FOLDACT: STABLE AND EFFICIENT TRAINING FOR CONTEXT-FOLDING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon reinforcement learning (RL) for large language model agents operates under fixed context windows while interaction histories keep growing. Context folding summarizes this history, but end-to-end folding changes the training problem because the same policy writes summaries and later acts from them. Summary tokens are policy-written memory–tokens that become future visible state–so a unified token-level objective creates credit dilution, summary-induced replay mismatch, and expensive per-turn replay. We introduce **FoldAct**[1], an RL training method for policy-written memory in long-horizon agents. FoldAct combines separated summary/action objectives, a consistency loss that stabilizes behavior after context folding, and selective segment training. At inference, the same policy writes summaries and produces subsequent actions, without a teacher summarizer, external memory model, or change to the deployed architecture. Our controlled claims are feasibility, update-side cost, and stability: FoldAct makes folded training feasible with a 5.19× speedup and stabilizes response length.

1 INTRODUCTION

Long-horizon RL for LLM agents supports multi-step reasoning, retrieval, and tool use (Guo et al., 2025; Chen et al., 2025b; Xi et al., 2025; Feng et al., 2025), but fixed context windows make policy-update replay over long contexts expensive as observations accumulate (Lu et al., 2025; Sun et al., 2025; Tan et al., 2025). **Context folding** bounds the visible history by generating summaries that replace earlier observations and actions. Some systems train this folding behavior end-to-end (Sun et al., 2025; Ye et al., 2025), while others rely on separate summarization or feedback models (Wan et al., 2025; Sun et al., 2025; Wu et al., 2025c). These approaches reduce context length, but they do not fully address the optimization issue that arises when summaries are generated by the same policy that later conditions on them.

We use SUPO (Lu et al., 2025) as the closest executable baseline because it also trains inline summaries with end-to-end RL; the controlled comparison fixes the backbone, SFT checkpoint, data, tools, rollout interface, and evaluator. FoldGRPO (Sun et al., 2025) provides the closest complementary RL design: it uses branch-return folding and process-reward rollouts, whereas FoldAct keeps rollout semantics fixed and separates summary/action credit with a consistency loss against the full context.

End-to-end folding creates a distinct optimization issue: the same policy writes summaries and later acts from them. Unlike ordinary reasoning tokens, a summary replaces part of the working history and becomes future visible state. Summary tokens are therefore **summary-write actions**: low-frequency in token space but high-impact in transition space. FoldAct targets this trainability bottleneck, while task metrics verify that the stability and efficiency gains preserve answer quality. This role of summary tokens introduces three structural challenges:

C1: Credit Assignment. Token-mean objectives compute a unified gradient over summary and action tokens. This causes **gradient dilution**: summary tokens may occupy a small fraction of the response while controlling the information available to many later actions.

C2: Summary-Induced Replay Mismatch. PPO updates replay folded contexts generated by the rollout summary policy. When summary-writing behavior drifts, later action learning is conditioned

[1] https://anonymous.4open.science/r/FoldAct-BEC2

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.

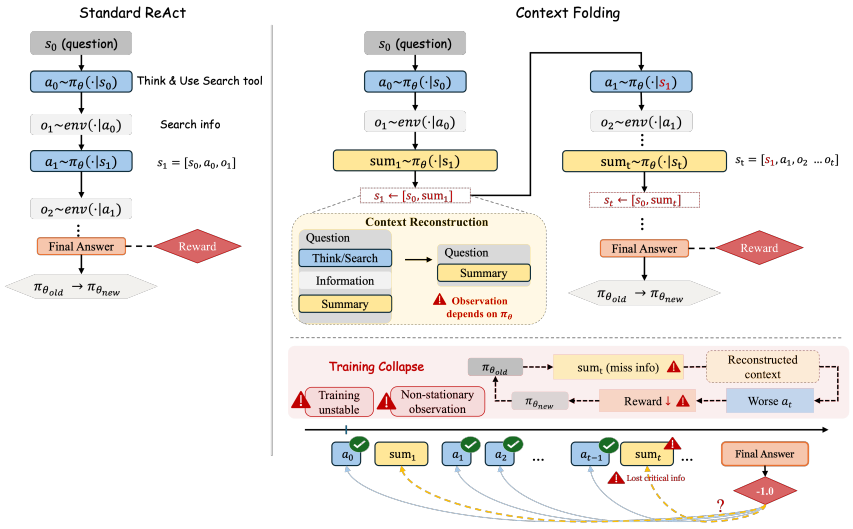

Figure 1: Recent-history replay treats collected observations as fixed conditioning contexts. In context folding, summaries $\text{sum}_t \sim \pi_\theta(\cdot \mid s_t)$ are generated by the policy and written into future visible contexts; summary-policy drift can therefore change later folded states and create replay mismatch.

on folded states that may omit or distort recoverable evidence, and the error can compound over long horizons (illustrated in Figure 1, *Right Bottom*).

C3: Per-Turn Replay Cost. Under the same fixed context budget $L_{\max}$, replay over all turns still requires many long contexts. With folding, each turn can contain a different summary and retained history, so the turns generally cannot be packed as prefixes of one causal sequence and instead require separate model passes.

Our contributions are:

1. We identify summaries in end-to-end folding as **policy-written memory** and isolate three resulting training challenges: credit dilution, summary-induced replay mismatch, and per-turn replay cost.
2. We propose **FoldAct** (see Figure 2), a training-side method with separated summary/action losses, a consistency loss against the full context $h_t = \text{Window}_{L_{\max}}(h_{0:t})$, and selective segment training; rollout semantics and the deployed model remain unchanged.
3. We provide cost, stability, ablation, and task evidence across Local RAG and Web Search: FoldAct makes folded training feasible with a 5.19× speedup, stabilizes response length, and preserves task quality in Local RAG and Web Search checks.

2 Related Work

Long-Horizon RL Agents. LLM agents commonly follow a ReAct-style loop that interleaves reasoning, tool actions, and observations (Yao et al., 2022). Recent RL systems extend this pattern to search, tool use, and web navigation (Xi et al., 2025; Jin et al., 2025; Chen et al., 2025c; Gao et al., 2025; Zheng et al., 2025; Wu et al., 2025a; Li et al., 2025; Feng et al., 2025). However, these agents typically retain evidence by appending observations and model replies; as trajectories grow, append-only histories make policy-update replay costly, stress fixed context windows (Lu et al., 2025; Sun et al., 2025; Tan et al., 2025), and can degrade behavior through context rot (Anthropic, 2025). WebSailor and ReSum improve web-search agents through web-agent training or context summarization (Li et al., 2025; Wu et al., 2025c), whereas FoldAct targets the policy-update replay cost and stability issue caused by policy-written folded contexts.

Memory and Context Management. External-memory methods preserve history by learning to write, retrieve, and update information outside the prompt (Yan et al., 2025; Yuan et al., 2025; Wang et al., 2025); context-folding methods instead compress the active trajectory itself. Representative strategies are threshold-triggered summary reset with end-to-end RL (SUPO (Lu et al., 2025) and ReSum (Wu et al., 2025c)), proactive multi-scale folding with SFT (AgentFold (Ye et al., 2025)), branch-return folding with process-reward RL (FoldGRPO (Sun et al., 2025)), optimized observa-

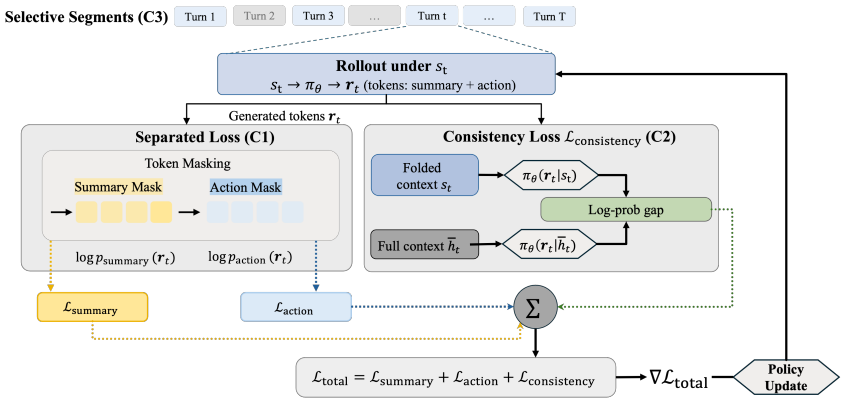

Figure 2: FoldAct addresses policy-written memory with separated summary/action losses, a consistency loss, and selective segment training.

tion/history compression with a distilled compressor (ACON (Kang et al., 2025)), and iterative state reconstruction (Chen et al., 2025a). FoldAct keeps state inside the active prompt and combines separated summary/action losses, a consistency loss, and selective segment training.

3 Method

3.1 Problem Formulation

We model a ReAct-style LLM agent under a fixed context budget $L_{\max}$. Let $h_{0:t}$ be the raw interaction history, s_t the folded visible context with $|s_t| \le L_{\max}$, and sum_t the summary write. In context folding, the policy writes sum_t to replace part of the raw history and insert it into future visible contexts. We use a_t and o_{t+1} for the action and resulting observation, τ for a rollout trajectory, $\mathbf{r}_t$ for the generated response at turn t, and $r_{t,i}$ for its i-th token. Operationally, a folded rollout alternates between environment interaction and summary writing:

$$h_0 \to a_0 \to o_1 \to \cdots \to \mathrm{sum}_t \Rightarrow s_{t+1} = \mathrm{Fold}(h_{0:t}, \mathrm{sum}_t) \to a_{t+1} \to o_{t+2} \to \cdots .$$

The key distinction is that sum_t is generated by the policy and then written into future state. Let x denote the augmented agent state, $K_{\pi,i}$ the turn-i transition kernel, and d_t^π the turn-t state distribution under policy π; β denotes the rollout policy. Let u be the policy output at a turn and $A_t^\beta(x,u)$ its advantage under β. To state the policy-improvement consequence, define the true-occupancy term $\Delta I_\beta(\pi) = \sum_t \mathbb{E}_{x\sim d_t^\pi, u\sim\pi}[A_t^\beta(x,u)]$ and its rollout-occupancy surrogate $L_\beta(\pi) = \sum_t \mathbb{E}_{x\sim d_t^\beta, u\sim\pi}[A_t^\beta(x,u)]$. The following result makes the resulting replay mismatch explicit.

Lemma 3.1 (Folded-state occupancy drift). *Assume $d_0^\pi = d_0^\beta$ and let $\epsilon_i = \sup_x D_{\mathrm{TV}}(K_{\pi,i}(\cdot \mid x), K_{\beta,i}(\cdot \mid x))$, where D_{TV} is total-variation distance. Then*

$$D_{\mathrm{TV}}(d_t^\pi, d_t^\beta) \le \sum_{i=0}^{t-1} \epsilon_i. \tag{1}$$

Consequently, if $|A_t^\beta| \le A_{\max}$ over horizon H and $\epsilon_i \le \epsilon$, then $|\Delta I_\beta(\pi) - L_\beta(\pi)| \le A_{\max} H(H-1)\epsilon = O(A_{\max} H^2 \epsilon)$.

Appendix E defines the augmented kernel and proves Lemma 3.1. Summary-policy drift contributes directly to ϵ_i because a changed write can hide recoverable evidence from all later replayed decisions. A second issue is credit dilution: summary tokens are only roughly 20% of optimized response tokens in our rollouts, yet their writes condition subsequent turns. FoldAct is designed around this summary-write structure: separated losses address token-level credit dilution, the consistency loss limits summary-induced folded-state drift, and selective segment training reduces the cost of replaying every folded turn.

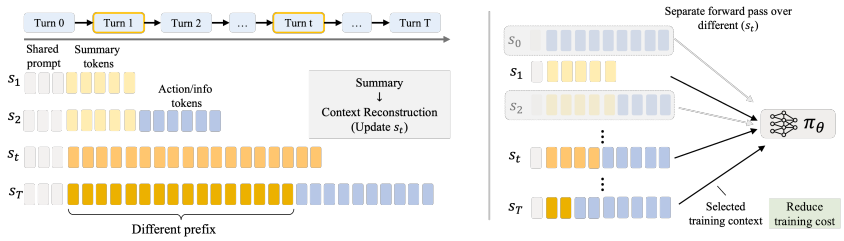

Figure 3: Summary-induced context variation and selective segment training. Summaries replace raw history, so retained turns observe distinct contexts s_t. Because these contexts generally cannot share one causal prefix, sampling turns reduces the number of per-turn model passes without changing rollout behavior.

3.2 Separated Loss Computation

Summary tokens form the write sum_t that determines later visible state, yet standard PPO assigns a single advantage to all tokens. This creates a token-frequency–causal-influence mismatch: for example, a unified average over 20 summary and 200 action tokens gives state writing only 9.1% of the token mass even though those 20 tokens condition many future decisions. At turn t, we partition the response $\mathbf{r}_t$ into summary spans and all remaining assistant tokens; for compact notation, we call the latter action tokens. We therefore use category-specific advantages and independently normalized objectives, with a binary summary-token mask:

$$\mathbf{c}_t^{\text{sum}}[i] = \begin{cases} 1, & \text{if } r_{t,i} \text{ is a summary token,} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The action-token mask is $\mathbf{c}_t^{\text{act}} = 1 - \mathbf{c}_t^{\text{sum}}$. For category $c \in \{\text{sum}, \text{act}\}$, $\mathbf{c}_t^{(c)}$ is the corresponding mask. For each token category $c \in \{\text{sum}, \text{act}\}$, we compute the category-wise clipped PPO objective using advantage estimates $\hat{A}_t^c$:

$$\mathcal{J}_c = \mathbb{E}_{\tau \sim \mathcal{D}_{\text{rollout}}} \left[\sum_t \frac{1}{N_t^c} \sum_i \mathbf{c}_t^{(c)}[i] \, \min\Big(\rho_{t,i} \, \hat{A}_t^c, \; \text{clip}\big(\rho_{t,i}, 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}}\big) \, \hat{A}_t^c \Big) \right], \tag{3}$$

where $N_t^c = \max(1, \sum_i \mathbf{c}_t^{(c)}[i])$ counts optimized category-c tokens, $\mathcal{D}_{\text{rollout}}$ is the rollout buffer, and ϵ_{clip} is the PPO clipping threshold. The importance ratio compares the updated policy π_θ with the frozen rollout policy $\pi_{\theta_{\text{old}}}$:

$$\rho_{t,i} = \frac{\pi_\theta(r_{t,i} \mid \mathbf{r}_{t,<i}, s_t)}{\pi_{\theta_{\text{old}}}(r_{t,i} \mid \mathbf{r}_{t,<i}, s_t)}, \tag{4}$$

where $\hat{A}_t^c$ uses category-specific rewards. Action tokens use the binary task reward r_{task} (Jin et al., 2025; Sun et al., 2025); summary tokens use $r_{\text{summary}}(t)$ defined below. Advantages are computed with standard estimators (Schulman et al., 2015). We use $\mathcal{J}$ for PPO maximization objectives and $\mathcal{L}$ for minimization losses; the masks only separate which tokens enter the summary and action objectives before converting them to losses in Eq. 6.

Summary Reward Design. Summary tokens receive lightweight shaping rewards for when to summarize and what answer-relevant evidence to retain (Figure 4). These rewards shape operational folding behavior rather than certifying full semantic correctness. We use $r_{\text{summary}}(t)$ defined as:

$$\begin{aligned} r_{\text{summary}}(t) &= r_{\text{unnec}}(t) + r_{\text{retention}}(t), \\ r_{\text{unnec}}(t) &= \begin{cases} -0.2, & \text{if } \mathrm{sum}_t \text{ is written but no observation is folded at turn } t, \\ 0, & \text{otherwise,} \end{cases} \\ r_{\text{retention}}(t) &= \begin{cases} +0.2, & \text{if } r_{\text{task}} = 1 \text{ and } \mathrm{match}(a^\star, \mathrm{sum}_t) = 1, \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

The unnecessary-summary penalty applies only when a summary write does not replace an observation in the next visible context. The predicate $\mathrm{match}(a^\star, \mathrm{sum}_t)$ uses the same answer normalization as

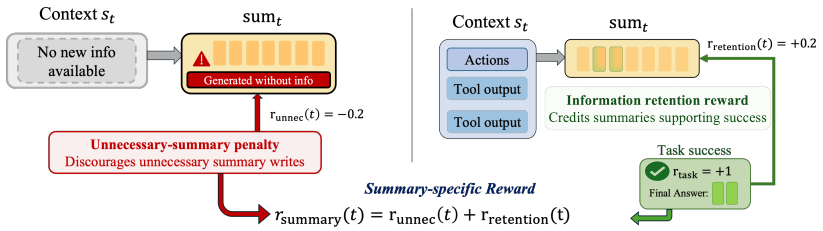

Figure 4: Summary reward. The unnecessary-summary penalty discourages writes that do not fold an observation, while the retention reward encourages answer-relevant evidence when the task succeeds.

evaluation. Because $r_{\text{retention}}$ is gated by terminal success, it provides task-linked process shaping rather than replacing the task reward. We use no learned summary scorer and fix ± 0.2 across experiments, following lightweight process-reward practice (Jin et al., 2025; Chen et al., 2025c; Sun et al., 2025). The corresponding minimization losses:

$$\mathcal{L}_{\text{summary}} = -\mathcal{J}_{\text{summary}}, \qquad \mathcal{L}_{\text{action}} = -\mathcal{J}_{\text{action}}, \qquad \mathcal{L}_{\text{separated}} = \mathcal{L}_{\text{summary}} + \mathcal{L}_{\text{action}}, \tag{6}$$

3.3 Consistency Loss

During RL updates, the model replays folded contexts written by the rollout summary policy. If summary-writing behavior drifts, later action learning may condition on folded states that omit or distort recoverable evidence. We therefore regularize s_t against the full context $\bar{h}_t = \text{Window}_{L_{\max}}(h_{0:t})$ using a squared sampled log-probability gap on rollout-generated optimized tokens $\mathcal{I}_t$:

$$\mathcal{L}_{\text{consistency}} = \mathbb{E}_{\tau \sim \mathcal{D}_{\text{rollout}}} \left[\sum_t \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} \left(\log \pi_\theta(r_{t,i} \mid \mathbf{r}_{t,<i}, s_t) - \text{sg}\left[\log \pi_\theta(r_{t,i} \mid \mathbf{r}_{t,<i}, \bar{h}_t) \right] \right)^2 \right] \tag{7}$$

Here $\text{sg}[\cdot]$ denotes stop-gradient. The model, sampled token, and response prefix are held fixed; only the conditioning state changes from the folded context to the full context. The loss is therefore a sampled control on folding-induced behavioral drift, rather than generic output distillation. It matches only rollout-token log probabilities rather than full-vocabulary distributions, avoiding dense-logit storage and keeping this branch forward-only. Appendix E gives the formal drift view. The total loss is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{summary}} + \mathcal{L}_{\text{action}} + \mathcal{L}_{\text{consistency}} = \mathcal{L}_{\text{separated}} + \mathcal{L}_{\text{consistency}} \tag{8}$$

We use unit weight on $\mathcal{L}_{\text{consistency}}$ in the main experiments; η_{cons} below refers only to how often this forward-only branch is computed.

3.4 Selective Segment Training

As illustrated in Figure 3, summary generation changes s_t across turns, so training every turn requires a separate replay pass for each distinct folded context. FoldAct drops replay losses, not rollout turns: all turns are executed, and unselected summaries still affect future contexts. During the policy update, we sample $z_t \sim \text{Bernoulli}(q)$, where $q = 1 - p_{\text{drop}}$, and optimize $\sum_t z_t \mathcal{L}_t$, where $\mathcal{L}_t$ denotes the per-turn contribution to $\mathcal{L}_{\text{total}}$ in Eq. 8. In the main experiments, $p_{\text{drop}} = 0.5$ ($q = 0.5$), reducing expected replayed contexts from T to qT while preserving the exact rollout context for retained turns.

Selective segment training is therefore different from trajectory truncation, sliding-window replay, or subsampling tasks. Rewards and advantages are computed on the complete rollout, later selected turns condition on summaries written by earlier unselected turns, and $z_t = 0$ only masks the optimization loss for that segment. An equal-cost sliding window can reduce length by training every turn under a truncated conditioning state, but then nonzero gradients may be computed after evidence has been removed. Selective replay instead gives either the exact rollout-context gradient for a retained turn or zero gradient for an omitted turn, preserving the folded-state distribution for every optimized segment. At inference time the mask disappears entirely: the same policy backbone writes summaries and actions at every turn, with no external summarizer, retrieval memory, or additional model call. Appendix F gives the full complexity derivation and a concrete six-turn example.

Table 1: Training cost comparison. Time/Step is update-side replay time; Peak Mem. is allocated memory over 16×NVIDIA L20 workers. Recent-history truncation retains the latest $L_{\max}$ tokens and replays every turn under the same model, update protocol, and hardware.

Training Strategy	Time/Step (s)	Peak Mem. (GB)	Mem. Reduction	Speedup
Recent-history truncation (all-turn replay)	∼4846.72	>441.04 (OOM)	–	–
SUPO	∼1212	∼419	∼5.0%	∼4.0×
FoldAct ($p_{\text{drop}} = 0.5$, w. cons.)	∼933.70	405.85	≥8.0%	5.19×
FoldAct ($p_{\text{drop}} = 0.5$, w/o cons.)	∼97.75	84.90	≥80.7%	49.6×

4 EXPERIMENTS

Scenario Scope. Local RAG and Web Search test complementary regimes. Local RAG uses fixed corpora and structured evidence, enabling controlled cost, stability, and objective ablations. Web Search produces longer, noisier tool traces with heterogeneous snippets and navigation artifacts, testing whether the same training recipe transfers when more irrelevant history must be folded.

4.1 EXPERIMENTAL SETUP

Tasks and Baselines. We train FOLDACT from Qwen-2.5-7B-Instruct (Qwen et al., 2024). Local RAG evaluates HotpotQA (Yang et al., 2018) and PopQA (Mallen et al., 2022). We reproduce SUPO (Lu et al., 2025) training from Qwen-2.5-7B-Instruct (Qwen et al., 2024). Web Search evaluates WebWalker (Wu et al., 2025b), GAIA (Mialon et al., 2023), BrowseComp (Wei et al., 2025), and XBench-DeepSearch[2] against open and API references (OpenAI, 2025; Team et al., 2025; Wu et al., 2025c; Alibaba-NLP, 2025); ASEARCHER-WEB-QWQ (32B) (inclusionAI, 2025) is only a larger-model reference.

Training Details. FoldAct first uses SFT to initialize two inline memory tags: `<think_summary>` for compact planning state and `<information_summary>` for factual, answer-bearing evidence. SFT uses Qwen3-30B-A3B-Instruct summaries on 200 annotated Local RAG trajectories from ASearcher data[3] and Web Search trajectories from WebExplorer data[4]; it teaches the tag format rather than discovering folding from scratch. During RL and inference, every summary is generated by the deployed 7B policy; the 30B teacher is not called. FoldAct uses $p_{\text{drop}} = 0.5$ ($q = 0.5$), where unselected turns drop replay loss only, not rollout.

Controlled Comparisons. All controlled rows fix the Qwen-2.5-7B-Instruct backbone, SFT checkpoint, training data, tools, rollout interface, and evaluator. Recent-history truncation and sliding-window replay are controlled heuristics, not aliases for SUPO, AgentFold, or other learned context-management methods.

Inference Protocol. At evaluation time, FoldAct runs a single 7B policy: summary tags are generated inline and folded into later contexts; no teacher summarizer, retrieval memory, or auxiliary compression model is used. Thus every FoldAct row uses the same inference interface; p_{drop} only controls how many folded segments are replayed during training.

4.2 MAIN RESULTS

4.2.1 TRAINING FEASIBILITY AND COST ANALYSIS

Context folding requires training on multiple reconstructed trajectory segments, so every distinct folded context s_t can require a separate replay pass. Table 1 compares strategies under the same context setting. The recent-history truncation baseline retains the latest $L_{\max}$ tokens and replays this full context at every turn; it exceeds 441.04 GB peak memory. FoldAct with $p_{\text{drop}} = 0.5$ ($q = 0.5$) replays selected compressed contexts and is feasible; with the consistency loss it keeps a 5.19× speedup, while removing this loss gives 49.6× speedup but collapses in Figure 5.

[2] https://huggingface.co/datasets/xbench/DeepSearch

[3] https://huggingface.co/datasets/inclusionAI/ASearcher-train-data

[4] https://huggingface.co/datasets/hkust-nlp/WebExplorer-QA

The measured speedup follows the fixed-budget cost model. Let $\bar{L}_t = \min(L_0 + \sum_{i<t}(g_i + o_i), L_{\max})$ be the full-context length, where L_0 is the initial prompt length, g_i and o_i are generated-token and observation-token counts, and $\bar{h}_t = \mathrm{Window}_{L_{\max}}(h_{0:t})$ retains the latest $L_{\max}$ tokens. Let $K_t \le \bar{L}_t$ be the folded context length, y_t the optimized response length, and $\Phi(n) = c_{\mathrm{lin}} n + c_{\mathrm{attn}} n^2$ the model-pass cost, with linear and attention coefficients $c_{\mathrm{lin}}, c_{\mathrm{attn}}$. With batch size B, sampled responses per prompt G, and PPO replay multiplier κ, we define the total update-side cost of replaying every turn under $\bar{h}_t$ as

$$C_{\mathrm{trunc}} = \kappa BG \sum_t \Phi(\bar{L}_t + y_t),$$

while FoldAct's update-side cost is

$$C_{\mathrm{FoldAct}}(z) = BG \sum_t z_t \left[\kappa \Phi(K_t + y_t) + \eta_{\mathrm{cons}} \Phi(\bar{L}_t + y_t)\right].$$

The second term is the forward-only consistency branch in Eq. 7; it computes full-context log probabilities only on rollout-generated tokens.

Corollary 4.1 (Cost-model agreement)**.** *If attention dominates, $K_t \approx \alpha_{\mathrm{fold}} \bar{L}_t$ with folding ratio $\alpha_{\mathrm{fold}} = K_t / \bar{L}_t$, response lengths are lower order, and $z_t \sim \mathrm{Bernoulli}(q)$, then the normalized expected update cost relative to recent-history truncation is*

$$R := \frac{\mathbb{E}_z[C_{\mathrm{FoldAct}}(z)]}{C_{\mathrm{trunc}}} \approx q\alpha_{\mathrm{fold}}^2 + q\frac{\eta_{\mathrm{cons}}}{\kappa}. \tag{9}$$

For $q = 0.5$, measured $\alpha_{\mathrm{fold}} \approx 0.25$, $\eta_{\mathrm{cons}} = 1$, and $\kappa \approx 3$, Eq. 9 predicts $R \approx 0.198$, or a 5.05× speedup, within 2.7% of the measured 5.19×.

Appendix F derives Corollary 4.1. Its replay and consistency terms are approximately 0.031 and 0.167, so the consistency branch accounts for about 84% of the remaining normalized update cost. The measured 5.19× is the stable net speedup after paying this cost, not the cheaper unstable ablation; all added complexity is training-side.

4.2.2 Training Stability Analysis

Figure 5 shows the impact of the consistency loss on response length during training. With this loss, response length stabilizes around 1200–1400 tokens. Without it, response length grows rapidly after step 50, producing repetitive outputs and training collapse. This supports its role as a stabilizer for per-turn replay; we interpret it as reducing behavioral drift between folded and full contexts.

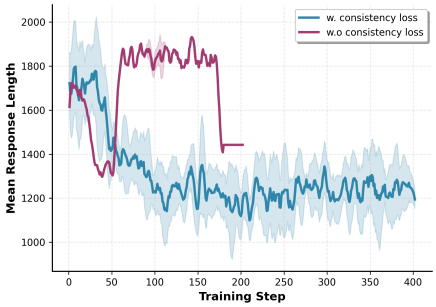

Figure 5: Response length with and without the consistency loss.

Appendix D gives a qualitative example: the consistency loss yields a concise, evidence-focused summary, whereas the ablated run accumulates redundant process traces.

4.2.3 Ablation Studies

Table 2 separates the objective components: the separated loss improves credit assignment and response length, while removing the consistency loss causes collapse, making its task scores pre-collapse diagnostics. Replay retention is not monotonic across tasks; we therefore use 50% as the cost-quality compromise.

Retaining all turns changes GAIA from 45.0 to 46.3, supporting $q = 0.5$ as the efficient setting rather than an accuracy improvement. Removing the success-gated retention bonus remains stable but changes GAIA from 45.0 to 43.6 and HotpotQA from 38.5/29.5 to 37.4/27.5. This ablation measures the downstream contribution of the lightweight retention signal, while the consistency loss provides the observed training-stability effect.

Panel A separates the two objective roles: token-type separation strengthens learning on sparse summary writes, whereas the consistency loss stabilizes behavior after those writes become future

Table 2: Ablations of training objectives (A) and replay retention (B). Scores without the consistency loss are pre-collapse diagnostics; bold and underline mark the best and second-best Panel B scores.

A. Training-objective ablations

Variant	Change	HotpotQA F1/EM	PopQA F1/EM	Avg. Resp. Len.
Full FoldAct	–	38.5/29.5	32.9/29.0	∼1300 tok
w/o separated loss	unified summary/action gradients	36.2/27.8	32.1/28.1	∼1600 tok
w/o consistency loss	remove consistency loss	34.9/26.7	33.3/29.2	unstable

B. Selective replay retention ablations

Drop/Retain (p/q)	Replayed turns	HotpotQA F1/EM	PopQA F1/EM	Setting
$0.8/0.2$	20%	37.6/29.5	**36.7**/32.0	aggressive
$0.5/0.5$	50%	<u>38.5</u>/29.5	<u>32.9</u>/29.0	default
$0.2/0.8$	80%	**39.2**/30.0	34.7/33.0	costly

context. Panel B shows that replaying more turns does not improve both retrieval tasks uniformly. We therefore use $q = 0.5$ as the default balance between update cost and task quality.

4.2.4 Task Quality Preservation

Table 4 reports controlled Local RAG answer quality. FoldAct reaches 38.5/29.5 on HotpotQA and 32.9/29.0 on PopQA, showing that the stability and cost gains preserve task performance on structured retrieval traces. Local RAG remains useful for controlled ablations and cost analysis, while Web Search tests whether the same training recipe transfers to noisier interaction.

Table 3 reports GAIA Pass@1 across methods. FoldAct's 6.6/3.5-point gains over SFT-only/unified PPO isolate post-SFT learning and the FoldAct objective under the common setup described in Experimental Setup.

The two benchmarks test complementary notions of quality: Local RAG measures answer extraction, whereas GAIA measures terminal success over multi-step tool-use trajectories. Under the controlled Qwen-2.5-7B setup, FoldAct improves over SUPO on both HotpotQA and GAIA, showing that the stability and cost gains preserve task quality. Its 8.2/10.8-point GAIA gains over truncation/sliding-window replay further favor preserving policy-written summaries over shortening replay context. All-turn replay adds 1.3 points but replays every turn, so $q = 0.5$ remains the primary cost–quality setting. Appendix C gives the remaining Web Search results.

Table 3: GAIA Pass@1. † marks reported results; bold and underline mark leading values.

Method	Model	Pass@1
GPT-4.1-mini	API	32.7
MiroThinker†	<u>8B</u>	<u>46.6</u>
SUPO	7B	43.0
ReSum†	7B	42.4
Unified PPO folding	7B	41.5
SFT-only folding	7B	38.4
Recent-history truncation	7B	36.8
Sliding-window replay	7B	34.2
WebSailor†	7B	33.0
FoldAct (ours, $q = 0.5$)	7B	<u>45.0</u>
All-turn FoldAct ($q = 1$)	7B	**46.3**

Table 4: HotpotQA/PopQA F1/EM; bold marks FoldAct.

Method	Hotpot F1/EM	PopQA F1/EM
SearchR1	37.9/21.2	30.9/27.5
ASearcher	41.5/22.7	29.9/25.3
DeepRes.	33.4/16.9	30.6/25.7
ReSearch	37.0/19.2	32.0/27.1
Qwen	32.8/15.7	26.3/21.4
Qwen-Fewshot	36.6/18.8	30.7/24.4
FoldAct w/o Cons.	34.9/26.7	33.3/29.2
FoldAct (ours)	**38.5/29.5**	**32.9/29.0**

Table 3 separates provenance from control: daggered values are source-reported, GPT-4.1-mini is an API scale reference, and SUPO is our reproduction. The controlled Qwen rows share the backbone, initialization, data, tools, and evaluator; external rows contextualize scores, while controlled rows support method-level attribution.

Local RAG uses token F1 and normalized exact match (EM), whereas GAIA uses terminal Pass@1. Because EM requires the complete normalized answer, the HotpotQA gain from 26.8 to 29.5 is not merely greater token overlap. Together, the metrics test structured answer fidelity and end-to-end tool-use completion.

4.2.5 Folding Efficiency Analysis

Table 5 reports mean folded visible length K_t and fold ratio $K_t/\bar{L}_t$ under the same context budget. Local RAG retains 1,892 tokens at a 0.65 ratio; Web Search falls from 0.42 for 1–5 turns to 0.25 beyond 10, showing that folding removes more accumulated history as trajectories grow. Selective replay trains each retained turn under its rollout-visible folded context rather than a sliding-window substitute.

Table 5: Folding efficiency.

Scenario	Len.	Avg Ctx.	Fold Ratio
Local RAG	1–10	1,892	0.65
Web Search	1–5	3,986	0.42
	6–10	5,311	0.31
	> 10	3,211	0.25

Local RAG uses one 1–10-turn bin because its traces are short; the Web Search bins expose the longer tail. Avg. Ctx. is measured after folding and need not increase with turn count: trajectories beyond 10 turns have undergone more folding events. The monotonic fold ratio is the direct cross-bin compression measure.

The savings are complementary: folding shortens redundant retrieval and noisy web histories, while selective replay reduces optimized segments without changing policy-visible state. The measured 5.19× speedup therefore comes from fewer, shorter replay contexts, not a different deployed model.

All-turn recent-history truncation, with cost C_{trunc}, defines the update-side speedup denominator and reaches the memory boundary because it replays every turn. The shorter sliding-window control costs less but deletes more recoverable state; its GAIA gap measures the quality trade-off rather than the speedup denominator.

5 Discussion & Limitations

FoldAct treats policy-written memory as an optimization problem rather than an inference-only compression heuristic. Its components target complementary failure modes: separated objectives prevent sparse summary writes from being diluted by action tokens, consistency stabilizes behavior after folding, and selective replay reduces optimized segments while preserving rollout-visible contexts. Because the components share the same PPO interface, they can be ablated independently, and the efficiency gain does not come from replacing the task with a shorter-context surrogate. FoldAct changes training, not deployment: the same policy writes summaries and produces reasoning, tool calls, and final answers without an auxiliary summarizer or memory model. Local RAG provides controlled mechanism evidence, while Web Search tests transfer to longer, noisier interactions; together, stability, profiling, ablations, and task metrics support improved feasibility and efficiency without material task-quality loss.

The retention signal uses deterministic task validators rather than a learned judge and does not certify unrestricted semantic correctness: it checks answer-string retention on successful trajectories and whether a write actually folds an observation. Its ablation measures shaping utility, while the consistency loss explains the observed stability effect. The analytical cost model abstracts kernel and hardware effects, so we use it only to explain where savings arise and ground efficiency claims in measured update time and peak memory under fixed conditions.

6 Conclusion

This work studies end-to-end context folding, where policy-generated summaries become inputs to later decisions and can propagate credit-assignment errors or behavioral drift. We introduced **FoldAct**, combining separated summary/action objectives, full-context consistency, and selective replay. Across Local RAG and Web Search, FoldAct prevents the response-length collapse observed without consistency and reduces measured update-side replay time by 5.19×, including consistency overhead. It maintains task quality under controlled comparisons while using the same policy at inference, without an auxiliary summarizer or memory model. These results support FoldAct as a stable and efficient training method for policy-written context with an unchanged inference architecture.

AI USE STATEMENT

Generative AI tools assisted with manuscript writing, LaTeX formatting, and coding support for experiments. The research ideas, methodological novelty, experimental design, analysis, and conclusions were developed by the authors. All AI-assisted text and code were reviewed and verified by the authors, who take full responsibility for the final content and claims.

REFERENCES

Alibaba-NLP. Alibaba-NLP/WebSailor-7B. https://huggingface.co/Alibaba-NLP/WebSailor-7B, 2025. Hugging Face model card; Apache-2.0 License.

Anthropic. Effective context engineering for ai agents, September 29 2025. URL https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents. Anthropic Engineering Blog.

Guoxin Chen, Zile Qiao, Xuanzhong Chen, Donglei Yu, Haotian Xu, Wayne Xin Zhao, Ruihua Song, Wenbiao Yin, Huifeng Yin, Liwen Zhang, Kuan Li, Minpeng Liao, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Iterresearch: Rethinking long-horizon agents via markovian state reconstruction, 2025a. URL https://arxiv.org/abs/2511.07327.

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. Reinforcement learning for long-horizon interactive llm agents. *arXiv preprint arXiv:2502.01600*, 2025b.

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, Fan Yang, et al. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*, 2025c.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*, 2025.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous rl. *arXiv preprint arXiv:2508.07976*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

inclusionAI. inclusionAI/ASearcher-Web-QwQ. https://huggingface.co/inclusionAI/ASearcher-Web-QwQ, 2025. Hugging Face model card; Apache-2.0 License.

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.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A. Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. Acon: Optimizing context compression for long-horizon llm agents, 2025. URL https://arxiv.org/abs/2510.00615.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, et al. Websailor: Navigating super-human reasoning for web agent. *arXiv preprint arXiv:2507.02592*, 2025.

Junteng Liu, Yunji Li, Chi Zhang, Jingyang Li, Aili Chen, Ke Ji, Weiyu Cheng, Zijia Wu, Chengyu Du, Qidi Xu, et al. Webexplorer: Explore and evolve for training long-horizon web agents. *arXiv preprint arXiv:2509.06501*, 2025.

Miao Lu, Weiwei Sun, Weihua Du, Zhan Ling, Xuesong Yao, Kang Liu, and Jiecao Chen. Scaling llm multi-turn rl with end-to-end summarization-based context management, 2025. URL https://arxiv.org/abs/2510.06727.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*, 2022.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*, 2023.

OpenAI. Introducing GPT-4.1 in the API. https://openai.com/index/gpt-4-1/, 2025. Large language model (part of GPT-4.1 family); proprietary API model.

A Yang Qwen, Baosong Yang, B Zhang, B Hui, B Zheng, B Yu, Chengpeng Li, D Liu, F Huang, H Wei, et al. Qwen2. 5 technical report. *arXiv preprint*, 2024.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon llm agent via context-folding. *arXiv preprint arXiv:2510.11967*, 2025.

Zheyue Tan, Mustapha Abdullahi, Tuo Shi, Huining Yuan, Zelai Xu, Chao Yu, Boxun Li, and Bo Zhao. Earl: Efficient agentic reinforcement learning systems for large language models. *arXiv preprint arXiv:2510.05943*, 2025.

MiroMind Team, Song Bai, Lidong Bing, Carson Chen, Guanzheng Chen, Yuntao Chen, Zhe Chen, Ziyi Chen, Jifeng Dai, Xuan Dong, Wenhan Dou, Yue Deng, Yunjie Fu, Junqi Ge, Chenxia Han, Tammy Huang, Zhenhang Huang, Jerry Jiao, Shilei Jiang, Tianyu Jiao, Xiaoqi Jian, Lei Lei, Ruilin Li, Ryan Luo, Tiantong Li, Xiang Lin, Ziyuan Liu, Zhiqi Li, Jie Ni, Qiang Ren, Pax Sun, Shiqian Su, Chenxin Tao, Bin Wang, Hellen Wang, Haonan Wang, James Wang, Jin Wang, Jojo Wang, Letian Wang, Shizun Wang, Weizhi Wang, Zixuan Wang, Jinfan Xu, Sen Xing, Chenyu Yang, Hai Ye, Jiaheng Yu, Yue Yu, Muyan Zhong, Tianchen Zhao, Xizhou Zhu, Yanpeng Zhou, Yifan Zhang, and Zhi Zhu. Mirothinker: Pushing the performance boundaries of open-source research agents via model, context, and interactive scaling, 2025. URL https://arxiv.org/abs/2511.11793.

Guangya Wan, Mingyang Ling, Xiaoqi Ren, Rujun Han, Sheng Li, and Zizhao Zhang. Compass: Enhancing agent long-horizon reasoning with evolving context. *arXiv preprint arXiv:2510.08790*, 2025.

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. Mem-{\alpha}: Learning memory construction via reinforcement learning. *arXiv preprint arXiv:2509.25911*, 2025.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Gang Fu, Yong Jiang, et al. Webdancer: Towards autonomous information seeking agency. *arXiv preprint arXiv:2505.22648*, 2025a.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, et al. Webwalker: Benchmarking llms in web traversal. *arXiv preprint arXiv:2501.07572*, 2025b.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, et al. Resum: Unlocking long-horizon search intelligence via context summarization. *arXiv preprint arXiv:2509.13313*, 2025c.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Honglin Guo, Jiaqi Liu, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, et al. Agentgym-rl: Training llm agents for long-horizon decision making through multi-turn reinforcement learning. *arXiv preprint arXiv:2509.08755*, 2025.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Kristian Kersting, Jeff Z Pan, Hinrich Schütze, et al. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*, 2025.

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. *arXiv preprint arXiv:1809.09600*, 2018.

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

Rui Ye, Zhongwang Zhang, Kuan Li, Huifeng Yin, Zhengwei Tao, Yida Zhao, Liangcai Su, Liwen Zhang, Zile Qiao, Xinyu Wang, et al. Agentfold: Long-horizon web agents with proactive context management. *arXiv preprint arXiv:2510.24699*, 2025.

Qianhao Yuan, Jie Lou, Zichao Li, Jiawei Chen, Yaojie Lu, Hongyu Lin, Le Sun, Debing Zhang, and Xianpei Han. Memsearcher: Training llms to reason, search and manage memory via end-to-end reinforcement learning. *arXiv preprint arXiv:2511.02805*, 2025.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *arXiv preprint arXiv:2504.03160*, 2025.

A Notation Summary

Table 6 summarizes the notation used in the method and analysis. We use $\mathcal{J}$ for maximization objectives, such as clipped PPO terms, and $\mathcal{L}$ for minimization losses. The summary write is denoted by sum_t throughout the paper.

Table 6: Notation used in FoldAct.

Symbol	Meaning
$h_{0:t}$	Raw interaction history up to turn t.
s_t	Bounded folded visible context given to the policy at turn t.
$\bar{h}_t$	Full context $\mathrm{Window}_{L_{\max}}(h_{0:t})$ under the model context budget.
sum_t	Summary write inserted into later folded contexts.
$a^\star$	Ground-truth answer string.
$\mathrm{match}(\cdot,\cdot)$	Normalized answer-string match used for the summary write.
a_t, o_{t+1}	Environment action at turn t and the resulting observation.
$\mathbf{r}_t, r_{t,i}$	Generated response at turn t and its i-th token.
$K_{\pi,i}, d_t^\pi, \beta$	Turn-i transition kernel, turn-t state distribution, and rollout policy.
$D_{\mathrm{TV}}, \epsilon_i$	Total-variation distance and one-step transition-kernel drift.
$A_t^\beta, H, A_{\max}$	Advantage under β, rollout horizon, and uniform advantage bound.
$\mathbf{c}_t^{\mathrm{sum}}, \mathbf{c}_t^{\mathrm{act}}$	Binary token masks for summary and action tokens.
$\mathcal{D}_{\mathrm{rollout}}$	Rollout buffer collected by the rollout policy.
$\theta, \theta_{\mathrm{old}}, \epsilon_{\mathrm{clip}}$	Updated-policy parameters, frozen rollout-policy parameters, and PPO clip threshold.
$N_t^c, \hat{A}_t^c, \rho_{t,i}$	Category token count, category advantage estimate, and token importance ratio.
$\mathcal{I}_t$	Rollout-generated token positions used by the consistency loss at turn t.
$\mathrm{sg}[\cdot]$	Stop-gradient operator.
$z_t, q, p_{\mathrm{drop}}$	Replay-selection mask, retention rate, and drop rate, with $q = 1 - p_{\mathrm{drop}}$.
$K_t, \bar{L}_t, y_t$	Folded-context length, full-context length, and optimized response length.
L_0, g_i, o_i	Initial prompt length and generated/observation token counts at turn i.
$\Phi(n), \alpha_{\mathrm{fold}}$	Model-pass cost at length n and context folding ratio $K_t/\bar{L}_t$.
B, G, κ	Batch size, sampled responses per prompt, and PPO replay-pass multiplier.
η_{cons}	Frequency of the forward-only consistency branch in the cost model.
$C_{\mathrm{trunc}}, C_{\mathrm{FoldAct}}(z)$	Update-side costs of all-turn recent-history truncation and FoldAct under replay mask z.

B FoldAct Training Update

Table 7 summarizes the training loop. The important distinction is that selective segment training masks replay losses only; rollout turns, observations, and summary writes remain in the trajectory and can affect later folded states.

Table 7: FoldAct training update.

1. Roll out the full multi-turn trajectory with the current 7B policy.
2. Generate inline `<think_summary>` and `<information_summary>` spans as part of context folding.
3. Construct each next folded visible state s_{t+1} from the raw history and generated sum_t.
4. Store all turns, observations, summaries, task rewards, and summary-shaping rewards.
5. Sample replay masks $z_t \sim \mathrm{Bernoulli}(q)$ after rollout.
6. For retained turns, build summary/action token masks and compute separated PPO objectives.
7. Compute the consistency branch against the full context with a stop-gradient target on rollout-generated tokens.
8. Update the policy with $\mathcal{L}_{\mathrm{separated}} + \mathcal{L}_{\mathrm{consistency}}$.

C Additional Web Search Results

Table 8: Additional results on Web Search benchmarks. † marks reference results from Liu et al. (2025) and Wu et al. (2025c); "–" indicates an unreported benchmark. Systems differ in data, model family, and tools, so these numbers provide transfer context across system configurations.

Method	Model scale	WebWalker Pass@1	GAIA Pass@1	BrowseComp en Pass@1	BrowseComp zh Pass@1	XBench DeepSearch Pass@1
GPT-4.1-mini	API	39.5	32.7	2.1	8.3	25.4
MiroThinker-8B-DPO-v0.1†	8B	45.7	46.6	8.7	13.6	–
WebSailor-7B†	7B	–	33.0	6.7	14.2	34.3
Resum-7B†	7B	–	42.4	12.3	27.1	–
FoldAct-7B $p_{\text{drop}} = 0.5$	7B	46.1	45.0	8.3	15.2	32.9
FoldAct-7B $p_{\text{drop}} = 0$	7B	45.7	46.3	8.4	14.3	35.4

D Qualitative Summary Example

Table 9: Summary comparison example.

w. consistency loss	w.o consistency loss
<think_summary> 1. I need to find out when Guns N Roses performed a promo for a movie starring Arnold Schwarzenegger as a former New York Police detective. </think_summary> <information_summary> The movie is called "End of Days" and it starred Arnold Schwarzenegger as a former New York Police Department detective named Jericho Cane. Guns N' Roses performed a promotional song for this movie during the editing phase. </information_summary>	<think_summary> 1. I need to find out the year when Guns N' Roses performed a promo for a movie starring Arnold Schwarzenegger as a former New York Police detective. 2. I searched for information about ... and found details about his filmography and a movie called "End of Days." </think_summary> <information_summary> 1. Arnold Schwarzenegger starred in "End of Days", ... 2. The film follows former New York Police Department detective Jericho Cane (played by Schwarzenegger) ... 3. "End of Days" had soundtrack songs, including "Oh My God" ... </information_summary>

E Formal View of Summary-Induced Replay Mismatch

This section makes precise the training issue discussed in the main text. Context folding does not violate the usual sequential-decision formulation: state visitation is policy-dependent in any RL problem. The additional issue is that, in an end-to-end folding agent, part of the generated text is also a summary write that changes future visible state.

Let the augmented state be $X_t = (e_t, h_{0:t}, s_t)$, where e_t is the environment state, $h_{0:t}$ is the raw interaction history, and s_t is the bounded folded visible context with $|s_t| \le L_{\max}$. Let sum_t denote the summary write. When a turn contains both a summary span and an environment action a_t, we represent the policy output as the composite action $u_t = (\mathrm{sum}_t, a_t)$; this abstraction does not require a separate summary call. For a generic state value $x = (e, h, s)$, where h denotes a raw history, the folded-state transition kernel induced by policy π is

$$K_\pi(x' \mid x) = \mathbb{E}_{\mathrm{sum}\sim\pi^M(\cdot|s),\, a\sim\pi^A(\cdot|s,\mathrm{sum})}[P(x' \mid x, \mathrm{sum}, a)], \tag{10}$$

where sum ranges over possible summary writes, π^M is the summary-writing distribution, π^A is the action distribution conditioned on the folded context and summary write, and P includes the environment transition and the deterministic folding update that constructs the next visible context. Thus a changed summary-writing policy changes future folded-state occupancy even before task actions differ.

The one-step kernel drift can be decomposed into summary-write drift and action drift:

$$\epsilon_t \le \epsilon_t^M + \epsilon_t^A,$$

where ϵ_t^M measures changes in the summary-writing distribution and ϵ_t^A measures changes in the action distribution conditioned on the written summary. Ordinary ReAct replay mainly exposes the

action channel, while end-to-end folding introduces ϵ_t^M as an additional source of future visible-state drift because the summary write is inserted into later contexts.

Proof of Lemma 3.1. Using $d_{i+1}^{\pi} = d_i^{\pi} K_{\pi,i}$ and $d_{i+1}^{\beta} = d_i^{\beta} K_{\beta,i}$, the total-variation contraction of Markov kernels gives

$$D_{\mathrm{TV}}(d_{i+1}^{\pi}, d_{i+1}^{\beta}) \le D_{\mathrm{TV}}(d_i^{\pi} K_{\pi,i}, d_i^{\pi} K_{\beta,i}) + D_{\mathrm{TV}}(d_i^{\pi} K_{\beta,i}, d_i^{\beta} K_{\beta,i}) \le \epsilon_i + D_{\mathrm{TV}}(d_i^{\pi}, d_i^{\beta}).$$

Unrolling the recursion from $d_0^{\pi} = d_0^{\beta}$ gives the result. For the surrogate claim, let $\Delta I_\beta(\pi) = \sum_t \mathbb{E}_{x\sim d_t^{\pi}, u\sim\pi}[A_t^{\beta}(x,u)]$ and $L_\beta(\pi) = \sum_t \mathbb{E}_{x\sim d_t^{\beta}, u\sim\pi}[A_t^{\beta}(x,u)]$. The bounded-expectation inequality and Eq. 1 give

$$|\Delta I_\beta(\pi) - L_\beta(\pi)| \le 2A_{\max} \sum_{t=0}^{H-1} \sum_{i=0}^{t-1} \epsilon_i \le A_{\max} H(H-1)\epsilon, \tag{11}$$

when $\epsilon_i \le \epsilon$. □

This bound is not meant to claim that summaries break RL assumptions; rather, it explains why summary-write tokens can amplify replay mismatch over long horizons. The consistency loss anchors folded-context behavior to $\bar{h}_t = \mathrm{Window}_{L_{\max}}(h_{0:t})$, regularizing the discrepancy between folded and full contexts that contributes to this mismatch channel.

F Selective Replay Complexity

All methods considered here operate under the same fixed context budget $L_{\max}$; we do not compare against an infinite-context oracle. The full context is $\bar{h}_t = \mathrm{Window}_{L_{\max}}(h_{0:t})$, obtained by retaining the latest $L_{\max}$ tokens without folding. The recent-history truncation baseline replays this context at every turn. Let T be the number of turns, and let its length at turn t be

$$\bar{L}_t = \min\left(L_0 + \sum_{i<t} (g_i + o_i), L_{\max} \right), \tag{12}$$

where L_0 is the initial prompt length and g_i and o_i denote generated-token and observation-token counts from earlier turns. Let K_t be the folded context length, y_t the optimized response length, and $\Phi(n) = c_{\mathrm{lin}} n + c_{\mathrm{attn}} n^2$ approximate the per-forward compute cost.

We denote the total update-side cost of this all-turn recent-history truncation baseline by

$$C_{\mathrm{trunc}} = \kappa B G \sum_{t=1}^{T} \Phi(\bar{L}_t + y_t), \tag{13}$$

where B is batch size, G is the number of sampled responses per prompt, and κ collects PPO replay passes. Training every folded turn requires separate model passes because each retained turn has its own reconstructed context.

FoldAct samples turns only for loss replay. With $z_t \sim \mathrm{Bernoulli}(q)$ and $q = 1 - p_{\mathrm{drop}}$, its random training-side cost is

$$C_{\mathrm{FoldAct}}(z) = BG \sum_{t=1}^{T} z_t \left[\kappa \Phi(K_t + y_t) + \eta_{\mathrm{cons}} \Phi(\bar{L}_t + y_t) \right], \tag{14}$$

where the second term represents the additional consistency branch and η_{cons} indicates how often it is applied per retained replay segment. This branch is forward-only and gathers log probabilities only for rollout-generated tokens, rather than materializing full-vocabulary distributions at each position. Importantly, $z_t = 0$ drops only the replay loss for turn t: the rollout still executes the turn, records its observation, and lets its summary affect later contexts.

As a concrete example, consider a six-turn Web Search trajectory. At turn 1 the agent searches for a person, at turn 2 it opens a relevant page and writes an information summary, at turn 3 it follows an ambiguous link, at turn 4 it visits an irrelevant page, at turn 5 it recovers the correct evidence from the folded context, and at turn 6 it answers. Suppose selective replay samples $z = (1, 0, 1, 0, 0, 1)$.

Turns 2, 4, and 5 are not optimized in this update, but they are not deleted: their actions, observations, and summary writes remain in the rollout buffer and determine the later folded states $s_3, \ldots, s_6$. The selected final-answer turn is therefore replayed under the same folded context it actually saw, including summaries produced by skipped turns. In contrast, truncating or deleting the skipped turns would change the visible context and train a different conditioning distribution. Since z_t is sampled after rollout and is independent of the turn content, $\mathbb{E}_z[\sum_t z_t \mathcal{L}_t] = q \sum_t \mathcal{L}_t$, up to the constant scale q; selective segment training is thus a stochastic estimator of the all-turn objective while preserving the full trajectory dynamics. Taking expectation over the replay mask gives $\mathbb{E}_z[C_{\mathrm{FoldAct}}(z)]$ by replacing each z_t with q.

If attention cost dominates, $K_t \approx \alpha_{\mathrm{fold}} \bar{L}_t$ with folding ratio $\alpha_{\mathrm{fold}} < 1$, and the response lengths are small relative to context lengths, the expected cost ratio is approximately

$$\frac{\mathbb{E}_z[C_{\mathrm{FoldAct}}(z)]}{C_{\mathrm{trunc}}} \approx q\alpha_{\mathrm{fold}}^2 + q\frac{\eta_{\mathrm{cons}}}{\kappa}. \tag{15}$$

Here $\alpha_{\mathrm{fold}} = K_t/\bar{L}_t$ denotes the folding ratio, approximated as constant for the ratio calculation. This expression captures the two efficiency sources used in the experiments: folded contexts shorten each retained replay, and selective segment training reduces the expected number of replayed turns from T to qT. In the main setting, $p_{\mathrm{drop}} = 0.5$ and therefore $q = 0.5$, while the consistency branch is applied once for each retained replay, giving $\eta_{\mathrm{cons}} = 1$. This establishes the cost expression used in Corollary 4.1; substituting the measured folding ratio gives the numerical prediction reported in the main text.

G Broader Impact

FoldAct can reduce the update-side cost of training long-horizon agents and improve stability when agents must preserve evidence across many turns. These benefits may make context-folding agents more accessible for retrieval, web search, and other information-intensive workflows. The same efficiency and stability could also lower the cost of training stronger tool-using agents, including agents that automate unwanted information gathering, spam, or unsafe web interactions. FoldAct itself does not introduce a new dataset, external memory system, or deployment privilege, but deployed systems should still use task-specific safety policies, source attribution, logging, and access controls around tool use. Because summaries become future policy state, high-stakes deployments should validate summary writes rather than treating FoldAct summaries as certified factual records.

H Detailed Related Work

H.1 Long-Horizon RL Agents with Context Constraints

LLM agents increasingly solve tasks by repeatedly reasoning, calling tools, observing results, and revising their plan. This interaction pattern appears in ReAct-style agents (Yao et al., 2022), multi-turn tool-use benchmarks (Xi et al., 2025), and retrieval or web-search agents (Jin et al., 2025; Gao et al., 2025; Wu et al., 2025a; Li et al., 2025). FoldAct shares this long-horizon setting: the policy must carry evidence from earlier tool calls into later actions. Much of the agent-RL literature studies how to improve action selection, tool-use behavior, or task-level performance over such trajectories. FoldAct studies a complementary constraint created by the same trajectory structure. Append-only histories preserve evidence and keep later decisions grounded, but they can exceed practical context windows, make policy-update replay expensive, and degrade model behavior through context rot (Anthropic, 2025). These constraints motivate methods that manage the working context rather than retaining every interaction verbatim.

H.2 Context Management in Long-Horizon RL

Prior work addresses this constraint through memory and compression. Memory-based methods add external storage and retrieval mechanisms so that an agent can write, search, and reuse information outside the prompt (Yan et al., 2025; Yuan et al., 2025; Wang et al., 2025). These methods share FoldAct's goal of retaining useful historical information under a context budget, but they place state behind a retrieval interface. FoldAct instead folds the trajectory inside the active prompt

and does not add inference-time modules. Compression-based methods reduce the active prompt directly. Within this family, heuristic summarization rules provide simple folding policies, AgentFold learns flexible selective folding with supervised training (Ye et al., 2025), and RL-based methods such as SUPO (Lu et al., 2025) and FoldGRPO (Sun et al., 2025) incorporate summarization into agent training. Related work also studies optimized context folding (Kang et al., 2025) and state reconstruction for long-horizon agents (Chen et al., 2025a).

FoldAct belongs to this folding family because it also replaces parts of an agent trajectory with compact textual state. Its focus is the optimization problem created when the same policy both writes summaries and acts from the folded context. In this setting, summary tokens are policy-written memory: they are generated tokens whose content becomes future visible state. FoldAct therefore isolates three training-specific questions: how to assign learning signal to summary tokens without diluting action learning, how to anchor folded-context behavior to the full context available during training, and how to reduce the number of per-turn model passes. Thus FoldAct is best viewed as a training method for folding policies, not as a new retrieval memory or inference-time agent architecture.