multi-News	ROUGE-1	**0.37**	0.35
	ROUGE-2	**0.12**	0.10
	ROUGE-L	**0.19**	0.17
LCC	CodeSim	**0.26**	0.22
	Token F1	**0.19**	0.12

Training costs and the accounting scope are reported in Appendix E.2.

Memory revision. In a recorded HotpotQA case, RL-MemAgent-7B associates Thirukkalacherry with Kumbakonam before direct evidence is available. When the third chunk identifies Tharangambadi as the relevant location, the baseline stores the new evidence but retains its incompatible earlier inference. MemAgent-Critic instead updates its core memory with the newly observed village–location relation. This example concerns how evidence changes a memory state, rather than whether the evidence is encountered. It illustrates one observed behavior without establishing its frequency or attributing it to online critic updates (Appendix E.3).

5 Conclusion

We presented **MemAgent-Critic**, which updates a 1.5B trajectory critic online using combined-reward comparisons to guide a 7B memory policy. The method retains the recurrent memory-agent architecture and augments sparse final-answer rewards with a learned score over memory traces. Its online labels are derived from existing within-question rollout groups, avoiding additional rollout collection or teacher annotation after critic initialization. The resulting policy achieves 93.75% accuracy at 7K and 80.47% at 3.5M tokens. Across ten context lengths, Adaptive-Critic obtains the highest average accuracy among the evaluated variants, at 88.75%. The memory-budget evaluation reports higher scores than both RL-MemAgent baselines across the tested limits, and the additional evaluations report gains on summarization and code completion.

The results also identify limits of the current approach. Frozen-Critic remains stronger at the two longest context lengths, and the unmatched training durations prevent attributing the full performance difference to online updates. The critic's labels depend on its own scores, so adaptation does not by itself establish independently calibrated memory-quality judgments. Moreover, critic-based training incurs substantially greater step time than No-Critic. These findings motivate further study of independently grounded critic supervision and the trade-off between adaptation, length generalization, and training cost.

AI USE STATEMENT

We used GPT-5.4 to generate synthetic positive memory traces for critic initialization, as described in Appendix B. We also used OpenAI Codex to assist with manuscript drafting and editing, literature lookup, interpretation and presentation of existing experimental results, checking mathematical claims and consistency with the local training implementation, and LaTeX formatting and compilation. The authors are responsible for the final text, claims, experimental results, and AI-assisted content of this paper.

REPRODUCIBILITY STATEMENT

Code and Data Availability. We will publicly release the code upon acceptance. The benchmark datasets used in this work are available from their respective original sources, as cited in Section 4.1. The construction of critic-training data is described in Appendix B, and additional evaluation protocols are provided in Section 4.4 and Appendix E.

REFERENCES

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. LongLoRA: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*, 2024.

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.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. LongRoPE: Extending LLM context window beyond 2 million tokens. *arXiv preprint arXiv:2402.13753*, 2024.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 10835–10866, 2023.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. ICAE: In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*, 2023.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Qihao Liu, Luoxin Ye, Wufei Ma, Yu-Cheng Chou, and Alan Yuille. Generative adversarial reasoner: Enhancing LLM reasoning with adversarial reinforcement learning. *arXiv preprint arXiv:2512.16917*, 2025.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*, 2024.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*, pp. 278–287, 1999.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, et al. RWKV: Reinventing RNNs for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023a.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yaorui Shi et al. Look back to reason forward: Revisitable memory for long-context llm agents. *arXiv preprint arXiv:2509.23040*, 2025.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.

Fanqi Wan, Weizhou Shen, Shengyi Liao, Yingcheng Shi, Chenliang Li, Ziyi Yang, Ji Zhang, Fei Huang, Jingren Zhou, and Ming Yan. Qwenlong-l1: Towards long-context large reasoning models with reinforcement learning. *arXiv preprint arXiv:2505.17667*, 2025.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2024.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992.

Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. *arXiv preprint arXiv:2203.08913*, 2022.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

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

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, et al. MemAgent: Reshaping long-context LLM with multi-conv RL-based memory agent. *arXiv preprint arXiv:2507.02259*, 2025a.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025b.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. *arXiv preprint arXiv:2007.14062*, 2020.

A Label Construction and Update Order

The implementation first collects verifier rewards and critic scores, normalizes the critic scores across the scored batch, and adds the weighted scores to the terminal rewards. It computes within-group advantages from these combined rewards. The actor uses the advantages on the batch retained by sparse-reward filtering; the critic replay pools use their signs over the original collected memory traces.

For critic optimization, the computed signs are detached and treated as fixed binary targets. This is a statement about the gradient used in an update, not independence of the targets from the model that produced them. Critic-score changes can alter both the ranking and its signs. In particular, a group with uniform verifier rewards can contribute self-generated critic labels even though it is excluded from actor updates. No critic-label-independence theorem applies.

The terminal reward is also batch-dependent and changes with the critic; no optimal-policy-invariance guarantee is asserted. The reported experiments evaluate this coupled training procedure and do not establish that feedback-induced label errors are eliminated.

A.1 Equal-Outcome Trajectories and Normalized Advantages

The critic can still distinguish equal-outcome trajectories within a retained group. For example, rewards $[1, 0, 0, 0]$ have nonzero variance, but the three failed traces share the same sparse advantage. For two such traces g, h, their combined-reward advantage difference is $\lambda(\tilde{s}_{i,g} - \tilde{s}_{i,h})/(\sigma_{R^{\text{combined}},i} + \epsilon)$. Thus $\lambda = 0.1$ does not by itself bound the normalized advantage change. Fully uniform-outcome groups are filtered from actor updates; this example concerns the actor advantage within a retained group.

B Training Pipeline

MemAgent-Critic trains in four sequential stages. Table 6 lists the shared hyperparameters.

Table 6: Shared training hyperparameters.

Hyperparameter	**Value**
Group size	16 rollouts
Max context per step	8,192 tokens
Chunk size	5,000 tokens
Max new tokens	1,024
Train batch size	128 samples
Actor learning rate	1×10^{-6}
LR warmup steps	20
KL coefficient	10^{-3} (low-variance KL)
Clip ratio high	0.20
Rollout temperature	1.0 (top-p 1.0)
Val temperature	1.0 (top-p 0.7)
Max memory length	1,024 tokens
Critic reward coef λ	0.1
Critic learning rate	2×10^{-5}
Critic update steps	10 per policy step
Critic replay pool	2,048 per class
Infrastructure	4×8 b200 GPUs

B.1 Stage 1: Critic Initialization with GPT-5.4 Supervision

Before policy RL, the critic is initialized using teacher- and policy-generated memory traces.

Positive traces (GPT-5.4 teacher). We run the MemAgent workflow (Eqs. 1–2) using GPT-5.4 at temperature 0, with chunk size 5 000 tokens and maximum 1 024 new tokens per memory update, on

32 000 HotpotQA training instances (Yang et al., 2018) (each ≈32K tokens of context). The resulting traces $\tau^+ \sim \pi_{\text{GPT-5.4}}(\cdot \mid q, D)$ are collected as positive examples $\mathcal{D}^+ = \{(\tau^+, 1)\}$.

Negative traces (early-stage policy). We sample traces from the Qwen2.5-7B RL-MemAgent policy at four intermediate checkpoints $t \in \{100, 200, 300, 400\}$ training steps using temperature 1.0, top-p 0.7. Each checkpoint is assigned a disjoint quarter of the training set (8 000 samples) so that negatives span a range of policy maturity levels. These form $\mathcal{D}^- = \{(\tau^-, 0)\}$.

Critic SFT. The critic D_ϕ (Qwen2.5-1.5B-Instruct) is pre-trained on $\mathcal{D}_{\text{init}} = \mathcal{D}^+ \cup \mathcal{D}^-$ via binary cross-entropy:

$$\mathcal{L}_{\text{init}}(\phi) = -\frac{1}{|\mathcal{D}_{\text{init}}|} \sum_{(\tau,y)\in\mathcal{D}_{\text{init}}} \left[y \log \sigma(s_\phi(\tau)) + (1-y) \log(1 - \sigma(s_\phi(\tau))) \right], \tag{14}$$

where y=1 for teacher traces and y=0 for policy traces. Optimization uses AdamW at learning rate 2×10^{-5} for 200 steps on a balanced subsample.

B.2 Stage 2: Frozen-Critic Policy RL

The initialized critic is *frozen* (ϕ fixed). The policy starts from RL-MemAgent-7B at step 110 ($\approx 84.4\%$ HotpotQA dev accuracy) and is trained with GRPO on the 32K HotpotQA training set using the combined reward (Eq. 10, λ=0.1). At each step the critic provides one forward pass per trajectory as a static memory-trace signal—no critic gradient updates occur. Infrastructure: 2 nodes (2×8 B200) for actor rollout+update via FSDP, 2 nodes (2×8 B200) for async critic inference (4×8 B200 total).

B.3 Stage 3: Adaptive-Critic Online Co-training

All policy hyperparameters are identical to Stage 2. In addition, every Δ=1 policy steps, D_ϕ is updated for S=10 gradient steps on a balanced sample of current-batch positive and negative traces. Rolling pools of 2 048 examples per class are maintained, but the update routine receives current-batch samples. The labels are the combined-reward within-group labels of Eq. 12 and the critic objective of Eq. 13. The critic optimizer is AdamW with learning rate 2×10^{-5} and gradient clipping at 1.0.

B.4 Stage 4: Curriculum Expansion to DocQA-RL-1.6K

After Stage 3 converges on HotpotQA, training continues on DocQA-RL-1.6K (Wan et al., 2025), comprising 1 600 long-context document QA problems across three domains: (i) *mathematical reasoning*—600 DocMath problems over financial reports; (ii) *logical reasoning*—600 multi-choice questions over legal, financial, insurance, and production documents; (iii) *multi-hop reasoning*—200 MultiHopRAG and 200 MuSiQue examples. Training follows the same protocol as Stage 3 (Adaptive-Critic, GRPO, G=16, λ=0.1), with the critic continuing to receive online within-group labels from the new task distribution.

B.5 Training Algorithm

Algorithm 1 gives the Stage 3 loop in detail. Stage 2 is identical with the critic-update block omitted.

C Additional Analysis

Critic training loss. Figure 4 reports the critic's training loss on online labels. It measures fit to the training targets rather than independently assessed memory quality.

Memory-update length. Figure 5 shows mean memory-update length during Stages 2 and 3. Length generally increases over each run, with substantial fluctuations during Stage 3. These curves describe writing length rather than evidence retention.

Algorithm 1 MemAgent-Critic: Adaptive-Critic training loop (Stage 3). Stage 2 omits the critic-update block.

Require: Policy π_θ, critic D_ϕ, reference policy π_{ref}, training data $\mathcal{T}$, group size G, critic coefficient λ, update steps S, update frequency Δ

for each training step t **do**
  Sample a batch $\{(D_i, q_i, a_i^*)\}$ from $\mathcal{T}$
  for each (i, g) with $g = 1, \ldots, G$ **do**
    Generate rollout $\tau_{i,g}$ via policy
    Compute $R_{\text{match},i,g}$
    $s_{i,g} \leftarrow D_\phi(\tau_{i,g})$ (Eq. 8)
  end for
  Batch-normalize s to $\tilde{s}$ over all scored trajectories
  For each (i, g), set $R_{i,g} \leftarrow R_{\text{match},i,g} + \lambda \tilde{s}_{i,g}$
  Compute $\hat{A}^{\text{combined}}_{i,g}$ from $R_{i,g}$ (Eq. 11)
  Use positive/negative signs of $\hat{A}^{\text{combined}}$ to label all collected traces
  Add detached positive/negative examples to the corresponding replay pools
  Retain nonzero sparse-reward-variance groups for actor updates
  if $t \bmod \Delta = 0$ **then** *// Critic update (Stage 3 only)*
    Sample balanced $\mathcal{B}$ from current-batch positive and negative traces
    for $s = 1, \ldots, S$ **do**
      Update ϕ with $\nabla_\phi \mathcal{L}_{\text{critic}}(\phi; \mathcal{B})$
    end for
  end if
  Update θ with GRPO/DAPO using $\hat{A}^{\text{combined}}$ on retained groups
end for

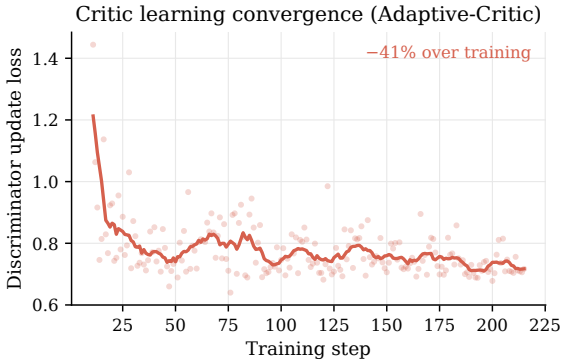
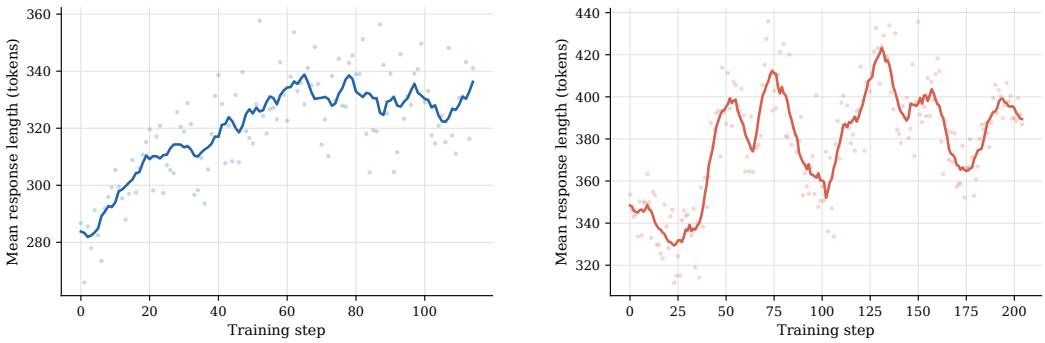


Figure 4: Critic update loss during Adaptive-Critic training. The curve measures training fit to online labels, not held-out classification accuracy or human-rated memory quality.

Removing the critic during curriculum expansion. A continuation from the Adaptive-Critic checkpoint removes the critic reward during the Stage 4 curriculum. HotpotQA dev accuracy remains at 91–93%, while LongBench-QA performance peaks shortly after resuming and then declines. Figure 6 reports memory-update length during this continuation.



(a) Stage 2: Frozen-Critic.

(b) Stage 3: Adaptive-Critic.

Figure 5: Mean memory-update response length (tokens) during training. Scatter points are raw per-step values; solid curves are smoothed.

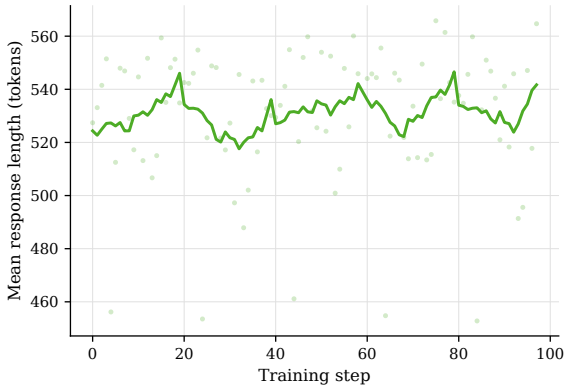


Figure 6: Mean memory-update response length during Stage 4 (No-Critic, curriculum expansion).

D PROMPT TEMPLATES

Policy (generator) prompt. At each memory-update step k, the policy receives the following prompt, which includes the question, the previous memory, and the current document chunk:

Memory-Update Prompt (step k)

```
You are presented with a problem, a section of an article that may
contain the answer to the problem, and a previous memory.  Please
read the provided section carefully and update the memory with the
new information that helps to answer the problem.  Be sure to
retain all relevant details from the previous memory while adding
any new, useful information.
<problem>
{question}
</problem>
<memory>
{previous memory m̂_{k-1}}
</memory>
<section>
{document chunk c^k}
</section>
Updated memory:
```

Figure 7: Policy memory-update prompt at step k. The policy produces an updated memory $\hat{m}_k$ by integrating the new chunk c^k with the previous memory $\hat{m}_{k-1}$.

Final answer prompt. After all K chunks are processed, the memory-update template is replaced by the following answer prompt, which asks the policy to produce the final answer from the accumulated memory $\hat{m}_K$:

Final Answer Prompt (step $K+1$)

```
You are presented with a problem and a previous memory.  Please
answer the problem based on the previous memory and put the answer
in \boxed{}.
<problem>
{question}
</problem>
<memory>
{final memory m̂_K}
</memory>
Your answer:
```

Figure 8: Final answer prompt at step $K+1$. The policy generates the answer from the accumulated memory $\hat{m}_K$; the answer is extracted from the `\boxed{}` environment and compared against the ground-truth answer a^* for reward computation.

Quality critic prompt. The critic receives the question and the complete sequence of memory updates (all K steps) as a single prompt, and outputs a single binary judgment:

Quality Critic Prompt

```
You are evaluating the memory management strategy of an agent
solving a multi-hop question answering task.
<question>
{question}
</question>
The agent progressively updated its memory after reading each
section:
step0:  {m̂_1}
step1:  {m̂_2}
⋮
stepK:  {m̂_K}
Compared with the average agent for this type of task, is this
memory management strategy above average or below average in
helping answer the question?  Answer with either "Good" (above
average) or "Bad" (below average).
```

Figure 9: Quality critic prompt. The critic receives the full memory trace across all K steps and outputs a single token: `Good` or `Bad`. The critic score $s_{i,g}$ is the log-odds $\text{logit}(\text{Good}) - \text{logit}(\text{Bad})$ at the final token position (Eq. 8).

E Additional Experiments and Controls

E.1 Random-Score Control

The random-score control replaces the learned critic's scores with random values in $\{-1, +1\}$ and uses GRPO policy training. The available measurements cover steps 0–62, rather than a completed 200-step run. The initial policy has 77.34375% HotpotQA accuracy; the reported checkpoints are spaced ten steps apart, with the observed low point at step 62 also included (Table 7). The decrease to 70.31250% is 7.03125 percentage points, or 9.09% relative to initialization. This random signal does not improve the policy in the observed run. It is not a shuffled-score or critic-logit-distribution-matched control, and its initial accuracy differs from the main Frozen-Critic run, so it does not by itself identify the source of every gain from the learned critic.

Table 7: Random-score control on HotpotQA. Relative changes use the unrounded reported accuracies.

Checkpoint	Accuracy (%)	Relative change (%)
Initial	77.34375	—
10	76.56250	−1.01
20	76.56250	−1.01
30	75.78125	−2.02
40	73.43750	−5.05
50	75.78125	−2.02
60	74.21875	−4.04
62	70.31250	−9.09

E.2 Training Cost and Comparison Scope

On the reported infrastructure, Adaptive-Critic takes 1,738.04 seconds per policy step, compared with 1,655.13 for Frozen-Critic and 408.54 for No-Critic. Online updating therefore adds 5.01% relative to the frozen critic, while the adaptive variant's total step time is approximately 4.25 times that of No-Critic. These measurements distinguish the incremental cost of updating the critic from the cost of adding a critic. The runs have different lengths and do not constitute equal-compute comparisons;

teacher generation and critic initialization are not included in the reported accounting. Table 8 gives the recorded run durations.

Table 8: Reported policy-training cost. Runs have different lengths; wall-clock values refer to the stated run/checkpoint durations.

Variant	Seconds/step	Hours	Recorded extent
No-Critic	408.54	24.29	214 steps
Frozen-Critic	1,655.13	51.95	checkpoint 113
Adaptive-Critic	1,738.04	≈93.18	checkpoint 193

E.3 Case Study: Memory Revision

We examine one recorded HotpotQA case comparing MemAgent-Critic with RL-MemAgent-7B. Both models process the same document in 5,000-token chunks and update their memory using deterministic decoding. This example illustrates a difference in memory revision, without establishing its frequency or attribution to a particular training component.

The question is "Thirukkalacherry is a village in which Indian town?" and the reference answer is *Tharangambadi*. The first two document chunks contain distracting place names but do not identify the village's location. Despite this absence of direct evidence, RL-MemAgent-7B infers that the village is in Kumbakonam because that town appears in the text. Its next memory update repeats this unsupported association. MemAgent-Critic, in contrast, does not commit to the distracting locations during these first two chunks.

The third chunk supplies direct evidence:

> "Thirukkalachery is a small village in Mayiladuthurai District, Tranquebar (Tharangambadi) Taluk, Tamil Nadu, India."

The baseline copies this sentence into its memory, yet retains its earlier conclusion that Thirukkalacherry is in Kumbakonam. Thus, its failure is not simply failure to encounter or store the relevant sentence: the updated memory contains the correct evidence alongside an incompatible prior inference. The final answer is `Kumbakonam`, recorded as incorrect.

MemAgent-Critic instead updates its core memory with the newly observed village–location relation. Its final answer includes `Tranquebar (Tharangambadi) Taluk` and is recorded as correct. In this example, MemAgent-Critic revises its memory after later evidence appears, whereas the baseline retains an earlier unsupported association. The MemAgent-Critic answer also includes additional nearby locations.