LatentHarness: Learning Latent Actions for Memory and Reasoning via Counterfactual Policy Distillation

Anonymous authors
Paper under double-blind review

Abstract

Long-context reasoning faces two complementary bottlenecks: retaining evidence across long inputs and sustaining computation across many reasoning steps. Existing approaches largely address them separately, with external memory extending access to distant evidence and latent reasoning compressing multi-step computation. We introduce **`LatentHarness`**, which unifies memory access and latent reasoning as sequential latent action selection. At each internal step, the model chooses THINK for further computation, RECALL from a fast-weight memory of input evidence and intermediate reasoning states, or EXIT to emit the next token. We train this policy with *counterfactual policy distillation*, which branches every action for one step and scores its effect on the emitted token. These gains teach the policy when memory is more useful than further reasoning, while gradients through counterfactual recall teach which intermediate states should be retained in memory for future use. Across six general and long-context reasoning benchmarks, `LatentHarness` at 1.4B improves on the strongest baselines by 2.8% and 10.0% relative, respectively, and runs 5.9× faster than the strongest long-context baseline.

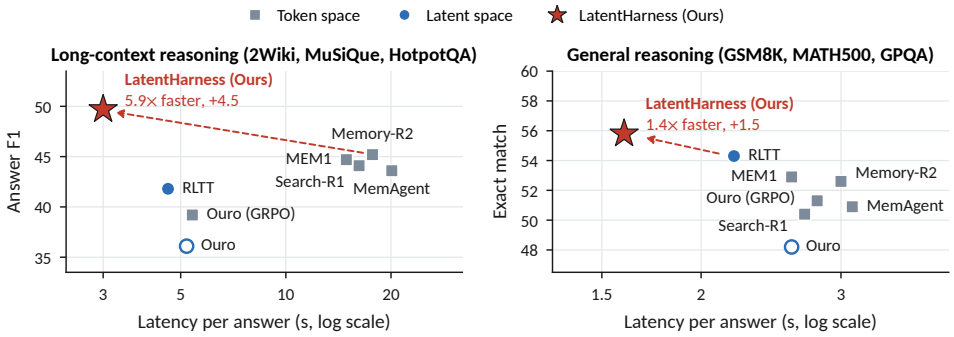

Figure 1: Accuracy versus latency of `LatentHarness` on general and long-context reasoning with Ouro-1.4B-Thinking. The dashed arrow points from the strongest baseline to ours.

1 Introduction

Large language models (Achiam et al., 2023; Yang et al., 2025; Guo et al., 2025) increasingly act as agents over long horizons (Yao et al., 2022; Yang et al., 2024a; Wang et al., 2024; Li et al., 2025b; Luo et al., 2025), so their reasoning becomes a long-context problem as trajectories accumulate observations, tool outputs, and intermediate reasoning states (Wu et al., 2025; Ye et al., 2025; Lu et al., 2025). This problem involves two distinct forms of length. First, relevant evidence may be distributed across long inputs (Bertsch et al., 2025; Bai et al., 2024; Hsieh et al., 2024) or multi-turn interactions (Maharana et al., 2024; Wu et al., 2024). Second, solving the task may require many intermediate reasoning steps (Wei et al., 2022; Guo et al., 2025; Team et al., 2025), and these steps may be interleaved with retrieval of the necessary evidence (Li et al., 2025a; Jin et al., 2025).

Existing approaches largely address these two forms of length separately. External-memory agents (Yan et al., 2026; Yu et al., 2026; Zhou et al., 2025; Yu et al., 2025) retain and retrieve

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.

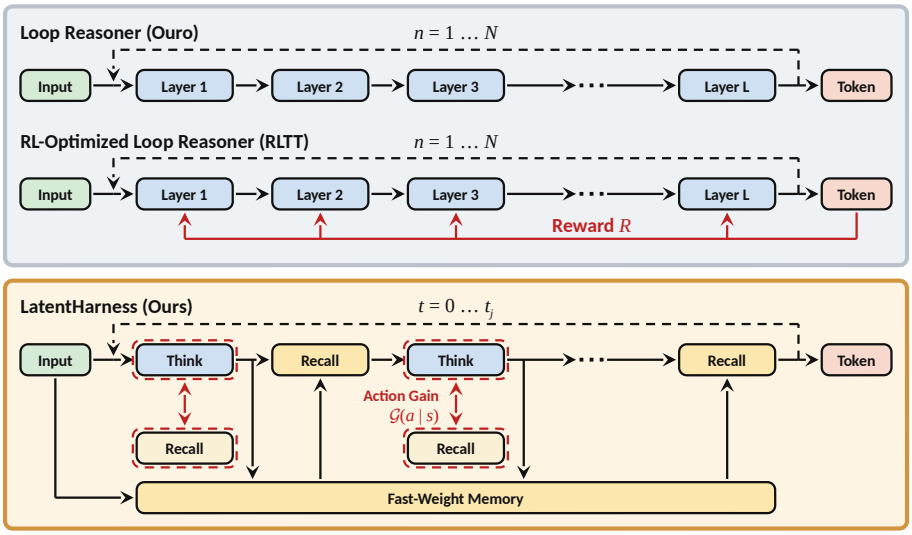

Figure 2: Comparison of Ouro (Zhu et al., 2025b), a looped reasoner that recurs over one weight-tied block, RLTT (Williams & Tureci, 2026), the same loop trained with an outcome reward, and `LatentHarness`, which interleaves latent reasoning and memory, with THINK writing the fast-weight memory that RECALL reads, and learns each latent action from its action gain $\mathcal{G}(a \mid s)$ (Eq. 7).

distant evidence across a trajectory, which extends access beyond the current context window. Latent-reasoning methods (Hao et al., 2024; Shen et al., 2025; Wang et al., 2025b; Zhu et al., 2025b) instead move long intermediate computation from decoded text into internal representations. More recent methods move toward learned memory management by training models to decide when to compress the context (Zhou et al., 2025; Wu et al., 2025; Sun et al., 2025), and when to read or write stored entries (Yan et al., 2025; Zhang et al., 2026b;a; Dong et al., 2026; Li et al., 2026). However, these policies still rely either on memory that remains external to the model and is read back through the context window, or on compression rules fixed in advance. This limitation motivates a more direct question: *can memory access and reasoning be unified natively within the model's latent computation?*

As illustrated in Figure 2, `LatentHarness` builds on the looped computation of latent-reasoning models and unifies memory access and reasoning as sequential latent action selection. At each internal step, a latent policy chooses THINK to continue computation, RECALL to access persistent memory, or EXIT to emit the next token. These actions operate over a shared fast-weight associative memory that stores both input evidence and intermediate reasoning states, allowing RECALL to recover distant evidence or reuse prior computation while THINK deepens reasoning.

Learning such a policy requires assigning credit to individual latent decisions beyond the final task reward. Dense objectives such as RLTT (Williams & Tureci, 2026) supervise what each latent state predicts, but sampled action credit evaluates only the chosen action and vanishes when rollouts stop splitting across actions. We introduce *counterfactual policy distillation*, which branches every allowed latent action, including EXIT, for one step at each latent state and measures the resulting change in the emitted token's log-probability. These counterfactual branches serve two roles. As *state-level action credit*, their gains teach the policy whether a memory read, further reasoning, or exiting yields the largest advantage-signed, cost-charged one-step gain. As *gain-credited memory writing*, their recall changes, differentiated through the fast-weight memory on positive-advantage trajectories, credit each earlier THINK write by how much it raises or lowers later recalls, which teaches the write gate which derived states to write for later reuse.

`LatentHarness` improves both reasoning quality and efficiency across three general and three long-context reasoning suites. It raises the six-suite average over the strongest baseline by 3.9 points at 1.4B and 4.1 points at 2.6B. At 1.4B, it improves on the strongest baseline in each family by 2.8% relative on general reasoning and 10.0% on long-context reasoning, while using 55% of the backbone's full-depth compute. As Figure 1 previews, on long-context reasoning `LatentHarness` is 5.9× faster than Memory-R2 and 1.5× faster than RLTT.

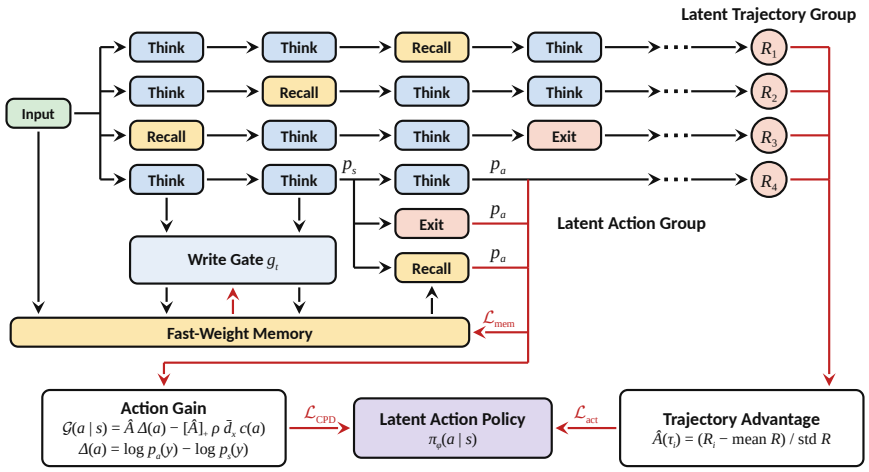

Figure 3: Illustration of counterfactual policy distillation (CPD) in `LatentHarness`. The G latent trajectories sampled from one input form the latent trajectory group, and their rewards determine the trajectory advantage $\hat{A}(\tau)$. The one-step branches at each latent state form the latent action group, and scoring their decodes p_a against p_s yields the action gain distilled into the latent action policy π_ϕ.

2 LatentHarness: Latent Actions for Memory and Reasoning

`LatentHarness` augments looped latent reasoning with memory recall, so a latent state can read missing evidence or reuse an earlier result instead of recomputing it. At every latent state, a latent action policy chooses among three actions. THINK applies the shared block once more and writes its result to a latent memory that also stores the memorized prompt, RECALL injects a read from this memory into the hidden state, and EXIT emits the next token. Because trajectory-level rewards credit latent actions and memory writes only coarsely, counterfactual policy distillation (CPD) branches every allowed action for one step at each latent state, as Figure 3 illustrates. The resulting action gains teach the policy which action to take, while the recall changes, differentiated through memory, teach the write gate which results to store.

2.1 Latent Actions and Memory

A loop can only recompute from its current hidden state. `LatentHarness` therefore augments it with a latent memory, a fixed-size fast-weight matrix (Schlag et al., 2021; Behrouz et al., 2024; Sun et al., 2024; Behrouz et al., 2026) written by the forward pass, not by gradient descent. It then treats each output token as the outcome of a decision process over the hidden state, memory, and number of THINK steps taken. Formally, given a prompt x, the model emits an answer $y = (y_1, \ldots, y_J)$ one position at a time. Within a position, latent steps are indexed by $t = 0, 1, \ldots$ with the position index suppressed. The state at step t is $s_t = (h_t, \mathbf{M}_t, n_t)$, where $h_t \in \mathbb{R}^d$ is the hidden state, $\mathbf{M}_t \in \mathbb{R}^{d \times d}$ is the latent memory, and n_t counts the THINK steps taken at the position. The latent action policy π_ϕ with parameters ϕ selects $a_t \in \{\text{THINK}, \text{RECALL}, \text{EXIT}\}$, the deterministic transition T_{a_t} produces the next state, and EXIT at step t_j leaves the state unchanged and emits the token at position j,

$$a_t \sim \pi_\phi(\cdot \mid s_t), \qquad s_{t+1} = T_{a_t}(s_t), \qquad y_j \sim p(\cdot \mid h_{t_j}), \tag{1}$$

where $p(\cdot \mid h) = \text{softmax}(W_{\text{LM}} h)$ decodes a hidden state through the output projection W_{LM}. Each position starts from its input hidden state h_0, $n_0 = 0$, and the previous position's memory $\mathbf{M}_0$.

Think. When the prediction requires further computation, THINK applies the shared block once more and stores the result under a key formed from the hidden state before the block, allowing a similar state to reuse it without another block application. Formally, a looped language model (Zhu et al., 2025b) applies one shared Transformer block F_θ with backbone parameters θ up to N times at each position, so that depth grows without added parameters, and uses a learned exit. We leave the block's attention over earlier positions implicit. After n applications from the input hidden state $h^{(0)}$, its state $h^{(n)} = F_\theta \circ \cdots \circ F_\theta(h^{(0)})$ decodes at any depth. `LatentHarness` replaces the exit with the latent action policy and exposes each block application as THINK. It also writes by the delta

rule (Schlag et al., 2021; Yang et al., 2024b) with the normalized key $k_t = W_k h_t / \|W_k h_t\|$, the value $v_t = W_v h_{t+1}$ of the block output h_{t+1}, and the write strength $g_t = \sigma(w^\top h_{t+1} + \kappa\, n_t/N)$,

$$h_{t+1} = F_\theta(h_t), \qquad n_{t+1} = n_t + 1, \qquad \mathbf{M}_{t+1} = \mathbf{M}_t + g_t\big(v_t - \mathbf{M}_t k_t\big)k_t^\top, \tag{2}$$

where W_k, W_v, and the gate vector w are learned, σ is the sigmoid, and the learned scalar $\kappa \geq 0$ favors results derived after more THINK steps. Without recalls, $h_t = h^{(n_t)}$. Before generation, one block application over the prompt writes every prompt token in order into the zero matrix by the same rule at unit write strength, memorizing the prompt for the first position.

Recall. When the hidden state needs information already in memory, RECALL reads it with one matrix-vector product, which is much cheaper than a block application. Formally, with the normalized query $q_t = W_q h_t / \|W_q h_t\|$ and learned projections W_q and W_o, the RECALL transition is

$$m_t = \mathbf{M}_t q_t, \qquad h_{t+1} = h_t + W_o m_t, \qquad \mathbf{M}_{t+1} = \mathbf{M}_t, \quad n_{t+1} = n_t, \tag{3}$$

where the read m_t superposes stored values whose keys resemble q_t.

Latent action policy. The policy must judge whether a recall would improve the prediction before the read changes the hidden state. It therefore observes the hidden state and a probe of the pending read. Computing this probe is not a latent step, whereas selecting RECALL is. Formally, let $u_t = (\|m_t\|_2, \cos(m_t, h_t))$ be the probe of the read m_t in Eq. 3. The learned linear head W_π defines $\pi_\phi(a \mid s_t) = \mathrm{softmax}(W_\pi[h_t; u_t])$ over the three actions, where $[h_t; u_t]$ denotes concatenation. The selected action updates the hidden state as

$$h_{t+1} = \underbrace{\mathbb{1}[a_t{=}\textsc{Think}]\, F_\theta(h_t)}_{\text{compute}} + \underbrace{\mathbb{1}[a_t{=}\textsc{Recall}]\,\big(h_t + W_o m_t\big)}_{\text{recall}} + \underbrace{\mathbb{1}[a_t{=}\textsc{Exit}]\, h_t}_{\text{emit } y_j \sim p(\cdot \mid h_t)}, \tag{4}$$

where $\mathbb{1}[\cdot]$ is the indicator function. THINK is masked when $n_t = N$, and RECALL is masked after N recalls at the position. An implicit per-position counter tracks recalls and is read by the admissible set $\mathcal{A}(s_t)$ of unmasked actions. EXIT is forced when both alternatives are masked. π_ϕ is renormalized over $\mathcal{A}(s_t)$, rollouts sample from it, and inference selects the most probable action. No gradient passes through selection, so $\phi = \{W_\pi\}$ learns only through $\log \pi_\phi$, while the backbone parameters θ and memory parameters $\psi = \{W_q, W_o, W_k, W_v, w, \kappa\}$ learn through realized states. The recall branches of Section 2.2 also train w and κ.

2.2 COUNTERFACTUAL POLICY DISTILLATION

Reinforcement learning over latent trajectories. We train whole latent trajectories with GRPO on the task reward. Formally, a rollout of Eq. 1 over the positions $j = 1, \ldots, J$ of a prompt x, with the position index restored, produces the latent trajectory $\tau = \big((s_{j,t}, a_{j,t})_{t=0}^{t_j},\, y_j\big)_{j=1}^{J}$. Because every transition is deterministic given the state and earlier positions, its likelihood factors as

$$\pi(\tau \mid x) = \prod_{j=1}^{J} \Big(\prod_{t=0}^{t_j} \pi_\phi(a_{j,t} \mid s_{j,t}) \Big)\, p(y_j \mid h_{j,t_j}), \tag{5}$$

where $a_{j,t_j} = \textsc{Exit}$. The reward $R(\tau) = R_{\text{task}}(y)$ is the answer's exact match or answer F1, without a length penalty, since a penalty could rank a short failure above a longer success. GRPO (Shao et al., 2024) samples $\tau_1, \ldots, \tau_G$ per prompt, as Figure 3 shows, and normalizes their rewards by the group mean and standard deviation into advantages $\hat{A}(\tau_i)$. The loss $-\hat{A}(\tau) \log \pi(\tau \mid x)$, whose gradient at the on-policy point equals that of the clipped surrogate of PPO (Schulman et al., 2017), decomposes by Eq. 5 into the sampled-action loss $\mathcal{L}_{\text{act}}$ and a token term $-\hat{A}(\tau) \sum_j \log p(y_j \mid h_{j,t_j})$, with each token and latent-action ratio clipped separately. Because the token term supervises only exit states, we follow RLTT (Williams & Tureci, 2026) and replace it with the dense latent loss $\mathcal{L}_{\text{latent}}$, yielding

$$\mathcal{L}_{\text{act}} = -\hat{A}(\tau) \sum_j \sum_t \log \pi_\phi(a_{j,t} \mid s_{j,t}), \qquad \mathcal{L}_{\text{latent}} = -\hat{A}(\tau) \sum_j \sum_t \omega_{n_{j,t}} \log p(y_j \mid h_{j,t}), \tag{6}$$

where the depth weights ω_n follow RLTT, are shared by states that RECALL reaches at the same THINK count, and are normalized over the realized states of each position. Two decisions still receive only coarse credit. $\mathcal{L}_{\text{act}}$ assigns one advantage to every latent action in a trajectory, and the write

gate, as a deterministic part of the transition, receives gradients only through predictions downstream of executed recalls, since the probe u feeds only the policy losses, which update only ϕ.

State-level action credit. At one latent state, the shared advantage of $\mathcal{L}_{\text{act}}$ provides credit that fades. Its gradient on the logit of an action taken by a fraction $\hat{p}$ of the sampled continuations scales with $\hat{p}(1-\hat{p})$, so it vanishes when they do not split and fades as the policy commits. CPD instead compares every admissible action from the same state using the one-step change it makes to the emitted token's log-probability, scaled by trajectory quality and charged an action cost for its latent length on positive-advantage trajectories. Formally, for each $a \in \mathcal{A}(s)$, including EXIT, the one-step change and the action gain are

$$\Delta(a \mid s) = \log p_{T_a(s)}(y) - \log p_s(y), \quad \mathcal{G}(a \mid s) = \hat{A}(\tau)\big(\Delta(a \mid s) - \mathbb{1}[\hat{A}(\tau) > 0]\, \rho\, \bar{d}_x\, c(a)\big), \tag{7}$$

where y is the token that τ emits at this position, $p_s(y)$ and $p_{T_a(s)}(y)$ are its probabilities decoded before and after a, ρ is the cost weight, and $\bar{d}_x$ is the median $|\Delta|$ over the THINK and RECALL branches of the prompt's positive-advantage rollouts. The latent lengths are $c(\text{THINK}) = 1$, $c(\text{RECALL}) = c_R < 1$, and $c(\text{EXIT}) = 0$. Recall has a shorter latent length because it injects an existing read and applies no block. Because T_{EXIT} is the identity, THINK or RECALL outscores EXIT on a positive-advantage trajectory exactly when its Δ exceeds its cost. Without the indicator, a negative advantage would reward latent length.

The gains define a counterfactual teacher over $\mathcal{A}(s)$ as the Boltzmann policy $\pi^*(a \mid s) \propto \exp\big(\mathcal{G}(a \mid s)/\zeta\big)$. CPD distills this teacher into the policy through policy distillation (Rusu et al., 2015),

$$\mathcal{L}_{\text{CPD}} = \frac{1}{|\mathcal{S}_\tau|} \sum_{s \in \mathcal{S}_\tau} D_{\text{KL}}\big(\pi^*(\cdot \mid s) \,\|\, \pi_\phi(\cdot \mid s)\big), \tag{8}$$

where $\mathcal{S}_\tau$ is the set of latent states of τ, the gains are held under stop-gradient, and $\mathcal{L}_{\text{CPD}}$ is averaged over the group. The temperature ζ controls how sharply the teacher favors higher gains, approaching the hard label $\arg\max_a \mathcal{G}(a \mid s)$ as $\zeta \to 0$ and the uniform distribution over $\mathcal{A}(s)$ as $\zeta \to \infty$. Because the teacher is fixed, $\mathcal{L}_{\text{CPD}}$ is cross-entropy to a soft label, with gradient $\pi_\phi(a \mid s) - \pi^*(a \mid s)$ on the logit of a, so its credit does not fade as the policy commits. Each branch ends after one action and one decode. An untaken branch is never continued and does not commit its write, while the taken branch reuses its next state. The gains therefore provide immediate state-level credit at every latent state, while $\mathcal{L}_{\text{act}}$ carries the downstream credit of the sampled path.

Gain-credited memory writing. A write helps only when a later recall reads it and raises the emitted token's probability. Standard delta-rule training therefore credits the gate only downstream of executed recalls, which are rare early in training. CPD provides denser credit through recall branches because every state where RECALL is admissible measures $\Delta(\text{RECALL} \mid s)$, whether or not the policy recalls there. This change depends on every earlier gate through $\mathbf{M}$. Formally, we train the gate to increase the recall changes of positive-advantage trajectories,

$$\mathcal{L}_{\text{mem}} = -\frac{[\hat{A}(\tau)]_+}{|\mathcal{S}_\tau|} \sum_{s \in \mathcal{S}_\tau} \Delta(\text{RECALL} \mid s), \tag{9}$$

where $[\hat{A}(\tau)]_+$ is the positive part of $\hat{A}(\tau)$ and masked states contribute zero. $\mathcal{L}_{\text{mem}}$ updates only the gate parameters w and κ, with every other quantity held fixed, so the token losses still train the keys and values. Restricting this signal to positive advantages is a design choice, since negative weights could make the gate lower recall changes by storing noise instead of correcting a harmful write.

The credit assigned to a write separates into whether its content improves a later prediction and whether the later query still addresses it after intervening writes. Formally, number the latent states of τ by $r = 0, 1, \ldots$ across positions. A realized THINK write at step r changes the recall change at a later state $s_{r'}$ where RECALL is admissible at the rate

$$\frac{\partial \Delta(\text{RECALL} \mid s_{r'})}{\partial g_r} = \underbrace{\big\langle \nabla_m \log p_{T_{\text{RECALL}}(s_{r'})}(y_{r'}),\ v_r - \mathbf{M}_r k_r \big\rangle}_{\text{content}}\ \underbrace{k_r^\top P_{r+1} \cdots P_{r'-1}\, q_{r'}}_{\text{addressing}}, \tag{10}$$

where $y_{r'}$ is the token emitted at the position of $s_{r'}$, the gradient is evaluated at the branch read $m = \mathbf{M}_{r'} q_{r'}$, $P_\ell = I - g_\ell k_\ell k_\ell^\top$ at each intervening THINK step and $P_\ell = I$ otherwise, and the empty

Table 1: Quantitative results of `LatentHarness` with Ouro-1.4B-Thinking and Ouro-2.6B-Thinking. MATH, GPQA, 2WQA, MSQ, and HQA denote MATH500, GPQA-Diamond, 2WikiMultihopQA, MuSiQue, and HotpotQA, † marks an out-of-distribution suite, and Lat. is wall-clock seconds per answer. **Bold**, underline, and green mark the best, the second best, and the margin over the second best. Ouro is the released backbone at full depth.

	General reasoning						Long-context reasoning					
Method	GSM8K	MATH	GPQA†	Avg.	FLOPs	Lat.	2WQA†	MSQ†	HQA	Avg.	FLOPs	Lat.
Ouro-1.4B-Thinking												
Ouro	74.6	42.8	27.3	48.2	100%	2.6	41.2	19.7	47.5	36.1	100%	5.2
Token space												
Ouro (GRPO)	78.1	46.3	29.4	51.3	106%	2.8	44.8	22.1	50.6	39.2	108%	5.4
+ Search-R1	77.4	45.6	28.3	50.4	133%	2.7	50.4	27.1	54.9	44.1	339%	16.2
+ MemAgent	78.0	45.7	29.0	50.9	130%	3.1	49.7	26.8	54.3	43.6	454%	20.1
+ MEM1	79.6	48.1	31.0	52.9	121%	2.6	50.9	27.8	55.5	44.7	307%	14.9
+ Memory-R2	79.4	47.6	30.8	52.6	125%	3.0	51.3	28.4	55.9	45.2	391%	17.7
Latent space												
RLTT	81.5	49.6	31.8	54.3	86%	2.2	47.6	24.5	53.2	41.8	82%	4.6
`LatentHarness`	**83.2**	**51.6**	**32.6**	**55.8** (+1.5)	**62%**	**1.6**	**55.9**	**33.1**	**60.1**	**49.7** (+4.5)	**48%**	**3.0**
Ouro-2.6B-Thinking												
Ouro	79.3	48.5	31.6	53.1	100%	4.4	45.9	23.4	51.8	40.4	100%	8.8
Token space												
Ouro (GRPO)	82.7	52.0	33.9	56.2	105%	4.7	49.2	26.1	55.0	43.4	107%	9.3
+ Search-R1	82.1	51.0	32.8	55.3	131%	4.5	54.6	31.0	59.0	48.2	335%	27.1
+ MemAgent	82.6	51.5	33.6	55.9	128%	5.2	54.1	30.7	58.6	47.8	448%	33.6
+ MEM1	84.3	53.5	35.3	57.7	120%	4.4	55.0	31.6	59.4	48.7	304%	24.6
+ Memory-R2	84.0	53.3	35.2	57.5	123%	4.9	55.6	32.2	60.1	49.3	387%	29.5
Latent space												
RLTT	85.4	55.3	36.2	59.0	87%	3.7	52.0	28.3	57.4	45.9	83%	7.7
`LatentHarness`	**87.3**	**57.2**	**38.3**	**60.9** (+1.9)	**64%**	**2.6**	**60.3**	**37.6**	**64.2**	**54.0** (+4.7)	**50%**	**5.0**

product is I. After scaling by $[\hat{A}(\tau)]_+/|\mathcal{S}_\tau|$, a write receives the sum of these products over later admissible states. Its credit is exactly zero when $v_r = \mathbf{M}_r k_r$ or when its addressing factor vanishes at every later admissible state. A later state contributes negatively when the addressed change lowers its prediction. Because hidden states are held fixed, Eq. 10 is a partial derivative along the realized trajectory. It ignores how a gate would alter later keys, values, gates, queries, and actions, but requires only one backward pass through branches CPD already computes.

Final objective. The objective is $\mathcal{L} = \mathcal{L}_{\text{latent}} + \beta\mathcal{L}_{\text{act}} + \gamma\mathcal{L}_{\text{CPD}} + \lambda\mathcal{L}_{\text{mem}} + \eta\mathcal{L}_{\text{ref}}$, where β, γ, λ, and η are loss weights and $\mathcal{L}_{\text{ref}}$ is the standard KL regularizer (Shao et al., 2024) from the token distribution at each emitting state to that of the frozen full-depth backbone. $\mathcal{L}_{\text{latent}}$ and $\mathcal{L}_{\text{ref}}$ train the backbone θ and the memory maps ψ through the realized states, $\mathcal{L}_{\text{act}}$ and $\mathcal{L}_{\text{CPD}}$ train the policy ϕ, and $\mathcal{L}_{\text{mem}}$ trains the write gate w and κ. We anneal γ linearly from one to a tenth over the first 1.5k steps, so the teacher shapes the policy early and later only prevents commitment. Please refer to Appendix B for all hyperparameters and to Appendix C for the derivation of the sampled credit.

3 Experiments

Backbones. We evaluate the two released looped reasoners, Ouro-1.4B-Thinking and Ouro-2.6B-Thinking (Zhu et al., 2025b), at their trained recurrent depth $N = 4$. The shared block F_θ is the full stack of 24 or 48 layers. The memory maps ψ and latent action policy ϕ are the only new parameters, adding 1.8% and 1.0% to the respective parameter counts.

Datasets. We train one model per backbone on the GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) training splits. Following MemAgent (Yu et al., 2025) and Search-R1 (Jin et al., 2025), we also train on HotpotQA (Yang et al., 2018) questions with supporting passages embedded among distractors and padded to 4k, 8k, and 16k tokens. We test GSM8K, MATH500 (Lightman et al., 2023), and HotpotQA in distribution. Following RLTT (Williams & Tureci, 2026), we test GPQA-Diamond (Rein et al., 2023) out of distribution, along with 2WikiMultihopQA (Ho et al., 2020) and MuSiQue (Trivedi et al., 2022). Each long-context test set is padded to

4k, 8k, 16k, 32k, and 64k tokens, so the two longest lengths exceed every training input. We report exact match on the general suites and answer F1 on the long-context suites.

Baselines. Token-space methods include GRPO (Shao et al., 2024), which trains the backbone with outcome RL without memory, Search-R1 (Jin et al., 2025), which interleaves retrieval and reasoning over the padded context, MemAgent (Yu et al., 2025) and Memory-R2 (Yan et al., 2026), which write and read memory as generated text, and MEM1 (Zhou et al., 2025), which combines memory and reasoning in one compact token-space state per turn. Latent-space methods include RLTT (Williams & Tureci, 2026), which gives the loop dense latent credit without memory, and `LatentHarness`. Every trained method uses the same backbone, training data, and number of RL steps. We report FLOPs relative to the full-depth backbone and per-answer latency following Wang et al. (2025b). Please refer to Appendix B for implementation details and efficiency accounting.

3.1 Main results

`LatentHarness` achieves the highest accuracy at both model scales. In Table 1, averaging the two family averages gives six-suite means of 52.8 at 1.4B and 57.5 at 2.6B, exceeding Memory-R2 by 3.9 and 4.1 points, respectively. The gains hold within both task families, consistent with a latent action policy that distinguishes states needing further computation from those missing information.

The largest gains come from supplying missing information on long inputs. Relative to RLTT, which uses the same latent loop without memory, `LatentHarness` raises the general-reasoning average from 54.3 to 55.8 and the long-context average from 41.8 to 49.7. Its long-context margin grows from 2.3 points at 4k tokens to 14.3 at 64k. On the general suites, the hidden state already carries the evidence, so the policy mainly reallocates THINK steps. On long inputs, RECALL recovers evidence that further latent computation cannot reconstruct.

`LatentHarness` improves accuracy while reducing inference cost. At 1.4B, `LatentHarness` averages 55% of full-depth FLOPs across the two task families, compared with 84% for RLTT, and answers long-context questions 5.9× faster than Memory-R2 in Figure 1. In Table 2, removing the action cost ($\rho = 0$) raises FLOPs from 55% to 93% while improving each family average by only 0.3 points. Applying the action cost to the action gain instead of the trajectory return suppresses low-gain latent steps without allowing a shorter failure to outrank a longer success.

Table 2: Ablations of `LatentHarness` on action credit, action space, write gate, and action cost. Gen. and Long denote family averages, FLOPs denotes the two-family average relative to full depth, and parentheses mark the change from ours. Training compute is 1.1×, 1.2×, and 3.6× GRPO only for ours, GiGPO groups, and VinePPO.

Variant	Gen.	Long	FLOPs
`LatentHarness`	55.8	49.7	55%
Action credit			
GRPO only	52.3 (−3.5)	44.3 (−5.4)	71%
GiGPO groups	52.9 (−2.9)	45.1 (−4.6)	68%
VinePPO branches	54.9 (−0.9)	48.6 (−1.1)	58%
Action space			
Think and Exit	53.6 (−2.2)	42.0 (−7.7)	63%
Always read	54.1 (−1.7)	45.3 (−4.4)	58%
Write gate			
Token-loss gate	55.5 (−0.3)	48.1 (−1.6)	57%
Write-time gate	55.4 (−0.4)	47.6 (−2.1)	58%
Action cost			
No cost	56.1 (+0.3)	50.0 (+0.3)	93%

3.2 In-depth analysis

RQ1: Does state-level credit matter? We retrain the policy with GRPO only, GiGPO groups, and VinePPO branches on the same data and steps. We compare their family averages and training compute with ours in Table 2, and track the entropy of π_ϕ and the recall change $\Delta(\text{RECALL} \mid s)$ on test rollouts.

One-step counterfactual branches outperform completed branches at a third of the training compute. VinePPO branches complete each branched action to the answer (Kazemnejad et al., 2025), while GiGPO groups pool latent states by output position and action prefix (Feng et al., 2025). On the long-context suites, `LatentHarness` reaches 49.7 at 1.1× training compute, compared with 48.6 at 3.6× for VinePPO branches. Every latent state decodes, so a one-step branch scores each action without completing a rollout.

CPD keeps the latent action policy from collapsing because its credit does not depend on sampled exploration. Under GRPO only, which drops $\mathcal{L}_{\text{CPD}}$, the entropy of π_ϕ falls to 0.21 nats within 200 steps, whereas `LatentHarness` holds it at 0.68, and its $\Delta(\text{RECALL} \mid s)$ from Eq. 7

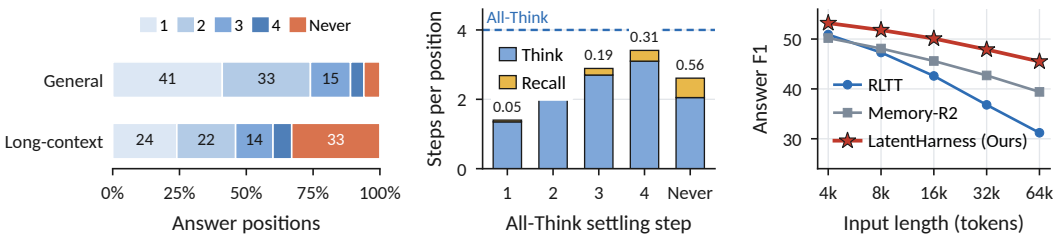

Figure 4: All-Think settling, latent actions, and long-context F1 of `LatentHarness`, RLTT, and Memory-R2 on Ouro-1.4B-Thinking. The left, middle, and right panels show positions by All-Think settling step, latent steps per position by settling step, and F1 by input length, respectively.

averages 0.36 nats at executed recalls against 0.03 at admissible states where it is not taken. Sampled credit fades as the policy commits, whereas counterfactual branches compare every action.

RQ2: When should the policy recall instead of think? We retrain the Think and Exit and Always read action-space ablations, and run RLTT at the full depth $N = 4$ as All-Think. A one-step THINK probe marks each position's settling step, and we count THINK and RECALL steps per position by that step and by input length. We measure each All-Think step's I/O cosine and, at every latent state of `LatentHarness`, the pending read's norm and cosines grouped by executed action.

Selective RECALL is necessary on long inputs. Always read injects a memory read at every step. Removing RECALL lowers the long-context average by 7.7 points to 42.0, while Always read remains 4.4 points below `LatentHarness`. An always-on read cannot distinguish missing evidence from missing computation, so the policy must decide when to read.

Full-depth thinking wastes loops yet leaves a third of long-context positions unsettled. The left panel of Figure 4 shows that 51% of loops change nothing on the general suites, while 33% of long-context positions never settle. Further loops are redundant after settlement and cannot recover evidence that the hidden state lacks, which only a read can supply.

THINK converges toward the current hidden state, while RECALL adds a direction unavailable to further THINK steps. Following Zhang et al. (2026c), a step's I/O cosine is the cosine between its input and output hidden states. Under All-Think, this cosine rises from 0.71 to 0.99 on the general suites but plateaus at 0.91 on the long-context suites. Later loops therefore rewrite either a converged state or one that cannot converge. In the left panel of Figure 5, the pending read at RECALL states has a relative norm of 0.58 and a cosine of 0.12 with the pending THINK update, compared with 0.14 and 0.36 at THINK states. This larger, less aligned read adds a component that further computation from the same state cannot produce.

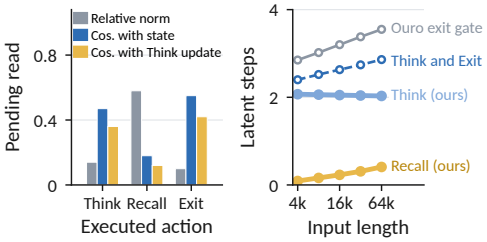

Figure 5: Pending reads by executed action and latent steps by input length on the Ouro-1.4B-Thinking long-context suites.

`LatentHarness` spends recalls, not extra THINK steps, where evidence is missing. In the middle panel of Figure 4, positions settled at step 1 average 0.05 recalls, compared with 0.56 for positions that never settle. In the right panel of Figure 5, recalls rise from 0.09 to 0.41 per position while THINK steps remain near 2.0. Because unsettled positions and long inputs lack evidence in the hidden state, RECALL has a larger one-step action gain than another THINK.

RQ3: What does RECALL read? We split every executed read into memorized-prompt and THINK-written components to label it derived or evidence, match each evidence recall to its best-matching passage and score hits against supporting-fact annotations, report F1 by hop count and rank passages at the final evidence recall of wrong MuSiQue answers with at least three hops.

RECALL returns supporting evidence and increasingly reuses derived states. At 64k tokens, evidence recalls hit 63.9% against a chance rate of 0.5%. The derived share rises from 27.4% at 4k to 33.2%. Prompt reads recover sparser evidence, and THINK writes return intermediate results.

Memory addressing dominates the remaining multi-hop errors. Splitting MuSiQue by hop count, F1 falls from 42.1 at two hops to 16.4 at four, and at the final evidence recall of 72% of wrong answers with three or more hops a distractor outranks the supporting passage, which ranks second. The query q scores the supporting key highly but fails to separate it from a nearby distractor.

RQ4: What does the write gate learn? We retrain the gate with the token-loss and write-time targets, record its value at each THINK step and whether a later positive-change recall addresses each write, and measure retention as the cosine between a memorized prompt value and its read-back after 256 to 2k generated tokens.

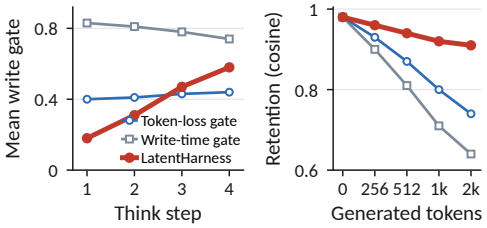

Figure 6: Mean write gate by THINK step and evidence retention by generated tokens on the long-context suites.

Gain credit opens the write gate for derived states that later recalls reuse. The write-time gate trains g by whether THINK beats RECALL at the writing state. In Table 2, these gates lower the long-context average by 1.6 and 2.1 points. Under gain credit, 58% of reused writes have $g > 0.7$, against 6% of unread writes. Eq. 10 credits a write only when a later RECALL branch, executed or not, addresses its key and gains from its content, so redundant and unaddressed writes receive none.

A write-time target keeps the gate open and overwrites memorized evidence. In the left panel of Figure 6, the write-time gate remains high at every THINK step. In the right panel, its retention after 2k generated tokens falls to 0.64 against 0.91 under gain credit. Because the target is scored only where computing already beats recalling, the gate remains open for most writes regardless of whether a later recall reads them. Please refer to Appendix B.5 for ablation definitions and analysis statistics and to Table 4 for per-suite values.

4 RELATED WORK

Memory for LLM reasoning. External memory systems (Packer et al., 2023; Xu et al., 2025a; Chhikara et al., 2025; Li et al., 2025c; Du, 2026) store evidence beyond an LLM's context. One line learns storage and retrieval policies (Yan et al., 2025; Yu et al., 2025; Zhou et al., 2025; Wang et al., 2025c; Yuan et al., 2025; Shen et al., 2026; Yu et al., 2026; Zhang et al., 2026b;a; Li et al., 2026; Ma et al., 2026; Dong et al., 2026), including Memory-R2 (Yan et al., 2026). Another interleaves retrieval with reasoning or applies long-context RL to extended inputs (Li et al., 2025a; Song et al., 2025; Wan et al., 2025; Wang et al., 2025a), including Search-R1 (Jin et al., 2025). `LatentHarness` moves memory access into the latent loop, so recall and reasoning share one latent trajectory.

Latent reasoning. Latent reasoning (Zhu et al., 2025a; Chen et al., 2025) performs intermediate computation in hidden states instead of decoded tokens. One line trains latent chains of thought (Goyal et al., 2023; Deng et al., 2023; Shen et al., 2025; Xu et al., 2025b; Zhang et al., 2025b; Tan et al., 2025; Wei et al., 2025; Butt et al., 2025; Wang et al., 2025b; Aichberger & Hochreiter, 2026; Jung et al., 2026; Zou et al., 2026; Zhao et al., 2026), including Coconut (Hao et al., 2024). Another scales latent computation through recurrent depth (Dehghani et al., 2019; Geiping et al., 2025; Saunshi et al., 2025; Williams & Tureci, 2026; Popescu et al., 2026; Schwethelm et al., 2026; Lin et al., 2026), including Ouro (Zhu et al., 2025b). Latent memory mechanisms (Schlag et al., 2021; Yang et al., 2024b; Behrouz et al., 2024; 2025; Kang et al., 2025; Zhang et al., 2025a) store hidden representations. `LatentHarness` adds RECALL to a loop that otherwise only continues or exits.

5 CONCLUSION

We propose `LatentHarness`, which unifies memory access and reasoning through latent action selection over THINK, RECALL, and EXIT and one fast-weight memory of input evidence and derived states. Counterfactual policy distillation branches every latent action once to credit the policy, and gradients through the same branches credit the write gate. It outperforms token-space and latent-space baselines at lower inference compute, recalls where further computation cannot converge, and retains the derived states that later recalls reuse.

AI USE STATEMENT

We used generative AI tools for literature search and grammar suggestions. Every suggestion was reviewed carefully, and all edits were made by the authors. The authors designed, ran, and checked all experiments, configurations, and measurements.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Lukas Aichberger and Sepp Hochreiter. Unlocking the working memory of large language models for latent reasoning. *arXiv preprint arXiv:2605.30343*, 2026.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*, 2024.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Atlas: Learning to optimally memorize the context at test time. *arXiv preprint arXiv:2505.23735*, 2025.

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. It's all connected: A journey through test-time memorization, attentional bias, retention, and online optimization. In *International Conference on Learning Representations*, volume 2026, pp. 131306–131333, 2026.

Amanda Bertsch, Adithya Pratapa, Teruko Mitamura, Graham Neubig, and Matthew R. Gormley. Oolong: Evaluating long context reasoning and aggregation capabilities. *arXiv preprint arXiv:2511.02817*, 2025.

Natasha Butt, Ariel Kwiatkowski, Ismail Labiad, Julia Kempe, and Yann Ollivier. Soft tokens, hard truths. *arXiv preprint arXiv:2509.19170*, 2025.

Xinghao Chen, Anhao Zhao, Heming Xia, Xuan Lu, Hanlin Wang, Yanjun Chen, Wei Zhang, Jian Wang, Wenjie Li, and Xiaoyu Shen. Reasoning beyond language: A comprehensive survey on latent chain-of-thought reasoning. *arXiv preprint arXiv:2505.16782*, 2025.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

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.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations (ICLR)*, 2019. arXiv:1807.03819.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023.

Jiajun Dong, Yutao Hu, Fengrui Fan, Shihan Dou, Yueming Wu, and Deqing Zou. MemArbiter: Decision-time memory arbitration for long-horizon LLM agents. *arXiv preprint arXiv:2608.02113*, 2026.

Pengfei Du. Memory for autonomous LLM agents: Mechanisms, evaluation, and emerging frontiers. *arXiv preprint arXiv:2603.07670*, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems*, 2025. arXiv:2505.10978.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. *arXiv preprint arXiv:2310.02226*, 2023.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.

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.

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.

Dongwon Jung, Peng Shi, Yi Zhang, Junshan Zhang, and Muhao Chen. Adaptive latent agentic reasoning. *arXiv preprint arXiv:2606.02871*, 2026.

Jikun Kang, Wenqi Wu, Filippos Christianos, Alex J. Chan, Fraser Greenlee, George Thomas, Marvin Purtorab, and Andy Toulis. Lm2: Large memory models. *arXiv preprint arXiv:2502.06049*, 2025.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *International Conference on Machine Learning*, 2025. arXiv:2410.01679.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*, 2025a.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yongkang Wu, Ji-Rong Wen, Yutao Zhu, and Zhicheng Dou. Webthinker: Empowering large reasoning models with deep research capability. *arXiv preprint arXiv:2504.21776*, 2025b.

Yilong Li, Suman Banerjee, and Tong Che. EMBER: Efficient memory via budgeted evidence retention for long-horizon agents. *arXiv preprint arXiv:2606.05894*, 2026.

Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, Qingchen Yu, Jihao Zhao, Yezhaohui Wang, Peng Liu, Zehao Lin, Pengyuan Wang, Jiahao Huo, Tianyi Chen, Kai Chen, Kehang Li, Zhen Tao, Huayi Lai, Hao Wu, Bo Tang, Zhengren Wang, Zhaoxin Fan, Ningyu Zhang, Linfeng Zhang, Junchi Yan, Mingchuan Yang, et al. Memos: A memory os for ai system. *arXiv preprint arXiv:2507.03724*, 2025c.

H. Lightman, V. Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John D. Schulman, I. Sutskever, and K. Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023. doi: 10.48550/arXiv.2305.20050.

Ruhai Lin, Yiyang Guo, Rui-Jie Zhu, Hao Ye, and Jason K. Eshraghian. Allocating recurrent compute in looped language models. *arXiv preprint arXiv:2608.18230*, 2026.

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. *arXiv preprint arXiv:2510.06727*, 2025.

Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, Rongcheng Tu, Xiao Luo, Wei Ju, Zhiping Xiao, Yifan Wang, Meng Xiao, Chenwu Liu, Jingyang Yuan, Shichang Zhang, Yiqiao Jin, Fan Zhang, Xian Wu, Hanqing Zhao, Dacheng Tao, Philip S. Yu, and Ming Zhang. Large language model agent: A survey on methodology, applications and challenges. *arXiv preprint arXiv:2503.21460*, 2025.

Yiwen Ma, Songjun Tu, Qichao Zhang, Dong Li, Linjing Li, and Dongbin Zhao. MemChain: Learning interpretable memory traces for memory-augmented LLM agents. *arXiv preprint arXiv:2607.24097*, 2026.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.

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.

Andrei Cristian Popescu, Haitz Sáez de Ocáriz Borde, and Pietro Liò. Adaptive depth in looped transformers: Diagnosing learned halting gates and trajectory readouts. *arXiv preprint arXiv:2607.20519*, 2026.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, J. Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, K. Kavukcuoglu, and R. Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. *arXiv preprint arXiv:2502.17416*, 2025.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning (ICML)*, 2021.

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

Kristian Schwethelm, Daniel Rueckert, and Georgios Kaissis. Depth-adaptive inference of looped language models via continuous depth batching. *arXiv preprint arXiv:2608.09444*, 2026.

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

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*, 2025.

Zhiyu Shen, Ziming Wu, Fuming Lai, Shaobing Lian, and Yanghui Rao. MemBuilder: Reinforcing LLMs for long-term memory construction via attributed dense rewards. *arXiv preprint arXiv:2601.05488*, 2026.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.

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.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Think silently, think fast: Dynamic latent compression of llm reasoning chains. *arXiv preprint arXiv:2505.16552*, 2025.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

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.

Siyuan Wang, Gaokai Zhang, Li Lyna Zhang, Ning Shang, Fan Yang, Dongyao Chen, and Mao Yang. Loongrl: Reinforcement learning for advanced reasoning over long contexts. *arXiv preprint arXiv:2510.19363*, 2025a.

Xiaoqiang Wang, Suyuchen Wang, Yun Zhu, and Bang Liu. System-1.5 reasoning: Traversal in language and latent spaces with dynamic shortcuts. *arXiv preprint arXiv:2505.18962*, 2025b.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*, 2024.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. Sim-cot: Supervised implicit chain-of-thought. *arXiv preprint arXiv:2509.20317*, 2025.

Jonathan Williams and Esin Tureci. Prioritize the process, not just the outcome: Rewarding latent thought trajectories improves reasoning in looped language models. *arXiv preprint arXiv:2602.10520*, 2026.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Minhao Cheng, Shuai Wang, Hong Cheng, and Jingren Zhou. Resum: Unlocking long-horizon search intelligence via context summarization. *arXiv preprint arXiv:2509.13313*, 2025.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025a.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot: Soft chain-of-thought for efficient reasoning with llms. *arXiv preprint arXiv:2502.12134*, 2025b.

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

Sikuan Yan, Ahmed Bahloul, Ercong Nie, Susanna Schwarzmann, Riccardo Trivisonno, Volker Tresp, and Yunpu Ma. Memory-R2: Fair credit assignment for long-horizon memory-augmented LLM agents. *arXiv preprint arXiv:2605.21768*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*, 2024a.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b. arXiv:2406.06484.

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.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Rui Ye, Zhongwang Zhang, Kuan Li, Huifeng Yin, Zhengwei Tao, Yida Zhao, Liangcai Su, Liwen Zhang, Zile Qiao, Xinyu Wang, Pengjun Xie, Fei Huang, Siheng Chen, Jingren Zhou, and Yong Jiang. Agentfold: Long-horizon web agents with proactive context management. *arXiv preprint arXiv:2510.24699*, 2025.

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*, 2025.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. *arXiv preprint arXiv:2601.01885*, 2026.

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.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. MemGen: Weaving generative latent memory for self-evolving agents. *arXiv preprint arXiv:2509.24704*, 2025a.

Qi Zhang, Shen Huang, Chu Liu, Shouqing Yang, Junbo Zhao, Haobo Wang, and Pengjun Xie. DeltaMem: Towards agentic memory management via reinforcement learning. *arXiv preprint arXiv:2604.01560*, 2026a.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, et al. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026b.

Xuanming Zhang, Sining Zhoubian, Yuxuan Chen, Tian-Yi Tang, An Yang, Sean Du, Chujie Zheng, Fei Huang, Dayiheng Liu, Gao Huang, and Jingren Zhou. Deeper is not always better: Mitigating the alignment tax via confident layer decoding. *arXiv preprint arXiv:2606.21906*, 2026c. doi: 10.48550/arXiv.2606.21906.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. *arXiv preprint arXiv:2505.15778*, 2025b.

Xuyang Zhao, Liting Zhang, Zichen Xu, Yong Chen, Wenjia Zeng, Shiwan Zhao, and Qicheng Li. Latent thought credit: Multi-answer credit assignment for latent reasoning. *arXiv preprint arXiv:2608.01593*, 2026.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*, 2025.

Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, Tianle Cai, Taylor Kergan, Assel Kembay, Andrew Smith, Chenghua Lin, Binh Nguyen, Yuqi Pan, Yuhong Chou, Zefan Cai, Zhenhe Wu, Yongchi Zhao, Tianyu Liu, Jian Yang, Wangchunshu Zhou, Chujie Zheng, Chongxuan Li, Yuyin Zhou, Zhoujun Li, Zhaoxiang Zhang, Jiaheng Liu, Ge Zhang, Wenhao Huang, and Jason Eshraghian. A survey on latent reasoning. *arXiv preprint arXiv:2507.06203*, 2025a.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason Eshraghian. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025b.

Xiandong Zou, Jing Huang, Jianshu Li, and Pan Zhou. Latent thought flow: Efficient latent reasoning in large language models. *arXiv preprint arXiv:2606.16222*, 2026.

A LIMITATIONS

The current design does not establish robustness to training variation or larger recurrent budgets, and its memory still has addressing failures on multi-hop questions. Every cell uses one training seed, so the reported margins provide no estimate of run-to-run variance or sensitivity to initialization and data order. The recurrent budget remains fixed at the backbone's inherited $N = 4$, so the experiments do not determine whether greater depth would resolve positions left unsettled after the available actions. The remaining MuSiQue errors concentrate on questions with three or more hops. Although the query addresses the right memory region, it does not separate the evidence needed across successive hops.

The credit used to train the policy and the write gate relies on explicit approximations. The action gain provides only one-step credit. Consequently, a THINK branch whose write is never committed receives no credit for what it would store, and a plan that pays off only after several steps is undervalued. The target token comes from the realized trajectory, which favors branches along that path. On a negative-advantage trajectory, the gain also rewards lowering the probability of every emitted token, including correct tokens within a wrong answer. The gate credit in Eq. 10 holds the hidden states fixed and uses only positive-advantage trajectories. It therefore ignores how a write changes later decisions and does not learn from harmful writes on failed trajectories. The keys, values, and read maps still learn only from executed recalls. Moreover, a $d \times d$ memory holds about d near-orthogonal keys, while memorization writes every prompt token of a long input. Branching adds about 10% training compute over the GRPO-only variant, although inference incurs none of this additional cost.

The experimental scope does not establish generality across serving regimes, natural long documents, or model families. Latency is measured in one serving configuration at batch size one on one accelerator, including prompt memorization for `LatentHarness`, so it does not establish throughput under batched serving. Inputs beyond the native long-context suites are padded with distractor passages, which tests whether memory can address relevant evidence as input length grows but does not reproduce the structure of naturally long documents. The evaluation also covers only two backbone scales from one family.

B Experimental setup

B.1 Backbones and what we add to them

Both backbones are released looped reasoners (Zhu et al., 2025b), Ouro-1.4B-Thinking and Ouro-2.6B-Thinking, and no number crosses between them. We keep the pre-trained recurrence budget $N = 4$ and leave the shared block, the language-model head and the tokenizer untouched. The new parameters are the read and write maps of the fast-weight memory and the latent action policy π_ϕ, which together add 1.8% and 1.0% to the two parameter counts. Ouro in Table 1 and the token-space baselines run the backbone at N on every token, RLTT keeps the backbone's own exit gate, and `LatentHarness` replaces it by π_ϕ. The Ouro exit gate series of Figure 5 and Table 4 runs the released backbone under its learned exit.

B.2 Benchmarks and scoring

Training uses the 7,473 GSM8K and 7,500 MATH training problems together with 15,000 HotpotQA training questions padded to 4k, 8k, or 16k tokens in equal proportion. The test sets comprise the 1,319 GSM8K test problems, MATH500, the 198 GPQA-Diamond questions, and 1,000 questions sampled from each of the HotpotQA distractor development set, the 2WikiMultihopQA development set, and the answerable MuSiQue development set. Each long-context test question is padded to all five lengths using distractor passages drawn from other questions in the same dataset. No test question or passage appears in training. The general suites are GSM8K (Cobbe et al., 2021), MATH500 (Hendrycks et al., 2021) and GPQA-Diamond (Rein et al., 2023), scored as exact match after answer extraction. The long-context suites are 2WikiMultihopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022) and HotpotQA (Yang et al., 2018), scored as answer F1 against the reference span, with the full distractor context supplied rather than a retrieved subset, so that the memory is the only mechanism that can reduce what the model must hold. Every arm is scored by the same extraction and matching code, and no arm is scored under two conventions.

B.3 Efficiency accounting

CPD skips prompts whose rollouts tie on reward, where every gain is zero, and its branches add about 10% to the training compute of the GRPO-only variant. Our efficiency accounting comprises two measured quantities and is independent of the training loss. We compute FLOPs per answer by counting the Transformer blocks executed over prompt and decoded tokens, and we report them relative to running the same backbone at N on every token. Following the FLOPs-reduction and wall-clock-speedup accounting of Wang et al. (2025b), we report wall-clock latency per answer at batch size one on one accelerator, including prompt memorization for `LatentHarness`.

B.4 Hyperparameters

Table 3 lists the values shared by both backbones. The distillation weight γ is annealed linearly over the first 1.5k steps, and every other value is fixed.

B.5 Analysis statistics

Statistics are computed on the Ouro-1.4B-Thinking test sets. Step counts are per-position averages over answer tokens and the derived share is a share of executed recalls and the hit rate a share of evidence recalls, with per-suite values of both in Table 4.

Table 3: Training hyperparameters of `LatentHarness` shared by both backbones.

Hyperparameter	Value
Maximum recurrent depth N	4
Rollouts per prompt G	8
Prompts per batch	64
Training steps	3k
Learning rate, backbone and memory (θ, ψ)	1×10^{-6}
Learning rate, latent action policy (ϕ)	1×10^{-4}
Teacher temperature ζ	0.5
Action-cost weight ρ	0.25
RECALL latent length c_R	0.15
Sampled-action weight β	1.0
Distillation weight γ	1.0 to 0.1 over 1.5k steps
Write-credit weight λ	0.5
Reference weight η	0.01

Ablation variants. Every variant in Table 2 retrains from the same initialization and data for the same number of steps, with one component changed. The credit variants change only how π_ϕ is credited and keep the trajectory advantage, dense latent credit, and $\mathcal{L}_{\text{mem}}$ with its RECALL branches. GRPO only uses the sampled-action term alone. GiGPO groups partitions the latent states of a prompt's rollouts by output position and preceding action sequence, then computes step advantages from within-group returns (Feng et al., 2025). Because continuous latent states rarely recur, only 14% of latent states share a group with another state. VinePPO branches selects one output position per rollout, branches each admissible action at a random 25% of its latent states, completes each branch to the answer four times, and replaces the action gain with the Monte Carlo return (Kazemnejad et al., 2025). Think and Exit removes RECALL. Always read injects the memory read at every step and restricts the policy to THINK and EXIT. The write-gate variants retain Eq. 2 but change how the gate is trained. Token-loss gate drops $\mathcal{L}_{\text{mem}}$, so the gate learns only from token losses at executed recalls, as in standard delta-rule memories. Write-time gate replaces $\mathcal{L}_{\text{mem}}$ with cross-entropy toward $\sigma\big((\Delta(\text{THINK} \mid s) - \Delta(\text{RECALL} \mid s))/\zeta\big)$ at the writing state, which opens the gate when thinking outperforms recalling there. No cost sets the cost weight ρ to zero.

Policy entropy and recall change. Entropy is the mean entropy of π_ϕ over the latent states of training rollouts at each training step. Both runs start at 1.08 nats. The recall change is $\Delta(\text{RECALL} \mid s)$ of Eq. 7, averaged over test latent states where RECALL is executed and where it is admissible but not taken.

Settling. A one-step THINK probe outside the latent trajectory marks a latent state as settled if it changes the decoded token distribution by less than 0.01 nats of KL. A position settles at the first step from which every later latent state is settled. Under All-Think, a position that settles at step $t^\star$ spends its remaining $N - t^\star$ loops without change, which gives the share of loops that change nothing.

Step dynamics. The I/O cosine of a latent step is the cosine between its input and output hidden states, averaged over answer positions. Under All-Think, we compute it separately for each THINK step. The pending read at a latent state is $W_o m_t$, with m_t from Eq. 3, and is computed whether or not RECALL is executed there. We report its norm relative to $\|h_t\|$. Its cosine with the state and its cosine with the pending THINK update is computed against $F_\theta(h_t) - h_t$ from the same one-step probe that defines settling. States are grouped by the action executed by the policy. Under All-Think, the I/O cosine of THINK steps one to four is 0.71, 0.91, 0.97, and 0.99 on the general suites and 0.69, 0.85, 0.90, and 0.91 on the long-context suites.

Recall sources. A read $m = \mathbf{M}q$ decomposes into the memorized-prompt component $\mathbf{M}_x q$, where $\mathbf{M}_x$ is the memory after prompt memorization, and the component written by THINK. A recall is derived when the written component has the larger norm. Otherwise, it is an evidence recall assigned to the passage of its best-matching prompt key, and it counts as a hit when that passage is annotated as supporting. Chance is the share of prompt tokens in supporting passages.

Reuse and retention. A write is reused when a later executed RECALL with positive Δ addresses its key with $k^\top q > 0.5$, and it is unread when no later RECALL addresses its key with $k^\top q > 0.5$.

Under gain credit, 58% of reused writes have $g > 0.7$ against 6% of unread writes. Following the associative-recall probes of fast-weight memories (Schlag et al., 2021; Yang et al., 2024b), retention is the cosine between a memorized prompt value v and its read-back $\mathbf{M}k$ after a given number of generated tokens, averaged over supporting passages.

Multi-hop errors. For each wrong answer to a MuSiQue question with three or more hops, we rank the prompt passages at the final evidence recall by the cosine $k^\top q$ of their best-matching key. The answer is an addressing error when the top passage is a distractor and a supporting passage ranks second.

B.6 Latent steps and recall sources per suite

Table 4 reports the per-suite values underlying the right panel of Figure 5 and the `LatentHarness` F1 curve of Figure 4, right. In that panel, RLTT scores 50.9, 47.3, 42.6, 36.8, and 31.2 F1 at 4k, 8k, 16k, 32k, and 64k tokens, and Memory-R2 scores 50.2, 48.1, 45.6, 42.7, and 39.4. Think and recall counts are averaged over the answer tokens of each test set, and the derived share, hit rate, and chance follow Appendix B.5. Step counts average over answer positions, and scores and chance over questions, the derived share over recalls, and the hit rate over evidence recalls, so the hop rows pool to the MuSiQue row under the MuSiQue hop mix and the length rows pool to the same totals as the long-context suite rows.

Table 4: Task scores, per-position latent steps with and without RECALL and under Ouro's exit gate, and recall sources of `LatentHarness` on Ouro-1.4B-Thinking by suite, hop count, and input length. Long-context, hop, and length rows are pooled over their other axis. Score denotes exact match on general suites and F1 otherwise, and – marks suites without supporting-fact annotation.

	`LatentHarness`			Think and Exit	Ouro exit gate	Recall sources (%)		
Suite	Think	Recall	Score	Think	Loops	Derived	Hit	Chance
General reasoning								
GSM8K	2.10	0.03	83.2	2.14	2.70	39.2	–	–
MATH500	2.55	0.05	51.6	2.57	3.30	35.6	–	–
GPQA	2.40	0.07	32.6	2.43	3.20	33.2	–	–
Long-context reasoning								
2WQA	2.00	0.20	55.9	2.46	3.10	29.3	69.8	2.8
MuSiQue	2.15	0.32	33.1	2.95	3.40	34.7	65.9	2.8
HotpotQA	2.00	0.20	60.1	2.47	3.10	27.7	72.0	2.7
MuSiQue by hops								
2-hop	2.09	0.24	42.1	2.78	3.24	29.5	74.0	2.2
3-hop	2.18	0.37	27.3	3.08	3.52	35.6	63.6	3.1
4-hop	2.27	0.50	16.4	3.25	3.67	40.8	55.6	4.1
Long-context by input length								
4k	2.07	0.09	53.2	2.40	2.85	27.4	78.4	7.0
8k	2.06	0.16	51.8	2.52	3.02	28.9	74.9	3.6
16k	2.05	0.23	50.1	2.63	3.20	30.3	71.1	1.8
32k	2.04	0.31	47.9	2.74	3.38	31.7	67.2	0.9
64k	2.03	0.41	45.5	2.86	3.55	33.2	63.9	0.5

B.7 The criterion every checkpoint had to meet

A shared-weight loop admits a constant-output fixed point, and a model that has fallen into one still reports a plausible validation loss while every ablation against it reads as zero. No number is quoted from a checkpoint that did not first pass a within-run check for that failure, comprising the mean total variation between each position's predictive distribution and the run's own mean distribution, the number of distinct argmax tokens across all probed positions, and the nats of context the model actually uses. Under collapse the first is below 10^{-3}, the second is one or two, and the third is zero.

C SAMPLED CREDIT AT A LATENT STATE

This section derives the sampled credit stated in Section 2.2. Fix a latent state s and an action a whose log-odds against the other two actions is z, so that $\pi_a = \pi_\phi(a \mid s) = \sigma(z)$. Consider K continuations sampled from s, let $b_i \in \{0, 1\}$ indicate whether continuation i takes a, let $\hat{p}$ be the mean of b, and let $\hat{A}$ be centered over the continuations. The sampled-action objective at s is $\mathcal{J} = \frac{1}{K} \sum_i \hat{A}_i \log \pi_\phi(b_i \mid s)$. Since $\partial_z \log \pi_\phi(b_i \mid s) = b_i - \pi_a$ and $\sum_i \hat{A}_i = 0$,

$$\partial_z \mathcal{J} = \frac{1}{K} \sum_i \hat{A}_i (b_i - \pi_a) = \frac{1}{K} \sum_i \hat{A}_i b_i = \widehat{\mathrm{Cov}}_K(b, \hat{A}) = \hat{p}(1 - \hat{p}) \Big(\widehat{\mathbb{E}}[\hat{A} \mid b{=}1] - \widehat{\mathbb{E}}[\hat{A} \mid b{=}0] \Big),$$

where the last step conditions the covariance on b and holds when both subsets are nonempty. When the continuations do not split, b is constant and the gradient is zero. The factor $\hat{p}(1 - \hat{p})$ shrinks as the policy commits, so sampled credit fades exactly when a committed but wrong action needs correction, whereas the action gain of Eq. 7 compares all actions at the same latent state without this factor.

D NOTATION

At output position j the model runs a latent trajectory of action steps t, with latent state $s_{j,t} = (h, \mathbf{M}, n)$, where h is produced by the shared block F_θ and decodable through W_{LM} as in Section 2.1, $\mathbf{M}$ is the fast-weight memory, and n counts the THINK actions executed at the position. The memory is initialized from the prompt by the memorization step of Section 2.1, carried across positions, and written only by THINK through the gated delta rule of Eq. 2, whose gate carries a depth coefficient κ on n and is trained by the RECALL changes through $\mathcal{L}_{\mathrm{mem}}$ in Eq. 9. The probe of the memory is u, and the latent action policy π_ϕ reads (h, u) of the state and chooses $a \in \{\textsc{Think}, \textsc{Recall}, \textsc{Exit}\}$. A group of G rollouts on one prompt gives task-only returns $R(\tau) = R_{\mathrm{task}}(y)$ and centered advantages $\hat{A}$ by group normalization as in Section 2.2. The action cost $\rho \, \bar{d}_x \, c(a)$ in the action gain weights the latent length $c(a)$, with $c(\textsc{Think}) = 1$, $c(\textsc{Recall}) = c_R$ and $c(\textsc{Exit}) = 0$, by the cost weight ρ and the prompt scale $\bar{d}_x$, and applies only on positive-advantage trajectories. A branch at s changes the emitted token's log-probability by $\Delta(a \mid s)$ in Eq. 7, the action gain of an action is $\mathcal{G}$ in Eq. 7, and the distilled target is π^* in Eq. 8.