REMPO: BEHAVIOR-CONSISTENT LATENT-DEPTH POLICY OPTIMIZATION FOR SEARCH AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Looped language models (LoopLMs) enable multi-step latent reasoning by repeatedly applying shared computation, offering a promising way to increase reasoning depth without increasing model parameters. However, their recurrent computation also creates a mismatch with standard reinforcement learning, which is defined around the final policy output. Recent work has taken an important first step by extending outcome supervision to intermediate predictions, allowing internal computation to receive learning signals. Yet reinforcement learning (RL) is fundamentally driven by the behavior policy that generates the trajectory: assigning credit to intermediate states does not by itself make them part of the policy that produces actions. We therefore introduce **Readout-Marginalized Policy Optimization** (**ReMPO**), which constructs the behavior policy by marginalizing token predictions across recurrent readouts. ReMPO uses the same readout-marginalized policy for both trajectory generation and policy optimization, allowing final-answer rewards to jointly train token prediction and readout selection. A terminal anchor and regularization stabilize this learned policy. Using only final-answer rewards, we train Ouro-1.4B and Ouro-2.6B as search agents for question answering. ReMPO outperforms the state-of-the-art method on all seven datasets at both model sizes; at 1.4B, it improves the average exact match by **11.30 points** and held-out multi-hop performance by **17.76 points**. It also yields more effective two-search trajectories, with second retrievals recovering answer-bearing evidence missed by the first.

1 INTRODUCTION

Reinforcement learning (RL) has become a central tool for post-training language models, but most existing methods (Jin et al., 2025; Song et al., 2025; Chen et al., 2025) implicitly assume that the computation used to produce each token is fixed. Looped language models (LoopedLMs) challenge this assumption by repeatedly applying shared computation before token prediction, producing intermediate predictions at multiple recurrent depths. This raises a fundamental question for post-training: **how can RL effectively leverage these intermediate predictions and the model's variable computational depth?**

Existing RL methods (Williams & Tureci, 2026) address this challenge only partially. Standard RL with verifiable rewards methods, such as Group Relative Policy Optimization (**GRPO**) (Shao et al., 2024), typically generates trajectories and optimizes the policy from the final recurrent readout, leaving intermediate predictions outside the behavior policy. Recent work takes an important step with Reward Latent Thought Trajectories (**RLTT**) (Williams & Tureci, 2026), extending outcome supervision along latent reasoning trajectories so that intermediate latent states at different depths can receive learning signals derived from the final reward. However, assigning credit to an intermediate state and allowing that state to influence behavior are two different questions(Yang et al., 2024). Credit assignment asks which latent states should learn from the outcome, whereas policy design asks which latent states should help determine the actions that produce that outcome. The latter remains largely unexplored for LoopedLMs.

This distinction becomes especially important in interactive settings such as search. In static reasoning tasks, differences across recurrent depths mainly affect the model's internal predictions before the final answer is produced. In a search agent, however, generated tokens can trigger retrieval

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.

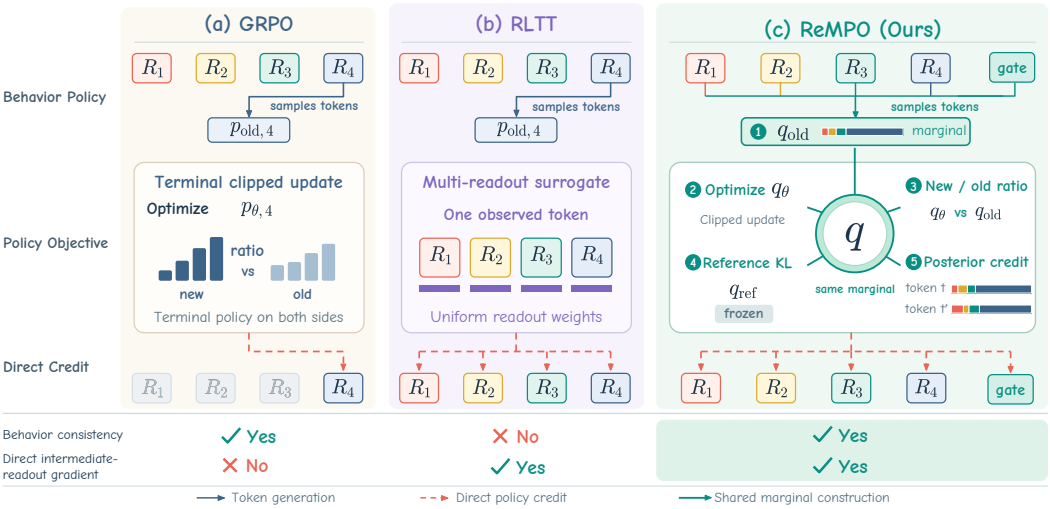

Figure 1: Behavior and credit assignment in looped search-agent RL. (a) GRPO samples from and optimizes R_4. (b) RLTT uses terminal sampling and a weighted multi-readout objective (uniform weights shown). (c) ReMPO uses one marginal-policy construction throughout training: all readouts contribute to sampling, and both readouts and the gate receive direct policy gradients.

actions, and those actions determine what information becomes available in subsequent decision steps (Yao et al., 2023; Trivedi et al., 2023; Jin et al., 2025). A different policy over recurrent depths can therefore lead not only to different token predictions, but also to different search queries, retrieved evidence, and future trajectories. This makes interactive search a natural setting for studying whether recurrent depth should participate directly in behavior generation.

Our key idea is to make recurrent depth part of the behavior policy itself. We introduce **Readout-Marginalized Policy Optimization** (**ReMPO**), which treats recurrent depth as a token-level latent variable and combines the token distributions from multiple readouts into a single policy. This readout-marginalized policy is used consistently for both trajectory generation and policy optimization (Figure 1). As a result, final-answer rewards can jointly optimize token prediction and the contribution of different readouts, without requiring separate supervision for each recurrent depth.

We evaluate ReMPO on search-based question answering with Ouro-1.4B and Ouro-2.6B (Zhu et al., 2025), comparing it against GRPO and RLTT across seven datasets. ReMPO outperforms both baselines on all seven datasets. The gains are especially pronounced on held-out multi-hop tasks, where effective use of additional retrieval is more important. Beyond final-answer accuracy, ReMPO also changes how the agent uses search: it produces more valid two-search trajectories, and the second retrieval can recover answer-bearing evidence that is missing after the first search. These results show that incorporating recurrent depth into the behavior policy improves both task performance and the agent's use of external information.

Our contributions are threefold. First, we distinguish *credit assignment over recurrent depth* from *policy design over recurrent depth*, identifying the latter as an underexplored problem for LoopedLMs in interactive settings. Second, we introduce **ReMPO**, which treats recurrent depth as a token-level latent variable and marginalizes multiple readouts into a unified policy for both rollout generation and RL optimization. Third, experiments on search-based question answering with Ouro-1.4B and Ouro-2.6B show that ReMPO outperforms all baselines across seven datasets while producing more effective multi-search behavior.

2 Related Work

Latent reasoning and looped language models. A growing line of work extends language-model computation beyond a single forward pass. Universal Transformers, PonderNet, and Huginn reuse computation across recurrent depth (Dehghani et al., 2019; Banino et al., 2021; Geiping et al., 2025),

while Coconut, CODI, and HRPO perform reasoning through continuous hidden states rather than explicit chain-of-thought tokens (Hao et al., 2025; Shen et al., 2025; Yue et al., 2025). In Ouro, each recurrent depth exposes a next-token distribution, together with a learned depth gate (Zhu et al., 2025). This makes recurrent depth not only an internal computation axis, but also a possible choice of token-generating readout. Recent work studies supervised tuning and inference-time depth selection for Ouro (Popescu et al., 2026).

Reinforcement learning for recurrent computation. Existing RL methods mainly differ in how intermediate recurrent states receive supervision. LSRL decodes intermediate states for process-level grading, while LoopRPT optimizes intermediate computation using teacher references and latent rollouts (Ren, 2025; Tang et al., 2026). RLTT applies outcome-weighted objectives to token distributions at multiple recurrent depths while retaining terminal-readout rollouts (Williams & Tureci, 2026). Inspired by latent-variable and mixture-policy optimization, where observed actions are optimized by marginalizing over latent components that may have generated them (Smith et al., 2018; Schulman et al., 2015; Liu et al., 2019; Wulfmeier et al., 2020), we treat recurrent depth as a latent decision within the behavior policy.

Reinforcement learning for agentic search. Search-R1, R1-Searcher, and ReSearch train language models to interleave search actions with answer generation using outcome rewards (Jin et al., 2025; Song et al., 2025; Chen et al., 2025), while subsequent work studies the effects of prompts, rewards, masking, and policy optimization (Xu et al., 2026). In agentic search, the behavior policy is important because query tokens determine the retrieved evidence, which affects later queries and the final answer (Yao et al., 2023; Trivedi et al., 2023; Jin et al., 2025).

3 Problem Setup

3.1 Search as an episodic token policy

Given a question x, the agent generates a trajectory $\tau = (s_t, a_t)_{t=1}^{T}$, where s_t is the visible history, a_t the model-generated token, and T the number of generation steps. Retrieved passages are treated as environment observations rather than policy actions, and a mask m_t selects only model-generated tokens for policy optimization.

Following Search-R1 (Jin et al., 2025), each trajectory receives only a terminal exact-match reward $R(\tau) \in \{0, 1\}$. For a group of G trajectories sampled for the same question, we compute

$$A_i = \frac{R(\tau_i) - \mu_x}{\sigma_x + \varepsilon}, \tag{1}$$

where μ_x and σ_x are the group reward mean and standard deviation, and $\varepsilon = 10^{-6}$. We broadcast A_i to all policy tokens in trajectory i.

3.2 From Credit over Depth to Policy over Depth

A looped language model exposes token distributions at K recurrent depths,

$$p_{\theta,1}(\cdot \mid s_t), \ldots, p_{\theta,K}(\cdot \mid s_t). \tag{2}$$

GRPO (Shao et al., 2024) uses only the terminal readout $p_{\theta,K}$ to generate and optimize behavior. RLTT (Williams & Tureci, 2026) extends outcome credit to intermediate readouts, but the trajectory is still generated by the terminal policy. Thus, RLTT addresses *credit assignment over depth*, while intermediate depths remain outside the behavior policy. This distinction matters in interactive search because generated actions, especially search queries, determine future observations and therefore alter the subsequent trajectory.

4 ReMPO

4.1 Making Recurrent Depth Part of the Policy

At each token position, a looped language model exposes K readout distributions $p_{\theta,z}(\cdot \mid s_t)$. ReMPO treats the recurrent depth z as a latent policy variable rather than using only the terminal

readout. We define a depth policy

$$\alpha_\theta(z \mid s_t) = (1 - \epsilon_{\phi(s_t)})\, \mathbb{I}[z = K] + \epsilon_{\phi(s_t)} g_\theta(z \mid s_t), \tag{3}$$

where $g_\theta(z \mid s_t)$ is the native depth gate and $\phi(s_t) \in \{0, 1\}$ indicates whether retrieved evidence has already been observed. The terminal anchor preserves a minimum probability mass on the final readout while allowing earlier depths to participate in behavior. We use stronger terminal anchoring after retrieval, with $(\epsilon_0, \epsilon_1) = (0.25, 0.15)$.

The resulting token policy marginalizes over recurrent depth:

$$q_\theta(a \mid s_t) = \sum_{z=1}^{K} \alpha_\theta(z \mid s_t) p_{\theta,z}(a \mid s_t). \tag{4}$$

This marginal distribution is the central object optimized by ReMPO. Unlike GRPO, which uses only the terminal readout, and RLTT, which assigns credit to multiple readouts while retaining terminal behavior, ReMPO makes all recurrent readouts components of a single environment-facing policy.

4.2 Generating Actions from the Marginal Policy

ReMPO samples each token using the latent-depth representation of Eq. (4):

$$z_t \sim \alpha_{\theta_{\text{old}}}(\cdot \mid s_t), \qquad a_t \sim p_{\theta_{\text{old}}, z_t}(\cdot \mid s_t), \tag{5}$$

where θ_{old} denotes the parameter snapshot used to collect the current rollout batch.

Marginalizing over the sampled depth gives

$$\Pr(a_t = a \mid s_t) = \sum_{z=1}^{K} \alpha_{\theta_{\text{old}}}(z \mid s_t) p_{\theta_{\text{old}}, z}(a \mid s_t) = q_{\theta_{\text{old}}}(a \mid s_t). \tag{6}$$

Therefore, the policy defined in Eq. (4) is exactly the policy that generates actions and interacts with the search environment. In particular, recurrent depth can directly affect search queries, retrieved observations, and the subsequent trajectory, rather than receiving credit only after a terminal-policy trajectory has already been generated.

For each generated token, we store its old-policy marginal log-probability $\ell_t^{\text{old}} = \log q_{\theta_{\text{old}}}(a_t \mid s_t)$, which is used by the policy update described next.

4.3 Optimizing the Same Marginal Policy

For each rollout token, the current model evaluates the same marginal policy, $\ell_t^\theta = \log q_\theta(a_t \mid s_t) = \log\Big(\sum_{z=1}^{K} \alpha_\theta(z \mid s_t) p_{\theta,z}(a_t \mid s_t)\Big)$. The corresponding importance ratio is

$$r_t(\theta) = \exp\big(\ell_t^\theta - \ell_t^{\text{old}}\big) = \frac{q_\theta(a_t \mid s_t)}{q_{\theta_{\text{old}}}(a_t \mid s_t)}. \tag{7}$$

ReMPO then applies the standard clipped GRPO objective directly to this marginal ratio:

$$\mathcal{L}_{\text{policy}}(\theta) = -\frac{1}{\sum_{i,t} m_{i,t}} \sum_{i,t} m_{i,t} \min(r_{i,t} A_i,\ \text{clip}(r_{i,t}, 1 - \delta, 1 + \delta) A_i)\,, \tag{8}$$

where δ is the clipping parameter and A_i is the group-relative trajectory advantage defined in Eq. (1). Thus, depths are marginalized before the policy ratio and clipping are applied, using the same marginal policy construction as rollout generation.

Because depth is latent, the marginal policy also induces a posterior responsibility for each readout,

$$\gamma_t(z) = \Pr_\theta(z \mid s_t, a_t) = \frac{\alpha_\theta(z \mid s_t) p_{\theta,z}(a_t \mid s_t)}{q_\theta(a_t \mid s_t)}. \tag{9}$$

At fixed mixture weights, treating readout log-probabilities as score coordinates gives

$$\frac{\partial \log q_\theta(a_t \mid s_t)}{\partial \log p_{\theta,z}(a_t \mid s_t)} = \gamma_t(z). \tag{10}$$

Hence, contributions from different recurrent depths arise directly from the likelihood of the action actually taken, without introducing a separate per-depth reward or heuristic credit weighting.

4.4 Stabilizing the Latent Policy

Because ReMPO jointly updates the depth policy and the marginal token policy, we regularize both sources of policy drift. First, we constrain changes in the latent depth distribution within each PPO batch:

$$\mathcal{L}_{\text{gate}}(\theta) = \frac{1}{\sum_{i,t} m_{i,t}} \sum_{i,t} m_{i,t} D_{\text{KL}}(\alpha_{\theta_{\text{old}}}(\cdot \mid s_{i,t}) \,\|\, \alpha_\theta(\cdot \mid s_{i,t})) \,. \tag{11}$$

This prevents abrupt redistribution of probability mass across recurrent depths.

Second, we regularize the visible marginal policy against a frozen reference policy q_{ref} using a low-variance KL surrogate $\mathcal{L}_{\text{ref}}$ computed on rollout tokens (Appendix D.1). Both policies use the same latent-depth construction. This term constrains overall token-policy drift, while the gate KL stabilizes how different recurrent depths are used.

The final ReMPO objective is

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{policy}}(\theta) + \beta_{\text{gate}}\mathcal{L}_{\text{gate}}(\theta) + \beta_{\text{ref}}\mathcal{L}_{\text{ref}}(\theta). \tag{12}$$

Together with the terminal anchor in Eq. (3), these regularizers keep depth adaptation controlled while preserving the behavior-consistent marginal policy introduced above. More implementation details of the reference KL estimator, algorithm, and loss reduction are provided in the Appendix A, B and D.1.

Complexity. All K recurrent loops are executed. Rollout adds gate sampling but no second model. The learner needs chosen-token log-probabilities for each readout, giving $O(KM)$ marginal arithmetic for M valid policy tokens; full-vocabulary logits are released loop by loop. Persistent additional tensors scale as $O(BTK)$ for batch size B and response layout T.

5 Experimental Setup

Models, data, and training. We use Ouro-1.4B-Thinking and Ouro-2.6B-Thinking with all methods initialized from the same raw checkpoint at each scale and no GRPO midpoint or agent SFT. Main experiments train on NQ and HotpotQA in Search-R1 with at most two search actions and top-3 retrieval, for up to 500 global steps using 512 questions and $G = 5$ trajectories per question. All methods share the question order, interaction and sequence limits, prompt, parser, exact-match reward, observation mask, and retriever.

Methods and evaluation. We compare Ouro, GRPO, RLTT and ReMPO and ablate the proposed components. Raw Ouro receives no search-specific training and performs zero-shot interactive search under the same protocol and retriever. In the main comparison, trained checkpoints are evaluated at common steps selected by a shared validation rule, using deterministic terminal-R_4 decoding. We report response-only exact match on all seven QA datasets, with search behavior and training GPU-hours reported separately. The GRPO, RLTT, and ReMPO rows in Table 1 each use 1 training seed per method and model size; Appendix E.1 reports additional Ouro-1.4B seed results. Section 7.4 separately studies ReMPO trained on Ouro-1.4B with search budgets from 1 to 4.

Training policy and evaluation readout. ReMPO samples training trajectories from q_θ and optimizes the marginal GRPO objective. Because all readouts share recurrent parameters and the terminal anchor retains $p_{\theta,K}$ in the training mixture, the objective updates both the depth gate and the backbone underlying the terminal readout. Evaluation fixes the readout to R_4 and deterministically decodes from $p_{\theta,K}$, so reported accuracy reflects the effect of each training method on this common terminal policy, while latent-depth choices affect trajectory generation and optimization. For the same ReMPO checkpoint, marginal and terminal greedy decoding yield similar aggregate accuracy and search behavior (Appendix D.2).

6 Main Results

6.1 ReMPO improves search-agent accuracy

Table 1 summarizes the main results. At 1.4B, ReMPO outperforms both GRPO and RLTT on all seven datasets, improving the overall average by **11.30 points**. The improvement is observed on both

Table 1: Seven-dataset exact-match results on a 0–1 scale. The Qwen2.5-3B/7B rows marked † are transcribed from Search-R1 Table 3 (Jin et al., 2025); they are external capability references and are excluded from within-Ouro ranking. All local Ouro rows use deterministic terminal-R_4 evaluation with at most two searches and the same protocol within each scale; each filled row aggregates 51,713 questions. Average is the unweighted mean of the seven datasets.

Method	NQ	HotpotQA	TriviaQA	PopQA	2Wiki	MuSiQue	Bamboogle	Average
Qwen2.5-3B								
Base (GRPO)†	0.421	0.297	0.583	0.413	0.274	0.066	0.128	0.312
Instruct (GRPO)†	0.397	0.331	0.565	0.391	0.310	0.124	0.232	0.336
Qwen2.5-7B								
Base (GRPO)†	0.395	0.326	0.560	0.388	0.297	0.125	0.360	0.350
Instruct (GRPO)†	0.429	0.386	0.623	0.427	0.346	0.162	0.400	0.396
Ouro-1.4B								
Raw Ouro	0.104	0.083	0.285	0.115	0.176	0.024	0.232	0.146
GRPO	0.434	0.300	0.609	0.434	0.287	0.068	0.128	0.323
RLTT	0.438	0.320	0.583	0.432	0.295	0.072	0.120	0.323
ReMPO	**0.485**	**0.452**	**0.631**	**0.464**	**0.427**	**0.177**	**0.416**	**0.436**
Ouro-2.6B								
Raw Ouro	0.174	0.163	0.454	0.165	0.201	0.039	0.344	0.220
GRPO	0.468	0.377	0.626	0.444	0.325	0.129	0.368	0.391
RLTT	0.473	0.397	0.600	0.442	0.334	0.132	0.360	0.391
ReMPO	**0.493**	**0.464**	**0.658**	**0.470**	**0.455**	**0.210**	**0.496**	**0.464**

the training-domain NQ and HotpotQA benchmarks and the held-out multi-hop datasets, where the average gain reaches **17.76 points**. Comparisons with Raw Ouro further indicate that these gains arise from reinforcement learning rather than the recurrent backbone alone.

The Qwen2.5-3B/7B results provide reference points for established search-agent performance. Since these models differ in both backbone and training setup, we do not use them for algorithmic comparisons; all method-level comparisons are made within matched Ouro settings.

RLTT applies outcome supervision to intermediate readouts while retaining the terminal readout as the behavior policy. ReMPO instead uses the marginal multi-depth policy for both rollout generation and proximal optimization, together with posterior responsibility, a stage-dependent terminal anchor, and gate KL. Across both Ouro scales, ReMPO obtains the highest EM on all seven datasets.

At 2.6B, the All-7 Average increases from 21.99% for Raw Ouro to 39.10% with GRPO and 39.12% with RLTT, while ReMPO reaches 46.37%. This corresponds to a 7.25-point improvement over RLTT. The same ordering is observed at 1.4B. Over 500 training steps on Ouro-1.4B with $B_{\text{search}} = 2$, ReMPO uses approximately $1.48\times$ the H100 GPU-hours of RLTT and $1.59\times$ those of GRPO (Appendix F). Component-wise effects are analyzed in Section 7.2.

6.2 ReMPO learns stable multi-round search

Figure 2 shows the training dynamics and corresponding search behavior over 500 steps. ReMPO reaches the highest and most stable reward level, while RLTT converges to a lower plateau. GRPO deteriorates after step 174, and its search behavior collapses at step 175. After a brief recovery, it produces almost exclusively zero-search trajectories from step 193 onward. Similar instability has been reported in RL for search agents (Jin et al., 2025; Xu et al., 2026).

The three methods also develop different search-count distributions. Early in training, all move from predominantly zero-search trajectories toward one-search behavior. ReMPO subsequently shifts toward two-search trajectories, whereas RLTT remains dominated by one-search trajectories. GRPO's reward collapse coincides with a return to zero-search outputs. Because all methods use the same final-answer reward, prompt, retriever, and two-search budget, this behavior is learned without an explicit reward for additional search calls. Section 7 further analyzes the role of second-search evidence in evaluation trajectories.

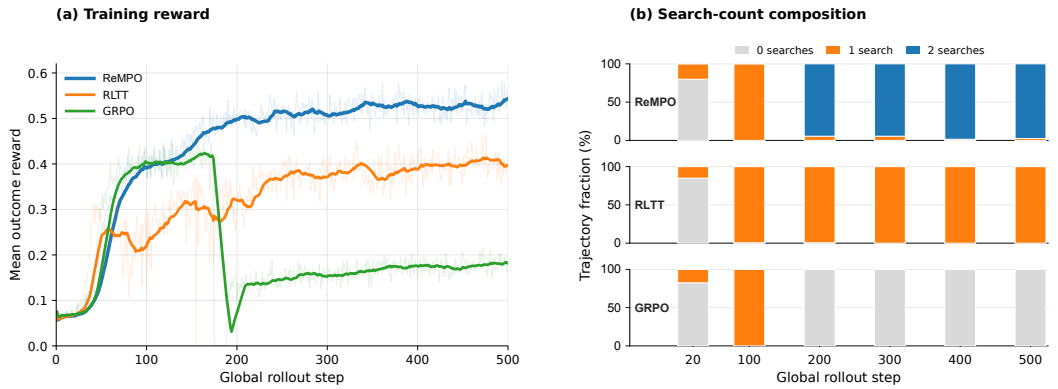

Figure 2: Training dynamics and search-count composition on Ouro-1.4B with a maximum of two searches. (a) Mean final-answer reward over 500 training steps; light lines show raw per-step means and dark lines show smoothed trends. (b) Fractions of training trajectories containing zero, one, or two searches at selected steps.

Table 2: Core ablations on Ouro-1.4B. ID averages NQ and HotpotQA; held-out multi-hop averages 2WikiMultiHopQA, MuSiQue, and Bamboogle; All-7 averages all seven datasets. Δ is the EM difference from Full ReMPO, and parser failure is the percentage of failed evaluation trajectories. Table 10 provides the failure definition, counts, and corresponding checkpoints.

Variant	ID avg.	Held-out multi-hop avg.	All-7 avg.	Δ vs. Full	Parser failure (%) $\downarrow$
GRPO	0.367	0.161	0.323	-0.113	0.15
A1: static responsibility	0.460	0.281	0.403	-0.033	14.79
A2: calibrated fixed gate	0.402	0.203	0.354	-0.082	1.04
A3: no terminal anchor	0.458	0.296	0.410	-0.026	16.67
A4: stage-blind anchor	0.405	0.191	0.351	-0.085	2.07
A5: no gate KL	0.460	0.320	0.426	-0.010	1.82
ReMPO (Full)	**0.468**	**0.340**	**0.436**	0.000	0.95

7 Why Does ReMPO Work?

7.1 How Latent Depth Supports Search Exploration

An additional search is useful when the new evidence improves the expected final answer. After a retrieval, ReMPO can use differences across recurrent readouts when deciding whether to continue searching or answer. Its behavior policy mixes the terminal readout with the depth-marginalized distribution $\widetilde{q}_\theta = \sum_z g_\theta(z \mid s) p_{\theta,z}(\cdot \mid s)$, while the terminal anchor keeps this deviation bounded.

At a fixed state s, hold the readout distributions and continuation policy fixed. Let V_z be the expected final reward of actions sampled from readout z, and $V_g = \sum_z g_z V_z$. The local expected reward is $\mathcal{J}_s = (1-\epsilon)V_K + \epsilon V_g$, where $\epsilon = \epsilon_{\phi(s)}$. Its derivative with respect to the local gate logits ξ is

$$\frac{\partial \mathcal{J}_s}{\partial \xi_z} = \epsilon g_z (V_z - V_g). \tag{13}$$

Thus, the gate places more weight on readouts whose action proposals have higher expected continuation value. In search, this allows final-answer reward to favor readouts that propose useful additional retrievals.

This differs from intermediate-readout credit assignment alone. RLTT extends reward supervision to intermediate readouts but retains terminal-readout rollouts. ReMPO additionally lets intermediate readouts influence the sampled trajectory, allowing alternative searches to be executed and evaluated by final reward. Appendix H provides the full derivation and the binary search–answer analysis.

7.2 Which Algorithmic Components Matter?

Table 2 reports single-run ablations on Ouro-1.4B. The largest accuracy drops come from fixing the learned gate (A2) or removing retrieval-stage conditioning (A4): the All-7 average falls from 43.59%

Case 1 Resolving the director's father (2WikiMultiHopQA).

Question Who is the father of the director of film Yodha (1991 Film)?

Search 1 evidence "Yodha is a 1991 Indian Bollywood film directed by **Rahul Rawail**."

Search 2 query Who is the father of **Rahul Rawail**

Search 2 evidence "Rahul Rawail is son of film director **Harnam Singh Rawail** (often credited as **H. S. Rawail**) [. . .]"

ReMPO answer **H. S. Rawail** — 2 searches; EM = 1

GRPO / RLTT answers Shafi Inamdar — 1 search each; EM = 0

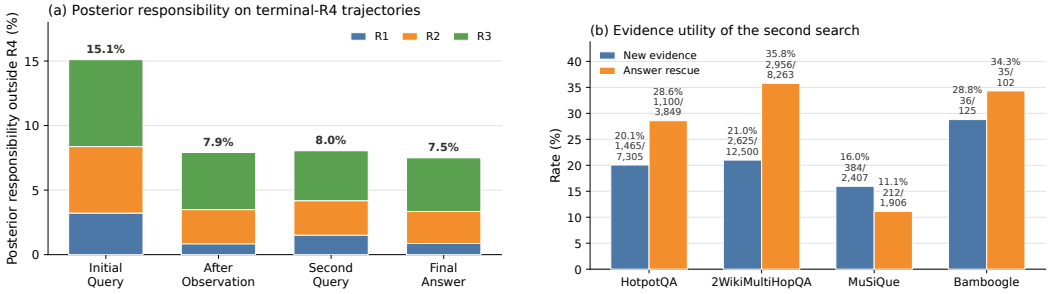

Figure 3: Posterior responsibility and second-search evidence. (a) Posterior responsibility assigned to intermediate readouts R1–R3; R4 is the omitted complementary mass. (b) *New evidence* measures whether the second retrieval adds answer-bearing evidence, and *Answer rescue* measures final strict-EM accuracy when the first retrieval lacks such evidence.

for Full ReMPO to 35.41% and 35.05%, respectively. The degradation is concentrated on multi-hop tasks, with the held-out multi-hop average dropping from 33.97% to 20.29% and 19.09%. These results indicate that both adaptive gating and stage-dependent mixing are important for effective search. Appendix D.3 provides the detailed calibration procedures.

A1 replaces posterior responsibility with the rollout-time anchored prior, while A3 removes the terminal anchor. Both retain relatively strong in-domain accuracy but degrade held-out multi-hop performance and sharply increase parser failures: from 0.95% for Full ReMPO to 14.79% for A1 and 16.67% for A3. This suggests that posterior responsibility and the terminal anchor are important not only for task accuracy but also for stable interaction behavior.

Removing gate KL (A5) has a smaller effect, reducing the All-7 average to 42.55% and increasing parser failures to 1.82%, suggesting that it mainly acts as a supporting regularizer. For reference, the selected GRPO checkpoint achieves 32.27% All-7 with a 0.15% parser-failure rate, while Full ReMPO reaches 43.59% with 0.95% parser failure. Table 9 reports full dataset-level results.

7.3 Posterior responsibility and second-search evidence

Figure 3(a) shows that R4 retains most posterior responsibility, while R1–R3 still contribute non-negligibly: 15.1% during initial-query formation and about 7.5–8.0% after retrieval. This indicates that intermediate readouts remain active in the marginal policy, consistent with the effects isolated by the A1 and A4 ablations.

Figure 3(b) shows that these additional interactions can acquire useful evidence. Across the four multi-hop benchmarks, 16–29% of legal two-search trajectories retrieve a normalized gold-answer alias absent from the first retrieval. Among legal two-search trajectories whose first retrieval misses such an alias, 11–36% end with a strict-EM-correct answer.

Case 1 provides a representative example, where the second query follows an intermediate entity discovered by the first retrieval. Additional examples are given in Appendix G.

7.4 Effect of the search budget

We evaluate ReMPO with search budgets $B_{\text{search}} \in \{1, 2, 3, 4\}$ on Ouro-1.4B, using the same budget during training and evaluation. All runs use the same training data and optimization setup; Appendix E.2 provides the full configuration and dataset-level results.

Increasing the budget from one to two yields the largest gain, raising Macro-7 EM from 36.65% to 43.59%. The improvement is concentrated on multi-hop tasks, with the held-out multi-hop average increasing from 21.41% to 33.97%. Larger budgets do not provide comparable gains: the four-search model reaches 43.38% Macro-7 while using more retrievals and increasing parser failures from 0.95% to 10.51%. Thus, additional interaction capacity is beneficial, but extending the horizon beyond two searches does not translate directly into higher accuracy.

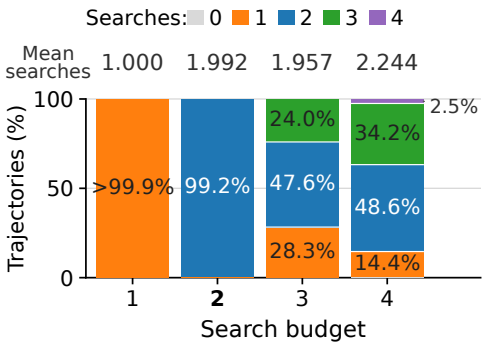

Figure 4: ReMPO search-count distributions on Ouro-1.4B. Means appear above the bars; Table 15 gives exact counts.

Figure 4 further shows that similar mean search counts can correspond to different learned behaviors. The two-search model uses both searches on nearly all trajectories, whereas the three- and four-search models distribute trajectories across different search depths and exhibit higher parser-failure rates. These results suggest that search budget affects not only the amount of interaction, but also the learned search policy and its stability.

Table 3: Search-budget sensitivity of ReMPO on Ouro-1.4B. Macro-7 averages all seven datasets; held-out multi-hop averages 2WikiMultiHopQA, MuSiQue, and Bamboogle. $B_{\text{search}} = 2$ is the main setting, and $B_{\text{search}} = 3$ uses an independent seed.

B_{search}	Macro-7	Held-out multi-hop	Mean searches	Parser failure (%)
1	0.367	0.214	1.000	0.51
2	0.436	0.340	1.992	0.95
3	0.405	0.289	1.957	8.88
4	0.434	0.333	2.244	10.51

8 Limitations

Our experiments are limited to Ouro-1.4B/2.6B on search-based open-domain QA, with the main setting capped at two retrievals; longer-budget runs show higher parser-failure rates, so generalization to other agents or long tool horizons remains unverified. ReMPO also relies on Ouro's recurrent gating mechanism, requires all recurrent loops to execute before depth selection, and therefore offers no inference-speedup guarantee. Evaluation uses a fixed retriever and exact-match reward, which does not capture citation faithfulness or explanation quality. Finally, seed coverage is limited, and the reported Qwen2.5-3B/7B results provide task context rather than a matched algorithmic comparison.

9 Conclusion

We study outcome-only reinforcement learning for looped language models in multi-turn search and introduce ReMPO, which incorporates recurrent depth into a unified behavior policy for both action generation and optimization. In experiment, ReMPO consistently outperforms GRPO and RLTT across seven datasets and two model sizes. Ablations and behavioral analyses further show that adaptive depth selection improves both task performance and interaction reliability, while learned second searches can acquire useful new evidence. These results support treating recurrent depth not only as a target of credit assignment, but also as part of the policy that generates behavior.

AI USE STATEMENT

Generative AI tools were used to assist with paper outlining, drafting, polishing, and LaTeX organization. The authors reviewed all AI-assisted text, equations, citations, and artifacts and take responsibility for the final content.

ETHICS STATEMENT

This work uses publicly described question-answering datasets, pretrained models, and retrieval infrastructure and does not involve human-subject data collection. Reinforcement-learned search agents may retrieve incomplete or misleading evidence, so their answers should not be treated as verified factual advice without source inspection. Training GPU-hours are reported in Appendix F.

REPRODUCIBILITY STATEMENT

Appendices A–I provide additional methodological details, experimental settings, and supporting analyses.

REFERENCES

Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021. URL https://arxiv.org/abs/2107.05407.

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. Research: Learning to reason with search for llms via reinforcement learning. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2503.19470.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1807.03819.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1380. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3b01972cf31e6fa0fe29e4b8b5c2a0a1-Abstract-Conference.html.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Itxz7S4Ip3.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Runjing Liu, Jeffrey Regier, Nilesh Tripuraneni, Michael I. Jordan, and Jon McAuliffe. Rao-blackwellized stochastic gradients for discrete distributions. In *Proceedings of the 36th International Conference on Machine Learning*, 2019. URL https://arxiv.org/abs/1810.04777.

Andrei Cristian Popescu, Haitz Sáez de Ocáriz Borde, and Pietro Liò. Looped language models improve compositional tool calling. arXiv preprint arXiv:2608.18171, 2026. URL https://arxiv.org/abs/2608.18171.

Hangliang Ren. LSRL: Process-supervised GRPO on latent recurrent states improves mathematical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 12534–12545, Suzhou, China, 2025. Association for Computational Linguistics.

doi: 10.18653/v1/2025.findings-emnlp.669. URL https://aclanthology.org/2025.findings-emnlp.669/.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, 2015. URL https://arxiv.org/abs/1506.05254.

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. URL https://arxiv.org/abs/2402.03300.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.36. URL https://aclanthology.org/2025.emnlp-main.36/.

Matthew Smith, Herke van Hoof, and Joelle Pineau. An inference-based policy gradient method for learning options. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4703–4712. PMLR, 2018. URL https://proceedings.mlr.press/v80/smith18a.html.

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. URL https://arxiv.org/abs/2503.05592.

Guo Tang, Shixin Jiang, Heng Chang, Nuo Chen, Yuhan Li, Huiming Fan, Jia Li, Ming Liu, and Bing Qin. Looprpt: Reinforcement pre-training for looped language models. *arXiv preprint arXiv:2603.19714*, 2026. URL https://arxiv.org/abs/2603.19714.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10014–10037, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.557. URL https://aclanthology.org/2023.acl-long.557/.

Jonathan Williams and Esin Tureci. Prioritize the process, not just the outcome: Rewarding latent thought trajectories improves reasoning in looped language models. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.10520.

Markus Wulfmeier, Abbas Abdolmaleki, Roland Hafner, Jost Tobias Springenberg, Michael Neunert, Noah Siegel, Tim Hertweck, Thomas Lampe, Nicolas Heess, and Martin Riedmiller. Compositional transfer in hierarchical reinforcement learning. In *Proceedings of Robotics: Science and Systems*, 2020. doi: 10.15607/RSS.2020.XVI.054. URL https://roboticsproceedings.org/rss16/p054.html.

Yinuo Xu, Shuo Lu, Jianjie Cheng, Meng Wang, Qianlong Xie, Xingxing Wang, Ran He, and Jian Liang. How to train your deep research agent? prompt, reward, and policy optimization in search-r1. *arXiv preprint arXiv:2602.19526*, 2026. URL https://arxiv.org/abs/2602.19526.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/b8402301e7f06bdc97a31bfaa653dc32-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2210.03629.

Zhenrui Yue, Bowen Jin, Huimin Zeng, Honglei Zhuang, Zhen Qin, Jinsung Yoon, Lanyu Shang, Jiawei Han, and Dong Wang. Hybrid latent reasoning via reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0195. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/087f7678af2abbe0c37ecc30bd7d1adf-Abstract-Conference.html.

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*, 2025. URL https://arxiv.org/abs/2510.25741.

APPENDIX

A FULL DERIVATION

A.1 ANCESTRAL SAMPLING EQUALS THE MARGINAL POLICY

For any token a and state s,

$$\Pr(a \mid s) = \sum_{z=1}^{K} \Pr(z \mid s) \Pr(a \mid s, z) \tag{14}$$

$$= \sum_{z=1}^{K} \alpha_{\theta_{\text{old}}}(z \mid s) p_{\theta_{\text{old}},z}(a \mid s) = q_{\theta_{\text{old}}}(a \mid s). \tag{15}$$

ReMPO forms its proximal ratio from the marginal probability of the visible token. To relate this ratio to a latent-action objective, define the joint policy and its ratio at a fixed state s:

$$\pi_\theta(z, a \mid s) = \alpha_\theta(z \mid s) p_{\theta,z}(a \mid s),$$

$$r_{\text{joint}}(z, a \mid s) = \frac{\alpha_\theta(z \mid s) p_{\theta,z}(a \mid s)}{\alpha_{\theta_{\text{old}}}(z \mid s) p_{\theta_{\text{old}},z}(a \mid s)}.$$

Using the old-policy posterior $\gamma_{\text{old}}(z \mid s, a)$ from Eq. (9), the marginal ratio satisfies

$$\sum_z \gamma_{\text{old}}(z \mid s, a) r_{\text{joint}}(z, a \mid s) = \sum_z \frac{\alpha_{\theta_{\text{old}}}(z \mid s) p_{\theta_{\text{old}},z}(a \mid s)}{q_{\theta_{\text{old}}}(a \mid s)} \frac{\alpha_\theta(z \mid s) p_{\theta,z}(a \mid s)}{\alpha_{\theta_{\text{old}}}(z \mid s) p_{\theta_{\text{old}},z}(a \mid s)}$$

$$= \frac{q_\theta(a \mid s)}{q_{\theta_{\text{old}}}(a \mid s)}.$$

Thus, the marginal ratio averages the joint ratios over the old posterior conditioned on the observed token. A component-only ratio $p_{\theta,z}/p_{\theta_{\text{old}},z}$ omits the gate ratio. It equals the joint ratio when the corresponding gate probabilities are unchanged. ReMPO first marginalizes and then applies the clipped surrogate in Eq. (8). Applying that surrogate to a sampled joint ratio generally gives a different objective because clipping is nonlinear.

A.2 POSTERIOR RESPONSIBILITY IDENTITY

Let $\chi_z = \log \alpha_\theta(z \mid s) + \log p_{\theta,z}(a \mid s)$. Then

$$\log q_\theta(a \mid s) = \log \left(\sum_{z=1}^{K} \exp \chi_z \right), \tag{16}$$

and

$$\frac{\partial \log q_\theta(a \mid s)}{\partial \chi_z} = \frac{\exp \chi_z}{\sum_j \exp \chi_j} = \gamma(z). \tag{17}$$

At a fixed observed prefix s, the chain rule gives the full parameter gradient:

$$\nabla_\theta \log q_\theta(a \mid s) = \sum_z \frac{\partial \log q_\theta(a \mid s)}{\partial \chi_z} \nabla_\theta \chi_z$$

$$= \sum_z \gamma(z) \left[\nabla_\theta \log \alpha_\theta(z \mid s) + \nabla_\theta \log p_{\theta,z}(a \mid s) \right].$$

Each derivative includes the paths through the shared recurrent backbone. The posterior is the coefficient of a component's log-score contribution. The resulting parameter gradient also depends on the direction and magnitude of that contribution. The learner sums over all four depths exactly.

For a token whose clipped surrogate selects the differentiable $r_t A_i$ branch, holding the stored old-policy probability and advantage fixed gives

$$\nabla_\theta (r_t A_i) = A_i r_t \nabla_\theta \log q_\theta(a_t \mid s_t).$$

The policy loss multiplies this expression by $-m_{i,t} / \sum_{j,u} m_{j,u}$. On the constant clipped branch, the token's policy-loss gradient is zero away from the branch boundary. Posterior responsibility therefore allocates the marginal score within the active PPO update.

A.3 STATIC-RESPONSIBILITY ABLATION

A1 uses a surrogate log-probability with the same forward value as $\ell_t^\theta = \log q_\theta(a_t \mid s_t)$. Let sg denote stop-gradient and define

$$\rho_t(z) = \mathrm{sg}[\alpha_{\mathrm{old}}(z \mid s_t)]\,,$$
$$b_t(\theta) = \sum_z \rho_t(z)\,[\log \alpha_\theta(z \mid s_t) + \log p_{\theta,z}(a_t \mid s_t)]\,.$$

The learner replaces the differentiable log-probability with

$$\widetilde{\ell}_t^\theta = \mathrm{sg}\big[\ell_t^\theta\big] + b_t(\theta) - \mathrm{sg}[b_t(\theta)]\,.$$

Its forward value and the resulting proximal ratio are unchanged at fixed parameters and observed tokens. In the backward pass, ρ_t weights both the gate and readout log-score paths. Each stored prior depends on the rollout context and retrieval stage and is held fixed during learner updates. This replacement tests token-conditioned posterior responsibility while retaining the marginal sampling rule and forward probabilities.

B ALGORITHM AND TENSOR SEMANTICS

Algorithm 1 ReMPO training.

Input: Raw Ouro parameters θ, frozen reference θ_{ref}, question pool $\mathcal{D}$, group size G, stage coefficients (ϵ_0, ϵ_1).
Output: Trained policy parameters θ.

for each training step **do**
  Snapshot $\theta_{\mathrm{old}} \leftarrow \theta$ and sample a question batch from $\mathcal{D}$.
  for each question in the batch **do**
    Generate G trajectories, alternating model tokens and retrieval observations. At each policy token, sample $z_t \sim \alpha_{\mathrm{old}}(\cdot \mid s_t)$, then $a_t \sim p_{\mathrm{old},z_t}(\cdot \mid s_t)$.
    Record retrieval stages and mask out observations, prompts, and padding.
    Parse model-generated answers and assign terminal EM rewards.
    Compute group-relative advantages using Eq. (1).
  end for
  Store $\log q_{\mathrm{old}}(a_t \mid s_t)$ and α_{old} at valid policy tokens.
  for each PPO mini-batch update **do**
    Recompute current per-depth chosen-token log-probabilities and α_θ.
    Compute the marginal ratio and posterior responsibility using Eqs. (7)–(9).
    Evaluate the masked policy loss, gate KL, and frozen-reference penalty.
    Update θ to minimize the total loss in Eq. (12).
  end for
  Save and evaluate checkpoints at the common global steps used for all methods.
end for

Tensor layout. Let B be the learner micro-batch size, T the response-layout length, M the number of valid policy tokens, V the vocabulary size, and $K = 4$ the number of recurrent loops. The learner retains chosen-token scores across depths; full-vocabulary logits are released loop by loop.

Table 4: Core tensor semantics.

Tensor	Shape	Meaning
Policy-token data	$[B, T]$	Responses, advantages, old log-probabilities, and mask
Retrieval stage	$[B, T]$	Before/after retrieval indicator
Old/current depth distribution	$[B, T, K]$	α_{old} and α_θ
Chosen-token log-probabilities	$[K, B, T]$	Per-readout scores of realized tokens
Posterior responsibility	$[B, T, K]$	$\gamma_t(z)$
Marginal log-probability and ratio	$[B, T]$	$\log q$ and $q_\theta / q_{\mathrm{old}}$
Conceptual loop logits	$[K, M, V]$	Full-vocabulary scores, released loop by loop

C Strict Baseline Adaptation

GRPO uses

$$r_t^{\text{term}}(\theta) = \frac{p_{\theta,K}(a_t \mid s_t)}{p_{\theta_{\text{old}},K}(a_t \mid s_t)} \tag{18}$$

inside the clipped PPO surrogate. RLTT applies the same trajectory advantage to a weighted collection of loop log-probabilities:

$$\mathcal{L}_{\text{RLTT}} = -\frac{1}{\sum_{i,t} m_{i,t}} \sum_{i,t} m_{i,t} A_i \sum_{z=1}^{K} w_{i,t,z} \log p_{\theta,z}(a_{i,t} \mid s_{i,t}) + \beta_{\text{ref}} \mathcal{L}_{\text{KL}}. \tag{19}$$

Here i indexes trajectories and t indexes token positions; $\mathcal{L}_{\text{KL}}$ denotes the reference-policy KL regularization term for this baseline. We use uniform loop weighting for the RLTT baseline, with $w_{i,t,z} = 1/K = 1/4$ for every policy token. The original RLTT study reports broadly comparable performance across uniform, progressive, and exit-probability weighting on its reasoning benchmarks (Williams & Tureci, 2026, Appendix A.3).

Table 5: Baseline objectives and shared experimental settings for adapting GRPO and RLTT to search-based question answering.

Component	GRPO	RLTT
Behavior rollout	terminal R_4	terminal R_4
Outcome signal	five-sample group-normalized final EM	same group-normalized final EM
Loop objective	terminal clipped ratio	official multi-readout weighted surrogate
Observation mask	prompt/observation/padding excluded	excluded from every loop loss and KL
Search environment	same parser, retriever, top-3, and two-search limit	same
Compatibility changes	Ouro interface and terminal readout	Ouro per-loop readouts plus Search-R1 trajectory adapter
Preserved scientific variable	fixed terminal policy	uniform loop weighting and RLTT gradient semantics

D Training and Evaluation Details

Table 6: Canonical training configuration.

Field	Value
Models	Ouro-1.4B-Thinking; Ouro-2.6B-Thinking
Train data	NQ + HotpotQA
Global questions / step	512
Trajectories / question	5
Maximum global steps	500
Maximum search actions	2
Retriever top-k	3
Maximum response/observation length	500 / 500
Depth count	4
Stage coefficients	$(\epsilon_0, \epsilon_1) = (0.25, 0.15)$
Gate KL coefficient	0.01
Reference KL coefficient / estimator	0.001 / `low_var_kl`
PPO clip	0.2
Learning rate / warmup	10^{-6} / 0.285
Optimizer / precision	AdamW / BF16
Primary decode	deterministic terminal R_4

Checkpoint selection. All methods are evaluated at the same global steps. We select one checkpoint per seed by the highest validation EM on a fixed set of 200 examples held out from the training split; test sets are evaluated only after selection.

Seven-dataset evaluation. The evaluator reports response-only exact match on NQ, HotpotQA, TriviaQA, PopQA, 2WikiMultiHopQA, MuSiQue, and Bamboogle. 'All-7 Average' is the unweighted mean of the seven dataset-level EM values. 'Held-out multi-hop Average' is the unweighted mean of 2WikiMultiHopQA, MuSiQue, and Bamboogle. Average searches, generated policy tokens, and protocol failure are reported separately from accuracy.

D.1 Frozen-reference regularization

The reference model has frozen parameters and uses the actor variant's prior and anchor configuration. At the observed prefix s_t, it constructs

$$\alpha_{\mathrm{ref}}(z \mid s_t) = \epsilon_{\phi(s_t)} g_{\mathrm{ref}}(z \mid s_t) + (1 - \epsilon_{\phi(s_t)}) \mathbb{I}[z = K],$$
$$q_{\mathrm{ref}}(a \mid s_t) = \sum_{z=1}^{K} \alpha_{\mathrm{ref}}(z \mid s_t) p_{\mathrm{ref},z}(a \mid s_t).$$

For Full ReMPO and A1, the frozen model computes g_{ref} from the current prefix and applies $(\epsilon_0, \epsilon_1) = (0.25, 0.15)$. A4 uses the same construction with $(0.192, 0.192)$. In A2, both actor and reference replace the native gate with g_{fixed} from Appendix D.3 and apply $(0.25, 0.15)$. The reference computes its probabilities independently of the stored actor prior α_{old}.

For each model-generated rollout token, define the log-probability difference

$$d_{i,t} = \log q_{\mathrm{ref}}(a_{i,t} \mid s_{i,t}) - \log q_\theta(a_{i,t} \mid s_{i,t}).$$

Let a PPO mini-batch of N sequences be partitioned into dynamic micro-batches. Let $\mathcal{B}_b$ denote the sequence-index set of micro-batch b, with $n_b = |\mathcal{B}_b|$ sequences and $M_b = \sum_{i \in \mathcal{B}_b} \sum_t m_{i,t}$ valid policy tokens. The reference term in Eq. (12) is

$$\mathcal{L}_{\mathrm{ref}}(q_\theta, q_{\mathrm{ref}}) = \sum_b \frac{n_b}{N} \frac{1}{M_b} \sum_{i \in \mathcal{B}_b} \sum_t m_{i,t} \left[\exp(d_{i,t}) - d_{i,t} - 1 \right]. \tag{20}$$

The mask selects model-generated tokens and excludes prompts, retrieved observations, and padding. The coefficient of this term is $\beta_{\mathrm{ref}} = 0.001$. Its token values are evaluated on rollouts from $q_{\theta_{\mathrm{old}}}$ without advantage weighting or additional importance-ratio weighting. The implementation uses `low_var_kl` with no clipping of $d_{i,t}$, its exponential, or the resulting penalty. The gate probability floor is zero. Reference log-probabilities are computed with gradients disabled, and the regularizer is differentiated through the current $\log q_\theta$.

D.2 Terminal and marginal decoding comparison

We compare two decoding rules using the same frozen, validation-selected Ouro-1.4B ReMPO checkpoint. Both evaluations cover the same 51,713 questions across seven datasets, with identical prompts, retriever, parser, a two-search budget, and generation and context limits. Terminal-R_4 greedy decoding selects the highest-probability token under $p_{\theta,K}$, whereas marginal greedy decoding selects it under q_θ. The marginal retains the checkpoint's learned gate and retrieval-stage anchor settings. Both rules execute all four recurrent loops.

Table 7: Terminal and marginal greedy decoding of the same Ouro-1.4B ReMPO checkpoint. EM uses a 0–1 scale with three decimal places. All-7 is the unweighted mean across the seven datasets, computed before rounding. Mean searches is averaged over evaluation trajectories; parser failure is reported as a percentage.

Benchmark / metric	R_4-greedy	q-greedy
Exact match		
NQ	0.485	0.486
HotpotQA	0.452	0.451
TriviaQA	0.631	0.631
PopQA	0.464	0.464
2WikiMultiHopQA	0.427	0.428
MuSiQue	0.177	0.180
Bamboogle	0.416	0.432
All-7 average	0.436	0.439
Search behavior		
Mean searches	1.992	1.990
Parser failure (%)	0.95	1.01

Table 7 shows a small increase in All-7 EM with marginal decoding, while average search counts and parser-failure rates remain close. The main search experiments use terminal-R_4 decoding throughout.

D.3 Ablation configurations and calibration

Full ReMPO and all five ablations start from the same raw Ouro-1.4B checkpoint with training seed 20260805. Each run completes 500 global steps using the shared data, reward, optimizer, retriever, and interaction limits described above. Training samples from the marginal policy at temperature 1 without top-p or top-k truncation, and all test results use deterministic terminal-R_4 decoding with at most two searches. Table 8 records the modified components and validation-selected checkpoints. A2 attains its best validation score at step 300, which supplies its reported test row.

Table 8: Configurations and selected checkpoints for Full ReMPO and A1–A5. Gate KL lists the penalty coefficient where active. The fixed prior makes the A2 gate KL term zero.

Variant	Native prior g	Responsibility	(ϵ_0, ϵ_1)	Gate KL	Step
Full	Learned	γ_t	$(0.25, 0.15)$	0.01	400
A1	Learned	$\alpha_{\mathrm{old},t}$	$(0.25, 0.15)$	0.01	500
A2	Calibrated fixed	γ_t	$(0.25, 0.15)$	0	300
A3	Learned	γ_t	$(1, 1)$	0.01	400
A4	Learned	γ_t	$(0.192, 0.192)$	0.01	500
A5	Learned	γ_t	$(0.25, 0.15)$	0	400

A1: static responsibility. A1 keeps the learned gate and stage-dependent anchor. The learner uses the anchored prior saved for each rollout token as its backward responsibility, following Appendix A. The weights are fixed during the update and retain their dependence on the preceding context. Gate KL remains active with coefficient 0.01, and the reference penalty retains coefficient 0.001.

A2: calibrated fixed gate. The fixed native prior is calibrated from Full ReMPO at step 400. Teacher-forced replay of deterministic terminal-R_4 trajectories for 200 frozen validation questions yields 597 model actions and 15,338 response-only policy tokens. We average the native, pre-anchor gate distributions with equal weight per policy token, excluding retrieved observation and environment tokens. This produces

$$g_{\mathrm{fixed}} = (0.10866105,\ 0.21712378,\ 0.29279166,\ 0.38142351).$$

A2 uses this distribution throughout training and applies the original stage coefficients through Eq. (3), with gate probability floor zero. The posterior in Eq. (9) remains token-conditioned. The

fixed prior replaces the current native gate in both rollout and learner computations, so the native gate receives no direct policy gradient through the prior. Gate KL contributes zero because the prior is fixed within each stage.

A4: stage-blind anchor. A4 retains the learned native gate, exact posterior responsibility, and gate KL. It sets $\epsilon_0 = \epsilon_1 = 0.192$. Across Full's 499 finite training-step metric records, 41.99% of model-generated token weight occurs before retrieval and 58.01% after retrieval. Weighting 0.25 and 0.15 by these proportions yields $\bar{\epsilon} \simeq 0.192$. This calibration matches the mean mixing coefficient on Full's recorded stage distribution. Both stage labels map to the same coefficient in A4, while the native gate continues to depend on the current context.

E ADDITIONAL RESULT TABLES

Table 9: Complete single-run ablation results on Ouro-1.4B, reported as EM on a 0–1 scale. Macro-2 averages NQ and HotpotQA; Macro-5 averages the remaining five datasets; Macro-7 averages all seven. Macro-5 includes TriviaQA and PopQA in addition to the three held-out multi-hop datasets. Bold marks the highest EM in each column before rounding.

Variant	NQ	HotpotQA	TriviaQA	PopQA	2Wiki	MuSiQue	Bamboogle
ReMPO (Full)	**0.485**	**0.452**	0.631	**0.464**	**0.427**	**0.177**	**0.416**
A1: static responsibility	0.479	0.442	0.618	0.439	0.375	0.141	0.328
A2: fixed gate	0.462	0.341	0.615	0.452	0.319	0.090	0.200
A3: no terminal anchor	0.481	0.436	0.625	0.439	0.340	0.165	0.384
A4: stage-blind anchor	0.469	0.342	0.618	0.452	0.303	0.094	0.176
A5: no gate KL	0.480	0.440	**0.634**	0.464	0.419	0.173	0.368

Variant	Macro-2	Macro-5	Macro-7
ReMPO (Full)	**0.468**	**0.423**	**0.436**
A1: static responsibility	0.460	0.380	0.403
A2: fixed gate	0.402	0.335	0.354
A3: no terminal anchor	0.458	0.391	0.410
A4: stage-blind anchor	0.405	0.329	0.351
A5: no gate KL	0.460	0.412	0.426

Table 10: Parser failures for main-comparison and ablation checkpoints. Every row contains 51,713 trajectories, and a failure is recorded when `termination_reason=parser_failure`. Rates pool trajectories across the seven datasets.

Method	Step	Parser failures	Rate (%)
GRPO	100	76	0.15
RLTT	400	4,406	8.52
Full ReMPO	400	493	0.95
A1: static responsibility	500	7,647	14.79
A2: calibrated fixed gate	300	538	1.04
A3: no terminal anchor	400	8,620	16.67
A4: stage-blind anchor	500	1,068	2.07
A5: no gate KL	400	940	1.82

E.1 SENSITIVITY TO TRAINING SEEDS

We evaluate ReMPO, GRPO, and RLTT on Ouro-1.4B with training seeds 20260805, 20260912, and 20260913, using a two-search budget and deterministic terminal-R_4 decoding. Seed 20260805 supplies the main-table rows. Tables 11 and 12 give per-seed EM and summary statistics. Ouro-2.6B uses one training run per method because of training cost.

Table 11: Ouro-1.4B results by training seed. EM uses a 0–1 scale with three decimal places. All-7 is the unweighted mean across the seven datasets. Rows marked † are the runs reported in Table 1.

Seed	NQ	HotpotQA	TriviaQA	PopQA	2Wiki	MuSiQue	Bamboogle	All-7
ReMPO								
20260805†	0.485	0.452	0.631	0.464	0.427	0.177	0.416	0.436
20260912	0.482	0.456	0.628	0.463	0.425	0.166	0.408	0.432
20260913	0.483	0.453	0.627	0.464	0.425	0.169	0.408	0.433
GRPO								
20260805†	0.434	0.300	0.609	0.434	0.287	0.068	0.128	0.323
20260912	0.426	0.296	0.598	0.427	0.278	0.061	0.152	0.320
20260913	0.428	0.294	0.598	0.427	0.277	0.065	0.120	0.315
RLTT								
20260805†	0.438	0.320	0.583	0.432	0.295	0.072	0.120	0.323
20260912	0.414	0.295	0.543	0.403	0.278	0.058	0.104	0.299
20260913	0.408	0.299	0.542	0.405	0.275	0.060	0.088	0.297

Table 12: Mean and sample standard deviation across the training seeds listed in Table 11. Statistics are computed before rounding, with standard deviation using denominator $n-1$. Held-out multi-hop averages 2WikiMultiHopQA, MuSiQue, and Bamboogle within each run.

Method	Seeds (n)	All-7 EM	Held-out multi-hop EM
ReMPO	3	0.434 ± 0.002	0.335 ± 0.004
GRPO	3	0.319 ± 0.004	0.159 ± 0.005
RLTT	3	0.306 ± 0.014	0.150 ± 0.011

Across three seeds, ReMPO's All-7 EM is 0.434 ± 0.002, and it exceeds both baselines on every dataset. GRPO has higher All-7 EM than RLTT for both additional seeds, whereas their main-table results are nearly tied.

Table 13: Search behavior and parser failures for the two additional Ouro-1.4B training seeds. Searches are averaged over evaluation trajectories; parser failure is reported as a percentage.

	Seed 20260912		Seed 20260913	
Method	Avg. searches	Parser failure (%)	Avg. searches	Parser failure (%)
ReMPO	1.991	0.37	1.990	0.42
GRPO	1.000	0.31	1.000	0.29
RLTT	0.987	24.65	0.987	24.47

Table 13 shows that both additional ReMPO runs retain nearly two searches per trajectory with parser-failure rates below 0.5%. GRPO averages one search with low parser-failure rates. RLTT also averages close to one search, but approximately one quarter of its trajectories end in parser failure. Average search count alone therefore does not capture protocol reliability.

E.2 SEARCH-BUDGET RESULTS

Table 14 reports the dataset-level EM values underlying Section 7.4, with each model evaluated under its training search budget. The three-search budget uses the independently seeded run selected at step 300 by the frozen 200-question validation evaluation before seven-dataset testing. The one-, two-, and four-search rows use step 400.

Table 16 separates parser failure, turn-budget exhaustion, and generation truncation. The four-search run has the highest parser-failure rate and the lowest generation-truncation rate, while turn-budget exhaustion is most frequent in the independent three-search run. These event rates describe distinct aspects of the generated trajectories.

Table 14: Seven-dataset search-budget results for ReMPO on Ouro-1.4B, reported as EM on a 0–1 scale. Macro-7 is the unweighted mean across datasets. Each row represents one run; the three-search row is the independent-seed run. Bold marks the highest EM in each column before rounding.

B_{search}	NQ	HotpotQA	TriviaQA	PopQA	2Wiki	MuSiQue	Bamboogle	Macro-7
1	0.478	0.353	**0.633**	0.459	0.330	0.096	0.216	0.367
2	**0.485**	0.452	0.631	0.464	**0.427**	0.177	**0.416**	**0.436**
3	0.478	0.421	0.622	0.448	0.345	0.138	0.384	0.405
4	0.483	**0.456**	0.630	**0.468**	0.410	**0.198**	0.392	0.434

Table 15: Search-count breakdown over all 51,713 evaluation trajectories per run. Counts include unsuccessful trajectories. A zero denotes no observed trajectory with that search count.

B_{search}	0 searches	1 search	2 searches	3 searches	4 searches	Mean
1	1	51,712	0	0	0	1.000
2	1	430	51,282	0	0	1.992
3	2	14,634	24,641	12,436	0	1.957
4	113	7,448	25,146	17,708	1,298	2.244

Table 16: Reported protocol-event rates (%) in the search-budget evaluations. The columns retain their individual event definitions.

B_{search}	Parser failure	Turn-budget exhaustion	Generation truncation
1	0.51	0.00	0.50
2	0.95	0.01	0.95
3	8.88	3.53	1.48
4	10.51	0.16	0.02

F Compute Accounting

Table 17 reports cumulative training cost for Ouro-1.4B with a two-search budget. Each method runs for 500 training steps on 16 NVIDIA H100 GPUs. GPU-hours cover the full 500-step run; the accuracy results in Table 1 use validation-selected checkpoints.

Table 17: Training cost for Ouro-1.4B with $B_{\text{search}} = 2$ on 16 NVIDIA H100 GPUs. GPU-hours are approximate measured totals over 500 training steps.

Method	Training steps	H100 GPU-hours
GRPO	500	140.1
RLTT	500	150.6
ReMPO	500	222.6

G Mechanism Analysis and Successful Search Cases

Posterior analysis on terminal trajectories. Figure 3(a) uses 200 validation trajectories generated with deterministic terminal-R_4 decoding by the Ouro-1.4B ReMPO checkpoint at step 400. We replay the recorded trajectories with teacher forcing and compute the posterior responsibility $\gamma_t(z)$ under q_θ using Eq. (9). The analysis scores the recorded tokens under their observed prefixes and groups their posterior responsibilities by agent stage. R1–R3 retain their absolute posterior mass, with R4 supplying the complement.

Posterior responsibility characterizes the marginal model's attribution of the recorded tokens. Component effects on task accuracy are evaluated separately by the ablations in Table 2.

Successful second-search trajectories. The three cases come from the Ouro-1.4B ReMPO step-400 evaluation with $B_{\text{search}} = 2$ and deterministic terminal-R_4 decoding. We select successful two-search trajectories whose first observation lacks a normalized gold-answer alias and whose second query uses an intermediate entity from the first observation. Queries and answers retain the recorded wording; evidence is excerpted from the retrieved observations, with omissions marked by [...]. Blue highlights the intermediate entity carried into the second query, and green highlights the answer. Same-question comparisons use GRPO step 100 and RLTT step 400.

Case 1 uses sample `2wikimultihopqa:dev_9379`, with gold aliases "Harnam Singh Rawail" and "H. S. Rawail". Its first query repeats the question without the final question mark. The first observation also contains a plot description naming actor Shafi Inamdar; both baselines return this name after one search. ReMPO's second query instead follows the director relation to Rahul Rawail's father.

Case 2 Following a family relation (2WikiMultiHopQA, `dev_7001`).

Question When did Camillo Ruspoli, Duke Of Sueca's father die?

Search 1 query When did Camillo Ruspoli, Duke Of Sueca's father die

Search 1 evidence "He was born in Rome on March 20, 1788, the third son of **Francesco Ruspoli, 3rd Prince of Cerveteri**, and Countess Maria Leopoldina von Khevenhüller-Metsch."

Search 2 query When did **Francesco Ruspoli, 3rd Prince of Cerveteri** die

Search 2 evidence "Francesco, Prince Ruspoli, [...] (February 19, 1752 – **March 8, 1829**) [...]"

ReMPO answer / gold **March 8, 1829** — 2 searches; EM = 1

GRPO / RLTT answers July 30, 1864 — 1 search each; EM = 0

Both baselines return Camillo's own death date; ReMPO follows the father relation before answering.

Case 3 From a village to its county seat (HotpotQA, `dev_80`).

Question What is the county seat of the county where East Lempster, New Hampshire is located?

Search 1 query What is the county seat of the county where East Lempster, New Hampshire is located

Search 1 evidence "Lempster is a town in **Sullivan County, New Hampshire**, United States. [...] Settlements include the town center (Lempster village) and the village of East Lempster."

Search 2 query What is the county seat of **Sullivan County, New Hampshire**

Search 2 evidence "Its county seat is **Newport**."

ReMPO answer / gold **Newport**	2 searches; EM = 1
GRPO answer Sullivan	1 search; EM = 0
RLTT outcome Parser failure; no parsed answer	1 search; EM = 0

H Local Analysis of Search Exploration

H.1 Continuation value and information from retrieval

For a fixed continuation policy π, define $Q^\pi(s,a) = \mathbb{E}[R(\tau) \mid s, a; \pi]$, where a is a model-generated token and the expectation includes subsequent tokens and retrieval transitions. The only reward is the terminal response-only exact match defined in Section 3. A search can therefore affect expected reward through the evidence and future states it produces.

For the local search/answer reduction, let H_1 be a history following the first retrieval. Use $\mathcal{Q}^\pi_{\mathrm{S}}(H_1)$ and $\mathcal{Q}^\pi_{\mathrm{A}}(H_1)$ for the expected final rewards of two specified continuation branches: attempting another search and answering directly. The search branch includes query generation, retrieval, and the subsequent answer, including unsuccessful execution. Its value difference is

$$\Delta^\pi_2(H_1) = \mathcal{Q}^\pi_{\mathrm{S}}(H_1) - \mathcal{Q}^\pi_{\mathrm{A}}(H_1). \tag{21}$$

These branch values describe a local abstraction of the token policy. A search call in the implemented agent is produced by a sequence of tokens parsed by the environment.

Lemma (Value of additional information). Let Y denote the correct normalized answer class in a finite answer space, and let O_2 be a second retrieval observation under a fixed query rule. An ideal answerer that maximizes expected exact match after observing O_2 has information value

$$\mathrm{VoI}_2(H_1) = \mathbb{E}_{O_2|H_1}\left[\max_y P(Y = y \mid H_1, O_2)\right] - \max_y P(Y = y \mid H_1) \geq 0. \tag{22}$$

Proof. The maximum can be taken after seeing the observation rather than before it:

$$\begin{aligned}\mathbb{E}_{O_2|H_1}\left[\max_y P(Y = y \mid H_1, O_2)\right] &\geq \max_y \mathbb{E}_{O_2|H_1}[P(Y = y \mid H_1, O_2)] \\ &= \max_y P(Y = y \mid H_1).\end{aligned}$$

The last equality follows from conditional expectation. This establishes the value of information when the answerer can optimally use or ignore it. For a learned search agent, query quality, evidence use, and execution reliability determine the sign of Δ^π_2.

H.2 Behavioral change under a terminal anchor

Fix a prefix s and write $\epsilon = \epsilon_{\phi(s)}$. The marginal policy has the form

$$q_\theta = (1-\epsilon)p_{\theta,K} + \epsilon\widetilde{q}_\theta, \qquad \widetilde{q}_\theta = \sum_z g_\theta(z \mid s)p_{\theta,z}.$$

All distributions in this subsection are conditional on s.

Proposition (Anchored behavior bounds). For any token set $\mathcal{U}$ and $0 \leq \epsilon < 1$,

$$q_\theta(\mathcal{U}) - p_{\theta,K}(\mathcal{U}) = \epsilon\big[\widetilde{q}_\theta(\mathcal{U}) - p_{\theta,K}(\mathcal{U})\big], \tag{23}$$

$$\mathrm{TV}(q_\theta, p_{\theta,K}) \leq \epsilon, \qquad D_{\mathrm{KL}}(p_{\theta,K} \| q_\theta) \leq -\log(1-\epsilon). \tag{24}$$

Proof. The token-set identity follows by summing the mixture over $\mathcal{U}$. Defining total variation as half the ℓ_1 distance,

$$\mathrm{TV}(q_\theta, p_{\theta,K}) = \epsilon\,\mathrm{TV}(\widetilde{q}_\theta, p_{\theta,K}) \leq \epsilon.$$

For every token with $p_{\theta,K}(a) > 0$, the mixture gives $q_\theta(a) \geq (1-\epsilon)p_{\theta,K}(a)$. Hence

$$\sum_a p_{\theta,K}(a)\log\frac{p_{\theta,K}(a)}{q_\theta(a)} \leq \sum_a p_{\theta,K}(a)\log\frac{1}{1-\epsilon} = -\log(1-\epsilon).$$

Thus readout differences can change the next-token behavior within the anchor's probability budget. These are conditional one-token bounds. A complete search and answer also depend on the subsequent trajectory.

H.3 GATE GRADIENTS AND USEFUL SEARCH

Local model. Fix s, parameters θ_0, $p_z = p_{\theta_0,z}(\cdot \mid s)$, $\epsilon = \epsilon_{\phi(s)}$, and continuation policy $\pi_0 = q_{\theta_0}$. Let $\xi \in \mathbb{R}^K$ be independently variable gate logits at this state. Define

$$g_z(\xi) = \frac{\exp \xi_z}{\sum_j \exp \xi_j}, \qquad \alpha_z(\xi) = (1-\epsilon)\mathbb{I}[z = K] + \epsilon g_z(\xi),$$
$$q_\xi(a \mid s) = \sum_z \alpha_z(\xi) p_z(a), \qquad V_z = \sum_a p_z(a) Q^{\pi_0}(s,a).$$

At logits reproducing $g_{\theta_0}(\cdot \mid s)$, q_ξ equals the current behavior distribution. The full method has $0 < \epsilon < 1$, so each anchored gate weight is positive. Only the decision at s varies in this local model. The readouts and all continuation values remain fixed. Set

$$V_g(\xi) = \sum_z g_z(\xi) V_z, \qquad \mathcal{J}_s(\xi) = \sum_a q_\xi(a \mid s) Q^{\pi_0}(s,a) = (1-\epsilon)V_K + \epsilon V_g(\xi).$$

Proposition (Value-directed gate gradient). The local objective satisfies Eq. (13). *Proof.* Using $\partial g_j / \partial \xi_z = g_j(\mathbb{I}[j = z] - g_z)$ gives

$$\frac{\partial \mathcal{J}_s}{\partial \xi_z} = \epsilon \sum_j V_j g_j (\mathbb{I}[j = z] - g_z) = \epsilon g_z (V_z - V_g).$$

For positive policy probabilities, the same derivative can be expressed through the marginal score:

$$\nabla_\xi \mathcal{J}_s = \mathbb{E}_{a \sim q_\xi(\cdot \mid s)} \left[Q^{\pi_0}(s,a) \nabla_\xi \log q_\xi(a \mid s) \right]. \tag{25}$$

An action-independent baseline leaves this expectation unchanged because $\sum_a \nabla_\xi q_\xi(a \mid s) = 0$. The posterior identity in Appendix A decomposes this score. In the local model it gives

$$\nabla_\xi \log q_\xi(a \mid s) = \sum_z \gamma_\xi(z \mid s, a) \nabla_\xi \log \alpha_z(\xi), \qquad \gamma_\xi(z \mid s, a) = \frac{\alpha_z(\xi) p_z(a)}{q_\xi(a \mid s)}.$$

With trainable readouts, the full score also includes the readout derivatives already derived in Appendix A.

Corollary (Local growth of valuable second searches). In the binary model at $s = H_1$, $\mathcal{U}_\mathrm{S}$ and $\mathcal{U}_\mathrm{A}$ partition the action space into search and answer branches, respectively. Let $u_z = p_z(\mathcal{U}_\mathrm{S})$, $\bar{u} = \sum_z g_z u_z$, and

$$u_q = q_\xi(\mathcal{U}_\mathrm{S} \mid H_1) = (1-\epsilon)u_K + \epsilon \bar{u}.$$

Assume the expected continuation reward conditional on a branch is the same for every readout assigning positive probability to that branch. Writing these two values as $\mathcal{Q}_\mathrm{S}^{\pi_0}$ and $\mathcal{Q}_\mathrm{A}^{\pi_0}$ yields

$$V_z = \mathcal{Q}_\mathrm{A}^{\pi_0} + \Delta_2^{\pi_0} u_z.$$

Under Euclidean gate-logit gradient ascent in continuous optimization time η, namely $\mathrm{d}\xi_z / \mathrm{d}\eta = \partial \mathcal{J}_s / \partial \xi_z$, the search probability obeys

$$\frac{\mathrm{d}u_q}{\mathrm{d}\eta} = \epsilon^2 \Delta_2^{\pi_0} \sum_z g_z^2 (u_z - \bar{u})^2. \tag{26}$$

It is strictly positive when $\Delta_2^{\pi_0} > 0$, $\epsilon > 0$, and search propensities differ among readouts with positive gate weight. *Proof.* Since u_z is fixed,

$$\frac{\partial u_q}{\partial \xi_z} = \epsilon g_z (u_z - \bar{u}), \qquad V_z - V_g = \Delta_2^{\pi_0}(u_z - \bar{u}).$$

Multiplying the two gradients and summing over z gives Eq. (26). Every summand after factoring out $\Delta_2^{\pi_0}$ is nonnegative. Their sum vanishes exactly when all positive-weight readouts have the same search propensity.

When branch-conditional query quality or execution outcomes differ across readouts, retain the general V_z rather than the binary value reduction. The same chain rule gives

$$\frac{\mathrm{d}u_q}{\mathrm{d}\eta} = \epsilon^2 \sum_z g_z^2 (u_z - \bar{u})(V_z - V_g). \tag{27}$$

Search then grows when propensity and continuation value have positive weighted alignment. The gate objective favors expected final reward, rather than search frequency itself.

Relation to the implemented update. The local model isolates gate selection. The implementation instead produces ξ through a shared neural network whose updates also change the readouts and future policy. Moreover, Eq. (8) uses sampled group-normalized advantages, active clipping branches, and a mean over valid tokens. Eq. (12) adds two KL terms. Equation (25) does not identify that full update with an unbiased gradient of $\mathcal{J}_s$. The exact connection retained by the implemented policy term is its marginal score decomposition on each active PPO branch, as shown in Appendix A.

H.4 Terminal exploration and search suppression

RLTT and encountering a successful two-search trajectory. For a fixed batch generated by terminal-R_4 rollout, the reward part of the uniform RLTT ascent direction in Eq. (19) is

$$G_{\mathrm{RLTT}} = \frac{1}{\sum_{i,t} m_{i,t}} \sum_{i,t} m_{i,t} A_i \frac{1}{K} \sum_{z=1}^{K} \nabla_\theta \log p_{\theta,z}(a_{i,t} \mid s_{i,t}). \tag{28}$$

This expands readout supervision on the sampled tokens while the terminal policy determines which trajectories are encountered. Let $\mathcal{E}_2$ be a trajectory event comprising an executed second search and a correct final answer. At a fixed terminal policy and question distribution, write $\varrho_K = \Pr(\mathcal{E}_2)$. For N independent trajectories from that distribution,

$$\Pr(\mathcal{E}_2 \text{ occurs at least once}) = 1 - (1 - \varrho_K)^N \le N\varrho_K. \tag{29}$$

Proof. Independence gives $(1 - \varrho_K)^N$ for no occurrence. The upper bound follows from the union bound. When $N\varrho_K \ll 1$, that batch is unlikely to contain such a trajectory, regardless of how many readouts receive its reward signal. Shared-parameter updates can subsequently alter terminal behavior, but intermediate readout proposals do not directly change the current rollout distribution. This provides a possible exploration bottleneck for terminal-sampled multi-readout learning.

GRPO and a low-search state. In a binary terminal-policy model with search logit ζ, the search probability is $u_{\mathrm{term}} = (1 + \exp(-\zeta))^{-1}$. With branch values held fixed, define

$$\mathcal{J}_{\mathrm{term}}(\zeta) = u_{\mathrm{term}} \mathcal{Q}_{\mathrm{S}} + (1 - u_{\mathrm{term}}) \mathcal{Q}_{\mathrm{A}}.$$

Direct differentiation gives

$$\frac{\partial \mathcal{J}_{\mathrm{term}}}{\partial \zeta} = u_{\mathrm{term}}(1 - u_{\mathrm{term}})(\mathcal{Q}_{\mathrm{S}} - \mathcal{Q}_{\mathrm{A}}). \tag{30}$$

Low-quality queries, poor evidence use, or execution failures can make the current search branch less valuable than answering. In that case the local reward gradient reduces search probability. A small search probability then limits opportunities to encounter successful search trajectories and attenuates this local gradient, even if the value difference later becomes positive.

The sampled group advantages in Eq. (1) give a complementary observation. In a group containing both successful and failed trajectories, every failed trajectory has negative advantage, including a failed search. When all rewards in a group are equal, including an all-failure group, all advantages are zero, so the policy-reward term supplies no update. Such a group supplies no reward-driven recovery signal. It does not itself generate a negative search gradient. These mechanisms describe conditions compatible with a low-search state, rather than identify the cause of a particular recorded collapse.

H.5 Connection to the empirical analysis

A2 fixes the native gate while retaining the stage anchor, removing its learned state-dependent adjustment. A1 replaces the posterior coefficients in the marginal score. A3 sets $\epsilon = 1$, making the total-variation bound vacuous and removing the finite KL bound. The ablations measure the accuracy and protocol consequences of these changes; the bounds themselves concern token distributions.

Figure 3(b) and Appendix G document answer-bearing retrieval on terminal-R_4 evaluation trajectories. The local condition $\Delta_2^{\pi_0} > 0$ instead concerns branch values under a fixed training-policy continuation. The terminal anchor and shared recurrent parameters provide a training path from marginal-policy exploration to the terminal readout, whose resulting performance is measured in the main results and Appendix E.1.

I Additional Evaluation Beyond Search

We additionally train ReMPO from Ouro-1.4B-Thinking on the same mathematics training data as RLTT (Williams & Tureci, 2026), following its training settings and selecting the checkpoint with the highest validation score. Without retrieval stages, we use $\epsilon = 0.25$ throughout generation, so $\alpha_\theta(z \mid s) = 0.25 g_\theta(z \mid s) + 0.75\mathbb{I}[z = K]$ with $K = 4$. The terminal readout therefore retains at least 75% of the mixture weight. Evaluation uses greedy decoding from q_θ. All other evaluation settings follow RLTT. Table 18 compares this run with RLTT's published terminal-readout results.

Table 18: Additional evaluation of mathematics-trained Ouro-1.4B-Thinking models. RLTT scores are taken from the original paper; ReMPO uses q-greedy decoding. Scores use a 0–1 scale: answer accuracy for mathematics and knowledge tasks, and pass@1 for MBPP. The final row is the unweighted mean over these eight benchmarks, computed before rounding. Bold marks the higher score in each row.

Benchmark	RLTT (reported)	ReMPO (math-trained)
Mathematics		
MATH500	0.812	**0.822**
GSM8K	**0.903**	0.859
AIME24	0.267	**0.333**
BeyondAIME	0.120	**0.130**
AIME26	**0.200**	0.133
Knowledge and reasoning		
ARC-C	0.919	**0.920**
MMLU-STEM	0.838	**0.845**
Code generation		
MBPP	0.567	**0.754**
Eight-benchmark mean	0.578	**0.600**

ReMPO scores higher on MATH500, AIME24, and BeyondAIME, while RLTT scores higher on GSM8K and AIME26. ARC-C and MMLU-STEM scores are close. The largest difference is on MBPP, where ReMPO reaches 0.754 compared with the reported RLTT score of 0.567. This code-generation gain accounts for the higher eight-benchmark mean, while the mathematics results vary across benchmarks.