TEMPO: Scaling Test-time Training for Large Reasoning Models

Qingyang Zhang [*1,2] **Xinke Kong** [*1] **Haitao Wu** [1] **Qinghua Hu** [1] **Minghao Wu** [3] **Baosong Yang** [3] **Yu Cheng** [4]
Yun Luo [†2] **Ganqu Cui** [†2] **Changqing Zhang** [†1]

Abstract

Large reasoning models (LRMs) remain static after training, limiting their ability to adapt to novel problems. Test-time training (TTT) addresses this by adapting model parameters on unlabeled test instances, continuously extending capabilities beyond offline training. However, existing TTT methods that rely on self-generated reward signals plateau quickly: without external grounding, the reward signal drifts as the policy evolves, causing both performance saturation and diversity collapse. We propose TEMPO, a dual-agent self-evolution framework that interleaves policy refinement by a *reasoning agent* on unlabeled questions with periodic recalibration of a *verification agent* (critic) on a labeled dataset. By formalizing this alternating procedure through the Expectation-Maximization (EM) algorithm, we reveal that prior single-agent self-rewarding methods are incomplete EM variants that omit the crucial verification step. Restoring this step tightens the evidence lower bound (ELBO) and enables sustained self-evolution. Across diverse model families (Qwen3 and OLMO3) and reasoning tasks, TEMPO improves OLMO3-7B on AIME 2024 from 33.0% to 51.1% and Qwen3-14B from 42.3% to 65.8%, while maintaining high output diversity. These results demonstrate that collaborative dual-agent self-evolution can push LRMs substantially beyond their static training boundaries. Code is available at https://github.com/QingyangZhang/TEMPO.

[*]Equal contribution [†]Co-supervised. [1]Tianjin University [2]Shanghai AI Lab [3]Tongyi Lab, Alibaba Group [4]The Chinese University of Hong Kong. Correspondence to: Yun Luo <luoyun1@pjlab.org.cn>, Ganqu Cui <cuiganqu@pjlab.org.cn>, Changqing Zhang <zhangchangqing@tju.com>.

The 3rd *AI for Math Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

1. Introduction

A fundamental challenge in AI for science is building reasoning agents that can autonomously self-evolve when dealing with novel problems beyond their training distribution. Large reasoning models (LRMs) have demonstrated remarkable capabilities on complex reasoning tasks, including logic puzzle-solving and Olympiad-level mathematics and physics (Cui et al., 2025; Chen et al., 2025; Luo et al., 2026). These models achieve their performance by allocating extensive computation at test time through extended reasoning chains. However, their parameters remain static after training, which prevents them from acquiring new knowledge during test-time experience.

Test-time training (TTT) addresses this limitation by enabling models to update their parameters on unlabeled test data, thereby extending their reasoning capabilities beyond the original training distribution. Recent methods such as EMPO (Zhang et al., 2025c), TTRL (Zuo et al., 2025), and Theta-Evolve (Wang et al., 2025) employ self-generated reward signals such as entropy, majority voting, or self-consistency to refine reasoning policies via reinforcement learning without ground-truth labels. These can be understood as *single-agent self-evolution* approaches: the model generates its own supervision and iteratively refines itself. Practical implementations such as Cursor's real-time reinforcement learning for Composer demonstrate the promise of test-time adaptation in dynamically adapting models to complex, interactive environments (Jackson et al., 2026).

Despite promising initial results, these single-agent self-evolution methods exhibit two fundamental limitations. First, they rely on self-generated reward signals that are intrinsically bounded by the model's initial capabilities, leading to performance plateaus as self-improvement saturates (Zuo et al., 2026). Second, these methods tend to collapse output diversity in pursuit of higher average performance, ultimately degrading reasoning quality (Zhang et al., 2025d). Both issues share a common root cause: without an independent verification mechanism, the reward signal drifts as the model becomes increasingly confident in a narrow set of reasoning patterns. These heuristic signals systematically overestimate the quality of self-generated responses, creating a self-reinforcing loop that drives both saturation and

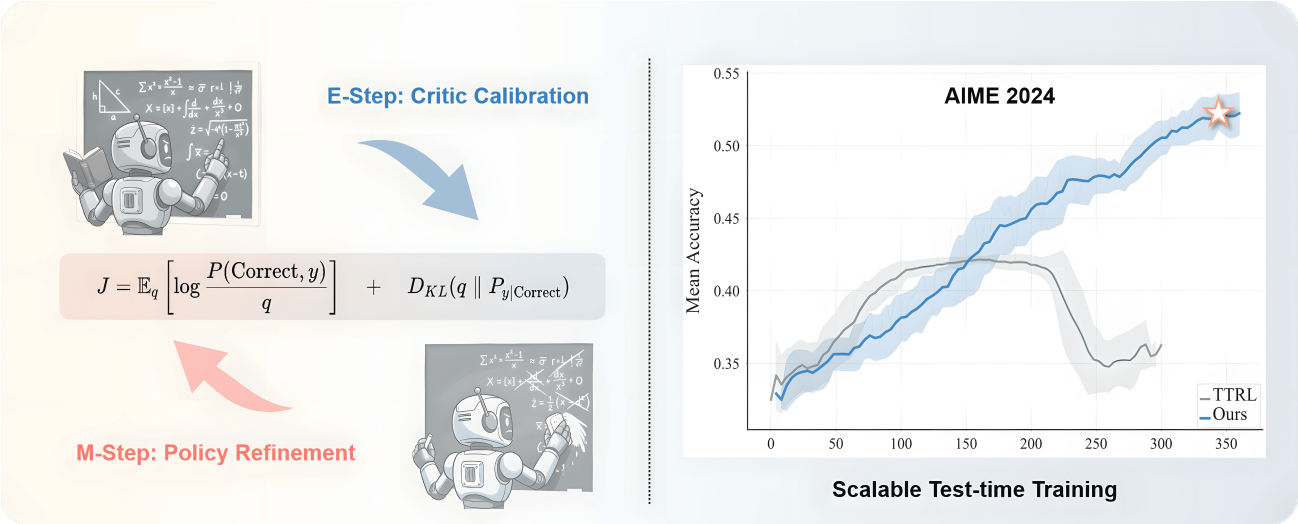

Figure 1. **Scalable self-evolution of TEMPO on the AIME benchmark.** TEMPO alternates between an E-Step (verification agent recalibration on labeled data) and an M-Step (reasoning agent refinement on unlabeled test questions). Representative single-agent self-rewarding baselines such as TTRL (grey curves) plateau and collapse after initial gains. In contrast, the TEMPO dual-agent framework (blue curve) sustains a consistent upward trajectory over 350 steps by periodically grounding the verification agent in external supervision, demonstrating that additional test-time compute translates into scalable self-evolution on unlabeled open problems.

diversity collapse.

To this end, we propose Test-time Expectation-Maximization Policy Optimization (TEMPO), a *dual-agent self-evolution* framework that decouples reward generation from policy optimization through a *reasoning agent* (actor) and a *verification agent* (critic) that collaborate at test time (Figure 1). TEMPO interleaves *Reasoning Agent Refinement* (M-step), where the actor optimizes against verification agent signals on unlabeled test prompts, with *Verification Agent Recalibration* (E-step), where the critic is periodically updated on a labeled dataset to stay grounded in external truth. We motivate this design through the EM framework: response correctness is a latent variable at test time, and maximizing the policy objective is naturally framed as tightening an evidence lower bound (ELBO). This reveals that existing single-agent self-rewarding methods are degenerate EM variants executing only the M-step, causing the posterior estimate to drift. Restoring the E-step through the verification agent sustains improvement over extended self-evolution.

Experimental results validate both the effectiveness and scalability of TEMPO as a dual-agent self-evolution system. On AIME 2024 and 2025, TEMPO improves OLMO3-7B from 33.0% and 26.3% to 51.1% and 37.0%. For Qwen3-14B, TEMPO pushes the accuracy from 42.3% and 37.1% to 65.8% and 44.6%, achieves an absolute gain of 23.5 and 7.5 percentage points, respectively. More importantly, TEMPO maintains high pass@K scores where single-agent baselines suffer from diversity collapse. Beyond mathematics, TEMPO generalizes to non-math reasoning domains, including logic puzzles and STEM tasks, confirming that the collaborative dual-agent design is not domain-specific and represents a step toward general self-evolving scientific reasoning agents.

Our contributions are summarized as follows

- We propose TEMPO, a dual-agent self-evolution framework that alternates between verification agent recalibration and reasoning agent refinement, enabling sustained self-improvement without the performance plateaus and diversity collapse inherent in prior single-agent self-rewarding approaches (Section 4.2, 4.3).
- We provide a unified EM-theoretic analysis showing that existing TTT methods can be interpreted as degenerate EM instances that omit the verification step (E-step), which theoretically accounts for their observed saturation and diversity collapse.
- We demonstrate strong results across different model scales (7B, 8B, 14B) and reasoning benchmarks spanning math, logic, and STEM, demonstrating consistent gains in average accuracy while preserving output diversity (Sections 4.2, 4.4).

2. Related Work

Single-agent self-rewarding for reasoning. Reinforcement learning with verifiable rewards (RLVR) has emerged as the dominant paradigm for incentivizing reasoning capability in LLMs (Lambert et al., 2024; Guo et al., 2025;

Shao et al., 2024), but its reliance on labeled data motivates self-rewarding alternatives. Prior works leverage intrinsic rewards such as entropy (Zhang et al., 2025c; Gao et al., 2025b), self-certainty (Zhao et al., 2025), majority voting (Yang et al., 2025b) or reasoning topology (Wang et al., 2026) for self-training. These can be viewed as *single-agent self-evolution* approaches: a single model generates, evaluates, and refines itself without external oversight. The pioneering TraPO (Yang et al., 2025b) demonstrates that a modest labeled set is sufficient to guide stable RLVR on unlabeled reasoning data, achieving results comparable to fully supervised training. However, these single-agent methods tend to collapse the output distribution and plateau as the self-generated reward signal becomes self-reinforcing (Zhang et al., 2025d; Zuo et al., 2026). Our approach overcomes this limitation by introducing a separate verification agent whose periodic external grounding breaks the self-reinforcement loop, enabling genuine self-evolution rather than self-repetition.

Test-time training as self-evolution for reasoning agents. Test-time training (TTT) originated in computer vision, where models continuously update their parameters on each test instance at inference time to bridge distribution shifts (Grandvalet & Bengio, 2004; Wang et al., 2021; Zhang et al., 2025b). For reasoning models, recent methods apply test-time RL using self-generated signals: TTRL (Zuo et al., 2025) uses majority voting for pseudo-labels, Intuitor (Zhao et al., 2025) and EMPO (Zhang et al., 2025c) use entropy-based rewards. These single-agent self-evolution methods share a structural deficiency: the same model both generates reasoning and evaluates its quality, causing the reward signal to drift from true correctness as the policy evolves—a phenomenon analogous to the mode collapse observed in models trained on their own outputs (Shumailov et al., 2024). From the perspective of building autonomous scientific agents, this is a critical limitation: a self-evolving system that cannot verify its own knowledge will eventually diverge. TEMPO addresses this by introducing a dual-agent architecture where a verification agent periodically grounds the self-evolution loop in external truth, maintaining a tight ELBO and enabling sustained improvement beyond the ceilings of single-agent methods.

3. Method

We propose Test-time Expectation-Maximization Policy Optimization (TEMPO), a dual-agent self-evolution framework that initializes a *reasoning agent* (actor) and a *verification agent* (critic) via RLVR on labeled data D_L, then continuously improves on unlabeled test questions by alternating between verification agent calibration and reasoning agent refinement, inspired by the EM algorithm. This section is organized as follows: we first formalize the problem setup (Section 3.1), then derive an EM-inspired variational lower bound as the self-evolution objective (Section 3.2), and finally detail the alternating E-step verification agent calibration (Section 3.3) and M-step reasoning agent optimization (Section 3.4). The dual-agent self-evolution procedure is illustrated in Figure 2.

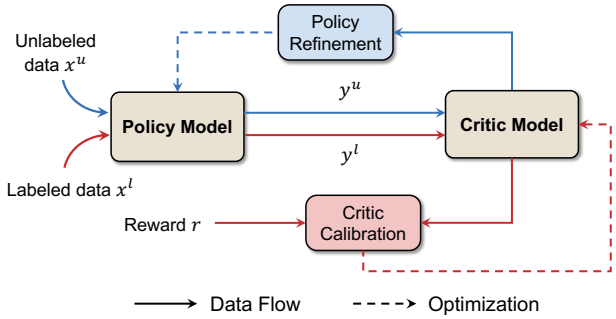

Figure 2. TEMPO alternates between (i) *Verification Agent Recalibration (E-step)*: the verification agent (critic) is periodically updated using verifiable rewards from D_L to maintain a grounded and informative reward signal, and (ii) *Reasoning Agent Refinement (M-step)*: the reasoning agent (actor) generates reasoning trajectories on unlabeled questions D_u and optimizes against verification agent signals. This alternating dual-agent EM-style procedure enables sustained self-evolution beyond the RLVR plateau.

3.1. Problem Setup

We consider a setting where a self-evolving agent has access to a labeled dataset D_L containing ground-truth answers and a set of unlabeled test questions D_u where the correct responses are unknown. Our goal is to enable LRMs to continuously self-evolve during the test phase. Formally, we aim to maximize the expected log-probability of generating a correct response given a question x. Let θ denote the parameters of the reasoning agent (LRM), and $P(\text{Correct}|x;\theta)$ represent the probability that the agent produces a correct output for a given input x. The global self-evolution objective is:

$$J(\theta) = \mathbb{E}_x[\log P(\text{Correct}|x;\theta)], \tag{1}$$

where the marginal probability $P(\text{Correct}|x;\theta)$ is obtained by marginalizing over all possible generated responses y as

$$P(\text{Correct}|x;\theta) = \sum_y P(\text{Correct}|x,y)\pi_\theta(y|x). \tag{2}$$

Here, $\pi_\theta(y|x)$ denotes the reasoning policy, which represents the output distribution of the reasoning agent.

3.2. Variational Lower Bound Objective

We draw on the evidence lower bound (ELBO) from variational inference to enable self-evolution when ground-truth correctness is unobserved, showing how the EM-inspired

objective decomposes into an estimable lower bound and a KL divergence term.

The fundamental challenge in test-time self-evolution is that the response correctness for $x \in D_u$ is unobserved. In such scenarios, the optimal response distribution $P(y|x, \text{Correct})$ is a latent variable. To optimize $J(\theta)$ under these conditions, we draw motivation from the Expectation-Maximization (EM) framework. By introducing an auxiliary distribution $q(y|x)$, we derive the Evidence Lower Bound (ELBO):

$$\begin{aligned} J(\theta) &= \sum_{x \in D_u} \log \sum_y q(y|x) \frac{P(\text{Correct}|y,x)\pi_\theta(y|x)}{q(y|x)} \\ &= \sum_{x \in D_u} \Bigg(\sum_y q(y|x) \log \frac{P(\text{Correct}|y,x)\pi_\theta(y|x)}{q(y|x)} \\ &\qquad + KL\big(q(y|x) \,\|\, P(y|x, \text{Correct})\big) \Bigg). \end{aligned} \tag{3}$$

Intuitively, this decomposition says: maximizing the lower bound corresponds to (i) increasing the expected log-likelihood of responses that are likely to be correct, and (ii) bringing the auxiliary distribution q closer to the true posterior $P(y|x, \text{Correct})$.

By omitting the non-negative KL divergence term, we obtain the objective $\mathcal{L}(q, \theta)$, where $J(\theta) \geq \mathcal{L}(q, \theta)$. The equality holds if and only if $q(y|x)$ perfectly matches the posterior distribution $P(y|x, \text{Correct})$. This allows for iterative self-evolution by alternating between estimating the distribution of correct responses and maximizing model likelihood.

3.3. Expectation Step: Verification Agent Calibration

We now describe how to train a verification agent (critic) on labeled data to approximate the posterior distribution over correct responses, thereby grounding the self-evolution loop in external supervision. In the E-step, we keep the current reasoning agent π_{θ_0} fixed and seek the optimal auxiliary distribution $q^*(y|x)$ that maximizes the lower bound. This optimal distribution corresponds to the posterior probability of the response conditioned on its correctness:

$$q(y|x) = P(y|x, \text{Correct}) = \frac{P(\text{Correct}|y,x)\pi_{\theta_0}(y|x)}{P(\text{Correct}|x)}. \tag{4}$$

To approximate the unknown term $P(\text{Correct}|y, x)$, we train a verification agent $V_\phi(x, y_t) \in \mathbb{R}$ using the labeled data D_L, where $t \geq 0$ is the token index. The verification agent is optimized by minimizing the MSE between its token-level predictions and the ground-truth outcomes. For each response y associated with a query $x \in D_L$, the verification agent is trained to perform token-level value estimation. Formally, the verification agent parameters ϕ are updated by

$$\mathcal{L}_{\text{critic}}(\phi) = \mathbb{E}_{(x,y,\mathcal{I}) \sim D_L} \|V_\phi(x, y_t) - \mathcal{I}\|_2^2, \tag{5}$$

where $\mathcal{I} \in \{0, 1\}$ denotes the binary correctness indicator. This training regime ensures the verification agent serves as a reliable proxy for the expected correctness of generated response y, providing an external grounding signal for self-evolution.

Once optimized, the verification agent provides a tractable surrogate for the posterior distribution. Since it is trained to predict outcome correctness, its last-token value $V_\phi(x, y_T)$ directly reflects the likelihood of a correct response, enabling the optimal auxiliary distribution to be approximated by reweighting the reasoning agent's current outputs with the verification scores:

$$q(y|x) \propto V_\phi(x, y_T)\pi_{\theta_0}(y|x), \tag{6}$$

where T is the response length of y. This step effectively identifies high-quality responses from the reasoning agent's own generations to serve as a surrogate for the missing ground-truth labels.

3.4. Maximization Step: Reasoning Agent Refinement

Finally, we formulate the reasoning agent update as a weighted maximum likelihood estimation using verification agent rewards, and implement it via a policy gradient RL framework with token-level advantage estimation. In the M-step, we fix the auxiliary distribution $q(y|x)$ and update the reasoning agent parameters θ by maximizing the lower bound $\mathcal{L}(q, \theta)$. The optimization problem is:

$$\theta_{\text{new}} = \arg\max_\theta \sum_{x \in D_u} \sum_y q(y|x) \log \left(P(\text{Correct}|y,x)\pi_\theta(y|x) \right). \tag{7}$$

By removing terms independent of θ, the objective simplifies to a weighted maximum likelihood estimation. Substituting the approximation of $q^*(y|x)$ derived in the E-step, the update rule becomes

$$\theta_{\text{new}} = \arg\max_\theta \sum_{x \in D_u} \sum_y V_\phi(x, y_T)\pi_{\theta_0}(y|x) \log \pi_\theta(y|x). \tag{8}$$

Given that $\pi_{\theta_0}(y|x)$ represents the sampling distribution, the inner summation can be interpreted as an expectation. We further simplify the objective by focusing on the weighted log-likelihood of the sampled responses as

$$\theta_{\text{new}} = \arg\max_\theta \sum_{x \in D_u} \sum_y V_\phi(x, y_T) \log \pi_\theta(y|x). \tag{9}$$

This optimization is then solved using policy gradient methods, facilitating the continuous self-evolution of the reasoning agent on unlabeled open questions.

To effectively optimize this objective while ensuring variance reduction, we implement the M-step update via a policy gradient-based RL framework. In this setting, the value prediction of the verification agent at the terminal token of the

Algorithm 1 Test-time Expectation-Maximization Policy Optimization (TEMPO)

Input: Labeled dataset D_L, unlabeled test set D_u, reasoning agent π_{θ_0}, verification agent V_{ϕ_0}, total iterations N
Output: Self-evolved reasoning agent π_{θ_N}
{Stage 1: RLVR Initialization}
Train reasoning agent π_{θ_0} and verification agent V_{ϕ_0} via RLVR on D_L
for $k = 1$ **to** N **do**
  {Stage 2: Dual-Agent Self-Evolution}
  {E-step: Verification Agent Recalibration}
  Sample $(x, y, \mathcal{I})$ from D_L
  Update verification agent ϕ by minimizing $\mathcal{L}_{\text{critic}}(\phi)$ (Eq. (5))
  {M-step: Reasoning Agent Refinement}
  Sample $x \sim D_u$, generate responses $y \sim \pi_\theta$
  Compute advantages $A_t = V_\phi(x, y_T) - V_\phi(x, y_{1:t})$
  Update reasoning agent θ by minimizing $\mathcal{L}_{\text{policy}}(\theta)$ (Eq. (10))
end for
return π_{θ_N}

sequence is treated as the external reward $R = V_\phi(x, y)$ for the entire response. To derive a stable training signal, we utilize the verification agent's intermediate value predictions $V_\phi(x, y_{1:t})$ as token-varying baselines b_t. The advantage A_t for each token y_t is defined as the discrepancy between the final realized reward and the expected value at step t, i.e., $A_t = R - V_\phi(x, y_{1:t})$. This formulation ensures that tokens contributing to a higher last-token verification score receive a positive reinforcement signal, while those leading to deviations from the predicted value are penalized. The reasoning agent optimization objective for the M-step is

$$\mathcal{L}_{\text{policy}}(\theta) = -\mathbb{E}_{x \in D_u, y \sim \pi_\theta}\left[\sum_{t=1}^{T} A_t \log \pi_\theta(y_t | x, y_{<t})\right]. \tag{10}$$

By alternating between verification agent recalibration (E-step) and reasoning agent refinement (M-step), the dual-agent system achieves continuous, scalable self-evolution on unlabeled open reasoning problems. The full alternating procedure is summarized as Algorithm 1.

4. Experiments

We empirically validate TEMPO as a dual-agent self-evolution framework across four dimensions. We first describe the experimental setup (Section 4.1), then demonstrate sustained self-evolution beyond RLVR ceilings (Section 4.2), show that the dual-agent design preserves output diversity while single-agent baselines collapse (Section 4.3), validate generalization to non-math reasoning tasks (Section 4.4), and finally ablate the alternating dual-agent design to confirm the necessity of each component (Section 4.5).

4.1. Experimental Setup

We evaluate TEMPO on both mathematical reasoning and general domain reasoning tasks. Full hyperparameters, prompt format, and computational cost analysis are provided in Appendices B and C. ∘ **Math experiments.** Similar to prior works (Zheng et al., 2025; Yue et al., 2025; Gao et al., 2025a), we select Qwen3 (Yang et al., 2025a) and OLMO3 (Olmo et al., 2025) as base models, initializing actor and critic via PPO on DAPO-Math-17K (Yu et al., 2025). We perform TTT on AIME 2024, AIME 2025, and Beyond AIME (ByteDance-Seed, 2025), with AIME 2026 and OlymMath (Zhang et al., 2025a) as holdout sets. We compare against Zero-RL (PPO), TTRL (Zuo et al., 2025), and EMPO (Zhang et al., 2025c), reporting *avg@16* and *pass@8*. ∘ **General reasoning.** We initialize via PPO on Dolci-RL-Zero-General (Olmo et al., 2025) (12.8K labeled corpus) and perform TTT on BigBenchHard (Suzgun et al., 2023), AGI Eval (Zhong et al., 2024), and ZebraLogic (Lin et al., 2025), with GPQA-Diamond (Rein et al., 2024) as a holdout. Following common practice, we use `gpt-oss-120b` (Agarwal et al., 2025) as the judge model.

4.2. Self-Evolution Scalability (RQ1)

We first evaluate whether the TEMPO dual-agent framework can sustainably self-evolve beyond the RLVR training ceiling by leveraging unlabeled test data. We compare TEMPO against the RLVR baseline and two representative single-agent self-rewarding TTT methods (TTRL and EMPO) across three model families and five benchmarks. Results are shown in Table 1. TEMPO **significantly outperforms baselines across model scales and benchmarks.** On AIME 24, TEMPO improves OLMO3-7B-Base from 33.0% to 51.1% and Qwen3-14B-Base from 42.3% to 65.8%. Gains are particularly pronounced on the most challenging benchmarks where prior methods show the largest degradation. By periodically recalibrating the critic, TEMPO prevents reward drift and continues to provide informative gradients throughout training. We further evaluate TEMPO's capability to surpass the RLVR ceiling by continuing from a converged OLMO3 model (192 steps on DAPO-Math-17K). As shown in Figure 5, further PPO training yields negligible gains, whereas TEMPO produces a consistent 15+ point surge over 200 iterations. Extended mean@16 scalability curves are provided in Appendix D.

4.3. Diversity Preservation (RQ2)

TEMPO improves model performance without compromising diversity. While many TTT methods achieve short-

Table 1. **Main results on math reasoning benchmarks.** avg@16 and pass@8 across five benchmarks, with Δ over Zero-RL. Gray: TTT data; blue: held-out.

	Bey. AIME		AIME 24		AIME 25		AIME 26		OlymM.	
	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
↪ TEMPO	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	26.7	33.3	35.4	19.4	33.3	13.1	26.7
↪ TEMPO	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
↪ TEMPO	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

term accuracy gains, they frequently suffer from diversity collapse, i.e., the model converges to a narrow set of reasoning patterns, causing pass@k to degrade. As reported in Table 1, TEMPO consistently maintains high pass@k scores while baselines degrade significantly. On Qwen3-14B, TEMPO achieves pass@8 of 73.3 on AIME 24 vs. TTRL's 56.7 and EMPO's 59.7. This advantage stems from the critic's continuous scoring, which preserves diversity among high-quality solutions rather than favoring the most common reasoning path regardless of correctness. The sustained gap between pass@16 and avg@16 (Figure 3) is consistent with this: under diversity collapse, pass@k would converge toward avg@k as all samples produce near-identical outputs. An extended pass@k analysis across different sampling budgets k is provided in Appendix G.

Why single-agent methods collapse. To understand the root cause of diversity collapse, we analyze TTRL's majority voting mechanism. TTRL assigns pseudo-labels based on the most frequent answer among sampled responses. We track the *majority ratio*—the fraction of samples agreeing with the majority answer—throughout TTRL training. On OLMO3-7B, this ratio rises from 0.57 to 0.85; on Qwen3-14B, from 0.83 to above 0.95 (see Appendix E for full curves). This increasing concentration reveals a fundamental trade-off in self-consistency-based rewards: the mechanism that stabilizes the pseudo-label also suppresses alternative valid reasoning paths. Simultaneously, we track actor entropy during training. TTRL exhibits a steep entropy decline, reflecting the model becoming increasingly deterministic. In contrast, TEMPO maintains substantially higher entropy throughout training, consistent with preserved pass@k scores. By grounding the reward in an externally calibrated verification agent rather than self-consensus, TEMPO sidesteps this trade-off, decoupling reward stability from output diversity.

4.4. Versatility (RQ3)

TEMPO is applicable for general reasoning tasks beyond math. We evaluate OLMO3-7B and Qwen3-8B across BigBenchHard, AGI Eval, ZebraLogic, and GPQA-Diamond. As shown in Table 2, TEMPO achieves substantial gains of +21.4 on BBH and +24.5 on AGI Eval for OLMO3-7B, enabling the 7B model to surpass frontier models like General-Reasoner-7B. On Qwen3-8B, TEMPO still yields consistent improvements on ZebraLogic (+8.2) and GPQA-Diamond (+5.0 Avg@8), confirming that the alternating training design generalizes across domains.

4.5. Ablation (RQ4)

We conduct ablation studies to isolate the contributions of key design choices in TEMPO: (1) *Frozen verification agent*: the critic is trained once on D_L and kept fixed, removing E-step recalibration; (2) *Supervised continuation*: the model continues PPO on D_L without test-time updates; (3) *Labeled data size*: we vary $|D_L|$ to test E-step data efficiency.

As shown in Figure 5, supervised PPO from a converged checkpoint saturates immediately, while TEMPO achieves 15+ point gains over 200 iterations, confirming that only exposure to novel test-time problems can push beyond the RLVR ceiling. Figure 6 shows that a frozen-critic variant initially matches TEMPO but plateaus after ∼100 iterations

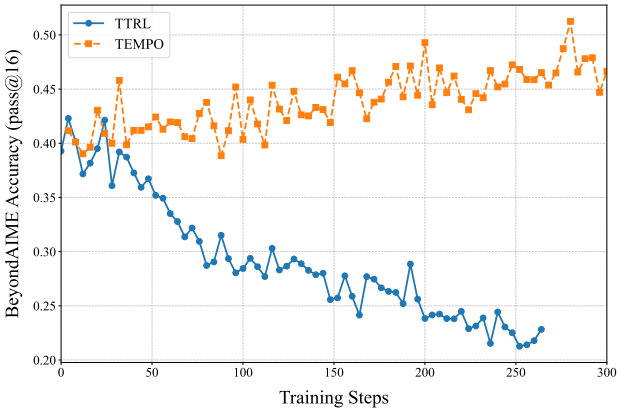

Figure 3. **TEMPO preserves diversity.** TTRL degrades pass@16 while TEMPO steadily improves it.

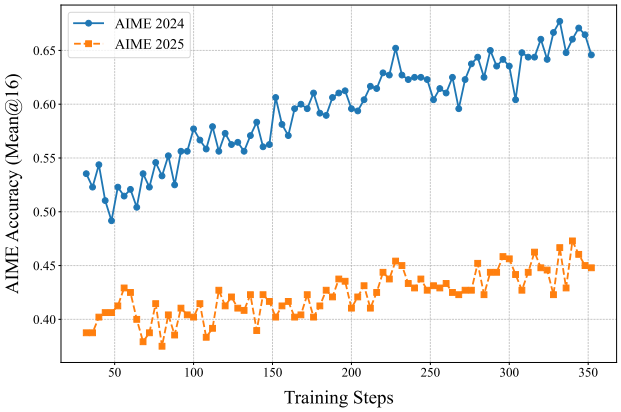

Figure 4. **TEMPO keeps improving.** On Qwen3-14B, performance has not plateaued at 224 steps.

Table 2. **Generalization beyond math.** Avg@1 for BBH, AGI Eval, ZebraLogic, and Avg@8/Pass@8 for GPQA-Diamond. Gray: TTT data; blue: held-out.

	BBH	AGI	Zebra	GPQA-D	
	(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
↪ TEMPO	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
↪ TEMPO	74.2	70.1	33.9	37.2	65.3
Δ	**+4.3**	**+4.4**	**+8.2**	**+5.0**	**+2.9**

as the static critic becomes misaligned with the evolving policy. This validates that E-step recalibration is necessary for sustained self-evolution.

For the data size ablation (3), we find that as few as 1024 labeled samples suffice for reliable verification agent recalibration, while 128 samples lead to overfitting (see Appendix F for detailed curves). This demonstrates that the E-step is data-efficient, making TEMPO practical when large labeled datasets are unavailable.

5. Discussion

The TEMPO framework offers a principled EM interpretation of test-time self-evolution for LRMs. Through this lens, existing TTT methods such as TTRL and EMPO can be understood as single-agent self-evolution variants that execute only the M-step (updating the reasoning agent using self-generated pseudo-labels) while entirely omitting the E-step that grounds the self-evolution loop in external verification. In TTRL, for instance, the auxiliary distribution $q(y|x)$ reduces to a binary indicator based on majority voting, reinforcing the agent's dominant reasoning pattern regardless of correctness—a form of self-reinforcement rather than genuine self-evolution. TEMPO avoids this through a dual-agent architecture: the verification agent provides dynamically calibrated reward signals grounded in external supervision, keeping the ELBO tight throughout the self-evolution process.

To make the EM connection concrete, we analyze TTRL's reward construction. In TTRL, the auxiliary distribu-

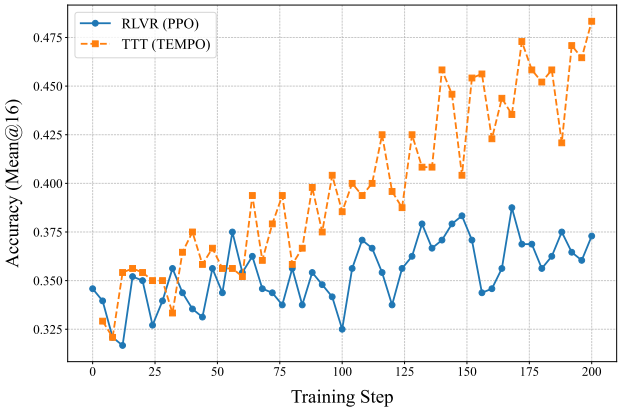

Figure 5. **Superiority of test-time self-evolution.** From converged OLMO3, supervised PPO (blue) saturates immediately; TEMPO (orange) gains 15+ points over 200 iterations.

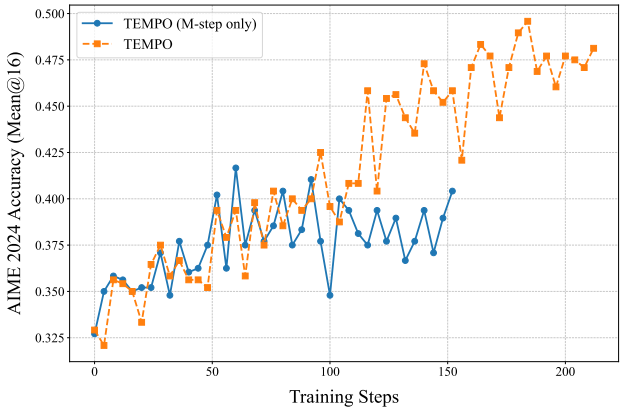

Figure 6. **Necessity of verification agent recalibration.** Frozen critic (blue) plateaus after ∼100 steps; full TEMPO (orange) continues improving.

tion $q(y|x)$, which should approximate the true posterior $P(y|x, \text{Correct})$, is reduced to a binary indicator:

$$q(y|x) \propto \mathbb{1}(y \in \mathcal{Y}_{\text{majority}}) \cdot \pi_{\theta_0}(y|x), \tag{11}$$

where $\mathbb{1}(\cdot)$ assigns unit weight to responses agreeing with the majority and zero to all others. This formulation is self-reinforcing: the consensus set $\mathcal{Y}_{\text{majority}}$ is determined by the current policy, and as the model becomes more confident in particular reasoning patterns, those patterns dominate the consensus, amplifying themselves in subsequent updates. This positive feedback loop—the same model evaluating its own outputs—is the root cause of both performance plateaus and diversity collapse. In contrast, TEMPO replaces the binary consensus indicator with a continuous, quality-aware score $V_\phi(x, y_T)$ from an independently calibrated verification agent, enabling fine-grained differentiation among generated samples and preventing reward drift.

This dual-agent perspective connects to a broader vision for autonomous scientific AI agents. A self-evolving agent operating in open-ended scientific domains must not only generate hypotheses but also verify them against empirical evidence. The reasoning-verification decoupling in TEMPO mirrors this scientific method: the reasoning agent proposes solutions (hypothesis generation), while the verification agent evaluates their correctness against known truths (empirical validation). By interleaving these two processes, the agent achieves sustained improvement without the self-delusion that afflicts single-agent self-training. This design is domain-agnostic: while our experiments focus on mathematical reasoning as a controlled testbed, the dual-agent self-evolution paradigm extends naturally to scientific reasoning tasks in physics, chemistry, and beyond, where verification against known principles can serve the same grounding role as our labeled dataset D_L.[1]

6. Limitations

Despite its advantages as a dual-agent self-evolution framework, TEMPO has several limitations that warrant discussion. First, the alternating E/M-step procedure requires maintaining both a reasoning agent and a verification agent, which increases GPU memory and computational overhead compared to single-agent methods. While our timing analysis (Section C) shows that the verification agent recalibration accounts for only 13.2% of per-step compute with rollout generation dominating the cost, TEMPO introduces approximately 1.3 to 1.6× per-step overhead. This overhead is a natural cost of the dual-agent architecture and may be mitigated through model sharing or distillation in future work. Second, the verification agent recalibration relies on access to a labeled dataset D_L. While our data size ablation (Section 4.5) shows that as few as 1024 labeled samples suffice for effective recalibration, the domain gap between D_L and unlabeled test questions may affect generalization. Developing verification agents that can calibrate from weaker or more diverse forms of supervision is an important direction for more autonomous self-evolving agents.

7. Conclusion

We present TEMPO, a dual-agent self-evolution framework for LRMs that alternates between verification agent recalibration and reasoning agent refinement. By framing test-time training as an EM-style procedure, we identified the missing verification step (E-step) as the key deficiency underlying the performance plateaus and diversity collapse of

[1] In standard EM, the E-step computes the exact posterior $P(y|x, \text{Correct})$ given fixed parameters, which is intractable for LLMs. Our E-step is an operational approximation: we learn a verification agent V_ϕ on labeled data to approximate this posterior. Despite this approximation, the alternating dual-agent procedure preserves the essential EM mechanism that prevents the self-reinforcing reward drift observed in single-agent methods.

prior single-agent self-rewarding methods. Our theoretical perspective unifies existing TTT approaches as incomplete EM variants, and our empirical results demonstrate that TEMPO consistently outperforms baselines across model scales and reasoning domains while preserving output diversity.

Acknowledgement

This work is supported by Shanghai Artificial Intelligence Laboratory. The authors thank the P1 team in Shanghai AI Lab for their extensive support of this work, including computational resources, training recipes and insightful discussions.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning, particularly toward enabling self-evolving AI agents for mathematical reasoning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Agarwal, S., Ahmad, L., Ai, J., Altman, S., Applebaum, A., Arbus, E., Arora, R. K., Bai, Y., Baker, B., Bao, H., et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

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

Chen, J., Cheng, Q., Yu, F., Wan, H., Zhang, Y., Zheng, S., Yao, J., Zhang, Q., He, H., Luo, Y., et al. P1: Mastering physics olympiads with reinforcement learning. *arXiv preprint arXiv:2511.13612*, 2025.

Cui, G., Yuan, L., Wang, Z., Wang, H., Li, W., He, B., Fan, Y., Yu, T., Xu, Q., Chen, W., et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Gao, X., Lin, J., Li, Z., Zhou, A., Li, P., Yang, H., Li, J., Ma, J., Xu, K., Liu, X., Zhang, K., Cai, D., Li, W., Wu, F., Wu, Y., Zhou, J., Yu, H., Chen, X.-H., and Yu, B. Soft Adaptive Policy Optimization. *arXiv:2511.20347*, 2025a. URL https://arxiv.org/abs/2511.20347.

Gao, Z., Chen, L., Luo, H., Zhou, J., and Dai, B. One-shot entropy minimization. *arXiv preprint arXiv:2505.20282*, 2025b.

Grandvalet, Y. and Bengio, Y. Semi-supervised learning by entropy minimization. *Advances in neural information processing systems*, 17, 2004.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jackson, J., Trapani, B., Wang, N., and Zhu, W. Improving composer through real-time rl. https://cursor.com/blog/real-time-rl-for-composer, March 2026. Accessed: 2026-04-12.

Lambert, N., Morrison, J., Pyatkin, V., Huang, S., Ivison, H., Brahman, F., Miranda, L. J. V., Liu, A., Dziri, N., Lyu, S., Gu, Y., Malik, S., Graf, V., Hwang, J. D., Yang, J., Bras, R. L., Tafjord, O., Wilhelm, C., Soldaini, L., Smith, N. A., Wang, Y., Dasigi, P., and Hajishirzi, H. Tulu 3: Pushing frontiers in open language model post-training. 2024.

Lin, B. Y., Le Bras, R., Richardson, K., Sabharwal, A., Poovendran, R., Clark, P., and Choi, Y. Zebralogic: On the scaling limits of llms for logical reasoning. In *International Conference on Machine Learning*, pp. 37889–37905. PMLR, 2025.

Luo, Y., Wang, F., Cheng, Q., Yu, F., Lei, H., Yan, J., Li, C., Chen, J., Zhao, Y., Wan, H., et al. P1-vl: Bridging visual perception and scientific reasoning in physics olympiads. *arXiv preprint arXiv:2602.09443*, 2026.

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

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Shumailov, I., Shumaylov, Z., Zhao, Y., Papernot, N., Anderson, R., and Gal, Y. Ai models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759, 2024.

Su, Y., Li, Y., Liu, N., Jia, K., Yang, X., Foo, C.-S., and Xu, X. 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.

Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q., Chi, E., Zhou, D., 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.

Wang, D., Shelhamer, E., Liu, S., Olshausen, B., and Darrell, T. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=uXl3bZLkr3c.

Wang, Y., Su, S.-R., Zeng, Z., Xu, E., Ren, L., Yang, X., Huang, Z., He, X., Ma, L., Peng, B., Cheng, H., He, P., Chen, W., Wang, S., Du, S. S., and Shen, Y. Thetaevolve: Test-time learning on open problems. *arXiv preprint 2511.23473*, 2025.

Wang, Y., Li, B., Cho, D., Zhang, R., Sui, F., and Grama, A. Sarl: Label-free reinforcement learning by rewarding reasoning topology. *arXiv preprint arXiv:2603.27977*, 2026.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a. URL https://arxiv.org/abs/2505.09388.

Yang, S., Zhu, G., Zheng, X., MA, Y., Chen, Z., Song, B., Wang, W., Zhao, J., Chen, G., and Wang, H. Trapo: A semi-supervised reinforcement learning framework for boosting llm reasoning. *arXiv preprint arXiv:2512.13106*, 2025b.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Fan, T., Liu, G., Liu, L., Liu, X., et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yue, Y., Chen, Z., Lu, R., Zhao, A., Wang, Z., Song, S., and Huang, G. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Zhang, B., Sun, H., Min, Y., Chen, Z., Zhao, W. X., Liu, Z., Wang, Z., Fang, L., and Wen, J.-R. Challenging the boundaries of reasoning: An olympiad-level math benchmark for large language models. *arXiv preprint arXiv:2503.21380*, 2025a. URL https://arxiv.org/abs/2503.21380.

Zhang, Q., Bian, Y., Kong, X., Zhao, P., and Zhang, C. Come: Test-time adaption by conservatively minimizing entropy. In *International Conference on Learning Representations*, 2025b.

Zhang, Q., Wu, H., Zhang, C., Zhao, P., and Bian, Y. Right question is already half the answer: Fully unsupervised llm reasoning incentivization. *Advances in neural information processing systems*, 2025c.

Zhang, Y., Zhang, Z., Guan, H., Cheng, Y., Duan, Y., Wang, C., Wang, Y., Zheng, S., and He, J. No free lunch: Rethinking internal feedback for llm reasoning, 2025d. URL https://arxiv.org/abs/2506.17219.

Zhao, X., Kang, Z., Feng, A., Levine, S., and Song, D. Learning to reason without external rewards. *arXiv preprint arXiv:2505.19590*, 2025.

Zheng, C., Liu, S., Li, M., Chen, X.-H., Yu, B., Gao, C., Dang, K., Liu, Y., Men, R., Yang, A., et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Zhong, W., Cui, R., Guo, Y., Liang, Y., Lu, S., Wang, Y., Saied, A., Chen, W., and Duan, N. Agieval: A human-centric benchmark for evaluating foundation models. In *Findings of the association for computational linguistics: NAACL 2024*, pp. 2299–2314, 2024.

Zuo, Y., Zhang, K., Qu, S., Sheng, L., Zhu, X., Qi, B., Sun, Y., Cui, G., Ding, N., and Zhou, B. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

Zuo, Y., He, B., Liu, Z., Zhao, S., Fu, Z., Yang, J., Zhang, K., Fan, Y., Cui, G., Qian, C., Chen, X., Sun, Y., Lv, X., Zhu, X., Sheng, L., Li, R., ang Gao, H., Zhang, Y., Yuan, L., Liu, Z., Zhou, B., and Ding, N. How far can unsupervised RLVR scale LLM training? In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=VesLZukY5E.

Appendices

A. Broader Impacts

Self-evolving reasoning agents that improve at test time hold significant promise for accelerating scientific discovery, enabling models to autonomously adapt to novel problems in mathematics, physics, and beyond. The dual-agent design of TEMPO, which decouples hypothesis generation from verification, aligns with the scientific method and could facilitate more trustworthy AI-assisted research. However, test-time self-evolution methods also 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). TEMPO's reliance on a fixed labeled calibration set D_L for verification agent grounding provides a partial safeguard, as the verification agent's reward signal remains anchored to a curated reference distribution. However, this does not eliminate the risk entirely if D_L itself is poorly curated. Developing robust verification mechanisms that can detect and resist adversarial manipulation is an important direction for the safe deployment of self-evolving scientific agents.

B. Training Details and Hyperparameters

B.1. Stage 1: RLVR Initialization

Table 3 lists the hyperparameters used to initialize the actor and critic via PPO on labeled data.

Table 3. Hyperparameters for RLVR initialization (Stage 1).

Hyperparameter	OLMO3-7B	Qwen3-14B
Labeled dataset	DAPO-Math-17K	DAPO-Math-17K
Batch size	256	256
Actor learning rate	1×10^{-6}	1×10^{-6}
Critic learning rate	5×10^{-6}	5×10^{-6}
Advantage estimator	GAE	GAE
KL penalty coefficient	0	0
Rollout samples per prompt	8	8
Sampling temperature	1.0	1.0
Max response length	16,384	16,384
Dual-clip ratio	3×10^{-4}, 5×10^{-4}	3×10^{-4}, 5×10^{-4}
Training steps	192	224
GPUs	$4\times$	$8\times$

B.2. Stage 2: Test-time Training

Table 4 lists the hyperparameters specific to the TEMPO alternating procedure. Most settings are shared with Stage 1; we highlight the differences.

Key differences from Stage 1. 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 advantage estimator switches from GAE to the EM token-level formulation described in Section 3.4.

B.3. Infrastructure

All experiments are conducted on $8\times$ 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 (TTT and Qwen3-14B initialization) use 8 GPUs.

Table 4. Hyperparameters for TEMPO test-time training (Stage 2).

Hyperparameter	Value
Actor learning rate	1×10^{-6}
Critic learning rate	1×10^{-6}
E-step frequency	every iteration
Rollout samples per prompt	8
Sampling temperature	1.0
Max response length	16,384
KL penalty coefficient	0
Batch size	256
Total TTT iterations	350
Evaluation frequency	every 4 steps
GPUs	8×

C. Computational Cost

Table 5 compares the per-step wall-clock time of TEMPO against baselines during the test-time training phase (Stage 2). All TTT runs use 8 GPUs. Because TEMPO and baselines differ in total training steps, we report per-step time as the primary comparison rather than total wall-clock.

Table 5. Per-step wall-clock time for test-time training (Stage 2 only, 8 GPUs).

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

Overhead analysis. TEMPO introduces approximately 1.3–1.6× per-step overhead over TTRL. This comes from two sources: (i) the E-step, which performs critic forward and backward passes on $\mathcal{D}_L$ (201s/step on OLMO3-7B, 13.2% of total step time as shown in Figure 7), and (ii) critic forward passes during M-step advantage computation. The dominant cost—rollout generation—is shared across all methods. The RLVR initialization (Stage 1) cost is amortized since it is performed once and shared by all TTT methods that start from the same checkpoint.

Stage 1 reference 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 7 shows the per-step time breakdown for TEMPO on OLMO3-7B. Rollout generation dominates at 1020s/step, while the critic update (E-step) takes 201s/step, accounting for 13.2% of total step time. The remaining time is spent on policy updates and system overhead.

Peak GPU memory. Table 6 reports peak GPU memory allocation (summed across all GPUs). On OLMO3-7B, TEMPO 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.

D. Extended Scalability Analysis

Figure 8 shows the mean@16 accuracy curves over training for both OLMO3-7B and Qwen3-14B on AIME 2024 and AIME 2025. These curves complement Figure 1 by showing the raw, unsmoothed trajectory on both models.

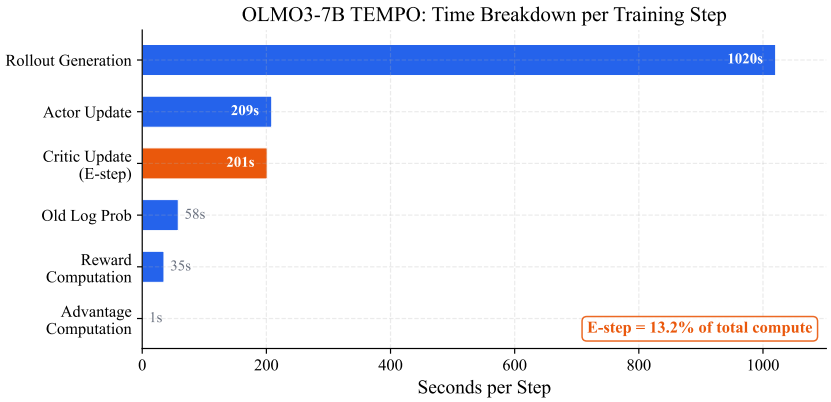

Figure 7. Per-step time breakdown for TEMPO (OLMO3-7B). The E-step (critic update, highlighted in orange) accounts for 13.2% of the total step time. Rollout generation is the dominant cost, shared with all TTT methods.

Table 6. Peak GPU memory allocation (total across 8× H200 GPUs).

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

E. Training Dynamics

E.1. Actor Entropy

Figure 9 compares the actor entropy between TEMPO and TTRL throughout training. A declining entropy indicates the model is becoming more deterministic—collapsing to fewer reasoning patterns. TTRL exhibits a steeper entropy decline, consistent with the diversity collapse observed in pass@k metrics. TEMPO maintains higher entropy throughout training, reflecting its ability to preserve diverse reasoning strategies.

E.2. Critic Calibration Quality

Figure 10 shows the critic's AUROC (area under the ROC curve for predicting response correctness) during TEMPO training. The AUROC remains consistently high (0.85–0.95), confirming that the periodic E-step recalibration keeps the critic well-calibrated as the policy evolves. Without this recalibration, the critic would drift and lose predictive accuracy, as demonstrated by the frozen-critic ablation in Figure 6.

E.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.

Figure 11 tracks this ratio over training for both OLMO3-7B and Qwen3-14B. On OLMO3-7B, the majority ratio rises from approximately 0.57 at the start of training to 0.85 by the end; Qwen3-14B exhibits a similar trend from 0.83 to above 0.95. This increasing concentration is consistent with the declining pass@k scores observed in Table 1: as the model converges toward a single dominant reasoning pattern for each question, the majority vote becomes more reliable as a reward signal, but at the cost of suppressing alternative valid solution paths.

This is not necessarily a flaw in TTRL's design—a higher majority ratio stabilizes the pseudo-label and reduces noise in the reward signal, which can benefit mean accuracy in the short term. However, it does illustrate a fundamental trade-off inherent to self-consistency-based rewards: the same mechanism that makes the reward more stable also drives the model toward lower diversity. TEMPO sidesteps this trade-off by grounding the reward in an externally calibrated critic rather than in the model's own output consensus.

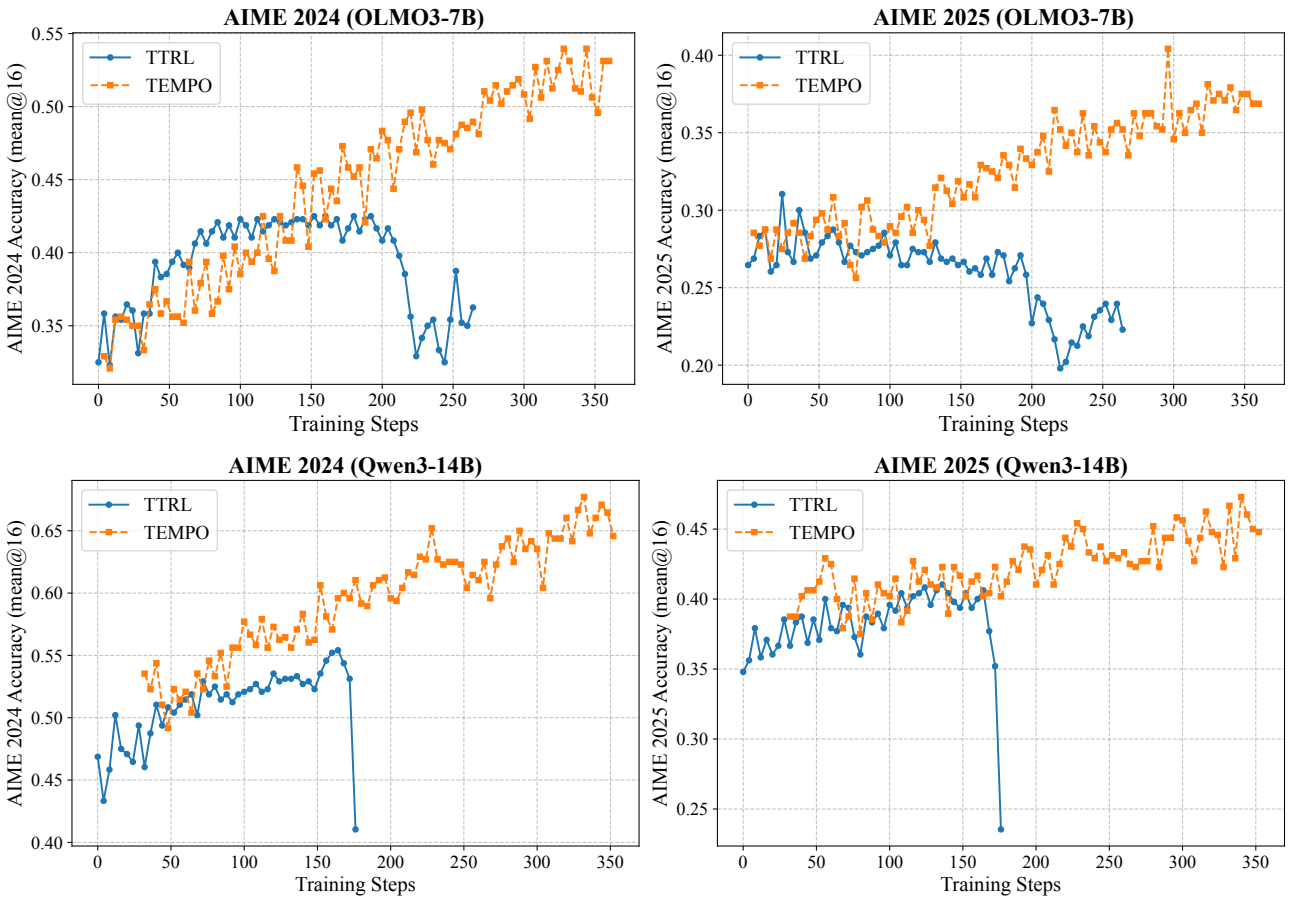

Figure 8. Mean@16 accuracy over training checkpoints. Top: OLMO3-7B. Bottom: Qwen3-14B. TEMPO (orange) consistently outpaces TTRL (blue) and continues improving where TTRL plateaus or degrades. Each point is one evaluation checkpoint (every 4 training steps).

F. Ablation on Labeled Data Size

A key practical question is how large D_L needs to be for the E-step to remain effective. We compare three sizes on OLMO3-7B: 128, 1024, and full. We use **critic AUROC** (discriminative ability) and **explained variance** (generalization gap) as metrics, since the 128- and 1024-shot runs train for fewer steps and have not yet converged in downstream accuracy.

As shown in Figures 12 and 13, with 128 samples AUROC degrades from $\sim$0.96 to below 0.75 by step 80, and test explained variance collapses—a clear signature of overfitting. In contrast, 1024 samples maintain stable AUROC (0.90–0.97) with a narrow train-test gap, matching the full-data behavior. This confirms that $\sim 10^3$ labeled samples suffice for reliable E-step recalibration.

G. Extended Diversity Analysis

The main paper reports pass@16 on Beyond AIME (Figure 3) and pass@8 across benchmarks in Table 1. Here we provide additional diversity analyses.

G.1. Pass@k Curves Across Benchmarks

Figure 14 shows how pass@16 evolves over training steps on AIME 2024 and AIME 2025 for OLMO3-7B, complementing the Beyond AIME result reported in the main paper (Figure 3).

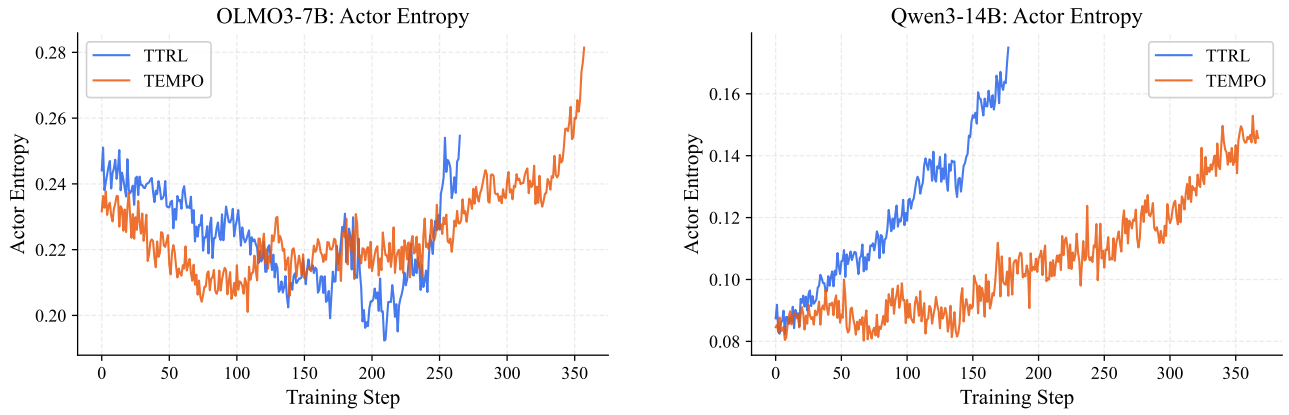

Figure 9. Actor entropy over training steps for OLMO3-7B (left) and Qwen3-14B (right). TTRL (blue) entropy declines more steeply, reflecting diversity collapse. TEMPO (orange) maintains higher entropy, consistent with preserved pass@k.

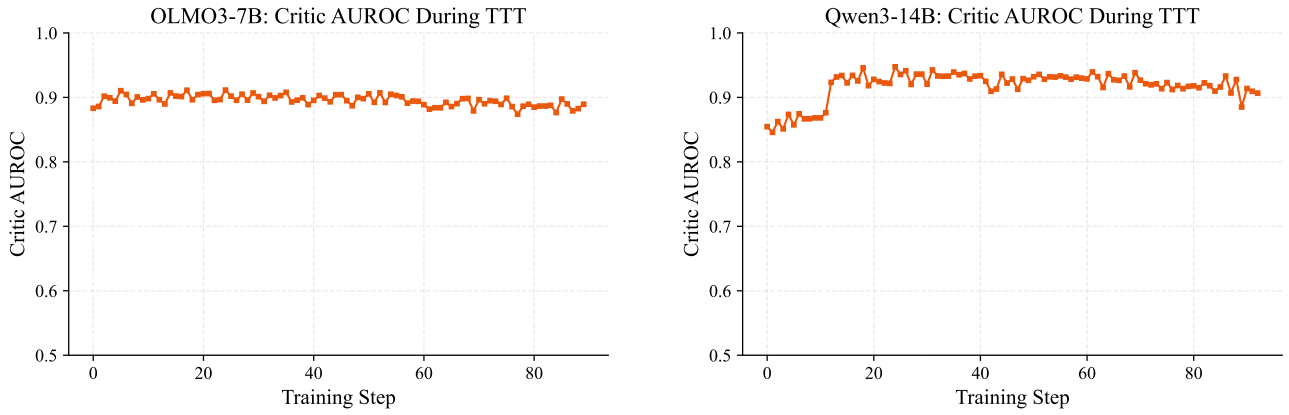

Figure 10. Critic AUROC during TEMPO training for OLMO3-7B (left) and Qwen3-14B (right). The critic maintains high classification accuracy (0.85–0.95) throughout training, confirming that periodic E-step recalibration keeps the reward signal grounded.

G.2. Pass@k at Different k

Table 7 reports pass@k for varying k on AIME 2024, showing that TEMPO's diversity advantage holds across all sampling budgets. Values are from the final checkpoint of each method.

Table 7. 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
TEMPO	69.0	70.6	71.3	72.1
Δ	+18.7	+15.5	+15.0	+15.5

TEMPO's advantage is substantial even at small k: at pass@2 (only two samples), TEMPO already achieves 69.0% compared to TTRL's 50.3%. This indicates that TEMPO's outputs are not only more diverse but also individually higher quality—it does not need many samples to find a correct solution.

H. Case Study

We present two examples from OLMO3-7B where the model answers incorrectly during early TTT but correctly after extended TEMPO training. Both examples are drawn from the same test set used during TTT. Responses are truncated to

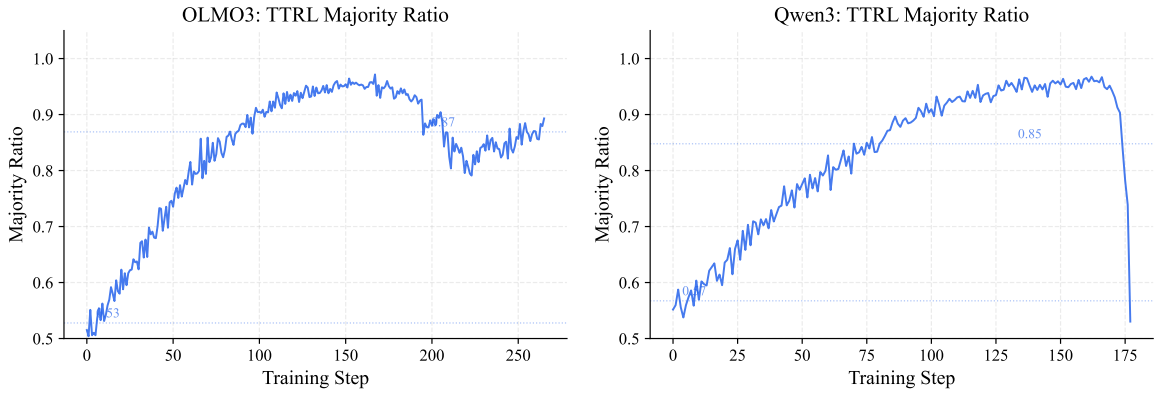

Figure 11. TTRL majority ratio over training steps for OLMO3-7B (left) and Qwen3-14B (right). The increasing ratio indicates that the model progressively concentrates on a single dominant answer per question, which stabilizes the majority-vote reward signal but reduces output diversity.

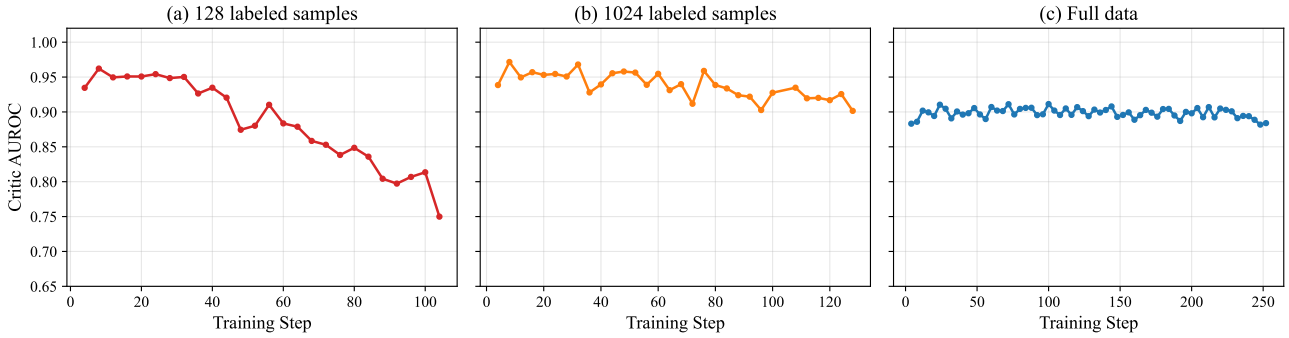

Figure 12. Critic AUROC over training for 128 (left), 1024 (center), and full (right) labeled samples.

show the critical reasoning steps and final answer.

H.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 (early TEMPO, 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.

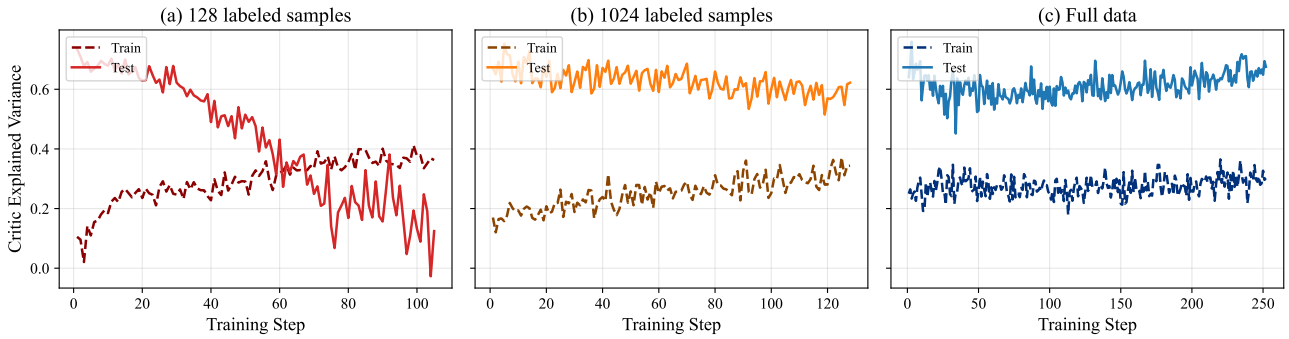

Figure 13. Critic explained variance (train: dashed, test: solid) for 128 (left), 1024 (center), and full (right). The widening 128-shot train-test gap reveals overfitting.

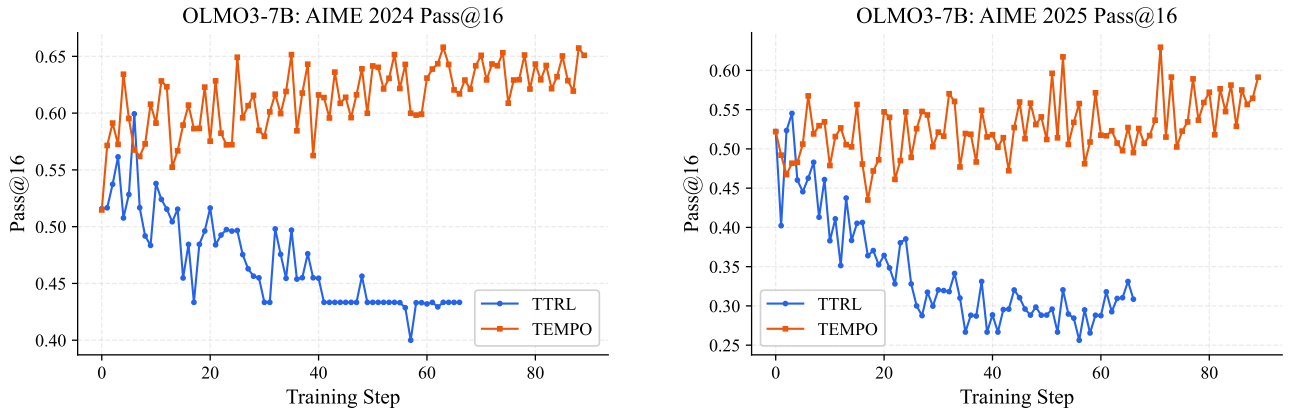

Figure 14. Pass@16 over training steps on AIME 2024 (left) and AIME 2025 (right) for OLMO3-7B. TTRL (blue) diversity degrades as training progresses; TEMPO (orange) maintains and steadily improves pass@16 throughout training, consistent with the Beyond AIME results in the main paper.

After TTT (late TEMPO, correct). After 300+ steps of TEMPO 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: $m + n = 1 + 257 = \boxed{258}$

H.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 (early TEMPO, 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 TEMPO, correct). After TEMPO 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:
$a - b - c = 1/2, \quad -a + b - c = 1/3, \quad -a - b + c = 1/4$
Solving: $a = -13/24$, $b = -7/24$, $c = -1/24$
$|4a + 3b + 2c| = |\frac{-52-21-2}{24}| = 25/8$
$m + n = 25 + 8 = \boxed{33}$

I. 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.