AGENTFRONTIER: STABILIZING LEARNING FRONTIERS IN REINFORCEMENT LEARNING FOR ACTIVE REASONING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Active reasoning requires large language model (LLM) agents to interact with external environments and strategically acquire missing information across multiple turns. Reinforcement learning (RL) with outcome rewards has become a standard approach for training such agents, particularly through group-relative algorithms that compare multiple rollouts from the same prompt to form relative learning signals. However, we observe that RL training for active reasoning often exhibits a sharp collapse after rapid initial improvement. We trace this instability to learning-frontier non-stationarity: under group-relative outcome rewards, a prompt contributes effective gradient signals only when its rollout group has non-zero reward variation. As easier prompts become mastered, they silently exit the effective gradient computation, causing the learning frontier to shrink and concentrate on a narrow subset of hard prompts. Continued optimization then overadapts to this shifting frontier while leaving mastered capabilities without gradient protection. To bridge this gap, we introduce `AgentFrontier`, a simple framework that stabilizes learning frontiers during RL training. `AgentFrontier` combines Frontier Retention, which anchors the policy on mastered prompts through lightweight behavioral regularization, and Frontier Stabilization, which constructs difficulty-aware batches to smooth frontier composition across updates. Extensive experiments on five active reasoning tasks show that `AgentFrontier` is **(1) high-performing**, raising the six-metric average of GSPO from 59.58 to 66.00 with gains on all reported metrics and reaching 67.10 when combined with T^3, with per-metric gains over GSPO of up to 9.2 points, and **(2) stabilizing**, reducing reward variance from 0.041 to 0.017. These results highlight learning-frontier stability as a key principle for building reliable LLM agents capable of active reasoning.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable reasoning capabilities across diverse domains, further advanced by reinforcement learning (RL) with outcome rewards (Shao et al., 2024; Guo et al., 2025; Yu et al., 2025b). Recently, the community has extended RL from single-turn reasoning to multi-turn and long-horizon settings (Chen et al., 2025; Xi et al., 2026; Luo et al., 2025). A particularly important capability in this direction is *active reasoning*, where an LLM agent interacts with external environments across multiple turns, strategically raises questions, and actively acquires missing information to solve problems (Zhou et al., 2025; Badola et al., 2025).

Recent RL recipes for reasoning models often construct learning signals by comparing multiple sampled rollouts for the same prompt (Shao et al., 2024; Zheng et al., 2025). While originally developed for single-turn reasoning, this paradigm has also been extended to active reasoning agents (Zou et al., 2026). It often leads to rapid initial progress: training rewards rise quickly, and the model begins to exhibit useful multi-turn behaviors, such as asking targeted questions and refining its decisions through environmental feedback. However, we observe a recurring pattern across tasks and optimization algorithms. After reaching a high reward level, the training curve can suddenly drop and then recover only partially. The collapse follows a high-reward regime in which the policy has already produced effective multi-turn behaviors.

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.

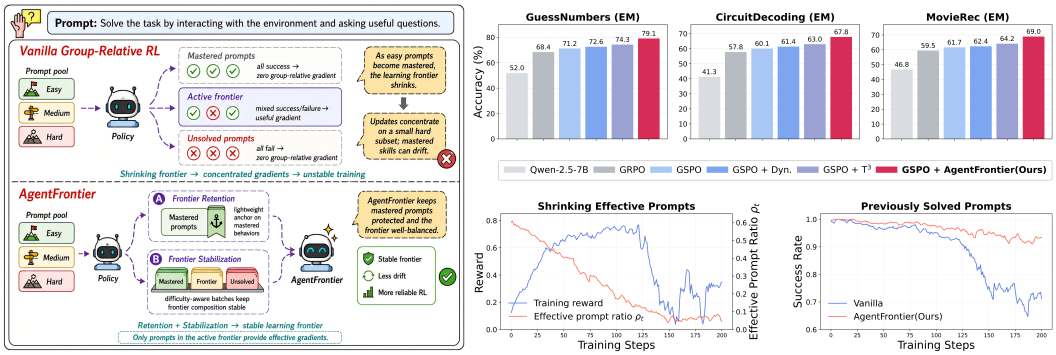


Figure 1: **Overview and empirical motivation of `AgentFrontier`. Left:** In vanilla group-relative RL, prompts only contribute effective gradients when their rollout groups contain both successes and failures. As easy prompts become consistently solved, the learning frontier shrinks and concentrates on hard cases, causing silent drift on mastered prompts. **`AgentFrontier`** stabilizes this through Frontier Retention (behavioral anchor on mastered prompts) and Frontier Stabilization (difficulty-aware batch construction). **Right-top:** `AgentFrontier` consistently improves over vanilla RL and existing stabilization methods across three active reasoning tasks. **Right-bottom:** The effective-prompt ratio ρ_t declines steadily before reward collapse (left), and vanilla training gradually loses accuracy on previously mastered prompts while **`AgentFrontier`** preserves them (right).

This observation raises a concrete question:

What changes in the training signal after the model starts performing well, and why can continued RL training damage previously acquired capabilities?

We trace this instability to a structural property of group-relative advantage computation under outcome rewards. A prompt produces an effective learning signal only when its rollout group contains both successful and failed trajectories. More generally, if the within-group reward variance is zero, the prompt contributes little or no gradient to the update. Binary outcome rewards form the clearest special case of this condition. As the model improves and solves easier prompts consistently, these prompts silently leave the effective gradient computation. Therefore, the training dataset remains fixed, but the prompts that actually drive parameter updates keep changing. We refer to this moving subset as the learning frontier. To quantify it, we introduce the active frontier ratio, defined as the fraction of training prompts that currently contribute non-zero group-relative gradient signals. This ratio provides a direct, real-time diagnostic of the effective training distribution. An overview of the framework and empirical motivation is shown in Fig. 1.

This learning-frontier dynamics explains the observed collapse. As fewer prompts contribute gradients, the active frontier becomes narrow and increasingly concentrated around the hardest remaining cases. The resulting updates are dominated by a small and changing subset of prompts. Meanwhile, mastered prompts have already left the effective gradient computation and receive no explicit protection. Their previously learned behaviors may drift under frontier-dominated updates. Such drift can accumulate silently until a tipping point is reached, after which performance on previously mastered prompts can degrade and the overall reward suddenly drops. In multi-turn active reasoning, this issue becomes more severe. The same prompt can appear easy or hard across different rollouts, depending on early-turn actions and environmental feedback. As a result, the frontier composition can fluctuate sharply between consecutive training steps.

In response to this question, we propose **`AgentFrontier`**, a framework for stabilizing learning frontiers in RL for active reasoning agents. The core idea is that prompts leaving the active frontier should not be ignored entirely, since they encode capabilities the model has already acquired. **`AgentFrontier`** introduces two complementary mechanisms. First, *Frontier Retention* applies a lightweight behavioral anchor to mastered prompts, preventing the policy from silently drifting away from previously successful strategies. Second, *Frontier Stabilization* constructs difficulty-aware training batches to smooth the frontier composition across updates and prevent gradients from collapsing onto a narrow band of hard prompts. **`AgentFrontier`** preserves the core policy

optimization algorithm and integrates into standard RL frameworks, including PPO, GRPO, and GSPO (Schulman et al., 2017; Shao et al., 2024; Zheng et al., 2025), as a practical training wrapper.

Our contributions are summarized as follows:

❶ ***Problem Identification.*** We identify learning-frontier non-stationarity as a concrete source of training collapse in group-relative RL for active reasoning. As the policy improves, mastered prompts silently exit gradient computation, causing the effective training distribution to shrink and concentrate.

❷ ***Mechanism Analysis.*** We show that a narrowing frontier leads to gradient concentration on a small prompt subset, while mastered capabilities lose parameter protection and drift until sudden failure. This establishes a direct link between frontier dynamics and reward collapse.

❸ ***Practical Solution.*** We propose `AgentFrontier`, a lightweight training wrapper that stabilizes the learning frontier through two mechanisms: *Frontier Retention* anchors the policy on mastered prompts, and *Frontier Stabilization* balances difficulty composition in each training batch.

❹ ***Experimental Validation.*** Across five active reasoning tasks, `AgentFrontier` raises the six-metric average of GSPO from 59.58 to 66.00 and maintains the active frontier ratio above 30% throughout training. Benefits hold across RL algorithms and model scales from 3B to 14B.

2 Related Work

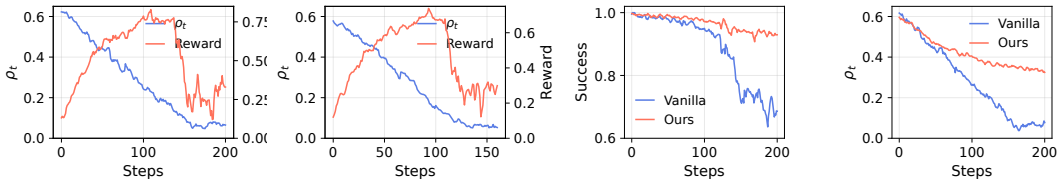
2.1 Reinforcement Learning for Active Reasoning

Reinforcement learning has become a standard approach for improving the reasoning capabilities of LLMs. Early policy-gradient methods provide the basis for stable language-model post-training (Schulman et al., 2017), while later studies improve mathematical reasoning after supervised warm-up (Luong et al., 2024) and elicit strong reasoning with group-relative outcome rewards (Shao et al., 2024). Other works revisit credit estimation through Monte-Carlo-style comparisons (Kazemnejad et al., 2024), scale outcome-based RL to large reasoning models (Guo et al., 2025), release open-source reasoning systems (Hu et al., 2025), and study sequence-level objectives and large-scale sampling details (Zheng et al., 2025; Yu et al., 2025b).

Active reasoning extends this setting with external information acquisition over multiple turns. Prior work studies clarification, proactive questioning, and uncertainty-aware planning (Deng et al., 2023; Hu et al., 2024). Recent benchmarks evaluate information seeking, strategic dialogue, planning, and interactive reasoning (Zhou et al., 2025; Badola et al., 2025). Agent benchmarks and training frameworks further cover self-evolving environments, tool-use synthesis, stateful digital environments, long-horizon interaction, decoupled training interfaces, and social reward design (Xi et al., 2025; Yin et al., 2025; Chen et al., 2025; Xi et al., 2026; Luo et al., 2025; Yu et al., 2025a). Our work builds on these lines and studies how the group-relative training signal evolves when interaction strategies are optimized across multi-turn rollouts.

2.2 Training Stability in LLM Reinforcement Learning

Recent work has revealed that the success of LLM reinforcement learning depends on reward design together with stable training dynamics. Classical advantage estimation studies how delayed returns are transformed into lower-variance update signals (Schulman et al., 2015). In LLM reasoning, multi-turn online RL has been used to reduce behavior collapse in self-correction (Kumar et al., 2024), and turn-level reward design has been explored to improve credit assignment in multi-turn tool-use scenarios (Zeng et al., 2025). For agentic tasks, recent reward redistribution methods estimate stepwise progress from delayed outcomes (Wang et al., 2025), and process-reward frameworks aim to guide long-horizon decision making through intermediate feedback (Choudhury, 2025). In active reasoning, trajectory-level instability has also been linked to belief deviation and uninformative tails that distort credit assignment (Zou et al., 2026). Our work complements these studies by identifying a prompt-level source of instability: under group-relative rewards, the set of prompts that contribute effective gradients changes substantially as training progresses. This makes the effective training distribution non-stationary even when the dataset itself remains fixed, providing a new explanation for the collapse patterns observed in multi-turn RL.



(a) GN (GSPO) (b) CD (PPO) (c) CD (PPO) (d) GN (GSPO)

Figure 2: Overview of empirical verification for key analytical components (details in Appendix A). **(a)(b)** Active frontier ratio ρ_t (blue, left axis) and training reward (orange, right axis) over training steps for the GN and CD tasks. In both tasks, ρ_t drops steadily before the reward collapse and falls below 0.08 at the point of collapse, consistent with Proposition 1. **(c)** Success rate of a cohort of prompts that were mastered ($k_j = G$) at step t_0, tracked over subsequent training steps. Without `AgentFrontier`, the success rate declines gradually and then drops sharply. With `AgentFrontier`, the success rate remains stable, consistent with Proposition 2. **(d)** Comparison of ρ_t throughout training with and without `AgentFrontier`. Vanilla training shows ρ_t collapsing below 0.08, while `AgentFrontier` maintains ρ_t above 0.28 throughout optimization.

3 Learning Frontier Dynamics in Group-Relative RL

3.1 Setup and Analysis

We begin by formalizing the key quantities that govern training stability in group-relative RL for active reasoning, and then analyze how the learning frontier evolves during optimization. To ground our analysis in observed training behavior, we conduct empirical studies that directly examine the quantities introduced below and summarize the findings in Appendix A (an overview in Fig. 2).

Group-relative RL. Let π_θ denote the policy parameterized by θ, x a training prompt, and $\mathcal{D}$ the training set with $|\mathcal{D}| = N$. For each prompt x, we sample G rollouts $\{y_1, \ldots, y_G\} \sim \pi_\theta(\cdot \mid x)$ and obtain scalar outcome rewards $r_i = r(x, y_i) \in \mathbb{R}$. The group-relative advantage is computed as:

$$\hat{A}_i = \frac{r_i - \bar{r}}{\sigma_r}, \quad \text{where} \quad \bar{r} = \frac{1}{G}\sum_{j=1}^{G} r_j, \quad \sigma_r = \text{std}(\{r_j\}_{j=1}^{G}). \tag{1}$$

When all rollouts in a group share the same reward, the centered term satisfies $r_i - \bar{r} = 0$ for all i. In common implementations, such groups are either skipped or normalized with a small numerical constant, yielding no effective group-relative gradient for that prompt.

Active frontier. Let $v_j^{(t)}$ denote the within-group reward variance for prompt x_j at training step t. For binary rewards, let $k_j^{(t)} \in \{0, 1, \ldots, G\}$ denote the number of successful rollouts. We define the frontier set and the active frontier ratio as follows.

Definition 1 (Frontier Set and Active Frontier Ratio)**.** *At training step t, the frontier set is $\mathcal{F}_t = \{j : v_j^{(t)} > \varepsilon_v\}$, and the active frontier ratio is $\rho_t = |\mathcal{F}_t|/N$. For binary rewards and $\varepsilon_v = 0$, this reduces to $\mathcal{F}_t = \{j : 0 < k_j^{(t)} < G\}$. A prompt is called mastered when its rolling success estimate exceeds the mastered threshold, and unsolved when the estimate remains below the unsolved threshold.*

Only prompts in $\mathcal{F}_t$ contribute non-zero gradients under group-relative advantage computation. The remaining prompts, whether mastered or unsolved, produce zero effective gradient and are invisible to the optimizer.

Frontier shrinkage is structural. As the policy improves on a subset of prompts, their success probabilities move toward one. We now show that such prompts are increasingly likely to leave the active frontier.

Proposition 1 (Frontier Shrinkage)**.** *Let $p_j \in [0, 1]$ denote the per-rollout success probability of prompt x_j under the current policy. The probability that prompt x_j belongs to the frontier set is $\Pr(x_j \in \mathcal{F}_t) = 1 - p_j^G - (1 - p_j)^G$. This quantity is maximized at $p_j = 1/2$ and decreases monotonically as $p_j \to 1$ or $p_j \to 0$. In particular, for a prompt with success probability $p_j \geq 1 - \epsilon$ (near-mastered), we have $\Pr(x_j \notin \mathcal{F}_t) \geq (1 - \epsilon)^G \geq 1 - G\epsilon$.*

Proof. Under independent Bernoulli rollouts, $k_j^{(t)} \sim \text{Binomial}(G, p_j)$. The prompt exits the frontier set when $k_j = 0$ or $k_j = G$, which occurs with probability $p_j^G + (1-p_j)^G$. The remaining claims follow from elementary calculus on $f(p) = 1 - p^G - (1-p)^G$. □

Proposition 1 establishes frontier shrinkage as a structural consequence of group-relative advantage in the binary-outcome special case. As prompts approach mastery, they exit the frontier set with high probability, reducing ρ_t over time. Fig. 2a and 2b confirm this empirically: in both GN and CD tasks, ρ_t drops steadily throughout training and falls below 0.08 at the point of collapse.

Multi-turn amplification. In active reasoning, the per-rollout success probability p_j is not a fixed property of the prompt. Different rollouts of the same prompt may follow different interaction trajectories depending on early-turn actions, leading to different outcomes. This means that $k_j^{(t)}$ exhibits higher variance across consecutive training steps compared to single-turn tasks. A prompt may appear mastered in one batch ($k_j = G$) but active in the next ($k_j = G-2$), making the frontier composition volatile.

Remark 1 (Frontier Volatility in Multi-Turn Settings)**.** *Let $q_j^{(t)} = 1-(p_j^{(t)})^G-(1-p_j^{(t)})^G$ denote the probability that prompt x_j belongs to the frontier set at step t. Then $|q_j^{(t+1)} - q_j^{(t)}| \le G \cdot |p_j^{(t+1)} - p_j^{(t)}|$. In multi-turn active reasoning, $p_j^{(t)}$ can change sharply because early-turn actions alter subsequent observations and outcomes. This makes the expected frontier composition more volatile than in single-turn settings.*

Gradient concentration. When the frontier set shrinks to a small subset concentrated on hard prompts, the gradient direction becomes dominated by this subset. Let $g_j^{(t)}$ denote the gradient contribution of prompt x_j at step t, with $g_j^{(t)} = 0$ for $j \notin \mathcal{F}_t$. We define gradient concentration as:

$$\text{GC}_t = \frac{\max_{c \in \mathcal{C}} \sum_{j \in c} \|g_j^{(t)}\|}{\sum_{j=1}^{N} \|g_j^{(t)}\|}, \tag{2}$$

where $\mathcal{C}$ is a partition of prompts by structural properties (e.g., difficulty level or task type). When $\text{GC}_t \to 1$, gradient directions collapse onto a single cluster.

Parameter drift on mastered prompts. The central consequence of frontier shrinkage is that mastered prompts lose all gradient protection. We now formalize the scale of possible drift.

Proposition 2 (Unprotected Drift Scale for Mastered Prompts)**.** *Let x_j be a prompt that exits the frontier set at step t_0 and remains outside until step $t_0 + T$. Let $\ell_j(\theta) = -\frac{1}{|y_j^*|} \sum_{s=1}^{|y_j^*|} \log \pi_\theta(y_{j,s}^* \mid x_j, y_{j,<s}^*)$ denote the negative log-likelihood of a reference successful trajectory y_j^* under the current policy. Assume that the per-step parameter update satisfies $\|\theta_{t+1} - \theta_t\| \le \eta_{\text{lr}} \cdot M$ for learning rate η_{lr} and gradient norm bound M. If ℓ_j is L-Lipschitz in θ, then after T unprotected steps:*

$$|\ell_j(\theta_{t_0+T}) - \ell_j(\theta_{t_0})| \le L \cdot \eta_{\text{lr}} \cdot M \cdot T. \tag{3}$$

Proof. By the triangle inequality and Lipschitz continuity: $|\ell_j(\theta_{t+1}) - \ell_j(\theta_t)| \le L\|\theta_{t+1} - \theta_t\| \le L \cdot \eta_{\text{lr}} \cdot M$. Summing over T steps via the triangle inequality yields the bound. □

The bound characterizes the drift scale after a mastered prompt stops contributing gradients: without a prompt-specific update, the possible drift grows linearly with the number of unprotected updates. When the frontier is narrow and concentrated, the effective M can be large relative to the diversity of gradient directions, increasing the drift potential. Fig. 2c validates this prediction: a cohort of prompts mastered at step t_0 shows gradual success-rate decline followed by a sharp drop under vanilla training, while `AgentFrontier` keeps the success rate stable.

From drift to collapse. When many mastered prompts experience sufficient likelihood drift, their rollout success rates may drop below the all-success regime. These prompts then re-enter the frontier set, often together, producing an abrupt drop in average reward and a sudden change in the gradient distribution. Under binary rewards, this transition is discontinuous: a prompt goes from contributing zero gradient ($k_j = G$) to contributing non-zero gradient ($k_j < G$) in a single step. The longer the

model trains without frontier protection, the more mastered prompts accumulate drift, and the more severe the eventual collapse. Furthermore, newly re-entered prompts may generate gradient signals that conflict with the current frontier direction, creating oscillatory dynamics during recovery.

3.2 Stabilizing the Learning Frontier

The analysis above identifies two sources of instability: (1) mastered prompts exit gradient computation and drift without protection, and (2) the frontier composition becomes narrow and volatile. We address each with a targeted mechanism.

Frontier Retention. A key structural property of group-relative RL is that mastered prompts can receive exactly zero gradient. This distinguishes the problem from general catastrophic forgetting in continual learning, where old tasks still contribute some training signal. In the binary-outcome case, the absence of signal is abrupt: when $k_j = G$, the prompt disappears from the optimization entirely. Standard replay or behavioral cloning would apply uniformly to all past data. Frontier Retention is activated for prompts that have entered the zero-gradient regime under group-relative advantage, targeting this structural property directly.

For each prompt x_j classified as mastered (rolling success estimate $\hat{p}_j^{(t)}$ exceeding τ_{master}), we cache the highest-likelihood successful rollout y_j^* from its most recent group and add a retention loss:

$$\mathcal{L}_{\text{retain}}(\theta) = \frac{1}{|\mathcal{M}_t|} \sum_{j \in \mathcal{M}_t} \left[-\frac{1}{|y_j^*|} \sum_{s=1}^{|y_j^*|} \log \pi_\theta(y_{j,s}^* \mid x_j, y_{j,<s}^*) \right], \tag{4}$$

where $\mathcal{M}_t = \{j : \hat{p}_j^{(t)} > \tau_{\text{master}}\}$. The total objective is $\mathcal{L}_{\text{total}}(\theta) = \mathcal{L}_{\text{RL}}(\theta) + \alpha \cdot \mathcal{L}_{\text{retain}}(\theta)$.

Proposition 3 (Retention Controls Aggregate Drift)**.** *At any stationary point of $\mathcal{L}_{\text{total}}$, we have $\|\nabla_\theta \mathcal{L}_{\text{retain}}(\theta)\| = \frac{1}{\alpha}\|\nabla_\theta \mathcal{L}_{\text{RL}}(\theta)\|$. A larger α enforces a stronger aggregate constraint: the retention gradient must be smaller at equilibrium, limiting how far mastered-prompt likelihoods can drift.*

Proof. At a stationary point, $\nabla_\theta \mathcal{L}_{\text{total}} = 0$, giving $\nabla_\theta \mathcal{L}_{\text{RL}} + \alpha \nabla_\theta \mathcal{L}_{\text{retain}} = 0$. Taking norms yields the result. □

The cached trajectory y_j^* is refreshed every K_{ref} steps using the current policy's best rollout, ensuring the retention target tracks the improving policy. Retention excludes unsolved prompts ($k_j = 0$), since these prompts produce zero group-relative gradient and provide no successful trajectory to anchor.

Frontier Stabilization. Uniform random sampling from the training set does not account for the non-stationary concentration of effective gradients. As training progresses, a uniformly sampled batch may contain mostly mastered prompts (zero gradient) and a few frontier prompts (all gradient), making each update direction highly dependent on which frontier prompts happen to appear. Frontier Stabilization corrects this by constructing batches with a controlled frontier density.

We maintain the exponential moving average $\hat{p}_j^{(t)} = \beta \cdot \hat{p}_j^{(t-1)} + (1-\beta) \cdot k_j^{(t)}/G$ and classify each prompt:

$$x_j \in \begin{cases} \mathcal{M} & \text{if } \hat{p}_j > \tau_{\text{master}}, \\ \mathcal{F} & \text{if } \tau_{\text{unsolved}} \le \hat{p}_j \le \tau_{\text{master}}, \\ \mathcal{U} & \text{if } \hat{p}_j < \tau_{\text{unsolved}}. \end{cases} \tag{5}$$

Each batch is then assembled by drawing n_{f} prompts from $\mathcal{F}$, n_{m} from $\mathcal{M}$, and n_{u} from $\mathcal{U}$, with proportions set as:

$$\frac{n_{\text{f}}}{B} = \min\left(\frac{|\mathcal{F}|}{N},\ \gamma_{\max}\right), \quad \frac{n_{\text{m}}}{B} = \frac{|\mathcal{M}|}{N} \cdot \gamma_{\text{m}}, \quad \frac{n_{\text{u}}}{B} = 1 - \frac{n_{\text{f}} + n_{\text{m}}}{B}, \tag{6}$$

where B is the batch size, $\gamma_{\max}$ caps the frontier fraction to prevent over-concentration, and $\gamma_{\text{m}} \in (0, 1]$ controls the mastered-prompt presence.

This differs from standard curriculum learning or balanced sampling in two respects. First, the categories are defined by group-relative gradient activity (whether a prompt currently produces non-zero advantage variance), not by static difficulty labels. Second, the proportions adapt implicitly as

the frontier moves: when $|\mathcal{F}|$ shrinks, n_{f} shrinks with it and more mastered/unsolved prompts enter the batch, preventing gradient collapse. No explicit scheduling or annealing is required.

As shown in Fig. 2d, this stabilization keeps ρ_t above 0.28 throughout training, compared with below 0.08 at the collapse point in vanilla baselines.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Tasks. We evaluate `AgentFrontier` on five active reasoning tasks from AR-Bench (Zhou et al., 2025) and Multi-Turn Puzzles (Badola et al., 2025). These tasks cover three representative regimes of active reasoning. *GuessNumbers* (GN) and *CircuitDecoding* (CD) have finite and enumerable hidden hypothesis spaces, where the agent must actively reduce candidate solutions through structured queries. *SituationPuzzles* (SP) requires natural-language information seeking under ambiguity. *PreferenceEstimation* (PE) and *MovieRecommendation* (MR) require the agent to infer latent user preferences through multi-turn feedback and make downstream decisions. This coverage allows us to examine whether `AgentFrontier` stabilizes group-relative RL across symbolic reasoning, structured querying, natural-language interaction, and preference-based decision making.

Baselines. We compare `AgentFrontier` with three groups of baselines. First, we include direct inference with Qwen-2.5-7B-Instruct (Qwen et al., 2025). Second, we evaluate standard RL post-training algorithms, including PPO (Schulman et al., 2017), GRPO (Shao et al., 2024), and GSPO (Zheng et al., 2025). Third, we compare with stabilization baselines, including dynamic sampling inspired by recent large-scale RL recipes (Yu et al., 2025b) and trajectory-level stabilization through T^3 (Zou et al., 2026). Dynamic sampling filters prompt groups with no within-group reward variance, while T^3 mitigates uninformative tails within each trajectory. We additionally evaluate global-KL regularization, self-paced sampling, and learning-progress sampling as controls for policy anchoring and curriculum selection. These controls isolate prompt-status tracking from global policy constraints and scalar progress heuristics. `AgentFrontier` stabilizes the prompt-level source of effective group-relative gradients. We also report the combination of T^3 and `AgentFrontier` to verify whether trajectory-level and prompt-level stabilization are complementary.

Implementation and Metrics. Unless otherwise stated, all RL experiments use Qwen-2.5-7B-Instruct as the policy model. GN, CD, PE, and MR use rule-based environment feedback, while SP follows prior active-reasoning setups and uses Qwen-2.5-14B-Instruct as a yes/no/unknown judge. Each prompt is sampled with $G = 8$ rollouts. For `AgentFrontier`, we use an EMA success estimator with coefficient 0.8, a mastered threshold of 0.875, an unsolved threshold of 0.125, a default batch composition of $20\%/60\%/20\%$ for previously solved, effective, and unsolved prompts, and a retention coefficient $\alpha = 0.05$. An unseen prompt is initialized at 0.5. A mastered status requires two consecutive all-success observations, successful retention caches are refreshed every 50 updates, and a cache that is unobserved for 100 updates decays toward 0.5 with factor 0.995^{Δ}. Shortages in one category are redistributed across the remaining categories while preserving the frontier cap. We report EM for GN, CD, and MR, word-level and character-level F1 for SP, and binary similarity for PE. All table metrics are scaled by 100. For training dynamics, we additionally report reward variance, effective-prompt ratio, solved-prompt success rate, and gradient concentration.

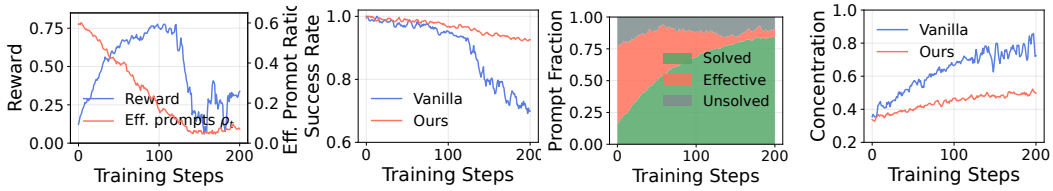
4.2 MAIN RESULT (**RQ1**)

RQ1: Does stabilizing the learning frontier improve active reasoning RL across tasks and algorithms?

Takeaway ❶: `AgentFrontier` consistently improves active reasoning RL across tasks and optimizers. As shown in Table 1, RL-trained agents outperform direct inference, confirming the value of outcome-reward optimization, but vanilla PPO, GRPO, and GSPO still remain unstable on multi-turn tasks where early interactions affect later observations. Adding `AgentFrontier` yields clear gains on all five tasks. For example, GSPO+`AgentFrontier` improves CD, GN, PE, and MR by 7.7, 7.9, 7.8, and 7.3 points over vanilla GSPO. Compared with stabilization baselines, dynamic sampling brings limited gains because it focuses on currently effective prompt groups and leaves solved-prompt protection unspecified, while T^3 improves within-trajectory credit assignment

Table 1: Main results across active reasoning tasks. All metrics are scaled by 100. ↑ indicates absolute improvement over the corresponding vanilla RL baseline. We report average rank across all task metrics.

Method	CD	SP		GN	PE	MR	Avg.
	EM	F1-word	F1-char	EM	Binary Sim	EM	Rank
Direct Inference							
Qwen-2.5-7B-Instruct	41.3	23.8	48.1	52.0	31.7	46.8	12.00
Vanilla RL							
PPO	55.6	31.2	72.5	64.8	39.2	55.4	11.00
GRPO	57.8	34.9	79.6	68.4	43.5	59.5	10.00
GSPO	60.1	36.5	81.0	71.2	47.0	61.7	9.00
Stabilization Baselines							
GSPO + Dyn.	61.4 ↑ 1.3	37.2 ↑ 0.7	81.4 ↑ 0.4	72.6 ↑ 1.4	48.3 ↑ 1.3	62.4 ↑ 0.7	8.00
GSPO + Global KL	61.8 ↑ 1.7	37.5 ↑ 1.0	81.6 ↑ 0.6	72.9 ↑ 1.7	48.8 ↑ 1.8	62.8 ↑ 1.1	7.00
GSPO + Self-paced	62.5 ↑ 2.4	38.0 ↑ 1.5	82.0 ↑ 1.0	73.8 ↑ 2.6	49.7 ↑ 2.7	63.6 ↑ 1.9	6.00
GSPO + Learning progress	63.0 ↑ 2.9	38.4 ↑ 1.9	82.3 ↑ 1.3	74.5 ↑ 3.3	50.4 ↑ 3.4	64.1 ↑ 2.4	4.25
GSPO + T^3	63.0 ↑ 2.9	38.1 ↑ 1.6	82.2 ↑ 1.2	74.3 ↑ 3.1	50.1 ↑ 3.1	64.2 ↑ 2.5	4.75
AgentFrontier							
GRPO + `AgentFrontier`	65.7 ↑ 7.9	39.2 ↑ 4.3	83.1 ↑ 3.5	77.2 ↑ 8.8	52.6 ↑ 9.1	66.1 ↑ 6.6	3.00
GSPO + `AgentFrontier`	67.8 ↑ 7.7	40.7 ↑ 4.2	84.6 ↑ 3.6	79.1 ↑ 7.9	54.8 ↑ 7.8	69.0 ↑ 7.3	2.00
GSPO + T^3 + `AgentFrontier`	**69.2** ↑ 9.1	**41.3** ↑ 4.8	**85.0** ↑ 4.0	**80.4** ↑ 9.2	**56.2** ↑ 9.2	**70.5** ↑ 8.8	**1.00**



(a) Effective Prompts (b) Solved Prompts (c) Prompt Composition (d) Gradient Concentration

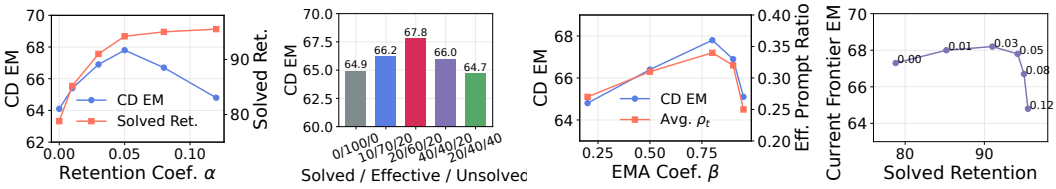
Figure 3: Model analysis of effective-prompt dynamics. (a) The effective-prompt ratio ρ_t drops before reward collapse, showing that non-zero group-relative gradients come from an increasingly narrow prompt subset. (b) Prompts solved at an earlier step degrade under vanilla training, while `AgentFrontier` preserves them more stably. (c)(d) Vanilla RL shifts the prompt pool toward ineffective groups and concentrates gradients on a narrow difficulty band. `AgentFrontier` maintains a broader prompt mixture and reduces gradient concentration.

and leaves prompt-level effective-gradient drift unchanged. The best average rank is achieved by combining T^3 with `AgentFrontier`, suggesting that trajectory-level and prompt-level stabilization are complementary. Overall, these results support our main finding: stabilizing the effective prompt distribution improves active reasoning RL beyond a single optimization recipe.

4.3 Model Analysis (**RQ2**)

RQ2: Are the gains tied to the effective-prompt dynamics identified in Sec. 3?

Model analysis. Fig. 3 shows that the effective-prompt ratio drops before reward collapse, previously solved prompts drift under vanilla training, and gradients concentrate on a narrow difficulty band. `AgentFrontier` maintains a broader prompt mixture, reduces gradient concentration, and preserves solved prompts more reliably. The temporal ordering in panels (a) and (b) places frontier shrinkage before the loss of solved-prompt accuracy, while panels (c) and (d) show the corresponding change in the training distribution. These results connect the final performance gain to effective-prompt shrinkage and solved-prompt drift.



(a) Retention strength (b) Batch composition (c) EMA coefficient (d) Retention trade-off

Figure 4: **Hyperparameter sensitivity.** A moderate retention coefficient, balanced prompt composition, and stable EMA estimator provide the best retention-learning trade-off. Together, these settings preserve solved-prompt coverage while sustaining current-frontier learning.

4.4 Hyperparameter Analysis (**RQ3**)

RQ3: How sensitive is `AgentFrontier` to the strength and composition of frontier stabilization?

Figure 4 identifies a stable operating region around $\alpha = 0.05$, an EMA coefficient of 0.8, and the $20/60/20$ batch composition. Moderate retention protects solved prompts while preserving current-frontier learning, and the EMA balances status noise against estimator lag. The batch-composition sweep shows that allocating most samples to the current frontier raises task performance while retaining solved-prompt coverage through the smaller mastered and unsolved fractions. The trade-off panel places these choices on the same retention-performance curve, so the default is selected from a joint operating region rather than from one metric alone.

4.5 Ablation Study (**RQ4**)

RQ4: Do Frontier Retention and Frontier Stabilization address complementary failure modes?

Table 2 shows complementary effects. Removing Frontier Retention reduces solved-prompt retention, while removing Frontier Stabilization lowers the effective-prompt ratio and increases reward variance. The stabilization-only variant reaches 78.8 retention with an effective ratio of 0.31, whereas the retention-only variant reaches 88.6 retention with an effective ratio of 0.19. Dynamic sampling and random replay improve over vanilla GSPO but remain below the full method on both frontier coverage and solved-prompt retention. Neither one-component variant reaches the full method's 94.2 retention and 0.34 effective ratio together. Their combination preserves solved behaviors and distributes learning signals across the current frontier, yielding the highest retention, effective ratio, and task scores together.

Table 2: **Ablation study.** Var., Eff., and Ret. denote reward variance, effective-prompt ratio, and solved-prompt retention.

Method	CD	GN	MR	Var.↓	Eff.↑	Ret.↑
Vanilla GSPO	60.1	71.2	61.7	0.041	0.07	71.5
+ Dyn. Samp.	61.4	72.6	62.4	0.034	0.18	74.3
+ Rand. Replay	62.0	73.1	63.0	0.031	0.16	79.6
w/o Retention	64.1	75.0	65.2	0.026	0.31	78.8
w/o Stabilization	63.4	74.1	64.5	0.030	0.19	88.6
`AgentFrontier`	**67.8**	**79.1**	**69.0**	**0.017**	**0.34**	**94.2**

5 Conclusion

We identified learning-frontier non-stationarity as a source of training collapse in active-reasoning RL. `AgentFrontier` combines Frontier Retention and Frontier Stabilization to preserve solved behaviors and maintain a balanced source of new learning signals. Together, the two mechanisms preserve learned competence while sustaining useful gradients on unresolved prompts. The model analysis traces frontier shrinkage and solved-prompt drift, while the hyperparameter and ablation results identify a stable operating region for the two mechanisms. Across five active-reasoning tasks, `AgentFrontier` raises the six-metric average of GSPO from 59.58 to 66.00, reduces reward collapse, and stabilizes training dynamics, lowering reward variance from 0.041 to 0.017.

AI Usage Statement. AI assisted only with language polishing.

REFERENCES

Kartikeya Badola, Jonathan Simon, Arian Hosseini, Sara Marie Mc Carthy, Tsendsuren Munkhdalai, Abhimanyu Goyal, Tomáš Kočiský, Shyam Upadhyay, Bahare Fatemi, and Mehran Kazemi. Multi-turn puzzles: Evaluating interactive reasoning and strategic dialogue in llms. arXiv preprint arXiv:2508.10142, 2025.

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

Sanjiban Choudhury. Process reward models for llm agents: Practical framework and directions. arXiv preprint arXiv:2502.10325, 2025.

Yang Deng, Lizi Liao, Liang Chen, Hongru Wang, Wenqiang Lei, and Tat-Seng Chua. Prompting and evaluating large language models for proactive dialogues: Clarification, target-guided, and non-collaboration. In Findings of the Association for Computational Linguistics: EMNLP, 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. URL https://arxiv.org/abs/2501.12948.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. In Advances in Neural Information Processing Systems, 2025.

Zhiyuan Hu et al. Uncertainty of thoughts: Uncertainty-aware planning enhances information seeking in large language models. In Advances in Neural Information Processing Systems, 2024.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. Vineppo: Unlocking rl potential for llm reasoning through refined credit assignment. arXiv preprint arXiv:2410.01679, 2024.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. arXiv preprint arXiv:2409.12917, 2024.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K. Qiu, and Yuqing Yang. Agent lightning: Train any ai agents with reinforcement learning. arXiv preprint arXiv:2508.03680, 2025.

Trung Quoc Luong et al. Reft: Reasoning with reinforced fine-tuning. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, 2024.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

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

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

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

Hanlin Wang, Chak Tou Leong, Jiashuo Wang, Jian Wang, and Wenjie Li. Spa-rl: Reinforcing llm agents via stepwise progress attribution. arXiv preprint arXiv:2505.20732, 2025.

Zhiheng Xi, Jixuan Huang, et al. Agentgym-rl: Training llm agents for long-horizon decision making through multi-turn reinforcement learning. In International Conference on Learning Representations, 2026.

Zhiheng Xi et al. Agentgym: Evolving large language model-based agents across diverse environments. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, 2025.

Fan Yin, Zifeng Wang, I-Hung Hsu, Jun Yan, Ke Jiang, Yanfei Chen, Jindong Gu, Long T. Le, Kai-Wei Chang, Chen-Yu Lee, Hamid Palangi, and Tomas Pfister. Magnet: Multi-turn tool-use data synthesis and distillation via graph translation. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, 2025.

Haofei Yu, Zhengyang Qi, Yining Zhao, Kolby Nottingham, Keyang Xuan, Bodhisattwa Prasad Majumder, Hao Zhu, Paul Pu Liang, and Jiaxuan You. Sotopia-rl: Reward design for social intelligence. arXiv preprint arXiv:2508.03905, 2025a.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. arXiv preprint arXiv:2503.14476, 2025b.

Siliang Zeng, Quan Wei, William Brown, Oana Frunza, Yuriy Nevmyvaka, and Mingyi Hong. Reinforcing multi-turn reasoning in llm agents via turn-level credit assignment. arXiv preprint arXiv:2505.11821, 2025.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. arXiv preprint arXiv:2507.18071, 2025.

Zhanke Zhou, Xiao Feng, Zhaocheng Zhu, Jiangchao Yao, Sanmi Koyejo, and Bo Han. From passive to active reasoning: Can large language models ask the right questions under incomplete information? In International Conference on Machine Learning, 2025.

Deyu Zou, Yongqiang Chen, Jianxiang Wang, Haochen Yang, Mufei Li, James Cheng, Pan Li, and Yu Gong. Reducing belief deviation in reinforcement learning for active reasoning of llm agents. In International Conference on Learning Representations, 2026.

Appendix

A Empirical Verification of the Frontier Analysis **13**

A.1 Estimating the Active Frontier Ratio 13

A.2 Mastered-Prompt Cohort Tracking 13

A.3 Gradient Concentration Measurement 13

B Implementation Details and Unified Algorithm **13**

B.1 Prompt Status Estimator Lifecycle 13

B.2 Full Configuration 14

B.3 Unified Training Procedure 14

B.4 Integration with PPO, GRPO, and GSPO 14

C Matched Controls for Regularization and Sampling **15**

D Robustness Beyond Binary Rewards **15**

E Additional Model Scales, Families, and Benchmarks **16**

E.1 Model Scale 16

E.2 Other Policy Families and Judge Independence 16

E.3 Single-Turn Reasoning and Long-Horizon Environments 16

F Complexity and Compute Cost **17**

G Limitations, Statistical Significance, and Compute Resources **17**

A Empirical Verification of the Frontier Analysis

This appendix reports the measurement protocols behind the four panels of Fig. 2 and the diagnostics referenced in Sec. 3. All quantities are logged during training on the CD and GN tasks with Qwen-2.5-7B-Instruct as the policy and the same configuration as the main runs, where the two CD panels of Fig. 2(b)(c) use PPO as the inner optimizer and every other panel uses GSPO.

A.1 Estimating the Active Frontier Ratio

At every optimization step we partition the training prompts by whether their rollout group in that step contains both a success and a failure. The frontier set $\mathcal{F}_t$ is the set of prompts in that mixed group, and $\rho_t = |\mathcal{F}_t|/N$ is reported directly. Because a single step gives a noisy view of a prompt's status, the curve in Fig. 2(a)(b)(d) is smoothed with the same exponential moving average used by Frontier Stabilization, with coefficient $\beta = 0.8$, and the raw per-step values are recorded alongside.

Two implementation points define this measurement. Prompts that are not sampled in a given step keep their previous status estimate. The denominator of ρ_t remains the fixed training-pool size N used in the main-text definition. Since the rewards are binary, zero within-group variance corresponds to uniform outcomes in a sampled group.

A.2 Mastered-Prompt Cohort Tracking

Fig. 2(c) follows one frozen cohort of prompts. At a chosen step t_0 we collect prompts whose rolling estimate exceeds τ_{master} and whose last two observed groups were all-success, following the main-text mastered-status rule. We freeze that set and evaluate its success rate at subsequent steps by re-rolling each prompt under the current policy with $G = 8$, averaging over three re-rollings. The shaded band represents the standard deviation across those re-rollings. The cohort composition stays fixed, so a decline reflects drift on prompts that were mastered at t_0.

A.3 Gradient Concentration Measurement

For GC_t in Eq. (2) we use the difficulty partition induced by the status estimator, so $\mathcal{C} = \{\mathcal{U}, \mathcal{F}, \mathcal{M}\}$, and we take $\|g_j^{(t)}\|$ as the ℓ_2 norm of the prompt-level policy gradient accumulated over the G rollouts of that prompt. Norms are read from the training framework's per-group gradient hooks at a single parameter slice, the final transformer block, and the same slice is used for every method. When $|\mathcal{F}_t|$ is small the numerator is dominated by one cluster and GC_t approaches one, which is the behavior plotted in Fig. 3d.

B Implementation Details and Unified Algorithm

B.1 Prompt Status Estimator Lifecycle

The status estimator $\hat{p}_j$ is the quantity on which both mechanisms depend, so its full lifecycle is given here.

1. **Initialization.** An unseen prompt starts with $\hat{p}_j = 0.5$.
2. **Update.** On each observation, $\hat{p}_j \leftarrow \beta\hat{p}_j + (1-\beta)\, k_j/G$ with $\beta = 0.8$.
3. **Classification.** The thresholds $\tau_{\text{master}} = 0.875$ and $\tau_{\text{unsolved}} = 0.125$ define the mastered, frontier, and unsolved regions. A mastered assignment additionally requires two consecutive all-success observations, as specified in the main-text experiment setup.
4. **Cache refresh.** A mastered prompt's successful cached trajectory y_j^* is refreshed every $K_{\text{ref}} = 50$ updates using the current policy.
5. **Staleness.** After 100 updates without observation, the cached estimate decays toward 0.5 by the factor 0.995^{Δ}, where Δ counts the updates beyond that threshold.

Retention therefore draws on mastered prompts only. Table 3 reports the sensitivity of the status-confirmation, refresh, and staleness settings, and Table 4 collects the complete configuration.

Table 3: Sensitivity to the estimator lifecycle parameters on CD and GN, both trained with GSPO as the inner optimizer. Each row changes one parameter from the default and keeps the rest of the configuration fixed.

Setting	CD	GN
Default (2 confirmations, $T_{\text{stale}} = 100$, $K_{\text{ref}} = 50$)	**67.8**	**79.1**
1 confirmation	66.9	78.4
3 confirmations	67.2	78.6
$T_{\text{stale}} = 50$	67.1	78.3
$T_{\text{stale}} = 150$	67.4	78.7
$K_{\text{ref}} = 25$	67.3	78.5
$K_{\text{ref}} = 100$	67.0	78.2

Table 4: Complete configuration of **`AgentFrontier`**. The estimator constants are those of Appendix B.1, the batch mixture is the default operating point of Frontier Stabilization, and the hardware matches Table 11.

Hyperparameter	Value	Role
Policy model	Qwen-2.5-7B-Instruct	all RL runs
Judge model (SP only)	Qwen-2.5-14B-Instruct	yes/no/unknown feedback
Rollout group size G	8	group-relative advantages
EMA coefficient β	0.8	status estimate smoothing
Mastered threshold τ_{master}	0.875	assigns $\mathcal{M}$
Unsolved threshold τ_{unsolved}	0.125	assigns $\mathcal{U}$
Initial estimate $\hat{p}_j$	0.5	unseen prompts
Mastered confirmation	2 all-success groups	status assignment
Cache refresh K_{ref}	50	retention target update
Staleness T_{stale}	100	begin estimate decay
Staleness decay	0.995^{Δ}	return toward 0.5
Retention weight α	0.05	$\mathcal{L}_{\text{total}}$
Frontier cap $\gamma_{\max}$	0.6	caps n_{f}/B
Mastered presence γ_{m}	1.0	scales n_{m}/B
Default batch mixture	$20/60/20$	solved/effective/unsolved
Hardware	8×A100 80GB	one training run

B.2 Full Configuration

We include the hyperparameters used for training **`AgentFrontier`** in Table 4, and no setting here changes across the main results, the ablation and the model scale sweep, beyond the single parameter that each sensitivity study varies.

With $G = 8$ rollouts per prompt and the caps above, a batch drawn at the default operating point holds 60% frontier prompts, and the mastered share follows $|\mathcal{M}|/N$, so the $20/60/20$ mixture is the composition observed once a fifth of the pool is mastered, which is the regime every main run spends most of its steps in.

B.3 Unified Training Procedure

Algorithm 1 states one step in full. Frontier Stabilization acts before sampling, Frontier Retention acts after the rollouts, and neither touches the advantage computation of the inner optimizer.

B.4 Integration with PPO, GRPO, and GSPO

The wrapper is optimizer agnostic because both mechanisms read only quantities that every group-relative optimizer already produces.

Retention adds one auxiliary loss and Stabilization adds one batch construction rule. No gradient path passes through the cached trajectories beyond the likelihood term in the retention loss of Sec. 3, and no additional model call is required. PPO enters through the same interface with one difference

Algorithm 1 `AgentFrontier` training step

Require: policy π_θ, pool $\mathcal{D}$, estimates $\{\hat{p}_j\}$, cache $\mathcal{B} = \varnothing$, thresholds $\tau_{\text{master}}, \tau_{\text{unsolved}}$, weights α, β, caps $\gamma_{\max}, \gamma_{\text{m}}$

draw a batch of prompts by Frontier Stabilization:
$n_{\text{f}} = \min(|\mathcal{F}|/N, \gamma_{\max})B,\ n_{\text{m}} = (|\mathcal{M}|/N)\gamma_{\text{m}}B,\ n_{\text{u}} = B - n_{\text{f}} - n_{\text{m}}$
for each sampled prompt x_j **do**
  roll out G trajectories, obtain binary rewards, compute group-relative advantages
  $\hat{p}_j \leftarrow \beta\hat{p}_j + (1-\beta)k_j/G$ ▷ update after the rollouts
  if $\hat{p}_j > \tau_{\text{master}}$ and two consecutive observed groups are all-success **then**
    add (x_j, y_j^*) to $\mathcal{B}$ and refresh the cached y_j^* every K_{ref} steps
  end if
end for
$\mathcal{L} \leftarrow \mathcal{L}_{\text{RL}} + \alpha\,\mathcal{L}_{\text{retain}}(\mathcal{B})$
update θ with the inner optimizer (PPO, GRPO, or GSPO)

Table 5: What each inner optimizer contributes, and which quantity the two mechanisms consume.

Optimizer	Advantage source	Consumed by
PPO	learned critic return	k_j, hence $\hat{p}_j$
GRPO	token-level group normalization	k_j and σ_r
GSPO	sequence-level group normalization	k_j and σ_r

worth stating: its advantage comes from a learned critic, so a mastered prompt keeps a non-zero update on PPO even when its group-relative signal, which is what $\hat{p}_j$ reads, has vanished. The two CD panels of Fig. 2 are therefore where the PPO frontier dynamics are visible, and Table 1 reports vanilla results for all three optimizers and **`AgentFrontier`** results on GRPO and GSPO.

C Matched Controls for Regularization and Sampling

A natural reading of the two mechanisms is that Frontier Retention behaves as a conditional behavior-cloning regularizer and that Frontier Stabilization behaves as a difficulty-aware sampler, so we test both readings directly. We hold the inner optimizer, the total number of sampled prompts, the rollout group size, and the update budget fixed and replace only the control signal, and each control is tuned over the same grid used for **`AgentFrontier`**.

Table 6 separates the two questions these controls answer. A global KL anchor does limit drift, and its retention of 82.1 sits well above vanilla training, but it constrains every sampled token including the frontier prompts that still need to move, and its $\rho_{\min}$ of 0.15 shows a frontier that keeps collapsing. Uniform BC replay reaches 85.7 on retention because it revisits past data, and it has no way to tell whether a prompt has lost its group-relative gradient, so part of its budget goes to prompts that their own non-zero advantage already protects. Self-paced and learning-progress sampling both raise $\rho_{\min}$ above 0.20 because they keep hard prompts in view, and neither opposes mastered-prompt drift, with retention staying below 84 in both cases. **`AgentFrontier`** is the only control that keeps $\rho_{\min}$ above 0.28 while also holding retention above 94, which is the pairing the analysis in Sec. 3 identifies as necessary.

D Robustness Beyond Binary Rewards

Generalizing the frontier definition. The binary form $\mathcal{F}_t = \{j : 0 < k_j < G\}$ in Definition 1 is what makes the closed-form analysis in Proposition 1 exact. For a general scalar reward the object that matters is whether a group produces nonzero centered advantages at all, so we define

$$\mathcal{F}_\epsilon = \{j : \sigma_r(x_j) > \epsilon\}, \tag{7}$$

where $\sigma_r(x_j)$ is the within-group reward standard deviation. Binary rewards are the special case in which $\sigma_r(x_j) = 0$ exactly when $k_j \in \{0, G\}$. The algorithm consumes only observed group

Table 6: Matched controls. Every row shares the GSPO backbone, the update budget, and the rollout count. $\rho_{\min}$ is the lowest active frontier ratio reached during training and Ret. is the mastered-prompt cohort retention of Appendix A.2, so the two rightmost columns are the Eff. and Ret. entries of Table 2 for the four rows the two tables share.

Control signal	CD	GN	MR	$\rho_{\min}\uparrow$	Ret.↑
None (vanilla GSPO)	60.1	71.2	61.7	0.07	71.5
Dynamic sampling	61.4	72.6	62.4	0.18	74.3
Uniform random replay	62.0	73.1	63.0	0.16	79.6
Global KL to reference	61.8	72.9	62.8	0.15	82.1
Uniform BC replay	63.6	74.5	64.8	0.18	85.7
Self-paced sampling	62.5	73.8	63.6	0.21	80.4
Learning-progress sampling	63.0	74.5	64.1	0.23	83.9
Online gradient activity (`AgentFrontier`)	**67.8**	**79.1**	**69.0**	**0.34**	**94.2**

Table 7: Robustness to non-binary rewards, correlated rollouts, and noisy judging. Each stress condition is applied to the task where it is naturally defined, holding every other setting fixed. The WebShop row is mean dense reward in $[0, 1]$ scaled by 100 and the other two rows are EM and F1-word respectively.

Stress condition	Vanilla	+AF	Gain
Shared-prefix correlated rollouts (CD)	60.1	66.3	$+6.2$
Dense partial reward in $[0, 1]$ (WebShop)	51.8	58.9	$+7.1$
Injected judge noise, 10% flips (SP, F1-word)	33.1	37.7	$+4.6$

statistics and requires no independence assumption of its own, since independence enters the proof of Proposition 1 only. Table 7 tests the three conditions under which that distinction matters.

The gain under dense reward of $+7.1$ is the largest of the three, which is consistent with the mechanism. A dense score keeps more groups in the interior of the reward range, so the frontier defined by $\sigma_r > \epsilon$ stays wider and Frontier Stabilization has more to balance. Judge noise compresses both columns because it corrupts the status estimate itself, which is the failure mode the Limitations paragraph of Appendix G describes.

E Additional Model Scales, Families, and Benchmarks

E.1 Model scale

Table 8 sweeps the Qwen-2.5 family across three policy sizes, from 3B to 14B. The frontier phenomenon and the correction both persist at every scale, and the gain over the vanilla optimizer stays inside the $+6.2$ to $+7.9$ interval at all three.

E.2 Other policy families and judge independence

Table 9 reports the five-task average for the three policy rows and the SituationPuzzles F1-word score for the judge row. The xLAM row uses the exact checkpoint `Salesforce/Llama-xLAM-2-8b-fc-r`, whose post-training targets multi-turn tool interaction. It starts from the highest vanilla average, 64.2 against 60.4 and 62.1, and ends with the smallest relative gain of the three, 9.2% against 10.6% and 9.5%, consistent with its stronger starting behavior on multi-turn interaction.

E.3 Single-turn reasoning and long-horizon environments

The single-turn gains of $+3.7$ and $+3.5$ fall below the $+7.8$ median of the 18 task gains in Table 1, and the long-horizon gains of $+6.7$ and $+7.1$ sit inside the $+6.2$ to $+7.9$ interval that the scale sweep of Table 8 reports. This ordering matches the analysis. Volatility in p_j is what makes a narrow frontier

Table 8: Model scale sweep. Same wrapper, same update budget, same $G = 8$ at every scale.

	CD		GN		MR	
Policy	GSPO	+AF	GSPO	+AF	GSPO	+AF
Qwen-2.5-3B	51.4	57.6	62.8	69.4	53.9	60.2
Qwen-2.5-7B	60.1	67.8	71.2	79.1	61.7	69.0
Qwen-2.5-14B	65.3	72.4	76.8	83.9	66.4	73.1

Table 9: Non-Qwen policy families, and the SituationPuzzles judge swapped away from the Qwen family. The policy rows average the five task metrics of Table 1, and four of the five tasks use rule-based feedback and are already judge independent. $\rho_{\min}$ is reported as vanilla $\rightarrow$ `AgentFrontier`.

Policy or judge	Vanilla	+AF	$\rho_{\min}$
Mistral-7B-Instruct	60.4	66.8	$0.09 \rightarrow 0.31$
Llama-3.1-8B-Instruct	62.1	68.0	$0.10 \rightarrow 0.32$
xLAM-2-8B-fc-r	64.2	70.1	$0.08 \rightarrow 0.33$
SituationPuzzles, Llama-3.3-70B judge	35.8	40.1	$0.12 \rightarrow 0.30$

damaging, and early-turn actions in interactive tasks produce more of that volatility than a single completion does.

F Complexity and Compute Cost

Per-step overhead. Frontier Stabilization requires $O(N)$ storage for the status estimates and $O(B \log B)$ to assemble one batch by bucketed sampling. Frontier Retention stores one cached trajectory per mastered prompt, bounded by N entries, and adds one forward and one backward pass over the retained tokens, which is at most the token count of a full rollout group. No additional model inference is triggered.

The 9% wall-clock overhead in Table 11 comes from the retention pass and the bucketed sampler. Since `AgentFrontier` needs no extra rollouts to reach its higher score, score per GPU-hour on CD rises from 1.30 under vanilla GSPO to 1.34 under the wrapper, and the extra compute is what buys the gain while the sampling budget stays fixed.

G Limitations, Statistical Significance, and Compute Resources

Limitations. As discussed in Sec. 3, our study centers on group-relative RL with binary outcome rewards, where a prompt contributes an update through the variance of its own rollout group. Dense partial credit and judge-scored feedback produce different dynamics for the effective-prompt ratio and mastered-prompt status, and Appendix D measures both conditions. Furthermore, both mechanisms estimate prompt status from repeated rollouts of the same prompt, and settings where a prompt is visited only a small number of times remain untested. A promising direction for future work is to extend the frontier view to those sparse-visit regimes by sharing the status estimate across related prompts. Our evaluation covers the five active reasoning tasks and the single-turn and long-horizon settings of Appendix E.3. Multimodal interaction and open-ended settings without a verifiable outcome stay outside this scope. The bookkeeping cost stays bounded by the prompt pool because the wrapper keeps $O(N)$ status estimates and one cached trajectory per mastered prompt. Excessive retention anchors the policy to past trajectories, which is why we sweep α in Fig. 4a and keep the retention weight at $\alpha = 0.05$.

Statistical significance. We use the same rollout protocol for all tasks, and the shaded band on the cohort curve of Fig. 2(c) represents the standard deviation across the three re-rollings of the frozen cohort described in Appendix A.2. For the estimator lifecycle we report the spread over the seven settings of Table 3, which stays within one point of the default on both CD and GN. We therefore hold the lifecycle fixed in the remaining tables and attribute their differences to the control signal under

Table 10: Settings outside the five active reasoning tasks. The single-turn rows locate frontier shrinkage inside group-relative optimization, where agent interaction is absent, and the long-horizon rows run the same wrapper where volatility is stronger. AIME24 and LiveCodeBench report accuracy and ALFWorld reports task success, while the WebShop pair is mean dense reward in $[0, 1]$ scaled by 100 and matches the dense-reward row of Table 7.

Setting	Regime	Vanilla	+AF	ρ before collapse
AIME24	single-turn math	28.7	32.4	0.10
LiveCodeBench	single-turn code	24.6	28.1	0.12
ALFWorld	long-horizon agent	51.6	58.3	0.08
WebShop	dense-reward agent	51.8	58.9	0.11

Table 11: Training cost on CD with GSPO as the inner optimizer on $8\times$A100 80GB. Overhead is the ratio to the vanilla GSPO row, computed from either of the two preceding columns, and GPU-hours are reported to one decimal.

Method	GPU-hours	Step time	Overhead
Vanilla GSPO	46.4	37.2 s	$1.00\times$
+ Dynamic sampling	48.1	38.6 s	$1.04\times$
+ AgentFrontier	50.7	40.7 s	$1.09\times$
+ T^3 + AgentFrontier	55.3	44.4 s	$1.19\times$

test. Tables 1, 2 and the appendix sweeps report one score per configuration, and every configuration is trained under the identical rollout budget listed in Table 4.

Compute resources. All experiments were conducted on NVIDIA A100 GPUs with 80GB memory, and each RL training run uses 8 such GPUs across rollout generation, environment interaction, reward computation, and parameter updates. The estimated GPU-hours required to train each configuration are detailed in Table 11, and the hardware of every run is listed in Table 4. Consequently, a typical GSPO run on CD takes approximately 46.4 GPU-hours, whereas the longest configuration, which stacks T^3 onto the wrapper, exceeds 55 GPU-hours. The wrapper triggers no additional model inference beyond the rollouts already sampled by the run. Its total cost scales linearly with the number of training prompts, the rollout group size G, and the number of optimization steps, as analyzed in Appendix F.

Broader impacts. `AgentFrontier` aims to improve the stability and efficiency of RL training for active reasoning agents. More reliable active reasoning agents may help reduce redundant interactions, improve information-seeking behavior, and lower the cost of training multi-turn LLM agents. At the same time, stronger interactive agents may also increase the capability of systems used for harmful purposes if deployed without safeguards. We therefore recommend using `AgentFrontier` together with standard model safety evaluation, controlled release practices, and downstream monitoring.