CONTINUOUS TEST-TIME TRAINING FOR LARGE REASONING MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling test-time computation improves the reasoning performance of large language models (LLMs) without improving the models themselves. Test-time training (TTT) instead updates model parameters on unlabeled test problems. Without ground-truth answers, existing methods often use agreement or confidence as proxy rewards, achieving early gains that can plateau or reverse with further training. We investigate whether TTT can sustain improvement beyond the plateau of reinforcement learning with verifiable rewards (RLVR) on labeled data. We propose Continuous Test-Time Reinforcement Learning (C-TTRL), which uses a learned critic to reward policy updates on unlabeled test problems. At each iteration, verified responses generated by the updated policy on a separate, non-overlapping labeled reference set are used solely to recalibrate the critic. Across OLMo3 and Qwen3 models, `C-TTRL` continues to improve over the evaluated training horizon while representative self-rewarded baselines plateau. On AIME 2024, it raises Avg@16 from 33.0% to 51.1% for OLMo3-7B and from 42.3% to 65.8% for Qwen3-14B. Controlled comparisons support the respective roles of unlabeled test problems and critic recalibration. Code is available at anonymous repository.

1 INTRODUCTION

Scaling computation at test time has become a predominant approach to improving the problem-solving capabilities of large reasoning models (LRMs). Longer reasoning traces, strategic sampling, and verification-guided search allow models to explore and select among candidate solutions before producing an answer (Creswell et al., 2023; Lewkowycz et al., 2022; Lightman et al., 2024; Guan et al., 2025; Chen et al., 2025; Luo et al., 2026; Guo et al., 2025). Yet this computation is temporary: after solving one problem, the model retains no experience and approaches the next problem from the same checkpoint. Test-time training (TTT) instead turns test-time experience into persistent improvement by updating model parameters on problems encountered after deployment without requiring their ground-truth answers (Gandelsman et al., 2022; Goyal et al., 2022; Sun et al., 2020; Wang et al., 2021; Zhang et al., 2025a; Zuo et al., 2025; Wang et al., 2025b).

Learning without ground-truth answers makes reliable reward estimation a central challenge in reasoning TTT. Existing methods often derive proxy rewards from agreement among generated responses or the model's own confidence (Zuo et al., 2025; Zhang et al., 2025b; Zhao et al., 2025). Although these signals can support early improvement, greater agreement or confidence does not necessarily indicate higher correctness. Optimizing these proxies can reinforce answers the model already favors, even when they are incorrect. This feedback can lead to performance plateaus or degradation while concentrating probability mass on fewer outputs (Press et al., 2024; Zhang et al., 2025c; He et al., 2026).

A natural alternative is to train the reward estimator with verified outcomes from a separate labeled reference set, rather than relying entirely on signals produced by the policy itself. This leads to a semi-supervised TTT setting (Yang et al., 2025b): the test problems remain unlabeled, while labeled reference problems are available only for maintaining the critic, not for updating the policy during TTT. However, a critic trained only once may become less accurate as the policy changes and generates different responses. This raises the central question of our work: *After labeled RLVR has plateaued, can TTT continue improving the policy using test problems whose answers remain unavailable?*

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.

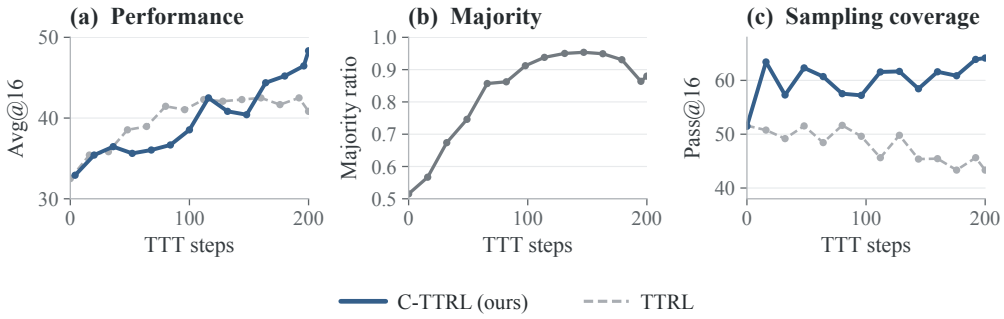


Figure 1: **Training dynamics of TTRL and `C-TTRL` on OLMo3-7B.** (a) On AIME 2024, TTRL's Avg@16 plateaus while `C-TTRL` continues to improve. (b) The majority ratio of TTRL increases during training. (c) Sampling coverage, measured by Pass@16, declines under TTRL but remains higher under `C-TTRL`. Markers denote evaluations at 16-step intervals.

We propose Continuous Test-Time Reinforcement Learning (`C-TTRL`) to study this problem. Unlike prior TTT methods that optimize a base or instruction-tuned model directly on unlabeled test problems (Zhang et al., 2025b; Zuo et al., 2025; Wang et al., 2025b), we first complete RLVR training on labeled data and begin TTT only after RLVR performance has plateaued. During TTT, a critic trained on verified outcomes supplies rewards for PPO updates on the unlabeled test set D_u, whose answers remain hidden. After each policy update, the updated policy generates responses on a separate, non-overlapping labeled reference set D_L, where verified outcomes update the critic before the next iteration. During TTT, D_L updates only the critic and never directly trains the policy. This differs from standard actor-critic PPO, which trains the critic from rewards observed on the same on-policy rollouts used for policy optimization.

Across Qwen3 and OLMo3 models with 7B, 8B, and 14B parameters, `C-TTRL` improves average accuracy beyond the RLVR initialization on both mathematical and general reasoning tasks. For instance, on AIME 2024, it raises Avg@16 from 33.0% to 51.1% for OLMo3-7B and from 42.3% to 65.8% for Qwen3-14B. In our experiments, self-rewarded baselines plateau or degrade after early gains, whereas `C-TTRL` continues to improve over the evaluated training horizon. Continued RLVR under the same schedule yields few gains, and a frozen-critic variant reaches an earlier plateau, supporting the respective roles of unlabeled test problems and critic calibration at every iteration. `C-TTRL` also achieves stronger Pass@k coverage than the self-rewarded baselines across the mathematical benchmarks and improves performance on general reasoning tasks.

Our contributions are summarized as follows:

- We identify reward reliability under policy evolution as a central challenge for continued improvement during TTT: policy-derived rewards can amplify existing predictions, while a fixed evaluator may become stale as the response distribution changes (Section 4.5).
- We propose `C-TTRL`, which separates policy and critic supervision by updating the policy on unlabeled test problems and the critic on verified responses from the latest policy over a labeled reference set. Our analysis gives a sufficient condition for correctness improvement under an idealized critic-guided update and explains why majority-vote rewards need not satisfy it.
- Across two model families, three model sizes, and mathematical and general reasoning tasks, `C-TTRL` continues to improve over the evaluated training horizon after self-rewarded baselines plateau while maintaining stronger Pass@k coverage (Sections 4.2, 4.3, and 4.4). Continued RLVR and frozen-critic controls support the contributions of unlabeled test experience and critic updates (Section 4.5).

2 Related Work

Test-time and label-free reasoning optimization. Test-time adaptation updates model parameters on unlabeled data after deployment to counter distribution shift (Wang et al., 2021; Zhang et al., 2025a). Recent work applies it to reasoning models that keep improving on unlabeled test problems. Since answers remain unavailable, most methods rely on reward signals from majority-voting (Zuo et al., 2025), entropy or high confidence (Zhang et al., 2025b; Zhao et al., 2025), agreement across paraphrased questions (Wang et al., 2025a), or analogous questions with a slowly updated teacher (Zhang et al., 2025d). Others pursue the same goal through program evolution (Wang et al., 2025b) or entropy minimization via unlabeled fine-tuning, reinforcement learning, and inference-time logit adjustment (Agarwal et al., 2025b). The common limitation is that the policy both produces and optimizes against these signals, so self-reinforcement can inflate agreement and confidence without improving correctness, and gains tend to plateau (Zhang et al., 2025c; He et al., 2026); supervision that is not grounded in verified outcomes, even from a stronger external model, may add little beyond output regularization (Ding & Zhang, 2026). Rather than adding another label-free reward, `C-TTRL` grounds rewards in verified outcomes: a separate critic is recalibrated on labeled reference problems after every TTT iteration, while the policy learns only from unlabeled test problems.

Semi-supervised reasoning optimization. RL with verifiable rewards trains reasoning policies using labeled answers or executable verifiers (Shao et al., 2024; Lambert et al., 2024; Guo et al., 2025), while semi-supervised methods combine limited labeled supervision with a larger unlabeled pool. Most closely related, TraPO identifies reliable unlabeled samples by comparing their learning trajectories with those of labeled examples (Yang et al., 2025b). `C-TTRL` shares the premise that limited supervision can stabilize learning from unlabeled reasoning data, but assigns it a different role: TraPO uses labeled trajectories for sample selection, whereas `C-TTRL` uses labeled outcomes to train a response-level critic and updates that critic at every iteration while the policy learns from unlabeled test problems. This also differs from standard actor–critic PPO, which fits the critic to returns observed on the same on-policy rollouts used for the policy update (Schulman et al., 2017); in `C-TTRL`, policy rollouts from D_u have no correctness rewards, so critic supervision comes from separate current-policy rollouts on D_L. Our setting is therefore semi-supervised TTT rather than fully label-free adaptation. The frozen-critic and supervised-continuation controls further distinguish repeated critic updates from merely initializing a reward model or replaying labeled data.

3 Method

We present `C-TTRL`, a test-time training method that alternates policy refinement with critic recalibration. Starting from an RLVR-initialized policy and critic, `C-TTRL` updates the policy on unlabeled adaptation problems and recalibrates the critic using verified responses from the updated policy.

3.1 Problem Setting and RLVR Initialization

Before TTT, we initialize the policy and critic via RLVR on a labeled reference dataset $D_L = \{(x, a)\}$, where each reasoning problem x is paired with its ground-truth answer a. During TTT, we adapt the policy on a test dataset $D_u = \{x\}$ that is disjoint from D_L, without access to its ground-truth answers. Let μ_L and μ_u denote the empirical sampling distributions over D_L and D_u, respectively.

We denote the reasoning model with parameters θ by π_θ and refer to it as the *policy*. Given a problem x, the policy generates a response $y = (y_1, \ldots, y_T)$ autoregressively according to $\pi_\theta(y \mid x)$. We define the binary reward $r(x, y)$ to be 1 if the final answer in y is correct and 0 otherwise. For $(x, a) \in D_L$, an outcome verifier computes this reward by comparing the generated answer with a. On D_u, this reward remains unavailable during TTT.

Our objective is to maximize the expected correctness of the policy on the test distribution:

$$J(\pi) = \mathbb{E}_{x \sim \mu_u,\, y \sim \pi(\cdot|x)}[r(x, y)]. \tag{1}$$

Because correctness rewards are unavailable on D_u during TTT, `C-TTRL` uses a critic V_ϕ to provide surrogate rewards. The critic maps a problem and a response prefix to a scalar score. We write

Table 1: **TTT supervision and performance gains.** Qwen3-14B results on AIME 2024, reproduced from Table 2. All methods use unlabeled adaptation problems from the target domain. Semi-supervised methods additionally reuse the same labeled reference problems from RLVR initialization, disjoint from the adaptation problems. Δ denotes percentage-point gains over Zero-RL, the initial policy π_0 (Avg@16: 42.3%; Pass@8: 69.1%). Pass@8 measures correct-solution coverage.

	Data used during TTT		Performance		
Method	**Adaptation data**	**Reference data**	**Avg@16**	**ΔAvg@16**	**ΔPass@8**
Unsupervised TTT					
TTRL	Unlabeled	Not used	53.1	+10.8	-12.4
EMPO	Unlabeled	Not used	55.6	+13.3	-9.4
Self-Harmony	Unlabeled	Not used	61.0	+18.7	-5.8
Semi-supervised TTT					
Semi-TTRL	Unlabeled	Reused labeled data	55.4	+13.1	-7.8
TraPO	Unlabeled	Reused labeled data	49.2	+6.9	+1.8
`C-TTRL`	Unlabeled	Reused labeled data	**65.8**	**+23.5**	**+4.2**

$y_{\le t} = (y_1, \ldots, y_t)$ for the first t tokens and $y_{<t} = (y_1, \ldots, y_{t-1})$ for the prefix before token y_t, with $y_{<1}$ denoting the empty response. The score $V_\phi(x, y_{\le t})$ predicts the final correctness reward given the prefix. For a complete response, $V_\phi(x, y)$ serves as the surrogate correctness reward for policy updates on D_u.

For RLVR initialization, we jointly train the policy and critic on D_L using proximal policy optimization (PPO): verifier rewards are used to optimize the policy and construct return targets for the critic. We use the final checkpoint from the fixed RLVR schedule reported in Appendix B as (π_0, V_0). During TTT, we write $\pi_k = \pi_{\theta_k}$ and $V_k = V_{\phi_k}$ for the policy and critic at iteration k.

This semi-supervised TTT setting reuses existing reference answers solely for critic updates, without requiring additional annotation. Table 1 compares supervision requirements and performance gains.

3.2 Policy Refinement with Critic Rewards

The policy update in `C-TTRL` uses standard clipped PPO; its non-standard component is the advantage constructed from the critic. At iteration k, we sample $x \sim \mu_u$ and generate $y \sim \pi_k(\cdot \mid x)$. The terminal score $V_k(x, y)$ replaces the unavailable correctness reward. For token y_t, we define its advantage A_t by subtracting the critic score available before that token is sampled:

$$A_t = \mathrm{sg}[V_k(x, y) - V_k(x, y_{<t})], \tag{2}$$

where $\mathrm{sg}[\cdot]$ stops gradients through the critic. Intuitively, A_t measures how much better the completed response scores than the critic predicted before choosing y_t. Because $V_k(x, y_{<t})$ is computed before the current token is sampled, it is an action-independent baseline and does not bias the expected on-policy gradient for the critic reward. We freeze V_k and optimize the standard clipped PPO surrogate with A_t, yielding π_{k+1}. Full optimization details are provided in Appendix B.

The policy update also changes the distribution of generated responses. Consequently, V_k, which was trained on responses from an earlier policy, may provide less reliable rewards for π_{k+1}. Before the next policy update, we therefore adapt the critic using verified responses generated by π_{k+1}.

3.3 Critic Recalibration

Critic recalibration prepares the reward signal for the next TTT iteration. We sample $(x, a) \sim \mu_L$, generate $y \sim \pi_{k+1}(\cdot \mid x)$, and evaluate $r(x, y)$ using the ground-truth answer a. We use $y_{\le 0}$ to denote the empty response. Starting from the current critic parameters ϕ_k, we minimize the recalibration loss

$$\mathcal{L}_V(\phi; \pi_{k+1}) = \mathbb{E}_{(x,a)\sim\mu_L,\, y\sim\pi_{k+1}(\cdot|x)} \left[\frac{1}{T+1} \sum_{t=0}^{T} \left(V_\phi(x, y_{\le t}) - r(x, y) \right)^2 \right], \tag{3}$$

Every prefix is trained to predict the verified outcome of the complete response. Optimizing Eq. (3) yields ϕ_{k+1} and the critic $V_{k+1} = V_{\phi_{k+1}}$ used in the next policy update. Generating the calibration responses with π_{k+1} exposes the critic to the behavior it will evaluate, while the ground-truth answer keeps its target independent of the policy's own confidence. During TTT, D_L is used only for critic updates; it neither reveals answers to D_u nor directly trains the policy.

3.4 Overall Training Procedure

Algorithm 1 summarizes the complete procedure: RLVR initializes π_0 and V_0 on D_L, after which TTT repeatedly updates the policy on D_u and recalibrates the critic on verified responses from the updated policy.

Algorithm 1: Continuous Test-Time Reinforcement Learning (`C-TTRL`)

Input: Labeled dataset D_L and unlabeled test dataset D_u with empirical sampling distributions μ_L and μ_u; initial policy π_θ; critic V_ϕ; TTT iterations N
Output: Adapted policy π_N and final critic V_N

`/* RLVR initialization */`
Train π_θ and V_ϕ on D_L using the fixed RLVR schedule; set $\pi_0 \leftarrow \pi_\theta$, $\phi_0 \leftarrow \phi$, and $V_0 \leftarrow V_{\phi_0}$;
`/* Test-time training */`
for $k = 0$ **to** $N - 1$ **do**
  `/* Refine the policy on unlabeled test problems */`
  Sample $x \sim \mu_u$ and $y \sim \pi_k(\cdot \mid x)$;
  Compute A_t with Eq. (2) and update π_k with PPO to obtain π_{k+1};
  `/* Recalibrate the critic for the updated policy */`
  Sample $(x, a) \sim \mu_L$ and $y \sim \pi_{k+1}(\cdot \mid x)$; evaluate $r(x, y)$ using a;
  Initialize $\phi \leftarrow \phi_k$, optimize Eq. (3) to obtain ϕ_{k+1}, and set $V_{k+1} \leftarrow V_{\phi_{k+1}}$;
end
return π_N, V_N;

3.5 Policy Improvement with Majority-Vote and Critic Rewards

Existing TTT methods such as TTRL (Zuo et al., 2025) use the policy's majority answer as a proxy for the unavailable ground-truth answer. Such a reward reinforces responses that match the current majority, regardless of whether that answer is correct. `C-TTRL` instead scores responses with a critic trained on verified outcomes. We formalize this difference and derive a sufficient condition under which a critic-guided update improves the true test objective J in Eq. (1): correct responses must receive a larger average multiplicative weight than incorrect responses.

We follow the idealized KL-regularized analysis of majority-vote TTT in He et al. (2026). For a fixed test problem x, we instantiate the same update with the critic V_k. Let $\overline{\pi}_{k+1}$ denote this exact update; the bar distinguishes it from the finite clipped-PPO update π_{k+1} used by `C-TTRL`. The idealized update maximizes the expected critic score while penalizing the KL divergence from the current policy π_k. For a coefficient $\beta > 0$ controlling this penalty,

$$\overline{\pi}_{k+1} = \arg\max_{\pi} \mathbb{E}_{y \sim \pi(\cdot|x)}[V_k(x, y)] - \beta D_{\mathrm{KL}}(\pi(\cdot \mid x) \,\|\, \pi_k(\cdot \mid x)), \tag{4}$$

where the maximization is over policies supported by π_k. The first term favors responses with high critic scores, while the second limits the change from the current policy. The solution is

$$\overline{\pi}_{k+1}(y \mid x) = \frac{\pi_k(y \mid x) \exp(V_k(x, y)/\beta)}{Z_k(x)}. \tag{5}$$

Here, $w_k(x, y) = \exp(V_k(x, y)/\beta)$ is the multiplicative weight assigned to response y, and $Z_k(x) = \mathbb{E}_{y \sim \pi_k(\cdot|x)}[w_k(x, y)]$ is a finite normalizing constant. Thus, responses with higher critic scores receive larger relative weights.

This reweighting exposes the limitation of a majority-vote reward. Responses matching the majority answer receive the larger weight even when that answer is wrong. Accordingly, He et al. (2026) show that, when the majority answer remains stable, repeated exact updates concentrate probability on that answer: correctness improves for a correct majority but degrades for an incorrect one. A critic reward

can avoid this failure only if its weights favor correct responses instead. To state this condition, define the current probability of generating a correct response on x as

$$p_k(x) = \Pr_{y \sim \pi_k(\cdot|x)}[r(x, y) = 1]. \tag{6}$$

When $0 < p_k(x) < 1$, define the average weights assigned to correct and incorrect responses as

$$W_k^+(x) = \mathbb{E}_{\pi_k}[w_k(x, y) \mid r(x, y) = 1], \qquad W_k^-(x) = \mathbb{E}_{\pi_k}[w_k(x, y) \mid r(x, y) = 0], \tag{7}$$

assuming both conditional expectations are finite.

Proposition 1 (Sufficient condition for correctness improvement)**.** *Let* $\overline{p}_{k+1}(x) = \Pr_{y \sim \overline{\pi}_{k+1}(\cdot|x)}[r(x, y) = 1]$ *be the correctness probability after the idealized update. Then*

$$\overline{p}_{k+1}(x) - p_k(x) = \frac{p_k(x)(1 - p_k(x))\left(W_k^+(x) - W_k^-(x)\right)}{p_k(x)W_k^+(x) + (1 - p_k(x))W_k^-(x)}. \tag{8}$$

Consequently, $W_k^+(x) \geq W_k^-(x)$ *implies* $\overline{p}_{k+1}(x) \geq p_k(x)$*, with strict improvement when the inequality is strict. If* $p_k(x) \in \{0, 1\}$*, the update leaves it unchanged. If the condition holds at every iteration for* μ_u*-almost every problem with* $0 < p_k(x) < 1$*, repeated idealized updates yield a monotonically non-decreasing value of* J*.*

Proof. By Eq. (5), the normalizer can be written as $Z_k(x) = p_k(x)W_k^+(x) + (1 - p_k(x))W_k^-(x)$. Therefore, $\overline{p}_{k+1}(x) = p_k(x)W_k^+(x)/Z_k(x)$. Subtracting $p_k(x)$ yields Eq. (8). □

Proposition 1 makes the contrast precise. Majority voting rewards agreement with the policy's current answer and therefore need not satisfy $W_k^+(x) > W_k^-(x)$. Training V_k with verified outcomes instead encourages higher scores, and hence larger weights, for correct responses. The critic need not be unbiased or perfectly calibrated: correctness improves whenever $W_k^+(x) > W_k^-(x)$ under the idealized update.

The quantities $W_k^+(x)$ and $W_k^-(x)$ are defined over responses from π_k. As the policy changes, a fixed critic may no longer satisfy $W_k^+(x) \geq W_k^-(x)$. We therefore update the critic on verified responses from the latest policy before the next iteration. Training on D_L does not guarantee that this condition holds on D_u, so we examine related behavior empirically. Figure 11 shows that the critic maintains an AUROC of 0.85–0.95 for distinguishing correct from incorrect responses over the evaluated training horizon. AUROC does not directly test the per-problem condition $W_k^+(x) \geq W_k^-(x)$; it provides related, but not equivalent, aggregate evidence that the critic distinguishes correct from incorrect responses. Figure 3 provides complementary downstream evidence: the frozen-critic variant initially matches `C-TTRL` but reaches an earlier performance plateau.

4 Experiments

4.1 Experimental Setup

Mathematical reasoning. We initialize OLMo3-7B (Team Olmo et al., 2025), Qwen3-8B, and Qwen3-14B (Yang et al., 2025a) policies and critics via PPO on DAPO-Math-17K (Yu et al., 2025). We adapt on AIME 2024, AIME 2025, and Beyond AIME (ByteDance Seed, 2025), and evaluate held-out transfer on AIME 2026 and OlymMath (Sun et al., 2026), reporting Avg@16 and Pass@8.

General reasoning. We initialize OLMo3-7B and Qwen3-8B policies and critics via PPO on Dolci-RL-Zero-General (Team Olmo et al., 2025) and adapt on BigBenchHard (Suzgun et al., 2023), AGI Eval (Zhong et al., 2024), and ZebraLogic (Lin et al., 2025), reporting Avg@1. GPQA-Diamond (Rein et al., 2024) is held out and evaluated using Avg@8 and Pass@8. We use `gpt-oss-120b` (Agarwal et al., 2025a) for answer verification and answer-equivalence judgments; details are provided in Appendix B and the verification prompt in Appendix J.1.

Table 2: **Main results on mathematical reasoning benchmarks.** We report avg@16 accuracy and pass@8 across five benchmarks, along with the absolute improvement Δ of C−TTRL over the Zero-RL baseline. Gray columns denote benchmarks used as TTT data, while blue columns denote held-out benchmarks never seen during training.

	Beyond AIME		AIME 24		AIME 25		AIME 26		OlymMath	
	Avg@16	**Pass@8**	**Avg@16**	**Pass@8**	**Avg@16**	**Pass@8**	**Avg@16**	**Pass@8**	**Avg@16**	**Pass@8**
Oat-Zero-7B	9.4	19.4	30.2	46.1	12.3	33.7	16.7	26.3	11.1	22.6
MiMo-Zero-7B	14.6	33.1	37.7	63.9	32.3	51.9	35.0	52.8	16.7	36.1
Olmo3.1-Zero-7B	13.8	32.1	31.9	56.3	26.5	39.7	24.0	42.4	14.3	42.3
Olmo3-7B-Base										
↪ Zero-RL	17.6	38.8	33.0	56.1	26.3	41.1	26.7	42.8	18.9	43.3
↪ TTRL	21.8	22.3	40.8	45.6	27.1	30.7	22.8	39.2	18.9	33.0
↪ EMPO	21.3	28.4	41.6	43.3	26.7	29.5	23.6	39.7	18.7	32.9
↪ Self-Harmony	13.7	27.2	40.0	42.6	29.4	30.0	20.6	38.1	16.0	29.5
↪ Semi-TTRL	17.8	18.0	33.5	35.0	26.5	26.7	21.9	29.7	17.7	29.6
↪ TraPO	15.9	33.1	29.6	47.5	25.4	41.8	20.8	38.8	18.0	39.5
↪ C−TTRL	24.5	44.0	51.1	61.6	37.0	52.5	30.1	49.4	23.5	51.6
Δ	**+6.9**	**+5.2**	**+18.1**	**+5.5**	**+10.7**	**+11.4**	**+3.4**	**+6.6**	**+4.6**	**+8.3**
Qwen3-8B-Base										
↪ Zero-RL	15.6	33.6	26.3	53.0	25.4	44.8	21.9	43.7	15.0	39.9
↪ TTRL	18.7	20.0	29.0	30.0	32.8	33.3	13.8	25.0	11.4	25.3
↪ EMPO	16.7	23.3	32.3	36.0	33.3	35.4	19.4	33.3	13.1	26.7
↪ Self-Harmony	14.9	33.1	32.9	47.0	30.4	42.3	21.9	44.0	14.9	34.9
↪ Semi-TTRL	18.3	24.0	34.0	43.0	28.1	36.7	19.2	39.0	15.1	28.9
↪ TraPO	15.2	29.7	34.6	50.7	29.2	42.5	22.3	44.3	15.4	32.6
↪ C−TTRL	20.0	36.7	42.7	61.1	40.8	60.4	24.2	50.7	18.7	43.3
Δ	**+4.4**	**+3.1**	**+16.4**	**+8.1**	**+15.4**	**+15.6**	**+2.3**	**+7.0**	**+3.7**	**+3.4**
Qwen3-14B-Base										
↪ Zero-RL	24.9	50.0	42.3	69.1	37.1	59.0	38.1	70.0	24.2	51.6
↪ TTRL	25.5	29.4	53.1	56.7	40.8	45.8	31.7	43.0	18.3	31.7
↪ EMPO	27.6	31.4	55.6	59.7	44.6	46.7	28.3	49.5	17.7	31.9
↪ Self-Harmony	23.2	41.0	61.0	63.3	42.3	48.3	34.8	57.3	20.3	40.7
↪ Semi-TTRL	26.3	32.7	55.4	61.3	38.5	53.1	37.7	54.8	21.0	39.5
↪ TraPO	23.9	41.6	49.2	70.9	36.5	58.2	37.1	56.8	21.5	45.1
↪ C−TTRL	29.3	46.3	65.8	73.3	44.6	60.0	38.8	70.0	25.8	50.2
Δ	**+4.4**	-3.7	**+23.5**	**+4.2**	**+7.5**	**+1.0**	**+0.7**	**0.0**	**+1.6**	-1.4

Baselines and protocol. Zero-RL denotes the RLVR-initialized policy before TTT. We compare with TTRL (Zuo et al., 2025), EMPO (Zhang et al., 2025b), Self-Harmony (Wang et al., 2025a), Semi-TTRL, and TraPO (Yang et al., 2025b) on mathematical reasoning, and TTRL and EMPO on general reasoning. Semi-TTRL augments TTRL with labeled examples. Within each domain, we adapt one policy per model–method pair on the combined unlabeled adaptation benchmarks. All semi-supervised methods use the same labeled reference set and unlabeled adaptation set. Adaptation answers are used only for offline evaluation and baseline checkpoint selection, not for policy or critic updates. For each TTT baseline, we select the checkpoint with the highest mean accuracy across the adaptation benchmarks, whereas C−TTRL is evaluated after 224 TTT updates. For each policy, the same checkpoint is evaluated on all adaptation and held-out benchmarks. Held-out benchmarks are excluded from both training and checkpoint selection. Training and checkpoint-selection details are provided in Appendix B.

4.2 Continued Improvement Beyond the RLVR Plateau (RQ1)

Adaptation performance. Table 2 shows that C−TTRL achieves the highest or tied-highest Avg@16 on all three adaptation benchmarks across all three models. This advantage holds against both unsupervised and semi-supervised TTT baselines, including methods using the same labeled reference data. On AIME 2024 with Qwen3-14B, for example, C−TTRL achieves 65.8%, compared with 61.0% for the strongest baseline, Self-Harmony.

Held-out transfer. The benefits extend beyond the problems used for TTT. On AIME 2026 and OlymMath, which are excluded from training, C−TTRL improves Avg@16 over Zero-RL across all three models and outperforms the compared TTT baselines. These results indicate that adaptation yields transferable improvements rather than gains confined to the adaptation benchmarks.

Sustained improvement during training. Figure 1(a) and the extended trajectories in Appendices E.2 and E.3 show that C−TTRL maintains an upward trend, while several self-rewarded baselines plateau or degrade after early gains. Together with the PPO initialization curves in Appendix E.1 and the supervised-continuation control in Section 4.5, these results support sustained improvement beyond the observed RLVR plateau.

4.3 Sampling Coverage (RQ2)

Following Yue et al. (2025), we use Pass@k to measure the coverage of problems solved within k sampled responses. Table 2 shows that C−TTRL achieves higher Pass@8 than all evaluated TTT baselines across the three models and five benchmarks. Importantly, improvements in average accuracy do not necessarily imply broader correct-solution coverage: on AIME 2024 with Qwen3-14B, TTRL and EMPO improve Avg@16 over Zero-RL but reduce Pass@8, whereas C−TTRL improves both, raising Pass@8 from 69.1% to 73.3%. Figure 1(c) complements these checkpoint comparisons by showing that C−TTRL maintains higher Pass@16 than TTRL during OLMo3-7B training. Additional coverage analyses are provided in Appendix H.

Table 3: **Results on reasoning tasks beyond math-only domains.** We report Avg@1 for BigBench-Hard, AGI Eval, and ZebraLogic, alongside Avg@8 and Pass@8 over 8 independent samples for GPQA-Diamond. We also highlight the absolute Δ of C−TTRL over the Zero-RL baseline across diverse domains. Gray columns denote benchmarks used as TTT data, while blue columns denote held-out benchmarks never seen during training.

	BBH	**AGI**	**Zebra**	**GPQA-Diamond**	
	(Avg@1)	(Avg@1)	(Avg@1)	(Avg@8)	(Pass@8)
Olmo-3-7B-RL-Zero-General	56.5	51.9	25.7	28.9	69.0
MiMo-Zero-RL-7B	61.4	53.6	30.3	18.8	45.8
General-Reasoner-7B	65.6	63.6	25.9	35.1	68.6
General-Reasoner-14B	78.2	73.4	44.5	44.4	70.3
OLMO3-7B-Base					
↪ Zero-RL	46.8	37.9	22.2	21.9	62.1
↪ TTRL	45.4	38.2	22.2	28.5	67.6
↪ EMPO	52.9	50.2	23.5	27.7	61.6
↪ C−TTRL	68.2	62.4	35.1	32.4	69.4
Δ	**+21.4**	**+24.5**	**+12.9**	**+10.5**	**+7.3**
Qwen3-8B-Base					
↪ Zero-RL	69.9	65.7	25.7	32.2	62.4
↪ TTRL	74.9	68.5	31.7	41.1	73.0
↪ EMPO	66.7	65.1	26.3	39.8	70.8
↪ C−TTRL	74.2	70.1	33.9	37.2	65.3
Δ	**+4.3**	**+4.4**	**+8.2**	**+5.0**	**+2.9**

4.4 Versatility (RQ3)

C−TTRL extends to general reasoning beyond mathematics. As shown in Table 3, C−TTRL improves over the RLVR-initialized policy on all four benchmarks for both OLMo3-7B and Qwen3-8B. On OLMo3-7B, it outperforms TTRL and EMPO across all reported metrics, with gains over Zero-RL of 21.4 percentage points on BBH, 24.5 on AGI Eval, and 12.9 on ZebraLogic. On Qwen3-8B, C−TTRL achieves the highest accuracy among the evaluated TTT methods on AGI Eval and ZebraLogic. It also improves held-out GPQA-Diamond Avg@8 by 5.0 points over Zero-RL, although TTRL and EMPO achieve higher scores on this benchmark. Together, these results show that C−TTRL can improve both adaptation performance and held-out transfer beyond mathematical reasoning.

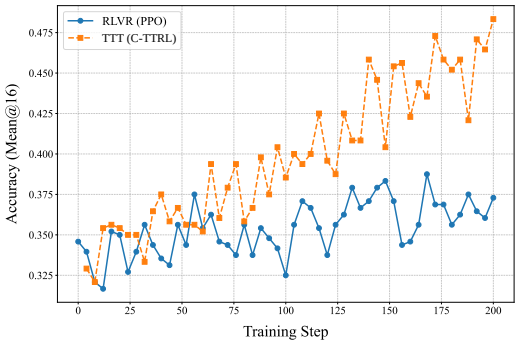


Figure 2: **Test-time training outperforms supervised continuation.** AIME 2024 Mean@16 accuracy on OLMo3-7B. Starting from the same converged RLVR checkpoint, continued PPO yields much smaller gains than C−TTRL, which improves by over 15 percentage points.

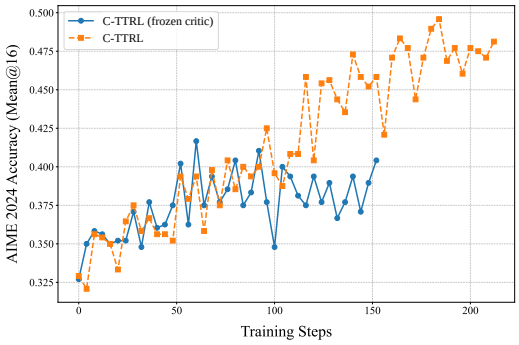


Figure 3: **Critic recalibration sustains improvement.** AIME 2024 Mean@16 accuracy on OLMo3-7B. A frozen critic plateaus after approximately 100 iterations, while the full method continues to improve.

4.5 Ablation (RQ4)

We conduct two ablation studies to isolate the contributions of key design choices in C−TTRL: (1) *Frozen critic*: the critic is trained once on D_L and kept fixed throughout all policy updates, removing per-iteration critic recalibration; (2) *Supervised continuation*: the model continues training on the labeled dataset D_L using standard PPO without any test-time updates on unlabeled data.

As shown in Figure 2, supervised PPO from the selected RLVR checkpoint yields much smaller gains, while C−TTRL improves by more than 15 points over 200 iterations. This comparison indicates that novel test-time problems provide useful learning signal beyond continued training on the same labeled data. Figure 3 shows that a frozen-critic variant initially matches C−TTRL but plateaus after $\sim$100 iterations, while the full method continues to improve. This supports the practical value of updating the critic as the policy evolves. We further investigate sensitivity to the size of the labeled reference set in Appendix G, showing that as few as 1024 labeled samples maintain strong critic quality over the evaluated training horizon.

5 Limitations

C−TTRL has two main limitations. First, maintaining both a policy and a critic increases GPU memory and computation relative to single-model TTT methods such as TTRL. Our timing analysis (Appendix C) shows that critic recalibration accounts for 13.2% of per-step compute, while the additional critic model raises total per-step cost by approximately 1.3–1.6$\times$ over TTRL. Second, critic recalibration requires a labeled reference set D_L. Its size and coverage may limit how well the critic transfers to out-of-domain test problems.

6 Conclusion

We present C−TTRL, a critic-guided framework for continuous test-time training for reasoning models. C−TTRL alternates policy updates on unlabeled test problems with critic recalibration on verified responses from the latest policy over a separate labeled reference set. Our analysis explains why majority-vote rewards can amplify an incorrect answer and gives a sufficient condition under which an idealized critic-guided update improves correctness. Across model scales and reasoning domains, C−TTRL continues to improve over the evaluated training horizon after self-rewarded baselines plateau, while continued-RLVR and frozen-critic controls support the roles of unlabeled test experience and critic updates. Future work will extend C−TTRL to settings with more efficient supervision and to multi-turn agentic tasks.

AI USE STATEMENT

We used generative AI tools to assist with literature retrieval, flag possible issues with wording and clarity in the manuscript for the authors to assess, and assist with LaTeX formatting. Generative AI was not used to formulate the research questions or hypotheses, design the method or experiments, implement the method, develop theoretical derivations, process data, analyze results, or draw conclusions. All AI-provided questions and suggestions were assessed by the authors, who wrote and revised the manuscript and take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Section 3 and Algorithm 1 describe the method and training procedure. Section 4.1 specifies the datasets, baselines, and evaluation protocol. Appendix B provides training hyperparameters, data and rollout allocation, and checkpoint-selection details; Appendix J gives the prompt format and answer-verification template. Code is available in the anonymous repository.

REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025a.

Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. The unreasonable effectiveness of entropy minimization in LLM reasoning. *arXiv preprint arXiv:2505.15134*, 2025b. URL https://arxiv.org/abs/2505.15134.

ByteDance Seed. Beyondaime: Advancing math reasoning evaluation beyond high school olympiads. https://huggingface.co/datasets/ByteDance-Seed/BeyondAIME, 2025.

Jiacheng Chen, Qianjia Cheng, Fangchen Yu, Haiyuan Wan, Yuchen Zhang, Shenghe Zheng, Junchi Yao, Qingyang Zhang, Haonan He, Yun Luo, et al. P1: Mastering physics olympiads with reinforcement learning. *arXiv preprint arXiv:2511.13612*, 2025.

Antonia Creswell, Murray Shanahan, and Irina Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3Pf3Wg6o-A4.

Yi Ding and Ruqi Zhang. Does on-policy distillation really distill? from noisy teacher to self-improvement. *arXiv preprint arXiv:2608.31046*, 2026.

Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei A. Efros. Test-time training with masked autoencoders. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-2130. URL https://papers.nips.cc/paper_files/paper/2022/hash/bcdec1c2d60f94a93b6e36f937aa0530-Abstract-Conference.html.

Sachin Goyal, Mingjie Sun, Aditi Raghunathan, and J. Zico Kolter. Test time adaptation via conjugate pseudo-labels. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-0449. URL https://papers.nips.cc/paper_files/paper/2022/hash/28e9eff897f98372409b40ae1ed3ea4c-Abstract-Conference.html.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. In *International Conference on Machine Learning*, 2025.

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.

Bingxiang He, Yuxin Zuo, Zeyuan Liu, Shangziqi Zhao, Zixuan Fu, Junlin Yang, Cheng Qian, Kaiyan Zhang, Yuchen Fan, Ganqu Cui, Xiusi Chen, Youbang Sun, Xingtai Lv, Xuekai Zhu, Li Sheng, Ran Li, Huan ang Gao, Yuchen Zhang, Lifan Yuan, Bowen Zhou, Zhiyuan Liu, and Ning Ding. How far can unsupervised RLVR scale LLM training? In *The Fourteenth International*

Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=VesLZukY5E.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024. URL https://arxiv.org/abs/2411.15124v1.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-0278. URL https://papers.nips.cc/paper_files/paper/2022/hash/18abbeef8cfe9203fdf9053c9c4fe191-Abstract-Conference.html.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. Zebralogic: On the scaling limits of llms for logical reasoning. In *International Conference on Machine Learning*, pp. 37889–37905. PMLR, 2025.

Yun Luo, Futing Wang, Qianjia Cheng, Fangchen Yu, Haodi Lei, Jianhao Yan, Chenxi Li, Jiacheng Chen, Yufeng Zhao, Haiyuan Wan, et al. P1-vl: Bridging visual perception and scientific reasoning in physics olympiads. *arXiv preprint arXiv:2602.09443*, 2026.

Ori Press, Ravid Shwartz-Ziv, Yann Lecun, and Matthias Bethge. The entropy enigma: Success and failure of entropy minimization. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 41064–41085. PMLR, 2024. URL https://proceedings.mlr.press/v235/press24a.html.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

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, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yongyi Su, Yushu Li, Nanqing Liu, Kui Jia, Xulei Yang, Chuan-Sheng Foo, and Xun Xu. On the adversarial risk of test time adaptation: An investigation into realistic test-time data poisoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=7893vsQenk.

Haoxiang Sun, Yingqian Min, Zhipeng Chen, Wayne Xin Zhao, and Ji-Rong Wen. Challenging the boundaries of reasoning: An olympiad-level math benchmark for large language models, 2026. URL https://arxiv.org/abs/2503.21380.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9229–9248. PMLR, 2020. URL https://proceedings.mlr.press/v119/sun20b.html.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

Team Olmo, Allyson Ettinger, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025. URL https://arxiv.org/abs/2512.13961.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=uXl3bZLkr3c.

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. *arXiv preprint arXiv:2511.01191*, 2025a. URL https://arxiv.org/abs/2511.01191.

Yiping Wang, Shao-Rong Su, Zhiyuan Zeng, Eva Xu, Liliang Ren, Xinyu Yang, Zeyi Huang, Xuehai He, Luyao Ma, Baolin Peng, Hao Cheng, Pengcheng He, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. Thetaevolve: Test-time learning on open problems. *arXiv preprint 2511.23473*, 2025b.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a. URL https://arxiv.org/abs/2505.09388.

Shenzhi Yang, Guangcheng Zhu, Xing Zheng, Yingfan Ma, Zhongqi Chen, Bowen Song, Weiqiang Wang, Junbo Zhao, Gang Chen, and Haobo Wang. TraPO: A semi-supervised reinforcement learning framework for boosting LLM reasoning. *arXiv preprint arXiv:2512.13106*, 2025b. URL https://arxiv.org/abs/2512.13106.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Qingyang Zhang, Yatao Bian, Xinke Kong, Peilin Zhao, and Changqing Zhang. Come: Test-time adaption by conservatively minimizing entropy. In *International Conference on Learning Representations*, 2025a.

Qingyang Zhang, Haitao Wu, Changqing Zhang, Peilin Zhao, and Yatao Bian. Right question is already half the answer: Fully unsupervised llm reasoning incentivization. *Advances in neural information processing systems*, 2025b.

Yanzhi Zhang, Zhaoxi Zhang, Haoxiang Guan, Yilin Cheng, Yitong Duan, Chen Wang, Yue Wang, Shuxin Zheng, and Jiyan He. No free lunch: Rethinking internal feedback for llm reasoning, 2025c. URL https://arxiv.org/abs/2506.17219.

Zizhuo Zhang, Jianing Zhu, Xinmu Ge, Zihua Zhao, Zhanke Zhou, Xuan Li, Xiao Feng, Jiangchao Yao, and Bo Han. Co-rewarding: Stable self-supervised RL for eliciting reasoning in large language models. *arXiv preprint arXiv:2508.00410*, 2025d. URL https://arxiv.org/abs/2508.00410.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. *arXiv preprint arXiv:2505.19590*, 2025. URL https://arxiv.org/abs/2505.19590.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models. In *Findings of the association for computational linguistics: NAACL 2024*, pp. 2299–2314, 2024.

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. *arXiv preprint arXiv:2504.16084*, 2025. URL https://arxiv.org/abs/2504.16084.

APPENDICES

A Broader Impacts **15**

B Training Details and Hyperparameters **15**

B.1 RLVR Initialization 15

B.2 Test-Time Training 15

B.3 Evaluated Checkpoints 16

B.4 Infrastructure 17

C Computational Cost **17**

D Cross-Domain Generalization to Code Generation **18**

E Extended Training-Horizon Analysis **19**

E.1 Labeled-Data PPO Initialization 19

E.2 Extended Training of C-TTRL 19

E.3 Additional Baseline Training Trajectories 21

F Training Dynamics **21**

F.1 Actor Entropy 21

F.2 Critic Discrimination Quality 21

F.3 TTRL Majority Ratio Dynamics 21

F.4 Response Length Dynamics 21

G Ablation on Labeled Data Size **25**

H Extended Sampling-Coverage Analysis **26**

H.1 Pass@16 Training Trajectories of C-TTRL and TTRL 26

H.2 Additional Baseline Pass@8 Trajectories 26

H.3 Pass@k at Different k 26

I Case Study **28**

I.1 Example 1: Probability (Beta-Binomial Integration) 28

I.2 Example 2: Logarithmic System of Equations 28

J Prompt Format **30**

J.1 Answer-Judging Prompt 30

A Broader Impacts

Test-time training methods that enable models to self-improve on user-provided data raise concerns about potential misuse. A system capable of adapting to arbitrary unlabeled inputs could be fine-tuned, intentionally or inadvertently, on harmful or misleading content, potentially reinforcing incorrect or biased reasoning patterns Su et al. (2025). While the fixed labeled reference set D_L used by C-TTRL provides a partial safeguard, the critic's reward signal remains anchored to a curated reference distribution. This does not eliminate the risk entirely if D_L itself is poorly curated.

B Training Details and Hyperparameters

B.1 RLVR Initialization

Table 4 summarizes the hyperparameters used to initialize the actor and critic via PPO on labeled data. The actor and critic backbones start from the same base-model checkpoint. The resulting checkpoints are used to initialize TTT.

Table 4: Hyperparameters for RLVR initialization.

Hyperparameter	OLMo3-7B	Qwen3-8B	Qwen3-14B
Labeled dataset	DAPO-Math-17K	DAPO-Math-17K	DAPO-Math-17K
Batch size	256	256	256
Actor learning rate	1×10^{-6}	1×10^{-6}	1×10^{-6}
Critic learning rate	5×10^{-6}	5×10^{-6}	5×10^{-6}
Advantage estimator	GAE	GAE	GAE
KL penalty coefficient	0	0	0
Rollout samples per prompt	8	8	8
Sampling temperature	1.0	1.0	1.0
Max response length	16,384	16,384	16,384
Dual-clip ratio	3×10^{-4}, 5×10^{-4}	3×10^{-4}, 5×10^{-4}	3×10^{-4}, 5×10^{-4}
Training steps	192	224	224
GPUs	4	8	8

B.2 Test-Time Training

Table 5 lists the hyperparameters specific to C-TTRL. Most settings are inherited from RLVR initialization; we highlight the differences.

Table 5: Hyperparameters for C-TTRL test-time training.

Hyperparameter	Value
Actor learning rate	1×10^{-6}
Critic learning rate	1×10^{-6}
Critic recalibration frequency	every iteration
Rollout samples per prompt	8
Sampling temperature	1.0
Max response length	16,384
KL penalty coefficient	0
Batch size	256
TTT checkpoint for mathematical main results	224
Evaluation frequency	every 4 steps
GPUs	$8\times$

Differences from RLVR initialization. The critic learning rate is reduced from 5×10^{-6} to 1×10^{-6} during TTT, as the critic only needs incremental recalibration rather than training from

scratch. The actor update uses the critic-guided token-level advantages described in Section 3.2 instead of GAE.

Baseline-specific settings. We follow the corresponding original papers and implementations for method-specific hyperparameters of TTRL, EMPO, Self-Harmony, and TraPO (Zuo et al., 2025; Zhang et al., 2025b; Wang et al., 2025a; Yang et al., 2025b). Semi-TTRL denotes a semi-supervised variant of TTRL that incorporates labeled examples alongside unlabeled problems.

Training data and rollout allocation. Within each experimental setting, all semi-supervised TTT methods use the same labeled reference set and unlabeled adaptation set. The labeled reference data are drawn from DAPO-Math-17K for mathematical reasoning and Dolci-RL-Zero-General for general reasoning, providing a common data basis for comparison. At each `C-TTRL` iteration, we sample equal numbers of labeled and unlabeled prompts and generate the same number of responses per prompt in both groups. Labeled responses are used for critic recalibration, while unlabeled responses are used for policy refinement.

General-reasoning verification. We use `gpt-oss-120b` both to verify responses against ground-truth answers on labeled training and evaluation data, and to assess answer equivalence among model-generated responses on unlabeled problems for baselines such as TTRL and EMPO. The latter supports answer grouping and agreement-based statistics without access to ground-truth answers. The exact answer-verification prompt used in our experiments is reproduced in Appendix J.1; the judge is instructed to check answer equivalence rather than independently solve the problem.

B.3 Evaluated Checkpoints

Existing self-rewarding TTT baselines often exhibit a common limitation in our experiments: performance initially improves, then plateaus, and subsequently degrades, sometimes collapsing under extended training (Section 4.2; Appendix E.3). To compare against the baselines at their strongest observed performance, we select each baseline's best-performing checkpoint from its plateau or pre-collapse phase, rather than a degraded final checkpoint. Selection is based on mean accuracy over the combined adaptation tasks, without using held-out benchmark scores. For each method and model, the same selected checkpoint is evaluated across all benchmarks.

For the mathematical-reasoning results in Table 2, `C-TTRL` uses the checkpoint after 224 TTT updates for all three models: OLMo3-7B, Qwen3-8B, and Qwen3-14B. This step count excludes RLVR initialization and denotes the evaluated checkpoint, not the endpoint of the extended training trajectories in Appendix E.2.

Table 6 summarizes the TTT checkpoints used for the comparisons in Table 2. Step indices count TTT updates only, excluding RLVR initialization, and indicate the selected checkpoint rather than the total training horizon. Complementary evaluation curves in Figure 9 show how baseline performance evolves over training.

Table 6: **Selected TTT checkpoints for mathematical reasoning.** Entries indicate the TTT step of the checkpoint evaluated in Table 2, excluding RLVR initialization.

Method	OLMo3-7B	Qwen3-8B	Qwen3-14B
TTRL	176	256	164
EMPO	128	104	184
Semi-TTRL	240	128	112
Self-Harmony	96	32	80
TraPO	144	256	160
`C-TTRL`	224	224	224

B.4 INFRASTRUCTURE

All experiments are conducted on 8× NVIDIA H200 GPUs (141GB HBM3e each, Hopper architecture) with CUDA 12.8. We use the verl framework with SGLang for rollout generation. OLMO3-7B RLVR initialization uses 4 GPUs; all other experiments use 8 GPUs.

C COMPUTATIONAL COST

Table 7 compares the per-step wall-clock time of C−TTRL against baselines during test-time training. All TTT runs use 8 GPUs. Because C−TTRL and baselines differ in total training steps, we report per-step time as the primary comparison rather than total wall-clock.

Table 7: Per-step wall-clock time during test-time training on 8 GPUs.

Model	Method	Per-step Time (seconds)
OLMO3-7B	TTRL	1,017
OLMO3-7B	C−TTRL	1,579
Qwen3-14B	TTRL	1,390
Qwen3-14B	C−TTRL	1,832

Overhead analysis. C−TTRL introduces approximately 1.3–1.6× per-step overhead over TTRL. This comes from two sources: (i) critic recalibration, which performs critic forward and backward passes on D_L (201s/step on OLMO3-7B, 13.2% of total step time as shown in Figure 4), and (ii) critic forward passes for advantage computation during policy refinement. The dominant cost—rollout generation—is shared across all methods. The RLVR initialization cost is amortized because it is incurred once and shared by all TTT methods that start from the same checkpoint.

RLVR initialization cost. For completeness, RLVR initialization takes approximately 170 wall-clock hours for OLMO3-7B (4 GPUs) and 116 wall-clock hours for Qwen3-14B (8 GPUs). This cost is identical across all methods since they share the same initialization checkpoint.

Timing breakdown. Figure 4 shows the per-step time breakdown for C−TTRL on OLMO3-7B. Rollout generation dominates at 1020s/step, while critic recalibration takes 201s/step, accounting for 13.2% of total step time. The remaining time is spent on policy updates and system overhead.

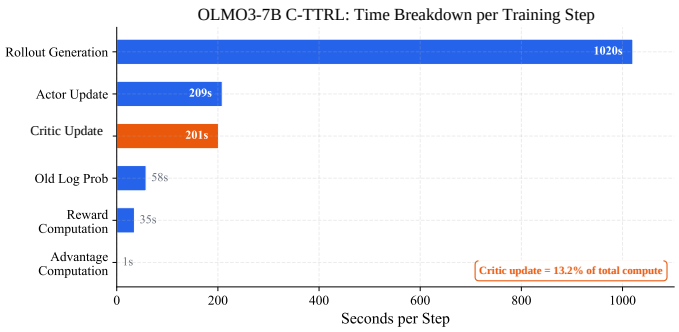


Figure 4: Per-step time breakdown for C−TTRL (OLMO3-7B). Critic recalibration (highlighted in orange) accounts for 13.2% of the total step time. Rollout generation is the dominant cost, shared with all TTT methods.

Peak GPU memory. Table 8 reports peak GPU memory allocation (summed across all GPUs). On OLMO3-7B, C−TTRL uses approximately 37% more memory than TTRL due to the additional critic model. On Qwen3-14B, memory usage is comparable as both methods are dominated by the larger model size.

Table 8: Peak GPU memory allocation (total across 8× H200 GPUs).

Model	TTRL (GB)	C−TTRL (GB)
OLMO3-7B	131.0	179.4
Qwen3-14B	159.5	159.0

D Cross-Domain Generalization to Code Generation

We evaluate the checkpoints trained on mathematical reasoning directly on HumanEval+, MBPP+, and LiveCodeBench, without any additional training or adaptation on code. As shown in Table 9, C−TTRL exhibits the clearest transfer on OLMo3-7B, improving all three benchmarks and raising their average from 43.9 to 48.1 over the corresponding Zero-RL checkpoint. Transfer on the stronger Qwen3 models is mixed at the individual-benchmark level, but the average increases from 55.6 to 56.3 on Qwen3-8B and from 59.9 to 60.6 on Qwen3-14B. Because no code-domain training is performed, these results provide evidence only of cross-domain generalization and should not be interpreted as demonstrating the effectiveness of test-time training on code tasks.

Table 9: **Cross-domain generalization to code generation.** Single-sample pass@1 of math-trained checkpoints evaluated without code-domain training or adaptation.

Method	**HumanEval+**	**MBPP+**	**LiveCodeBench**	**Avg.**
Oat-Zero-7B	55.5	51.1	2.3	36.3
MiMo-Zero-7B	**84.8**	50.8	**34.3**	**56.6**
Olmo3.1-Zero-7B	67.7	**61.9**	26.3	52.0
Olmo3-7B-Base	47.0	43.7	9.1	33.3
↪ Zero-RL	59.8	53.7	18.3	43.9
↪ C−TTRL	**62.8**	**60.8**	**20.6**	**48.1**
Qwen3-8B-Base	48.8	40.5	12.6	34.0
↪ Zero-RL	79.3	**66.9**	20.6	55.6
↪ C−TTRL	**81.1**	66.7	**21.1**	**56.3**
Qwen3-14B-Base	48.8	51.3	12.0	37.4
↪ Zero-RL	**84.1**	68.3	**27.4**	59.9
↪ C−TTRL	82.3	**72.2**	**27.4**	**60.6**

Evaluation details. HumanEval+ and MBPP+ use the additional EvalPlus tests; LiveCodeBench uses the `code_generation_lite` v6 split. Sampling uses $T = 1.0$, $p = 1.0$, and no top-k truncation. Oat-Zero-7B uses its native 4K context limit; all other models use up to 16K. Bold denotes the best result within each block.

E Extended Training-Horizon Analysis

This section examines mean accuracy over the recorded training horizons. We first inspect the labeled-data PPO initialization stage, then compare the extended-horizon behavior of C−TTRL and TTRL and report additional baseline trajectories. Complementary response-length and sampling-coverage analyses are collected in Appendices F.4 and H, respectively.

E.1 Labeled-Data PPO Initialization

We examine whether PPO performance has approached saturation before TTT, rather than using an initialization that still benefits substantially from continued labeled-data training.

Mathematical reasoning. Figures 5 and 6 show PPO training on DAPO-Math-17K with online AIME 2024 evaluation. Qwen3-8B improves rapidly at first, then fluctuates around 25–31% over the later checkpoints. Continuing PPO from step 192 through step 256 yields no sustained improvement; the step-224 checkpoint used to initialize TTT lies within this plateau region.

Qwen3-14B provides complementary evidence at a larger model scale. Its mean online accuracy is 44.26% across checkpoints at steps 196–224 and 44.28% at steps 228–268, showing little change in average performance after the step-224 initialization checkpoint. Together, these trajectories support an empirical PPO plateau for the two mathematical-reasoning models within the observed training horizons. Both curves report Avg@32, rather than the independent Avg@16 evaluations in Table 2.

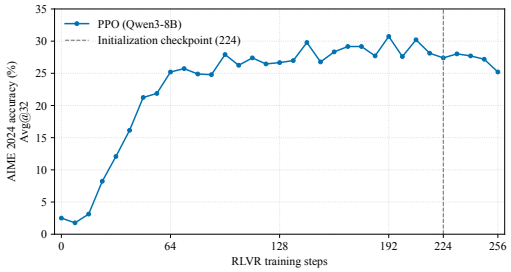


Figure 5: **Qwen3-8B PPO initialization.** Unsmoothed AIME 2024 Avg@32 accuracy over RLVR steps 0–256, evaluated every eight steps. The dashed line marks the step-224 checkpoint used to initialize TTT.

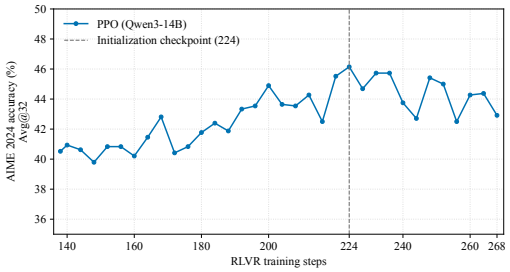


Figure 6: **Qwen3-14B PPO initialization.** Unsmoothed AIME 2024 Avg@32 accuracy over RLVR steps 138–268, showing the recorded late-stage trajectory. The dashed line marks the step-224 checkpoint used to initialize TTT.

General reasoning. Figure 7 examines saturation during PPO initialization on labeled general-reasoning data. On a 1,000-problem MMLU-Pro subset, OLMo3-7B improves rapidly from 22.7% before entering a late-stage regime with diminishing gains, where accuracy fluctuates between 38.9% and 42.1% over steps 316–360. This pattern is consistent with PPO performance approaching saturation within the observed horizon.

Implications for TTT. The mathematical-reasoning trajectories place the selected initializations within an empirical PPO plateau. Complementing these initialization curves, Figure 2 directly compares continued PPO with C−TTRL from the same converged OLMo3 checkpoint. The substantially larger gains achieved by C−TTRL support the benefit of the TTT procedure beyond simply extending labeled-data PPO training.

E.2 Extended Training of C-TTRL

Figure 8 extends the main-paper comparison in Figure 1 to OLMo3-7B and Qwen3-14B on both AIME 2024 and AIME 2025. Despite checkpoint-to-checkpoint fluctuations, C−TTRL exhibits an upward trend across all four settings. TTRL instead reaches a plateau or declines later in training.

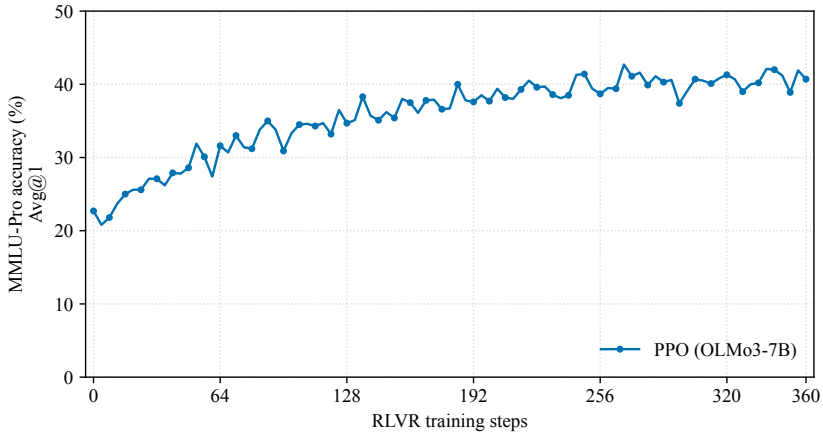


Figure 7: **OLMo3-7B PPO initialization on labeled general-reasoning data.** Unsmoothed MMLU-Pro subset Avg@1 accuracy, evaluated every four training steps from 0 to 360 with one sampled response per problem.

On Qwen3-14B, `C-TTRL` maintains an upward trend over the observed training horizon on both AIME 2024 and AIME 2025, whereas TTRL improves initially but subsequently degrades. These trajectories highlight the ability of `C-TTRL` to sustain improvement during extended test-time training. The extended trajectories include checkpoints beyond step 224, the TTT checkpoint used for the `C-TTRL` results in Table 2.

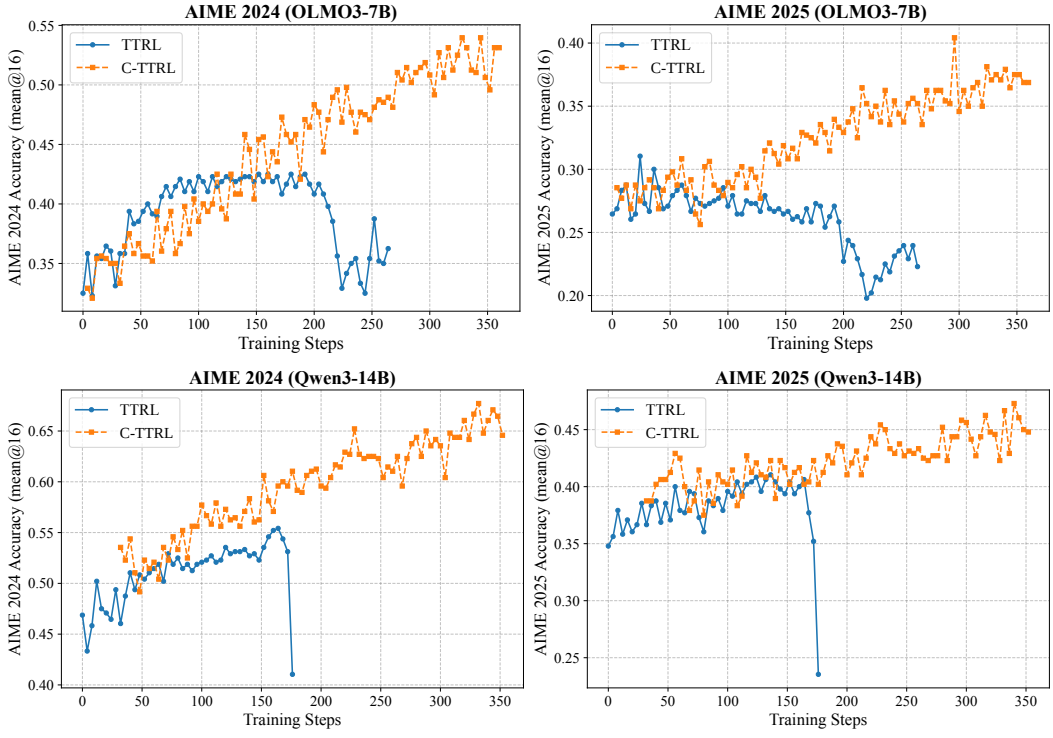


Figure 8: Avg@16 accuracy over training checkpoints. Top: OLMo3-7B. Bottom: Qwen3-14B. Left: AIME 2024; right: AIME 2025. `C-TTRL` (orange) maintains an upward trend over the evaluated horizon, whereas TTRL (blue) plateaus or degrades later in training. Each point is one evaluation checkpoint (every 4 training steps).

E.3 Additional Baseline Training Trajectories

The Qwen3-14B diagnostics below cover Self-Harmony, Semi-TTRL, TraPO, TTRL, and EMPO on AIME 2024 and AIME 2025. We show unsmoothed observations from step 4, retaining the original global training-step indices and joining TraPO's resumed runs along the continued checkpoint chain. Self-Harmony, Semi-TTRL, and TraPO retain their logged Mean@8 metric, while TTRL and EMPO use Avg@16. These curves describe within-run trends over different recorded horizons and supplement, rather than replace, the benchmark comparisons in Table 2.

Mean accuracy. Figure 9 reveals different training dynamics across the five baselines. Self-Harmony improves on both benchmarks, while Semi-TTRL shows smaller gains with persistent fluctuations and TraPO exhibits no sustained upward trend. TTRL and EMPO initially improve but lose accuracy near the end of their recorded runs on both AIME 2024 and AIME 2025. Taken together, these trajectories show that early gains do not necessarily translate into sustained improvement: several baselines subsequently plateau or degrade, contrasting with the continued upward trend of `C-TTRL` in Figure 8.

F Training Dynamics

F.1 Actor Entropy

Figure 10 compares actor entropy between `C-TTRL` and TTRL during training. On OLMO3-7B, entropy initially decreases and later rebounds for both methods. On Qwen3-14B, entropy generally increases, with TTRL showing a faster rise than `C-TTRL` over their shared training interval. These model-dependent trends do not support a uniform entropy ordering between the two methods. We therefore report entropy as a complementary training diagnostic, rather than direct evidence of semantic reasoning-path diversity.

F.2 Critic Discrimination Quality

Figure 11 shows the critic's AUROC for distinguishing correct from incorrect responses during `C-TTRL` training. The AUROC remains high (0.85–0.95) over the evaluated training horizon, showing that the critic retains strong discriminative performance under the full method as the policy changes. The frozen-critic ablation in Figure 3 provides complementary downstream evidence: leaving the critic unchanged leads to an earlier performance plateau.

F.3 TTRL Majority Ratio Dynamics

TTRL constructs its reward signal via majority voting: for each question, multiple responses are sampled and the most frequent answer is treated as the pseudo-label. The *majority ratio*—the fraction of sampled responses that agree with the majority answer—is a natural indicator of the model's output concentration.A higher majority ratio indicates greater agreement on an answer within the sampled group.

Figure 12 tracks this ratio over training for both OLMO3-7B and Qwen3-14B. On OLMO3-7B, the ratio rises from approximately 0.53 to around 0.95, then declines and fluctuates near 0.85–0.90. On Qwen3-14B, it starts near 0.55, rises to around 0.95, and drops sharply to approximately 0.53 at the end of the recorded run. Thus, answer agreement increases over much of training but does not rise monotonically or remain high through the final checkpoint.

F.4 Response Length Dynamics

We complement the Qwen3-14B accuracy trajectories in Appendix E with mean response lengths for `C-TTRL` and the five baselines in Figure 13 . Lengths are measured in tokens on each method's training batches, rather than separately on the AIME 2024 and AIME 2025 evaluation sets. All curves start at step 4, with the displayed horizons specified in the caption. Resumed runs are joined along their checkpoint chains without resetting global steps.

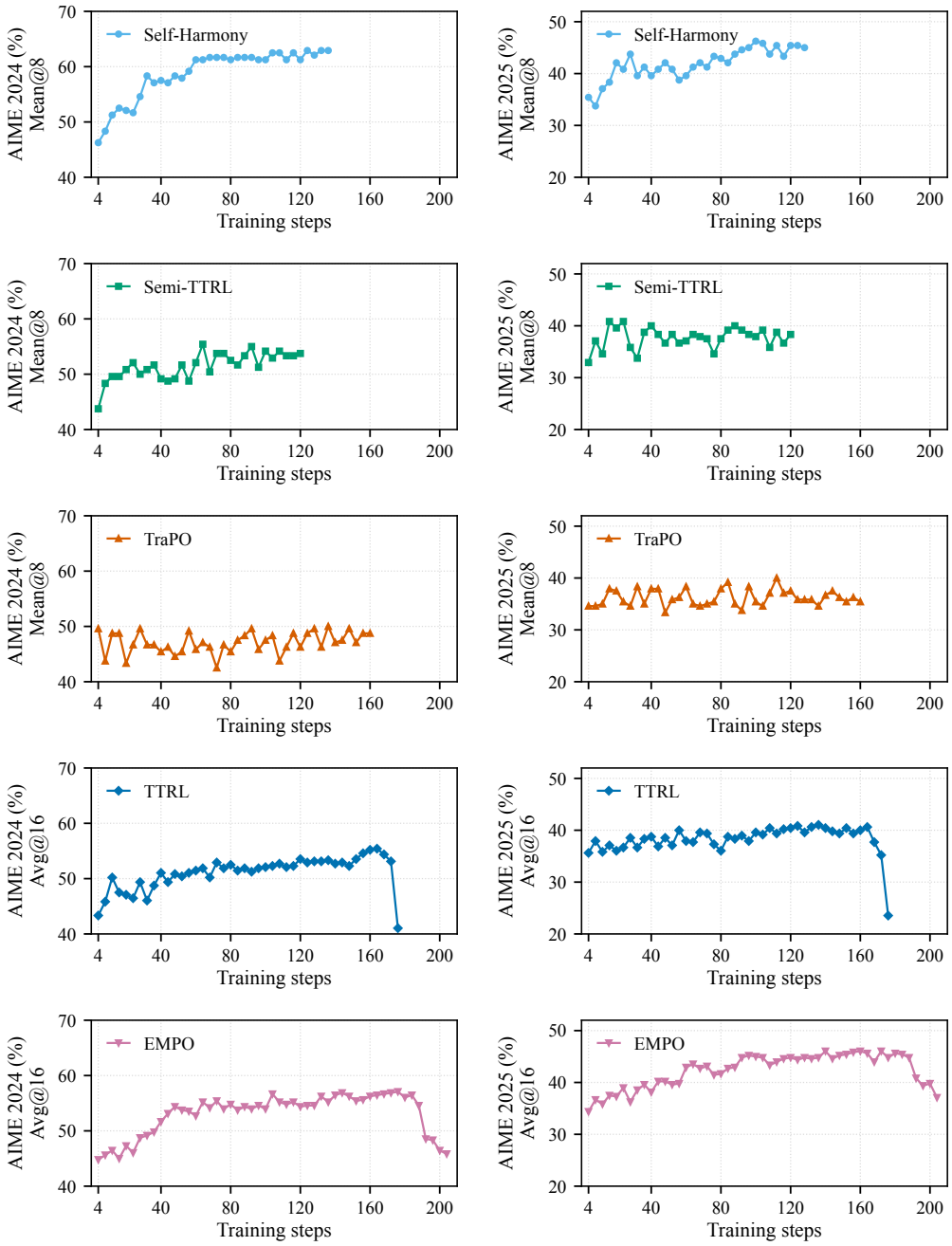


Figure 9: **Additional Qwen3-14B baseline training trajectories.** Left: AIME 2024; right: AIME 2025. Top to bottom: Self-Harmony (steps 4–136 on AIME 2024; 4–128 on AIME 2025), Semi-TTRL (4–120), TraPO (4–160), TTRL (4–176), and EMPO (4–204). The first three rows show Mean@8; the last two show Avg@16. Points are unsmoothed evaluations.

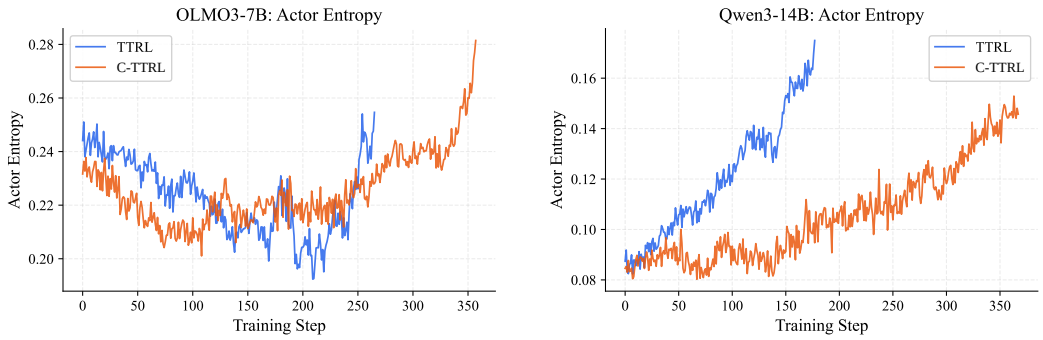


Figure 10: Actor entropy over training steps for OLMO3-7B (left) and Qwen3-14B (right). Blue: TTRL; orange: `C-TTRL`. Entropy initially decreases and later rebounds on OLMO3-7B, whereas it generally increases on Qwen3-14B.

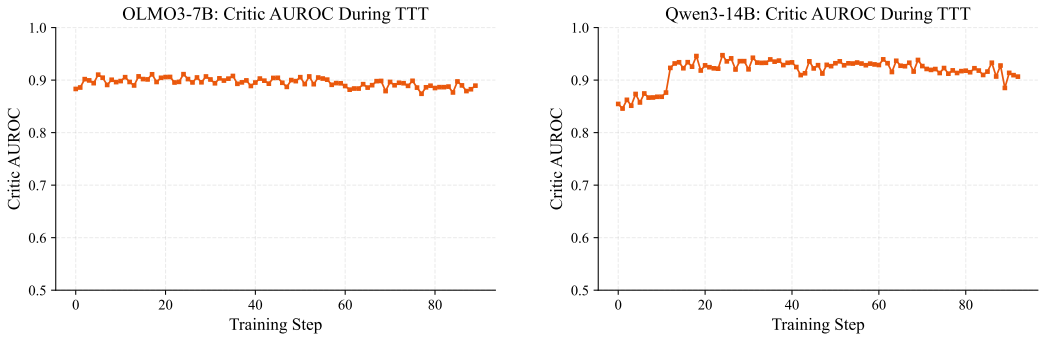


Figure 11: Critic AUROC during `C-TTRL` training for OLMO3-7B (left) and Qwen3-14B (right). Under the full method, the critic maintains strong discrimination between correct and incorrect responses (0.85–0.95) over the evaluated training horizon.

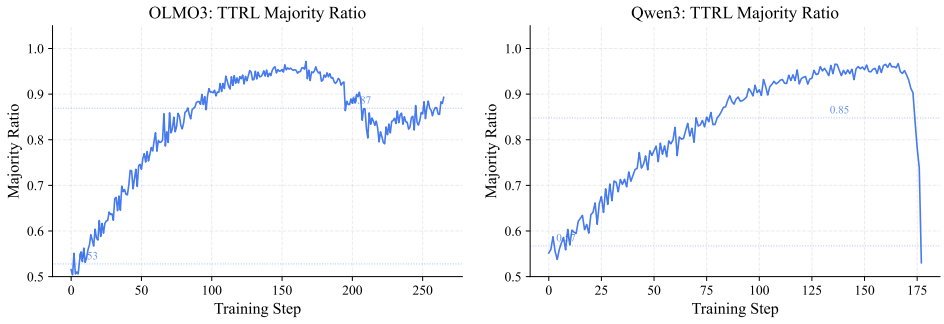


Figure 12: TTRL majority ratio over training steps for OLMO3-7B (left) and Qwen3-14B (right). Both curves rise during the earlier part of training. OLMO3-7B subsequently declines and fluctuates, while Qwen3-14B shows a sharp drop near the end. Majority ratio measures within-group answer agreement, not answer correctness or semantic reasoning-path diversity.

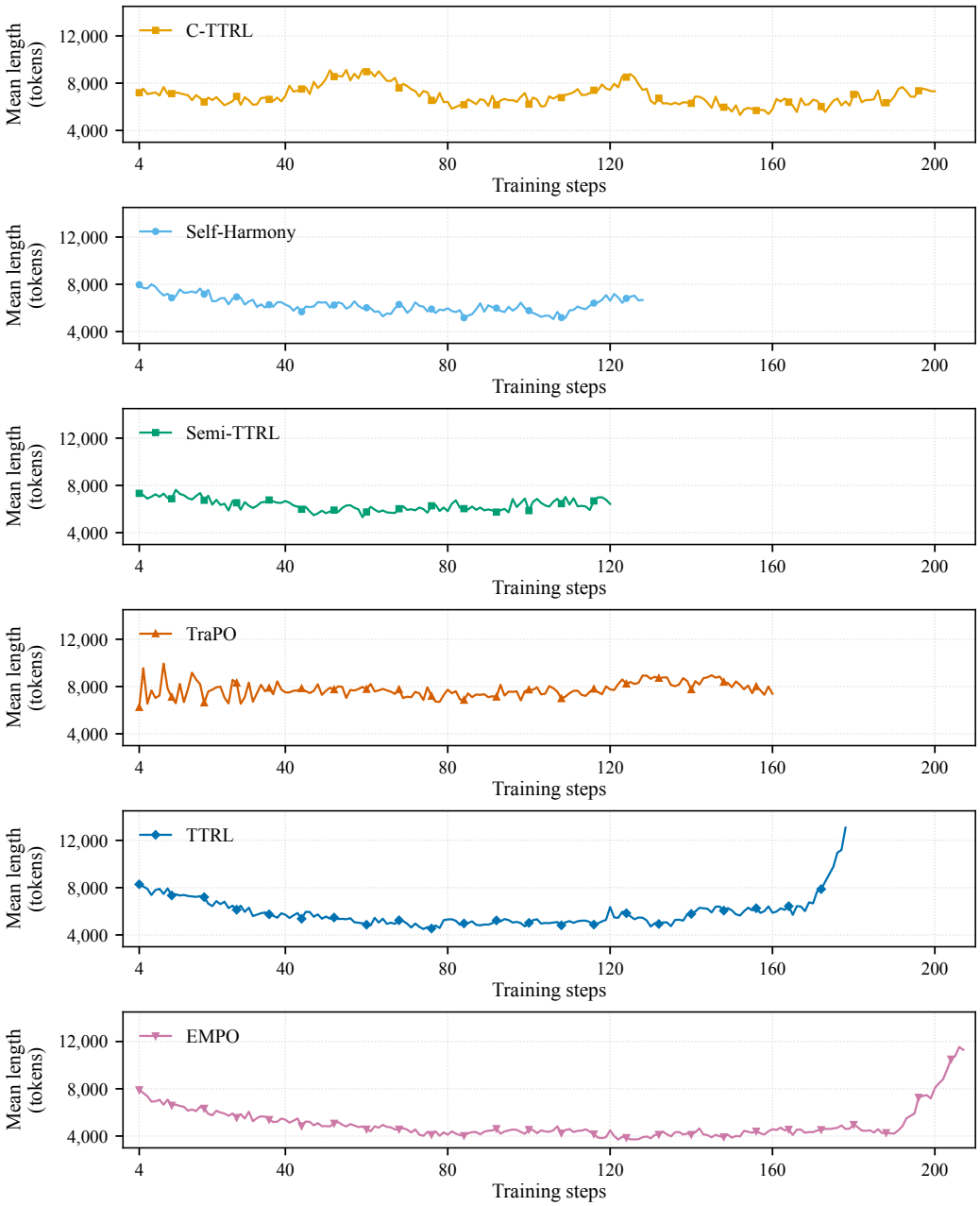


Figure 13: **Qwen3-14B response lengths.** Top to bottom: `C-TTRL` (steps 4–200), Self-Harmony (4–128), Semi-TTRL (4–120), TraPO (4–160), TTRL (4–178), and EMPO (4–207). Curves show unsmoothed mean generated response lengths on training batches, in tokens. Each panel shows one method; all panels share the same axis scales.

G Ablation on Labeled Data Size

A key practical question for C-TTRL is how large the labeled reference set D_L must be for critic recalibration to remain effective as the policy changes. If D_L is too small, the critic can overfit to the labeled examples and lose its ability to generalize to new policy responses. We compare three D_L sizes on OLMO3-7B: 128 samples, 1,024 samples, and the full labeled dataset.

We assess critic quality using two metrics rather than downstream accuracy, because the 128- and 1,024-shot runs train for fewer steps (105 and 128, respectively) than the full-data run (252 steps) and have not yet converged. **Critic AUROC** measures discriminative ability—whether the critic can distinguish correct from incorrect responses on held-out rollouts from the current policy. **Explained variance** on the reference training set versus held-out test rollouts measures generalization—a widening train–test gap indicates that the critic is memorizing the small labeled set rather than learning features that transfer to new policy outputs.

As shown in Figures 14 and 15, the 128-sample setting fails on both metrics: AUROC degrades from ∼0.96 to below 0.75 by step 105, and test explained variance falls near zero while training variance continues to rise—a clear signature of overfitting. In contrast, the 1,024-sample setting maintains AUROC in the 0.90–0.97 range over its evaluated training horizon, and the train–test explained-variance gap is broadly comparable to that observed in the full-data setting. Thus, 1,024 labeled samples are sufficient to maintain strong critic discrimination and generalization over this horizon, without using the full labeled dataset.

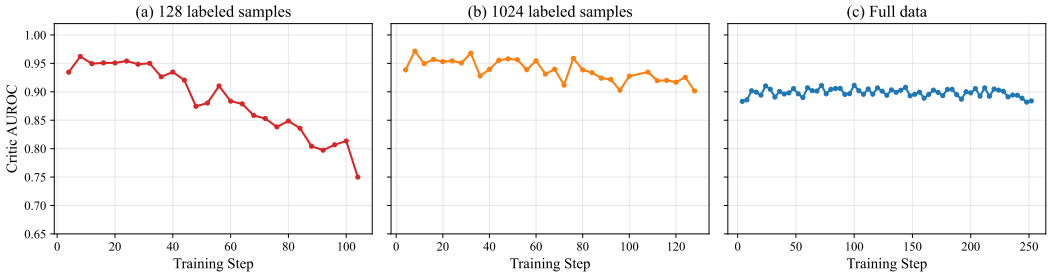


Figure 14: Critic AUROC over training steps for 128 labeled samples (left), 1,024 labeled samples (center), and the full reference set (right). With 128 samples, AUROC degrades below 0.75, indicating loss of discriminative ability. With 1,024 samples, AUROC remains between 0.90 and 0.97 over the evaluated training horizon, close to the full-data range.

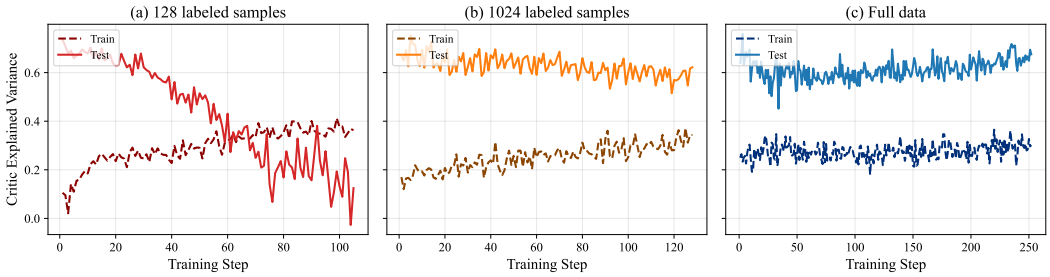


Figure 15: Critic explained variance (train: dashed, test: solid) for 128 labeled samples (left), 1,024 labeled samples (center), and the full reference set (right). The widening train–test gap in the 128-shot setting reveals overfitting. With 1,024 samples, the train–test gap over the evaluated training horizon is broadly comparable to that observed with the full reference set.

H Extended Sampling-Coverage Analysis

This section complements the mean-accuracy trajectories in Appendix E with problem-level sampling coverage. Pass@k measures the fraction of problems solved by at least one of k sampled responses. We report pass@16 trajectories for C-TTRL and TTRL, pass@8 trajectories for additional baselines, and final-checkpoint results at different sampling budgets. The trajectory panels use different model configurations and sampling budgets and should not be read as a matched cross-method comparison. The main paper summarizes the AIME 2024 pass@16 trend in Figure 1(c) and reports benchmark-level pass@8 results in Table 2.

H.1 Pass@16 Training Trajectories of C-TTRL and TTRL

Figure 16 shows the complete pass@16 trajectories on AIME 2024 and AIME 2025 for OLMO3-7B, extending the concise AIME 2024 view in Figure 1(c).

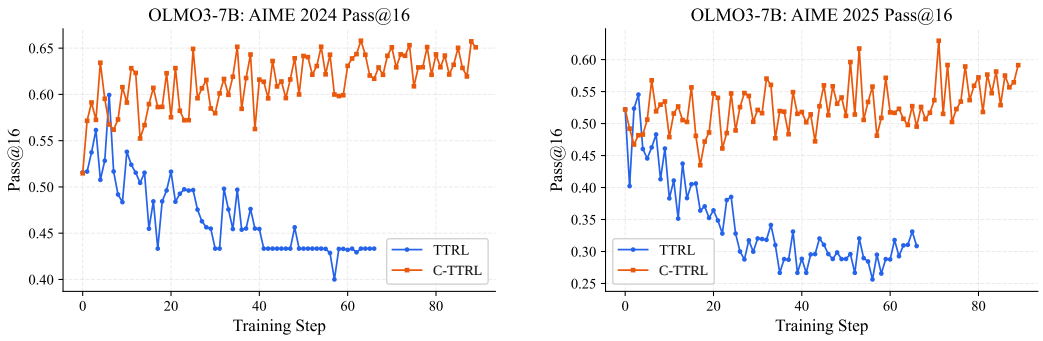


Figure 16: Pass@16 over training steps on AIME 2024 (left) and AIME 2025 (right) for OLMO3-7B. TTRL (blue) loses problem-level sampling coverage as training progresses, whereas C-TTRL (orange) maintains higher pass@16 coverage.

H.2 Additional Baseline Pass@8 Trajectories

Figure 17 reports pass@8 on AIME 2024 and AIME 2025 for the Qwen3-14B Self-Harmony, Semi-TTRL, and TraPO runs described in Appendix E.3. The curves retain the logged metric and the same displayed checkpoint ranges as the Mean@8 trajectories in Figure 9, starting from step 4 without smoothing.

Self-Harmony's later pass@8 values fall below its early peaks, especially on AIME 2025, despite its overall increase in Mean@8 accuracy (Figure 9). Semi-TTRL's coverage also fluctuates and ends below its early peaks; TraPO remains variable without a sustained increase. These observations show why increasing mean accuracy alone does not imply broader coverage of solvable problems.

H.3 Pass@k at Different k

Table 10 reports pass@k for varying k on AIME 2024, showing that the sampling-coverage advantage of C-TTRL holds across all evaluated budgets. The advantage of C-TTRL is substantial even at small k: at pass@2 (only two samples), C-TTRL already achieves 69.0% compared to TTRL's 50.3%. This result shows that the gain is not limited to large sampling budgets; C-TTRL finds correct solutions more often even with only two samples.

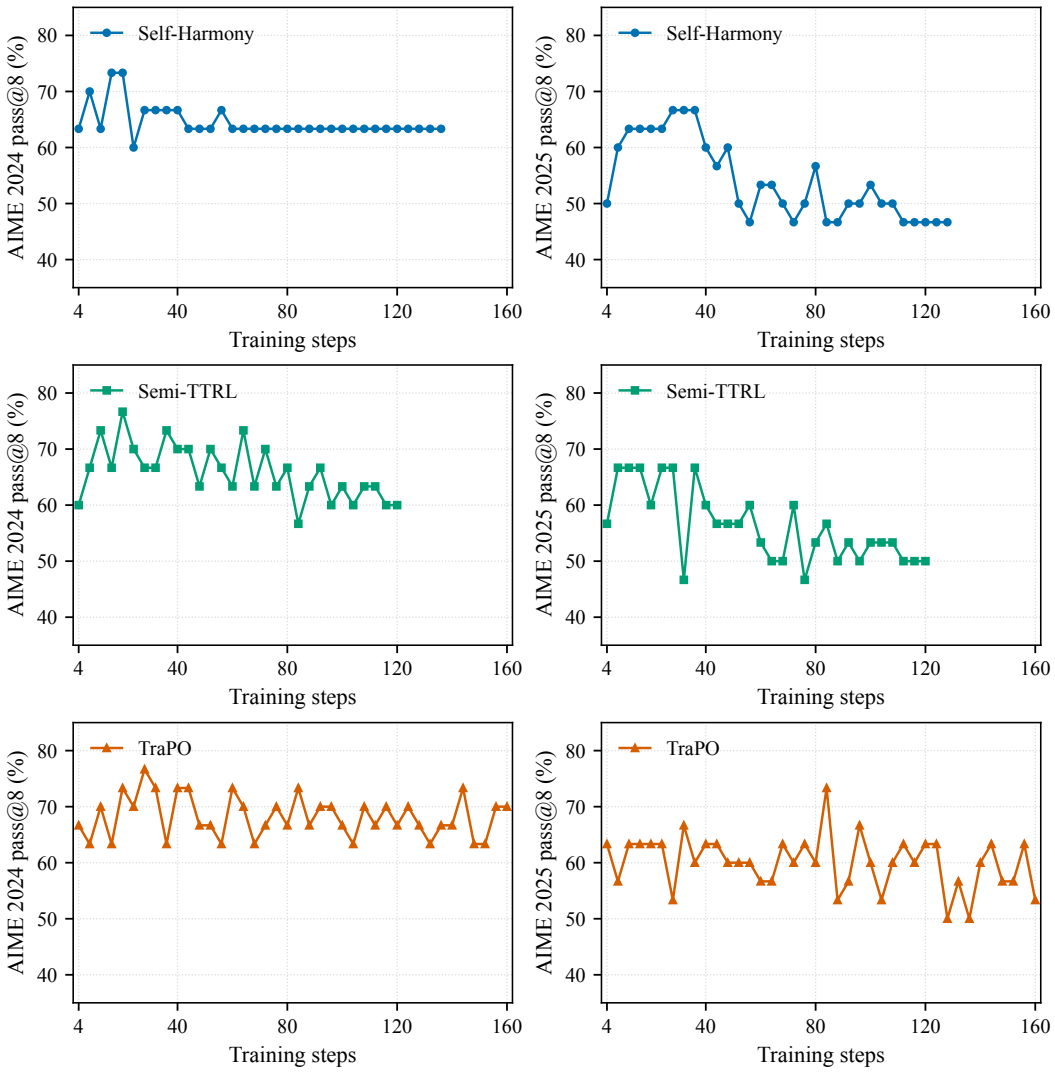


Figure 17: **Qwen3-14B baseline pass@8 trajectories.** Left: AIME 2024; right: AIME 2025. Rows show Self-Harmony, Semi-TTRL, and TraPO. Curves show unsmoothed pass@8 evaluations from step 4 over the same displayed checkpoint ranges as the corresponding first three rows of Figure 9.

Table 10: Pass@k on AIME 2024 (Qwen3-14B) at different sampling budgets k.

Method	pass@2	pass@4	pass@8	pass@16
TTRL	50.3	55.1	56.3	56.6
C-TTRL	69.0	70.6	71.3	72.1
Δ	+18.7	+15.5	+15.0	+15.5

I CASE STUDY

We present two examples from OLMO3-7B where the model answers incorrectly during early TTT but correctly after extended C−TTRL training. Both examples are drawn from the same test set used during TTT. Responses are truncated to show the critical reasoning steps and final answer.

I.1 EXAMPLE 1: PROBABILITY (BETA-BINOMIAL INTEGRATION)

Problem

A coin is randomly drawn from a population. The probability that this coin shows heads when tossed once is p, and p follows a uniform distribution $U[0, 1]$. Subsequently, this randomly drawn coin is tossed repeatedly for a total of 256 times. Let $p = \frac{m}{n}$ be the probability of getting exactly 128 heads where m and n are positive coprime integers. Find $m + n$.

Before TTT (Zero-RL, incorrect). The model correctly sets up the Beta-Binomial integral and arrives at $\frac{256!}{(128!)^2} \cdot \frac{(128!)^2}{257!} = \frac{1}{257}$, but then makes a critical error in the final step:

Incorrect: predicted $128! + 257$

`[...correct derivation leading to` $P = 1/257$`...]`
Thus, the probability is: $p = \frac{128!}{257}$
Since $m = 128!$ and $n = 257$ are coprime, we find: $m + n = 128! + 257$
Therefore, the final answer is: $\boxed{128! + 257}$

The model correctly computed $P = 1/257$ but then inexplicably rewrote the fraction as $128!/257$, confusing the intermediate computation with the final result.

After TTT (late C−TTRL, correct). After 300+ steps of C−TTRL training, the model solves the same problem cleanly:

Correct: predicted 258

`[...same Beta-Binomial setup...]`
$P(X = 128) = \frac{256!}{(128!\cdot 128!)} \cdot \frac{(128!\cdot 128!)}{257!} = \frac{256!}{257!} = \frac{1}{257}$
So, $m = 1$ and $n = 257$, and they are coprime. Therefore: $m + n = 1 + 257 = \boxed{258}$

I.2 EXAMPLE 2: LOGARITHMIC SYSTEM OF EQUATIONS

Problem

Let x, y and z be positive real numbers that satisfy the following system of equations: $\log_2\left(\frac{x}{yz}\right) = \frac{1}{2}$, $\log_2\left(\frac{y}{xz}\right) = \frac{1}{3}$, $\log_2\left(\frac{z}{xy}\right) = \frac{1}{4}$. Then the value of $|\log_2(x^4y^3z^2)|$ is $\frac{m}{n}$ where m and n are relatively prime positive integers. Find $m + n$.

Before TTT (Zero-RL, incorrect). The model enters a degenerate repetition loop, producing the same irrelevant text dozens of times before hitting the token limit:

Incorrect: predicted 2 (degenerate repetition)

`[...initial reasoning attempt diverges...]`
The value of k such that $22k = 44$ equals some polynomial is: $\boxed{2}$
The value of k such that $22k = 44$ equals some polynomial is: $\boxed{2}$
`[...same line repeated >30 times until token limit...]`

This is a clear example of reasoning collapse: the model fails to maintain coherent problem-solving and falls into a repetitive output pattern.

After TTT (late `C-TTRL`, correct). After `C-TTRL` training, the model solves the problem systematically by converting to a linear system:

Correct: predicted 33

Let $a = \log_2 x$, $b = \log_2 y$, $c = \log_2 z$. The system becomes:
$a - b - c = 1/2, \quad -a + b - c = 1/3, \quad -a - b + c = 1/4$
Adding the equations pairwise gives $-2a = 1/3 + 1/4$, $-2b = 1/2 + 1/4$, and $-2c = 1/2 + 1/3$.
Hence, $a = -7/24$, $b = -9/24$, and $c = -10/24$.
$|4a + 3b + 2c| = \left|\frac{-28-27-20}{24}\right| = \frac{75}{24} = \frac{25}{8}$
$m + n = 25 + 8 = \boxed{33}$

J PROMPT FORMAT

All models use the ChatML format during both RLVR initialization and test-time training. The prompt template is shown below. The problem statement is inserted between the instruction preamble and the closing reminder, and the model generates its response in the assistant turn.

Prompt Template

```
<|im_start|>user
Please answer the problem adhering to the following rules:
1. Please use LaTeX format to represent the variables and formulas
used in the solution process and results.
2. Please put the final answer(s) in \boxed{}, note that the unit
of the answer should not be included in \boxed{}.
3. If the problem requires multiple answers, list them in order,
each in a separate \boxed{}.
Problem: {problem text}
Remember to output your answer within \boxed{}.<|im_end|>
<|im_start|>assistant
```

For general reasoning tasks (BigBenchHard, AGI Eval, ZebraLogic), the instruction preamble is adjusted to match each benchmark's expected answer format, while keeping the same ChatML structure.

J.1 ANSWER-JUDGING PROMPT

The following is the exact answer-verification template used with `gpt-oss-120b` in our experiments. The placeholders `{problem}`, `{given_answer}`, and `{ground_truth}` denote the question, the model answer to be judged, and the reference answer, respectively. The same template verifies labeled training responses and evaluation responses.

Answer-Verification Prompt

```
User: ### Question: {problem}

### Ground Truth Answer: {ground_truth}

### Student Answer: {given_answer}

For the above question, please verify if the student's answer is
equivalent to the ground truth answer.
Do not solve the question by yourself; just check if the student's
answer is equivalent to the ground truth answer.
If correct, output [Correct]. If incorrect, output [Incorrect].
Assistant:
```