BEYOND SELF-CONSENSUS: MITIGATING SELF-REINFORCING EXPLORATION BIAS IN TEST-TIME REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time reinforcement learning (TTRL) enables language models to improve reasoning under unlabeled test-time settings through self-generated supervision. Yet this supervision is intrinsically coupled to the model's own exploration distribution, creating an endogenous feedback process distinct from conventional reinforcement learning with externally grounded rewards. We identify this feedback as *self-reinforcing exploration bias*, characterized by the amplification of existing policy preferences through endogenous pseudo-supervision and the progressive concentration of subsequent exploration around a restricted set of solution modes. To mitigate this bias, we introduce a two-dimensional structured exploration framework that expands reasoning-space coverage through multiple model-driven reasoning views and adapts exploration depth to instance-specific search demand. The resulting framework increases access to alternative solution trajectories while allocating test-time computation according to heterogeneous exploration requirements. Across multiple reasoning benchmarks and model scales, our approach consistently improves TTRL performance while reducing sampling cost by more than 30%. Our analysis shows that the resulting limitation is not solely a consequence of noisy pseudo-rewards or unstable optimization, but also reflects insufficient control over the solution space exposed during adaptation. These findings establish exploration as a quantitatively actionable dimension of TTRL, providing a principled route toward mitigating self-reinforcing adaptation under self-generated supervision.

1 INTRODUCTION

Test-time reinforcement learning (TTRL) Zuo et al. (2025) has emerged as a compelling framework for improving language-model reasoning under unlabeled test-time settings, supporting online policy adaptation through self-generated supervision. At its core, TTRL follows the same exploration-and-reinforcement structure as conventional reinforcement learning, leveraging sampled reasoning trajectories to guide subsequent policy updates. However, unlike conventional reinforcement learning, TTRL lacks an external ground truth reward signal that independently evaluates these trajectories, and instead typically derives supervision from consensus among the model's own generations Zuo et al. (2025); Wang et al. (2022). The resulting pseudo-supervision is thus inherently coupled to the model's existing exploration distribution, favoring solutions that are already prominent in its current generations. As illustrated in Figure 1, this initial preference can be further amplified by policy optimization Liao et al. (2026); Lin et al. (2026), progressively reducing exploration of alternative solutions and concentrating subsequent learning around the model's existing solution modes Zhao et al. (2026). We term this phenomenon *self-reinforcing exploration bias*, referring to the progressive concentration of exploration around policy-preferred solutions under self-generated supervision. In practice, self-reinforcing exploration bias can induce premature saturation of test-time adaptation and incomplete realization of the model's latent reasoning capabilities Lin et al. (2026), motivating our central question: *How can TTRL sustain effective exploration in the presence of self-reinforcing exploration bias?*

Existing studies in TTRL have largely centered on pseudo-reward construction and reinforcement-learning optimization. Recent work on pseudo-reward construction has improved supervision by

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.

filtering low-confidence generations, restricting updates to strongly supported consensus targets, and verifying candidate answers with auxiliary signals Wang et al. (2026b); Yan et al. (2026); Liao et al. (2026); Zhao et al. (2026); Xu et al. (2026). These approaches improve the fidelity of the supervision signal extracted from model generations, but largely remain limited to the candidate solutions already exposed by the current policy, leaving the coverage of the explored solution space largely unchanged Wang et al. (2026a;b); Zhao et al. (2026); Xu et al. (2026). A complementary line of work focuses on the optimization stage, improving policy updates through reward shaping, selective training, and modified rollout strategies Wu et al. (2026); Jin et al. (2026); Cui et al. (2025); Zhang et al. (2026b); Ziegler et al. (2019). These methods can stabilize learning and improve the efficiency of reinforcement updates under noisy self-generated rewards, but they primarily regulate optimization within the solution distribution already induced by the current policy. Consequently, the exploration dimension of test-time adaptation remains comparatively underexamined, limiting the model's ability to sustain informative search beyond its dominant solution modes.

To address this challenge, we explicitly structure exploration in TTRL along two orthogonal dimensions, broadening the range of reasoning modes explored and adapting the depth of search through instance-specific sampling allocation. First, we expand the reasoning space through complementary reasoning views, including natural-language chain-of-thought Wei et al. (2022) and executable program-based reasoning Chen et al. (2023), and aggregate their resulting solutions into a unified supervision target. Importantly, our program-based view differs from tool-augmented training paradigms that incorporate external tools as additional problem-solving agents. Instead, program generation remains entirely model-driven, while execution provides an alternative reasoning pathway that broadens the model's exploration of the same problem and exposes candidate solutions that may be underrepresented in natural-language reasoning.

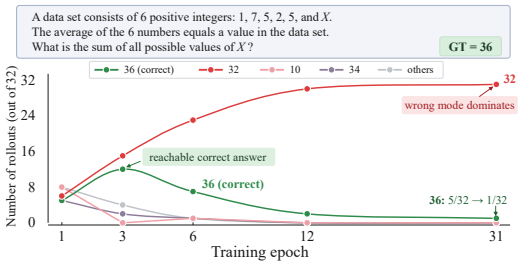

Figure 1: **Representative example of self-reinforcing exploration bias during TTRL**. An initially reachable correct solution is progressively suppressed during adaptation, widening the gap between available reasoning capability and realized policy behavior.

Beyond expanding the reasoning space, we further adapt exploration depth to the heterogeneous difficulty of individual instances. The amount of search required to expose useful solution trajectories varies substantially across problems, rendering uniform sampling an inefficient allocation of exploratory capacity. We therefore allocate exploration effort in proportion to instance-specific search demand, increasing sampling for instances with limited target accessibility and reducing it as additional rollouts provide diminishing benefit. This yields a more targeted use of test-time computation while maintaining sufficient exploration for challenging instances. In practice, the resulting allocation also improves computational efficiency, reducing sampling cost by more than 30% relative to standard TTRL. To prevent these exploratory gains from being absorbed into the policy's evolving generation bias, we further construct the supervision targets before adaptation and keep them fixed throughout training. Our contributions can be summarized as follows:

- **Problem Formulation.** We identify self-reinforcing exploration bias as a distinctive challenge of TTRL under self-generated supervision, induced by the endogenous dependence of the supervision signal on the model's current exploration distribution. This formulation establishes exploration as an independent design dimension of TTRL beyond pseudo-reward estimation and policy optimization.

- **Structured Exploration Framework.** We introduce a two-dimensional exploration framework for TTRL, combining multi-view reasoning-space expansion with instance-adaptive exploration depth. This design broadens the effective solution space available for learning while allocating search effort according to heterogeneous instance difficulty.

- **Empirical Validation.** We empirically validate the proposed exploration framework across multiple reasoning benchmarks and model families, showing consistent performance gains and more than 30% reduction in sampling cost through explicit control of exploration breadth and depth.

2 Related Work

Test-Time Reinforcement Learning with Self-Generated Supervision. TTRL (Zuo et al., 2025) extends self-consistency (Wang et al., 2022) from answer selection to adaptation on unlabeled test problems. Subsequent methods refine consensus and pseudo-label selection (Wang et al., 2026b; Yan et al., 2026), reshape exploration through reward design, hints, and branching (Du et al., 2026; Xu et al., 2026; Zhao et al., 2026), or co-evolve generation and self-verification (Pan et al., 2026). Analyses of spurious reward amplification and correct-answer extinction expose risks of this endogenous supervision (Yu et al., 2026; Lin et al., 2026). TEMPO addresses reward drift using labeled critic recalibration (Zhang et al., 2026a), differing from our label-free setting.

Self-Harmony combines original and paraphrased problem views (Wang et al., 2026a), program-of-thought prompting delegates computation to an interpreter (Chen et al., 2023), while T^3RL uses tool verification to refine online consensus (Liao et al., 2026). Our framework coordinates complementary candidate generation before adaptation with instance-adaptive sampling during learning. CoT and executable reasoning construct stable targets, while subsequent rollout budgets respond to target accessibility, linking exploration breadth and depth without continually redefining supervision.

Optimization Dynamics in Reinforcement Learning for Reasoning. Policy entropy and exploration depend on probability–advantage covariance, positive-advantage updates, and group-relative credit assignment. Interventions include KL-Cov, positive-advantage reweighting, quantile advantage estimation, adaptive entropy regularization, and KL-based policy regularization (Cui et al., 2025; Jin et al., 2026; Wu et al., 2026; Zhang et al., 2026b; Ziegler et al., 2019). Lower entropy does not universally imply poorer reasoning (Jin et al., 2026). In TTRL, stabilizing updates need not correct erroneous pseudo-targets or expose alternative answers. We therefore study optimization, target construction, and rollout allocation as complementary design choices.

3 Methodology

In this section, we develop a structured exploration framework that separates candidate discovery from subsequent policy adaptation to mitigate self-reinforcing exploration bias induced by endogenous supervision. As illustrated in Figure 2, the framework combines multi-view candidate exploration with stable target construction, followed by reachability-aware policy adaptation. This design jointly controls exploration breadth and depth while reducing the feedback between the evolving policy and its self-generated supervision.

We consider an unlabeled problem set $\mathcal{D} = \{x_i\}_{i=1}^{N}$ and adapt a policy π_θ initialized from a base policy π_{θ_0}. For each problem, the base policy first explores a broader candidate space through complementary reasoning and presentation views. The resulting candidates are consolidated into a stable target before adaptation begins. The policy is then optimized toward these targets using on-policy trajectories, with exploration effort adjusted according to target accessibility under the evolving policy. Ground-truth answers are not used during either target construction or policy adaptation.

3.1 Exploration Breadth: Multi-View Target Construction

Standard TTRL constructs supervision from solutions already prominent under a single generation mode, making the resulting target sensitive to the policy's initial preference. We instead broaden candidate discovery by eliciting the same base policy π_{θ_0} through a set of complementary views $\mathcal{V}(x)$. Each view $v \in \mathcal{V}(x)$ specifies a prompt transformation τ_v and an answer mapping ϕ_v. We sample responses from $\pi_{\theta_0}(\cdot \mid \tau_v(x))$; the mapping returns an element of the shared answer space $\mathcal{A}(x)$ or abstains when no valid answer can be extracted. This common space allows candidates produced by different reasoning procedures or option orders to contribute to the same vote.

For open-ended problems, we use natural-language chain-of-thought (CoT) and executable program-of-thought (PoT) as complementary reasoning views. CoT elicits natural-language reasoning trajectories, whereas PoT induces executable program representations that realize an alternative computational pathway before mapping the result to the shared answer space. Importantly, PoT introduces computational diversity through execution rather than an additional answer-generating model, and program execution is restricted to target construction. CoT and PoT are assigned matched generation budgets to control for differences in sampling scale across reasoning views.

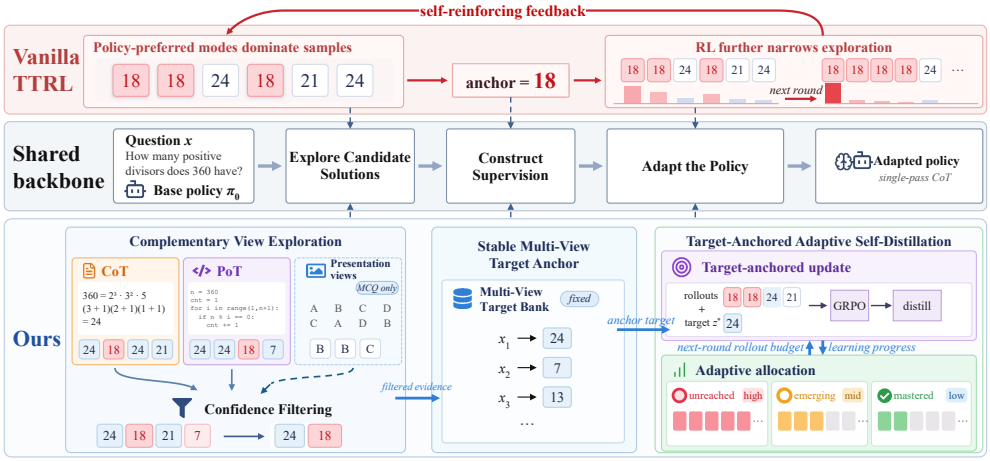

Figure 2: **Overview of our structured exploration framework for TTRL.** Our method structures exploration along two dimensions: (1) **breadth**, through complementary reasoning and presentation views that expand candidate solution coverage, and (2) **depth**, through adaptive rollout allocation based on target accessibility. A stable multi-view target bank connects candidate discovery to target-anchored policy updates, mitigating the self-reinforcing feedback between the evolving policy and its supervision.

For multiple-choice problems, we additionally introduce presentation diversity through cyclic permutations of the answer choices. Predictions from each permutation are mapped back to the original option identities before aggregation, ensuring that all views contribute to the same canonical answer space. The CoT generation budget is distributed across these presentation views while keeping the total CoT sampling budget fixed. PoT predictions are mapped to the same canonical option space, allowing reasoning and presentation views to be aggregated under a unified target construction procedure. Appendix C.1 provides the prompts and execution details.

To reduce the influence of low-confidence generations, responses are filtered independently within each view according to their mean token log-probability. For a response $y = (y_1, \ldots, y_T)$, we define

$$c(y) = \frac{1}{T} \sum_{t=1}^{T} \log \pi_{\theta_0}(y_t \mid \tau_v(x), y_{<t}), \tag{1}$$

where the dependence on x and v is implicit in $c(y)$. We retain the top fraction of valid samples separately within each view. This avoids directly comparing prose and program scores, whose distributions can differ.

Let $S_v(x)$ denote the retained responses from view v. The multi-view target is obtained by aggregating the retained candidates in the shared answer space,

$$\hat{a}(x) \in \arg\max_{a \in \mathcal{A}(x)} \sum_{v \in \mathcal{V}(x)} \sum_{y \in \mathcal{S}_v(x)} \mathbf{1}[\phi_v(y) = a], \tag{2}$$

For each $x_i \in \mathcal{D}$, this procedure defines a target

$$\hat{a}_i = G(x_i; \pi_{\theta_0}), \qquad x_i \in \mathcal{D}, \tag{3}$$

where G denotes the complete multi-view elicitation, confidence filtering, and aggregation procedure. The target set $\{\hat{a}_i\}_{i=1}^{N}$ is constructed once before adaptation and remains fixed throughout optimization. This temporal separation preserves the candidate evidence exposed during initial exploration and prevents subsequent policy updates from recursively redefining the supervision used for later updates.

3.2 Exploration Depth: Reachability-Aware Target-Anchored Adaptation

After constructing the fixed multi-view targets, we adapt the policy toward these targets using on-policy CoT rollouts. A target identified through complementary views may initially have low prob-

Algorithm 1 Structured Exploration with Multi-View Self-Distillation

Require: unlabeled problems $\mathcal{D}$, base policy π_{θ_0}, view set $\mathcal{V}$
1: construct $\hat{a}_i \leftarrow G(x_i; \pi_{\theta_0})$ for every $x_i \in \mathcal{D}$ using multi-view elicitation
2: initialize policy $\pi_\theta \leftarrow \pi_{\theta_0}$ and per-problem rollout states
3: **for** each adaptation iteration **do**
4: select active problems and scheduled probes; generate CoT groups with their assigned budgets
5: compute rewards against the fixed labels $\{\hat{a}_i\}$
6: update π_θ with GRPO on training groups with non-zero reward variance
7: update rollout budgets and suspension states from observed target hit rates
8: **end for**

ability under the policy's CoT distribution, creating a mismatch between the supervision available for adaptation and the trajectories currently accessible to the policy. We characterize this mismatch through target reachability. For problem x_i with fixed target $\hat{a}_i$, we define

$$p_i(\theta) = \Pr_{y \sim \pi_\theta(\cdot|\tau_{\mathrm{CoT}}(x_i))}[\phi_{\mathrm{CoT}}(y) \equiv \hat{a}_i] \tag{4}$$

where $\equiv$ denotes the task-specific answer-matching rule. Target reachability therefore measures the probability that the current policy produces a trajectory consistent with the fixed target under the training prompt. For each sampled trajectory y, we assign a binary target-consistency reward

$$r(y; x_i) = \mathbf{1}[\phi_{\mathrm{CoT}}(y) \equiv \hat{a}_i]\,. \tag{5}$$

Policy optimization is performed with GRPO using binary target-consistency rewards. Because GRPO relies on relative differences within each rollout group, the training signal is informative only when the group contains both target-consistent and target-inconsistent trajectories. Let p_i denote the current reachability of the fixed target for problem x_i. For a group of n independently sampled rollouts, the probability of obtaining such a mixed group is

$$1 - p_i^n - (1 - p_i)^n. \tag{6}$$

This expression shows that additional sampling can increase the probability of obtaining a non-degenerate training group when target reachability is limited. In practice, however, allocating the same large budget to targets that are already frequently observed is computationally inefficient. We therefore use observed target accessibility to adapt the rollout budget across problems.

Motivated by this dependence, we assign each problem an adaptive rollout budget n_i that evolves with its observed target accessibility. If a rollout group contains no target-consistent trajectory, the budget is increased on the next visit to expand the search depth for that problem. Once the target is observed with sufficient frequency, the budget is reduced to avoid unnecessary sampling. Repeated homogeneous groups provide limited additional training signal and are therefore temporarily suspended, including all-match groups and no-match groups that persist at the maximum budget. These problems are periodically revisited as the shared policy evolves, allowing their target accessibility to be reassessed without continuously allocating computation to uninformative rollouts. This procedure concentrates exploration on problems whose targets remain difficult to access while reducing redundant computation once additional search yields limited training signal. Reachability-aware allocation thus concentrates adaptation effort according to the accessibility of each fixed target. Combined with multi-view target construction, it completes the breadth–depth exploration framework summarized in Algorithm 1.

4 Experiments

This section covers the setup, performance comparisons, anti-collapse strategies, complementary views, ablations, and efficiency (Sections 4.1–4.6).

4.1 Experimental Setup

Benchmarks. We evaluate on five competition-mathematics benchmarks spanning both open-ended (OE) and multiple-choice (MCQ) reasoning. The OE suite contains AIME 2025, AMC, and

Table 1: **Comparison with state-of-the-art TTRL methods across multiple reasoning benchmarks and model backbones.** We report sampled pass@1 accuracy (%). **Bold** and underline denote the best and second-best results, respectively. Numbers in parentheses denote the absolute improvement of our method over the strongest baseline in each setting.

Backbone Model	Method	AIME 25	AMC	HMMT 26	AMC12-MC	JEEBench	Avg
Qwen3-1.7B-Base	Vanilla TTRL Zuo et al. (2025)	6.56	40.21	2.65	44.06	35.85	25.87
	ECHO Zhao et al. (2026)	6.67	39.61	1.89	50.90	39.62	27.74
	T3RL Liao et al. (2026)	5.83	40.66	2.56	46.41	42.29	27.55
	SCRL Yan et al. (2026)	9.17	39.16	0.38	21.29	41.51	22.30
	SCOPE Wang et al. (2026b)	6.25	40.96	2.27	45.13	41.04	27.13
	Ours	**22.50** (+13.33)	**42.70** (+1.74)	**7.39** (+4.74)	**58.28** (+7.38)	**43.93** (+1.64)	**34.96** (+7.22)
Granite-4.1-3B-Base	Vanilla TTRL Zuo et al. (2025)	30.00	71.99	15.91	61.32	61.77	48.20
	ECHO Zhao et al. (2026)	30.00	71.54	15.56	60.68	60.90	47.74
	T3RL Liao et al. (2026)	29.58	71.76	15.74	63.24	63.85	48.83
	SCRL Yan et al. (2026)	29.86	72.10	16.74	61.03	63.44	48.63
	SCOPE Wang et al. (2026b)	29.92	71.84	16.27	60.82	63.52	48.47
	Ours	**33.46** (+3.46)	**73.53** (+1.43)	**20.88** (+4.14)	**66.10** (+2.86)	**69.35** (+5.50)	**52.66** (+3.83)

Table 2: **Comparison with anti-collapse training methods under standard TTRL supervision.** Final accuracy and signal alignment are reported on AIME 2025 and AMC.

	AIME 2025		AMC	
Method	Acc.	Align. ($\times$100)	Acc.	Align. ($\times$100)
Vanilla TTRL Zuo et al. (2025)	6.56	13.85	40.21	40.29
+ QAE Wu et al. (2026)	6.67	10.64	39.31	38.11
+ Adaptive-KL Ziegler et al. (2019)	6.67	14.50	38.10	38.28
+ PAR Jin et al. (2026)	10.00	18.80	38.40	40.22
+ KL-Cov Cui et al. (2025)	5.00	16.70	37.50	39.43
+ AER Zhang et al. (2026b)	7.08	11.55	40.81	40.69
Ours	**22.50**	**40.94**	**42.70**	**49.95**

HMMT February 2026, while the MCQ suite contains AMC12-MC and JEEBench. We follow the transductive test-time adaptation setting: the model adapts on the same unlabeled problems on which it is evaluated, and ground-truth answers are used only for evaluation. Unless stated otherwise, we report sampled pass@1 estimated from eight independent generations per problem.

Models and baselines. We report results on two base-model families, Qwen3-1.7B-Base and Granite-4.1-3B-Base. We compare against vanilla TTRL and recent TTRL variants including ECHO, SCRL, SCOPE, and T^{3}RL. For mechanism studies, we additionally compare with optimization-side controls that preserve vanilla TTRL's online majority supervision while modifying the update, including QAE, positive-advantage reweighting (PAR), KL-Cov, adaptive entropy regularization (AER), and Adaptive-KL. All methods use the same backbone, prompt format, and GRPO configuration wherever their algorithms permit. Vanilla TTRL constructs supervision from fresh current-policy votes, whereas our method distills a stable multi-view target bank using adaptive per-instance rollout allocation. Full prompts, hyperparameters, and dataset preparation details are provided in Appendix C.

4.2 Performance Comparison across Models and Benchmarks

We first conduct a comprehensive comparison with recent TTRL methods across multiple model families and five reasoning benchmarks spanning both open-ended and multiple-choice settings. Table 1 reports sampled pass@1 performance against state-of-the-art TTRL methods, including ECHO, T3RL, SCRL, and SCOPE, across all backbone–benchmark combinations. As shown, our method achieves the strongest performance across the evaluated settings, outperforming the strongest competing baseline by 4.62 percentage points on average. Beyond this overall advantage, the magnitude of improvement varies systematically with task difficulty, with the largest gains observed on challenging open-ended benchmarks while remaining consistently positive on easier and multiple-choice settings. This pattern supports the interpretation that restricted exploration becomes more consequential when adaptation depends on recovering low-support solution trajectories rather than reinforcing already dominant modes. The persistence of this trend across both model families further indicates that the benefit of structured exploration is not contingent on a particular level of underlying reasoning capability.

4.3 Evaluating Anti-Collapse Training Strategies under TTRL Supervision

We next evaluate the effectiveness of existing anti-collapse training strategies in mitigating self-reinforcing exploration bias under standard TTRL supervision. In Table 2, we report final accuracy and signal alignment across five representative anti-collapse RL methods, covering entropy regularization (AER Zhang et al. (2026b)), policy-drift control (KL-Cov Cui et al. (2025) and Adaptive-KL Ziegler et al. (2019)), and advantage-based update dynamics (QAE Wu et al. (2026) and PAR Jin et al. (2026)). Signal alignment is defined as the cosine similarity between the advantage induced by a method's pseudo-reward and the advantage induced by ground-truth rewards, providing a direct measure of agreement between the resulting learning signal and the ground-truth update direction. Ground-truth rewards are used exclusively for diagnostic evaluation and are excluded from the training process. As shown, existing anti-collapse strategies yield only limited improvement under standard TTRL supervision. In contrast, our method consistently produces stronger final performance together with substantially improved signal alignment across both datasets. Notably, stabilizing the RL optimization process alone does not correct the learning direction induced by endogenous pseudo-supervision, as the resulting updates remain largely anchored to the policy's existing solution preferences.

We further examine the training dynamics in Figure 3 to characterize the temporal manifestation of self-reinforcing exploration bias during adaptation. Across both benchmarks, standard TTRL methods exhibit rapid early improvement followed by premature adaptation saturation, with limited gains from continued optimization. In contrast, our method sustains performance improvement over a substantially longer training horizon and converges to a consistently stronger solution. Notably, this pattern is less consistent with optimization instability, which typically manifests as noisy or degrading trajectories, and instead reflects an early loss of adaptation gain as exploration becomes progressively less informative under concentrated self-generated supervision. This interpretation is further supported by Table 2, where explicit anti-collapse strategies provide only limited recovery under unchanged self-generated supervision. These results indicate that *self-reinforcing exploration bias cannot be reduced to conventional reinforcement learning collapse and instead reflects a distinct feedback mechanism induced by the coupling between policy exploration and self-generated supervision.*

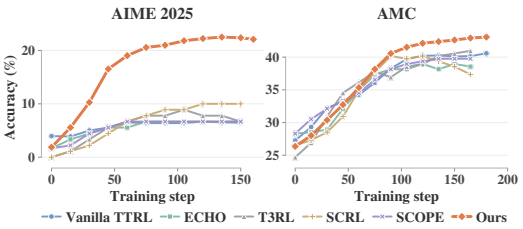

Figure 3: **Training Dynamics under Self-Generated Supervision.** Standard TTRL baselines exhibit rapid early improvement followed by premature adaptation saturation, whereas our method sustains performance gains over a longer training horizon. These results indicate that self-reinforcing exploration bias manifests as a progressive loss of informative adaptation, rather than conventional optimization collapse.

4.4 Complementary Reasoning and Surface Variation under Matched Budgets

We next examine the source of multi-view gains by comparing complementary reasoning views with answer-preserving surface variations under matched sampling budgets. Table 3 compares CoT-only exploration with CoT augmented by either question rewriting or executable program reasoning across multiple rollout budgets. Overall, surface-level rewriting provides limited and inconsistent gains, with its benefit diminishing as the sampling budget increases. In contrast, the program view delivers consistent improvements across budgets and benchmarks. These results indicate that effective multi-view exploration depends on computational complementarity rather than presentation diversity alone, enabling access to solution modes that remain underrepresented under standard CoT exploration.

Table 3: **Comparison of alternative multi-view constructions under matched sampling budgets.** Target accuracy (%) for CoT-only, rewrite-augmented, and program-augmented views are reported across multiple rollout budgets.

Budget	Views	AIME 25	AMC
64	CoT	6.67	45.78
	+ Rewrite	13.33 (+6.67)	45.78 (+0.00)
	+ PoT	**23.33** (+16.67)	**50.60** (+4.82)
128	CoT	16.67	40.96
	+ Rewrite	16.67 (+0.00)	46.99 (+6.02)
	+ PoT	**23.33** (+6.67)	**48.19** (+7.23)
256	CoT	20.00	44.58
	+ Rewrite	20.00 (+0.00)	43.37 (-1.21)
	+ PoT	**23.33** (+3.33)	**49.40** (+4.82)

4.5 Ablation Study

We further perform ablations to isolate the contribution of each design component. The upper block of Table 4 shows that removing any principal design component, including multi-view target construction, target freezing, or adaptive rollout allocation, consistently degrades performance. This indicates that exploration breadth, supervision decoupling, and adaptive search depth provide complementary benefits. Notably, online multi-view supervision remains substantially weaker than its stable counterpart, highlighting the importance of preventing the evolving policy from reshaping its own supervision. The lower block further examines complementary views in multiple-choice settings. Presentation variation provides modest gains over CoT alone, while adding program-based reasoning yields further improvement, supporting the benefit of combining presentation diversity with computationally distinct reasoning pathways.

Table 4: **Ablation study of the core design components and complementary reasoning views.** The upper block reports downstream accuracy after adaptation on open-ended tasks, while the lower block reports fixed-teacher target accuracy measured offline before adaptation on multiple-choice tasks.

Core design components Setting	AIME 25	AMC
Ours (stable MV, adaptive)	**22.50**	**42.70**
stable MV (fixed 32)	13.02	38.55
stable CoT (adaptive)	17.81	39.08
Online MV (CoT+PoT)	9.58	39.16
Online majority (CoT)	6.56	38.20
Complementary views on MCQ Setting	AMC12-MC	JEEBench-MC
CoT	46.68	38.30
CoT + presentation	47.33	39.99
CoT + presentation + PoT	**49.94**	**42.57**

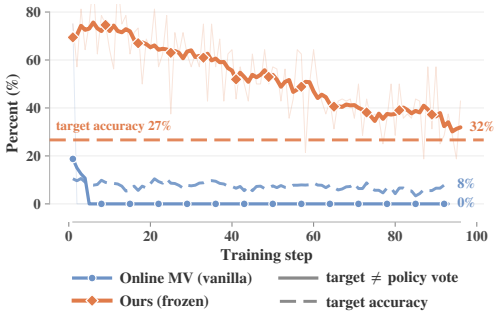

Figure 4: **Supervision dynamics under online and stable multi-view targets.** Online supervision rapidly aligns with the policy plurality, while stable supervision preserves target–policy divergence and maintains a persistent corrective signal throughout adaptation.

In Figure 4, we examine the supervision dynamics underlying self-reinforcing exploration bias to further explain the performance gap between online and stable multi-view supervision observed in Table 4. Supervision–policy divergence is quantified by the fraction of training steps showing disagreement between the supervision target and the policy plurality, with target accuracy reported in parallel. Under online multi-view supervision, target disagreement rapidly collapses toward zero, indicating that the supervision increasingly mirrors the policy's existing preference and provides limited corrective signal beyond the current dominant mode. In contrast, the stable target maintains substantial disagreement throughout adaptation while preserving higher target accuracy. The gradual reduction in divergence under stable supervision reflects the policy progressively absorbing a stationary target, rather than the target drifting toward the evolving policy. These dynamics provide direct evidence that online self-generated supervision becomes increasingly self-confirmatory during adaptation, while target freezing breaks this feedback path and preserves informative supervision over a substantially longer training horizon.

4.6 Computational Efficiency of Structured Exploration

We finally evaluate the computational cost of our framework to assess the efficiency of structured exploration relative to standard TTRL. In Table 5, we separately report policy rollouts, one-off teacher construction cost, total model calls, and generated tokens. Benefiting from targeted rollout allocation, our adaptive variant reduces policy rollouts by 43–54% relative to vanilla TTRL across the three benchmarks. Even after accounting for the one-off multi-view teacher construction, total sampling calls remain 25–44% lower, accompanied by a 22–31% reduction in generated tokens. These reductions demonstrate that targeted exploration improves sampling efficiency by concentrating computation on informative adaptation steps while reducing redundant rollouts. The resulting improvement in both reasoning performance and computational efficiency thus supports structured exploration as a practical test-time adaptation strategy under constrained inference budgets.

Table 5: **Computational cost comparison with vanilla TTRL.** Policy rollouts, one-off teacher construction cost, total sampling calls, and generated tokens are reported across three benchmarks.

Dataset	Method	Policy rollouts	Teacher (one-off)	Total calls	Tokens (B)
AIME25	Vanilla TTRL	161,792	0	161,792	0.1490
	Ours (fixed)	81,920	15,360	97,280	0.0834
	Ours (adaptive)	92,064	15,360	107,424	0.1025
AMC	Vanilla TTRL	184,320	0	184,320	0.1444
	Ours (fixed)	92,160	42,496	134,656	0.0808
	Ours (adaptive)	95,744	42,496	138,240	0.1025
HMMT26	Vanilla TTRL	168,960	0	168,960	0.1125
	Ours (fixed)	84,480	16,896	101,376	0.0605
	Ours (adaptive)	78,528	16,896	95,424	0.0875

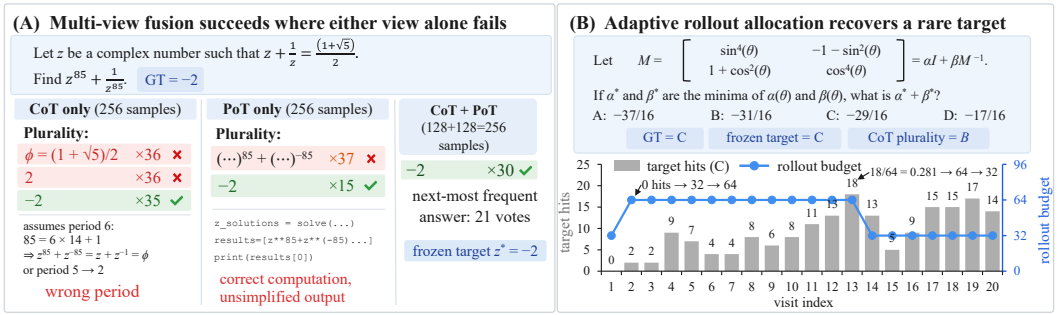

Figure 5: **Representative cases illustrating complementary reasoning views and reachability-aware adaptive rollout allocation. (A):** Multi-view fusion combines distinct CoT and PoT error patterns to recover the correct stable target. **(B):** Adaptive rollout allocation increases search depth for an initially low-reachability target and reduces the budget once target accessibility improves.

5 Case Studies

We further provide representative case studies to illustrate multi-view complementarity and adaptive rollout allocation. Figure 5 (A) shows that CoT and PoT exhibit distinct failure modes, with CoT concentrating on incorrect answers induced by erroneous periodicity assumptions and PoT favoring a mathematically valid but unsimplified output. Although neither view independently selects the correct answer as the plurality, their fusion disperses view-specific errors and reinforces shared support for the correct solution, allowing it to emerge as the stable target. Figure 5 (B) further examines adaptive allocation under target reachability mismatch, where the correct stable target is initially absent from 32 CoT rollouts and therefore triggers an increased rollout budget that is subsequently reduced as target reachability improves. These cases demonstrate that multi-view reasoning expands access to alternative solution modes, while adaptive allocation provides additional search depth for low-probability but learnable targets.

6 Conclusions

We characterize self-reinforcing exploration bias as a distinctive failure mode of TTRL arising from the endogenous dependence of supervision on the model's own exploration distribution. Our analysis identifies the explored solution space as a distinct source of limitation in TTRL, beyond the effects of pseudo-reward noise and optimization instability. We address this limitation through structured control of exploration breadth and depth, combining model-driven multi-view reasoning with instance-adaptive sampling. Across diverse reasoning settings, this design expands access to alternative solution trajectories, sustains search on more demanding instances, and reduces redundant computation. Our findings suggest that exploration in TTRL constitutes an explicit adaptation mechanism rather than an incidental consequence of rollout sampling, providing a principled pathway for improving test-time learning under self-generated supervision.

AI USE STATEMENT

We used generative AI tools, including language-model and coding assistants, to support research concept framing, experimental design, method implementation and debugging, result analysis and interpretation, literature search, figure design, and drafting and editing parts of the manuscript. All AI-assisted methodological choices, implementations, experimental results, citations, and paper content were reviewed and verified by the authors. The authors take responsibility for the final content of this work.

REFERENCES

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=YfZ4ZPt8zd.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025. URL https://arxiv.org/abs/2505.22617.

Bodong Du, Xuanqi Huang, and Xiaomeng Li. Distribution-aware reward estimation for test-time reinforcement learning. *arXiv preprint arXiv:2601.21804*, 2026. URL https://arxiv.org/abs/2601.21804.

Renren Jin, Pengzhi Gao, Yuqi Ren, Zhuowen Han, Tongxuan Zhang, Wuwei Huang, Wei Liu, Jian Luan, and Deyi Xiong. Revisiting entropy in reinforcement learning for large reasoning models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 25300–25322, 2026. URL https://aclanthology.org/2026.findings-acl.1266/.

Ruotong Liao, Nikolai Röhrich, Xiaohan Wang, Yuhui Zhang, Yasaman Samadzadeh, Volker Tresp, and Serena Yeung-Levy. Tool verification for test-time reinforcement learning. *arXiv preprint arXiv:2603.02203*, 2026. URL https://arxiv.org/abs/2603.02203.

Hongxiang Lin, Zhirui Kuai, Erpeng Xue, and Lei Wang. Detecting and mitigating the correct-answer extinction window in test-time reinforcement learning with majority voting. *arXiv preprint arXiv:2605.19444*, 2026. URL https://arxiv.org/abs/2605.19444.

Teng Pan, Yuchen Yan, Zixuan Wang, Ruiqing Zhang, Guiyang Hou, Wenqi Zhang, Weiming Lu, Jun Xiao, and Yongliang Shen. CoVerRL: Breaking the consensus trap in label-free reasoning via generator-verifier co-evolution. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29833–29853, 2026. URL https://aclanthology.org/2026.acl-long.1376/.

Ru Wang, Wei Huang, Qi Cao, Yusuke Iwasawa, Yutaka Matsuo, and Jiaxian Guo. Self-Harmony: Learning to harmonize self-supervision and self-play in test-time reinforcement learning. In *International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=ZzG6oJ5ehI.

Weiqin Wang, Yile Wang, Kehao Chen, and Hui Huang. Beyond majority voting: Towards fine-grained and more reliable reward signal for test-time reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 37251–37265, 2026b. URL https://aclanthology.org/2026.acl-long.1727/.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022. URL https://arxiv.org/abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Junkang Wu, Kexin Huang, Jiancan Wu, An Zhang, Xiang Wang, and Xiangnan He. Quantile advantage estimation: Stabilizing RLVR for LLM reasoning. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=WDP5b3mtFV.

Kunbin Xu, Xingzuo Li, Xuefeng Bai, and Kehai Chen. Hi-TTRL: Regulating consensus with hints for test-time reinforcement learning. *arXiv preprint arXiv:2608.03545*, 2026. URL https://arxiv.org/abs/2608.03545.

Dong Yan, Jian Liang, Yanbo Wang, Shuo Lu, Ran He, and Tieniu Tan. What if consensus lies? selective-complementary reinforcement learning at test time. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28957–28970, 2026. URL https://aclanthology.org/2026.acl-long.1337/.

Yongcan Yu, Lingxiao He, Jian Liang, Kuangpu Guo, Meng Wang, Qianlong Xie, Xingxing Wang, and Ran He. Understanding and mitigating spurious signal amplification in test-time reinforcement learning for math reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 37424–37436, 2026. URL https://aclanthology.org/2026.findings-acl.1865/.

Qingyang Zhang, Xinke Kong, Haitao Wu, Qinghua Hu, Minghao Wu, Baosong Yang, Yu Cheng, Yun Luo, Ganqu Cui, and Changqing Zhang. TEMPO: Scaling test-time training for large reasoning models. *arXiv preprint arXiv:2604.19295*, 2026a. URL https://arxiv.org/abs/2604.19295.

Xiaoyun Zhang, Xiaojian Yuan, Di Huang, Wang You, Chen Hu, Jingqing Ruan, Kejiang Chen, and Xing Hu. Revisiting entropy regularization: Adaptive coefficient unlocks its potential for LLM reinforcement learning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 18005–18020, 2026b. URL https://aclanthology.org/2026.findings-acl.895/.

Chu Zhao, Enneng Yang, Yuting Liu, Jianzhe Zhao, and Guibing Guo. ECHO: Entropy-confidence hybrid optimization for test-time reinforcement learning. *arXiv preprint arXiv:2602.02150*, 2026. URL https://arxiv.org/abs/2602.02150.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019. URL https://arxiv.org/abs/1909.08593.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. TTRL: Test-time reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 131459–131483, 2025. URL https://papers.nips.cc/paper_files/paper/2025/hash/be690ea16f005c174f6c4102a5970e67-Abstract-Conference.html.

A Backbone Models

All experiments use *base* checkpoints, never instruction-tuned ones: test-time RL has to operate on the model a practitioner would actually adapt, and an instruction-tuned checkpoint has already been exposed to supervision we do not control.

Granite-4.1-3B-Base has no chat template, so all of its prompts use the plain completion frame of Appendix C.1. Sampling temperature is 0.6 with top-$p = 1.0$ for both models, in training and in evaluation.

Model	Params	Context	Source
Qwen3-1.7B-Base	1.7 B	32 k	`Qwen/Qwen3-1.7B-Base`
Granite-4.1-3B-Base	3 B	128 k	`ibm-granite/granite-4.1-3b-base`

Table 6: Backbones; each resolves under `https://huggingface.co/`. The two differ in tokenizer, pre-training corpus and chat-template availability (Granite base ships none), so a result that holds on both is not an artefact of one model family.

B Datasets

We evaluate on five competition-mathematics sets: three open-ended (OE) and two multiple-choice (MCQ). All follow the transductive test-time-training protocol, i.e. `train` = `test`: the model is adapted on exactly the unlabelled problems it is scored on, and ground-truth answers are used only to compute the reported metrics, never in training.

Dataset	n	Type	Answer	Hugging Face source
AIME 2025	30	OE	integer 0–999	`yentinglin/aime_2025`
AMC	83	OE	numeric	`AI-MO/aimo-validation-amc`
HMMT Feb 2026	33	OE	exact LaTeX	`MathArena/hmmt_feb_2026`
AMC12-MC	197	MCQ	letter A–E	`edev2000/amc12-full`
JEEBench	53	MCQ	letter A–D	`daman1209arora/jeebench`

Table 7: Datasets. Sizes are after the selection rules described below. Each source resolves under `https://huggingface.co/datasets/`.

Selection and preparation. *AIME 2025* is the 30 problems of AIME 2025 I and II; `tikzpicture` figure blocks are stripped from the three problems that carry them, matching how the source ships figure problems. Answers were cross-checked against two independent releases (`math-ai/aime25`, `opencompass/AIME2025`) and all 30 agree. *AMC* is the 83-problem open-ended set used by prior test-time-RL work. *HMMT Feb 2026* is the `hmmt_feb_2026` subset of MathArena, taken verbatim; it postdates both backbones' pre-training cut-offs. *AMC12-MC* is every AMC 12A/12B problem from 2022–2025, kept in its original five-option multiple-choice form rather than converted to open-ended. The source ships 199 of the 200 problems (2025 12A P25 is missing); we drop the two whose options are figure labels rather than values (2022 12A P3, 2024 12B P4), leaving 197. Twenty-three carry their figure as Asymptote code, and every option was verified to round-trip through the answer matcher. *JEEBench* is the mathematics, single-correct-MCQ subset of JEEBench (53 of 515 items).

C Implementation Details

C.1 Prompt templates

Every view below is *answer-preserving*: it changes how the question is asked or how the answer is computed, never what the answer is.

Chain-of-thought (CoT), open-ended. The question followed by

```
\n Please reason step by step, and put your final answer
within \boxed{}.
```

Chain-of-thought, multiple choice. The question (stem, blank line, then `A.` ... option lines) followed by

```
\n Please reason step by step, and put ONLY the letter of
the correct option (A, B, C, or D) within \boxed{}.
```

AMC12-MC has five options and uses "`the letter of the correct option (A, B, C, D, or E)`".

Program-of-thought (PoT), open-ended. A separate user turn asking for a program rather than a derivation:

```
Solve the following math problem by writing a Python
program.
Requirements:
- Derive the answer with code (enumerate, simulate, or
compute exactly); do not hard-code a guessed answer.
- Use exact arithmetic:  fractions.Fraction or sympy
(Rational, sqrt, pi, symbols, solve, simplify) instead of
floating point whenever the answer is not an integer.
- The LAST line printed must be the final answer alone, in
the form
   \boxed{ANSWER}.  Print integers as integers and exact
values
   symbolically.
Problem:
<question>
```

The assistant turn is prefilled with ```` ```python\n ```` and generation stops at the closing fence. Generated programs are executed in a sandbox with a fixed preamble that imports `math`, `itertools`, `fractions`, `sympy` and related modules, under a wall-clock timeout; the printed $\boxed{\ldots}$ value is the program's answer. A program that raises, times out, or prints nothing parseable casts no vote. On the MCQ sets the options are shown in the program prompt and the printed value is mapped to the option carrying that value.

Two variants of this prompt are used and the choice is per backbone, not per dataset. Qwen3-1.7B uses a *two-shot* form, with two fixed exemplars prepended as dialogue history; they are drawn from the MATH *training* split, chosen for output format only, and never from any evaluation set. Granite has no chat template and no exemplars: it is prompted zero-shot in the completion frame below.

Option rotation (MCQ only). The option list is cyclically shifted and the letters relabelled; the model's chosen letter is mapped back through the inverse permutation before voting. This is a view of the same question, so it tests whether the model's answer depends on option position. We use all four non-identity shifts on the five-option set (AMC12-MC) and all three on the four-option set (JEEBench).

Completion frame (models without a chat template). Granite base has no chat template, so its prompts are rendered as

```
Question:  <question><suffix>\n Answer:  \n\think\n
```

with generation stopped at the next "`Question:`". The model's own `<think>` tag is prefilled after `Answer:` in every view, in pool generation and in training alike; without it the 3B model opens roughly half of its rollouts with a reflexive $\boxed{}$ guess before any reasoning.

C.2 Constructing the stable multi-view target

For each (backbone, dataset) pair, the target bank is constructed once, offline, from the *base* policy before any gradient update, and is then kept fixed throughout adaptation. The views used for each benchmark are listed in Table 8. The construction procedure is as follows:

1. **Sample each view independently.** We sample each view using the same decoding parameters as training ($T = 0.6$, top-$p = 1.0$, up to 3072 new tokens). For open-ended tasks, CoT and PoT each contribute up to 256 samples. For multiple-choice tasks, the natural-language budget is distributed across the original option order and its cyclic rotations, with 64 samples per presentation view, while PoT contributes up to 256 samples.

2. **Map predictions to a common answer space.** CoT responses are parsed into their final answers, generated programs are executed and their printed values are parsed, and predictions from rotated multiple-choice views are mapped back to the original option identities. For open-ended tasks, numerically equivalent answers within a relative tolerance of 10^{-6} are mapped to the same answer key.
3. **Discard invalid outputs.** Samples that do not yield a valid task answer do not vote. This includes unparseable CoT outputs and program generations that fail to execute or do not return a valid answer.
4. **Apply confidence filtering.** Within each view, valid samples are ranked by their mean chosen-token log-probability under the base model, and only the top $\eta = 50\%$ are retained. Filtering is performed separately within each view so that confidence scores from qualitatively different output formats, such as natural-language reasoning and programs, are not compared directly.
5. **Aggregate the retained evidence.** All retained predictions are mapped into the same canonical answer space and pooled across views. The stable target is the plurality answer among the retained candidates,

$$\hat{a}(x) \in \arg\max_{a \in \mathcal{A}(x)} \sum_{v \in \mathcal{V}(x)} \sum_{y \in S_v(x)} \mathbf{1}[\phi_v(y) = a]\,,$$

where $S_v(x)$ denotes the retained samples from view v. The resulting target is stored once and remains unchanged during policy adaptation.

Backbone	Dataset	Views in the stable bank	Samples/prompt
Qwen3-1.7B	AIME 2025, AMC, HMMT Feb 2026	CoT, PoT	512
	AMC12-MC	CoT, 4 rotations, PoT	576
	JEEBench	CoT, 3 rotations, PoT	512
Granite-4.1-3B	AIME 2025, AMC, HMMT Feb 2026	CoT, PoT	512
	AMC12-MC	CoT, 4 rotations, PoT	576
	JEEBench	CoT, 3 rotations, PoT	512

Table 8: Views used to construct each stable target bank and the corresponding number of base-policy samples per problem. Open-ended tasks use 256 CoT and 256 PoT samples. For multiple-choice tasks, 64 samples are drawn from each natural-language presentation view, while the PoT view contributes 256 samples. The target bank is constructed once before adaptation, and this one-off cost is included in the compute accounting in the main text.

C.3 Training configuration

Every arm in the paper — ours and all baselines — runs inside the same trainer with the settings in Table 9; a method changes the reward, the advantage, or the rollout budget, never the optimiser or the data pipeline.

C.4 Adaptive rollout allocation

The per-problem budget n_i changes only between visits, from the observed target hit rate of the group just sampled (Table 11). Two efficiency consequences are worth stating explicitly. First, a group whose rewards are all equal has an exactly zero GRPO advantage, so it is skipped before the actor and reference forward passes rather than contributing a zero gradient; this changes the cost of a step, not the update. Second, the pause rules are what pay for the enlarged groups: without them the budget on an unreachable target would keep doubling indefinitely.

C.5 Baselines

All baselines are implemented within the same training harness and use the same backbone, prompt format, answer extraction, grading procedure, decoding settings, and GRPO configuration wherever their algorithms permit. Method-specific rollout budgets and update rules follow the corresponding baseline design and are summarized in Table 9.

Setting	Value
RL algorithm	GRPO, group-relative advantages from binary rewards
Clip ratio	0.2
Loss aggregation	token-mean
KL	in-loss to the stable reference, low-variance estimator, coeff. 10^{-3} (no KL term in the reward)
Optimiser	AdamW, lr 5×10^{-7}, cosine decay, 3 % warm-up
Prompts per training step	16 (no last-batch drop)
Mini-batch	one prompt's rollouts, i.e. 16 optimiser updates per training step
Rollout engine	vLLM, $T = 0.6$, top-$p = 1.0$, ≤ 3072 new tokens
Prompt window	512 tokens (1024 on the two MCQ sets)
Rollouts per prompt	vanilla and most baselines: 64 voted, 32 trained; SCOPE 64/64; ours (fixed) 32/32; ours (adaptive) 32 initial, up to 256
Validation	sampled at $T = 0.6$, top-$p = 1.0$; 1 sample/prompt every 15 steps for the trajectory read-out, 8 samples/prompt at the last step (the reported metric)

Table 9: Shared training configuration. The reported accuracy is always the 8-sample read-out.

Dataset	Problems	Epochs	Steps	Visits per problem
AIME 2025	30	80	160	80
AMC	83	30	180	30
HMMT Feb 2026	33	55	165	55
JEEBench	53	30	120	30
AMC12-MC	197	30	390	30

Table 10: Adaptation schedule. Steps are $\lceil n/16 \rceil \times$ epochs. Every method on a dataset gets the same schedule. AMC12-MC is reported at 30 epochs; an earlier round of runs used 10, which gave that one dataset 10 visits per problem against 30–80 everywhere else, and those runs are not reported.

Released implementations. For ECHO, SCRL, and SCOPE, we follow the behavior of the authors' released implementations. Our ports preserve the components active in the released code while keeping the surrounding training pipeline fixed. In particular, ECHO uses entropy–confidence advantage shaping and confidence-adaptive clipping, while SCOPE follows the voting and update conventions implemented in its public release.

T3RL. We implement T3RL from the method described in the paper. Relative to vanilla TTRL, T3RL modifies pseudo-label construction through tool-assisted verification of current-policy rollouts, while retaining the same GRPO training framework and binary reward formulation used in our baseline harness.

Optimiser-side anti-collapse methods. QAE, PAR, KL-Cov, AER, and Adaptive-KL retain vanilla TTRL's online majority-vote supervision and modify only the optimization stage. We implement each method following its published formulation and available reference implementation. Because the entropy and KL scales in our training regime differ from those used in the original settings, controller hyperparameters are calibrated to the operating range observed in our runs rather than copied directly from their default configurations. We use $k = 0.4$ for QAE, a target entropy of 0.06 for PAR and AER, a covariance fraction of 0.002 for KL-Cov, and a target KL of 0.01 for Adaptive-KL.

C.6 Compute

Training used 80 GB NVIDIA A100, A800 and H800 GPUs, one GPU per run. A single run is one dataset $\times$ one method: 120–390 optimiser steps (Table 10) at roughly 2.5–6 minutes per step; the runs entering the cost accounting took 7–16 hours each. Wall-clock is the one quantity in the paper that is not hardware-independent, which is why the cost accounting in the main text is reported in sampling calls and generated tokens rather than in hours. The stable multi-view target bank for a dataset is built once, offline, before any training, and is reused by every arm and every hyper-

Parameter	Value
Initial budget n_{init}	32 (the fixed arm's group size)
Maximum budget $n_{\max}$	256
No target match in the group	double the budget, capped at $n_{\max}$
Hit rate ≥ 0.25	halve the budget, floored at n_{init}
All rollouts match, 3 visits running	pause for 10 visits, then one probe group
No match for 3 visits at full evidence	pause for 10 visits, then one probe group
Per-step cap	none ($n_{\max}$ bounds the worst case)

Table 11: Adaptive allocation. "Full evidence" means the visit was at $n_{\max}$, or $n_{\max}$ rollouts have been spent since the last match. A no-match group does not prove the target is unreachable and an all-match group does not certify it is correct, so both rules pause and re-probe rather than deciding.

parameter setting that reads it; its cost is reported separately in the main text rather than amortised silently.

D Additional Results

D.1 Supervision quality before any training

Table 12 measures the two supervision sources directly, on the base policy's own samples, before a single gradient step. The stable multi-view target is more accurate than the online majority on 9 of the 10 backbone $\times$ dataset cells.

The last column is the one that bounds the alternatives: *CoT reachability* is the fraction of problems whose ground truth appears *anywhere* in the CoT sample pool, and it is therefore a ceiling for every method that only ever re-weights CoT votes. Our target reaches that ceiling exactly on two cells and exceeds it on a third — on AMC12-MC with Granite, answers that are not present in the CoT pool at any sample count, so no voting rule over CoT could have produced them.

Backbone	Dataset	Online CoT maj.	stable multi-view	Δ	CoT ceiling
Qwen3-1.7B	AIME 2025	16.67	26.67	+10.00	43.33
	AMC	43.37	51.81	+8.43	90.36
	HMMT Feb 2026	6.06	12.12	+6.06	33.33
	JEEBench	35.85	56.60	+20.75	98.11
	AMC12-MC	46.19	61.93	+15.74	97.97
Granite-4.1-3B	AIME 2025	33.33	43.33	+10.00	43.33
	AMC	79.52	80.72	+1.20	86.75
	HMMT Feb 2026	21.21	30.30	+9.09	30.30
	JEEBench	84.91	83.02	−1.89	90.57
	AMC12-MC	77.16	87.31	+10.15	81.22†

Table 12: Label accuracy (%) of the two supervision sources, measured once on the base policy's pools, before any gradient step. †the target is *above* the CoT ceiling: the extra views supply answers the CoT pool does not contain.

D.2 How much of the gradient points at the truth

Accuracy is read at the end of training; the quantity in Table 13 is averaged over every training step, so it does not move with end-of-training evaluation noise. At each step we score the batch a second time under the ground truth and record the cosine between the advantage the arm's pseudo-reward induces and the advantage the ground-truth reward would have induced on the same rollouts. It is defined identically for every arm — all of them supervise with *some* pseudo-label — and higher means more of the update points at the truth. Ground truth is used only for this diagnostic and never for training.

This is the same measurement the main text reports beside accuracy for the optimiser-side controls: the AIME 2025 and AMC columns below are those numbers, not a second piece of evidence. What this table adds is the rest of the matrix. It is also not independent corroboration of our accuracy — our arm trains on a better target by construction, so agreeing more with the ground-truth advantage is close to restating that. It explains the accuracy gap; it does not confirm it.

Method	AIME 2025	AMC	HMMT Feb 2026	JEEBench
Vanilla TTRL	13.85	40.29	14.51	19.53
ECHO	12.99	37.32	7.84	21.52
SCRL	14.39	34.26	1.98	15.36
SCOPE	11.38	34.30	6.98	19.68
T^3RL	9.85	42.37	5.62	8.49
Vanilla TTRL with an optimiser-side control				
QAE	10.64	38.11	—	—
PAR	18.80	40.22	—	—
KL-Cov	16.70	39.43	—	—
AER	11.55	40.69	—	—
Target-KL	14.50	38.28	—	—
Ours (stable MV, fixed 32)	**45.38**	49.58	24.57	43.21
Ours (full)	40.94	**49.95**	**33.73**	**44.66**
Ours / vanilla	3.28×	1.24×	2.33×	2.29×

Table 13: Mean cosine between the induced advantage and the ground-truth advantage, Qwen3-1.7B, averaged over the 120–180 training steps of each run, $\times 100$ to sit on accuracy's scale (it is a scaled cosine, not a percentage). The five optimiser-side controls were run on the two mathematics sets only. AMC12-MC is omitted: it is reported at 30 epochs throughout the paper, and our run at that schedule does not log this diagnostic, so the column would compare baselines against nothing. The ratio is taken against vanilla rather than against the best baseline, because the baseline pool is not the same size in every column and a larger pool has a larger maximum.

D.3 The feedback loop, counted

For an online-majority method the training target *is* the plurality of the policy's own samples, so the reward asks the policy to produce what it already produces most. Table 14 counts how often the target is *not* the answer the policy would itself have voted for, per prompt visit.

Dataset	Supervision	whole run	first quarter	last quarter
AIME 2025	online majority	0.81	3.26	**0.00**
AMC	online majority	0.23	0.91	**0.00**
AIME 2025	stable multi-view	**52.80**	70.76	**36.79**

Table 14: Percentage of prompt visits on which the training target differs from the policy's own plurality, Qwen3-1.7B. The online arms reach exactly zero: past that point the objective carries no information the policy did not already have, and whatever it converged on — right or wrong — is fixed. The stable target still disagrees on a third of prompts at the end; the decline from 70.8 is the policy absorbing the target, which is what should happen when the target does not move. The count needs the per-prompt rollout dump, which was enabled on the three runs listed; the stable arm on AMC has no such dump, so that row is absent for a logging reason rather than a methodological one.

D.4 Reasoning length

Mean response length over the last eighth of training as a percentage of the first eighth (Qwen3-1.7B): the online-majority arms end their runs producing markedly shorter answers than they started with, and the stable-target arms hold their length — 91–94 % against 69–82 % on AMC, 68–70 % against 50–62 % on HMMT Feb 2026, 75–77 % against 67–73 % on JEEBench. An arm trained with the *ground-truth* reward holds 98 % on AMC, so the ordering tracks target quality. This is also

why our token saving is smaller than our rollout saving: on AIME 2025 our adaptive arm draws 43 % fewer rollouts than vanilla but generates only 31 % fewer tokens, because the rollouts it does draw are still full-length reasoning rather than the shortened answers vanilla converges to.