WHERE TO VERIFY, WHERE TO EXPLORE: TEST-TIME REINFORCEMENT LEARNING WITH CONFIDENCE-CONDITIONED VERIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time reinforcement learning (TTRL) has emerged as a promising paradigm for enhancing the complex reasoning abilities of large language models (LLMs) in a completely label-free manner. Despite existing studies focusing on Pass@1 performance, optimizing Pass@k remains under-explored yet critical in label-free settings, which measures generation coverage for sustained exploration. Optimizing Pass@k in label-free setting is highly non-trivial, as directly applying the Pass@k advantage designs effective for RLVR yields unsatisfactory performance. We show that this transfer utilizes samples uniformly across confidence regions, although samples in different regions differ in both exploration value and label reliability. Low-confidence samples carry the strongest exploration signal, but their mostly incorrect majority labels push exploration in the wrong direction; high-confidence samples have reliable labels, but their exploration value is nearly exhausted and their trajectory diversity collapses. To overcome these hurdles, we propose TTRL-CoCoV (**T**est-**T**ime **R**einforcement **L**earning with **Co**nfidence-**Co**nditioned **V**erification), a novel confidence-adaptive framework that expands Pass@k coverage and improves Pass@1 by conditioning the operation applied to each sample on its confidence. For high-confidence samples, it uses their reliable pseudo-labels to supervise the verifier and applies a length-diversity reward to prevent diversity collapse; for low-confidence samples, it delegates pseudo-label selection to the verifier to filter out incorrect pseudo-labels; and for medium-confidence samples, it bypasses verification entirely. In this way, reliable high-confidence supervision improves the verifier, which in turn corrects low-confidence pseudo-labels, allowing generation and verification to co-evolve within a single model. Extensive experiments show that TTRL-CoCoV outperforms competing label-free methods across six benchmarks, with average absolute gains of 10.42 points in Pass@1 and 20.25 points in Pass@16 over TTRL. Anonymous code repository for TTRL-CoCoV: `https://anonymous.4open.science/r/TTRL-CoCoV-C5BC`.

1 INTRODUCTION

Test-time reinforcement learning (TTRL)(Zuo et al., 2025; Liu et al., 2025; Wang et al., 2025a; Zhou et al., 2025) has emerged as a promising paradigm to enhance the reasoning capabilities of large language models (LLMs) in completely label-free settings. Compared to traditional Reinforcement Learning from Verifiable Rewards (RLVR)(Jaech et al., 2024)(Guo et al., 2025), which fundamentally relies on human annotations or oracle-provided labels, TTRL optimizes reasoning policies entirely through self-sampling and internal reward construction. Building upon standard architectures such as Group Relative Policy Optimization (GRPO)(Shao et al., 2024), recent TTRL methods effectively address the scalability bottlenecks of RLVR and achieve single-pass generation (Pass@1) performance that closely approaches fully supervised baselines. Beyond Pass@1 accuracy, maintaining a broad exploratory space is critical to prevent premature convergence during policy optimization. In the RLVR setting, recent studies (Chen et al., 2025b; Walder & Karkhanis, 2025) have demonstrated that explicitly optimizing for Pass@k generation coverage effectively balances exploration and exploitation.

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.

To advance the efficacy of TTRL, current methodologies mainly focus on establishing the reliability of self-generated pseudo-labels. For complex, low-consistency problems, recent sample-level methods employ consensus gating and confidence reweighting to secure label purity and stabilize the optimization process (Yan et al., 2026; Yu et al., 2025b). Other approaches leverage internal self-verification mechanisms or external verifiers to actively filter out spurious trajectories or perform autonomous error correction(Pan et al., 2026; Yan et al., 2026; Liao et al., 2026). Simultaneously, researchers have also recognized the importance of preventing premature convergence—often observed as a dramatic collapse in the standard deviation of response lengths on mastered problems (Chen et al., 2025b; Walder & Karkhanis, 2025). To address this, recent studies have introduced novelty or entropy bonuses to maintain a broad exploratory space in unsupervised TTRL(Liu et al., 2025; Zhou et al., 2025). Collectively, these efforts have advanced TTRL along two directions: managing label reliability and enhancing exploration. Yet, a fundamental dimension remains unexplored: in completely unsupervised TTRL, how to systematically optimize Pass@k generation coverage, which is a critical metric for sustained exploration.

A natural attempt is to transfer Pass@k Training (Chen et al., 2025b), originally developed for supervised RLVR, to TTRL by treating the majority answer as the label, but this recovers only part of the lost coverage. Our analysis (Section 3) shows that this transfer utilizes samples uniformly across confidence regions: the majority confidence governs both the exploration value that the Pass@k objective assigns to a sample and the reliability of its pseudo-label, and the two move in opposite directions, yet all samples are optimized in the same way. This leads to two mismatches. Low-confidence samples provide the strongest exploration signal, but their mostly incorrect majority labels push exploration toward incorrect trajectories. High-confidence samples provide reliable supervision, but their exploration value is nearly exhausted and their trajectory diversity collapses. Moreover, each region lacks what the other offers, yet the two remain uncoupled: the reliable labels of high-confidence samples could supervise a verifier that corrects low-confidence exploration, but this relationship is left unexploited.

To address these mismatches, we propose TTRL-CoCoV (**T**est-**T**ime **R**einforcement **L**earning with **Co**nfidence-**Co**nditioned **V**erification), in which a single shared-weight model serves as both generator and verifier, and the operation applied to each sample is conditioned on its confidence. Low-confidence samples are relabeled through confidence-conditioned verification, which replaces unreliable majority labels with verified candidates and skips samples for which no candidate can be verified. High-confidence samples receive a length-diversity reward that restores exploration under reliable pseudo-labels. The two regions form a feedback loop: high-confidence pseudo-labels provide reliable supervision for the verifier, the resulting verifier refines the low-confidence pseudo-labels used for Pass@k optimization, and the improved generator in turn produces more reliable high-confidence pseudo-labels. The verifier is trained with an asymmetric reward that penalizes false acceptance more heavily, and as both roles are learned by the same model, generation and verification co-evolve within a single label-free training process.

We summarize our primary contributions as follows:

- **Diagnosis of Label-Free Pass@k Dynamics:** We are the first to systematically investigate Pass@k optimization in label-free TTRL. We show that directly transferring Pass@k Training utilizes samples uniformly across confidence regions, although exploration value and label reliability move in opposite directions along majority confidence. As a result, low-confidence samples are explored in the wrong direction, high-confidence samples offer little exploration, and the complementary strengths of the two regions remain uncoupled.
- **A Novel Confidence-conditioned Framework (TTRL-CoCoV):** We propose TTRL-CoCoV, in which a single model serves as both generator and verifier and each sample is optimized according to its confidence. Low-confidence samples are relabeled through verification, high-confidence samples receive a length-diversity reward, and the two regions form a feedback loop in which reliable high-confidence pseudo-labels supervise the verifier that refines low-confidence labels, allowing generation and verification to co-evolve.
- **Comprehensive Evaluations and Performance:** We conduct exhaustive experiments across six complex reasoning benchmarks (e.g., MATH500, AIME24/25) using multiple backbone models (Qwen3-4B/8B). Empirically, TTRL-CoCoV completely reverses the unsupervised Pass@k degradation, yielding massive exploratory generation coverage (average absolute gains of +10.42

points in Pass@1 and +20.25 points in Pass@16), successfully narrowing the gap to the fully supervised GRPO reference.

2 Background and Preliminaries

Problem Setup. Let $\mathcal{D}$ be a dataset of reasoning questions. For a given prompt $\boldsymbol{x} \sim \mathcal{D}$, a target large language model (LLM) policy π_θ generates a response sequence $\boldsymbol{y} = (y_1, \ldots, y_{|\boldsymbol{y}|})$ autoregressively. Specifically, $\pi_\theta(y_t \mid \boldsymbol{x}, \boldsymbol{y}_{<t})$ denotes the probability of generating the t-th token y_t conditioned on the prompt $\boldsymbol{x}$ and prefix $\boldsymbol{y}_{<t}$. During reinforcement learning, responses are sampled via the old policy $\pi_{\theta_{\text{old}}}$ and evaluated against the ground-truth label y^{gt} to yield a reward R, guiding the policy optimization from a pre-trained reference model π_{ref}.

Group Relative Policy Optimization (GRPO). To eliminate the independent value network, GRPO (Shao et al., 2024) estimates the advantage by normalizing intra-group relative rewards. For a given $\boldsymbol{x}$, the policy samples N responses $\{\boldsymbol{y}_i\}_{i=1}^{N}$. Each receives a reward $R_i \in \{R_{\text{neg}}, R_{\text{pos}}\}$ corresponding to an incorrect or correct response based on y^{gt}. The sequence-level advantage $\hat{A}_i$ is then calculated by standardizing these rewards within the sampled group. This relative advantage is subsequently utilized to optimize the policy via a standard clipped surrogate objective. The complete mathematical formulations for the GRPO advantage estimation and policy loss are provided in Appendix B.2.

Pass@k Training. Recent works (Chen et al., 2025b; Walder & Karkhanis, 2025; Wan et al., 2026) directly optimize the Pass@k metric (Kazdan et al., 2025), defined as the probability of generating at least one correct answer within k samples. Specifically, Pass@k Training (Chen et al., 2025b) derives analytical advantages, denoted as $A^{pass@k}$, based on the combinatorial probabilities of sampling correct versus incorrect responses within a group. While these closed-form formulations (detailed in Appendix B.3) effectively encourage exploration, they rely strictly on ground-truth labels to accurately identify positive and negative responses. Extending such Pass@k optimization objectives to the completely label-free Test-Time Reinforcement Learning (Zuo et al., 2025) setting remains under-explored.

3 Analysis

Standard TTRL erodes Pass@k coverage, lowering the six-benchmark average Pass@16 of Qwen3-4B-Base from 64.41 to 51.40 (Table 1). A natural remedy is to transfer Pass@k Training (Chen et al., 2025b) to the label-free setting by using the majority answer as the label, which we denote as Pass@k TTA. However, it recovers only part of the lost coverage, e.g., raising the AIME25 Pass@16 from 14.40 (TTRL) to 24.6, compared with 32.50 for the base model (Appendix D.1).

We attribute this gap to a mismatch that uniform Pass@k optimization cannot resolve. With $N = 32$ rollouts and $k = 4$, the Pass@k advantage of a sample depends only on how many rollouts disagree with its label (Appendix B.3): it decreases with the majority confidence C_{maj} and vanishes once fewer than k rollouts disagree. We refer to this advantage as the sample's *exploration value*. In TTRL, however, C_{maj} also governs label reliability, and the two move in opposite directions (Figure 1).

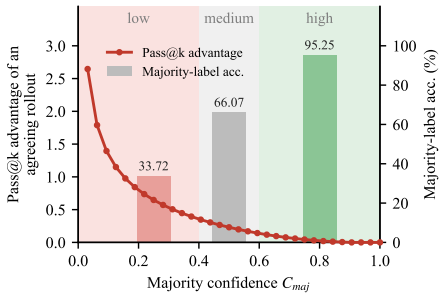


Figure 1: Exploration value (Pass@k advantage) and majority-label accuracy versus C_{maj} ($N = 32$, $k = 4$). Shaded areas mark the three confidence regions.

Low-confidence samples carry the highest exploration value, yet only 33.72% of their majority labels are correct, so Pass@k pushes their exploration in the wrong direction. Discarding them would forfeit the most valuable exploration signal; instead, their labels require a signal more reliable than majority voting, which motivates self-verification.

High-confidence samples have reliable labels (95.25% correct), but their advantage is at most 0.12 and becomes exactly zero once 29 of the 32 rollouts agree. Their exploration value is thus nearly exhausted, and trajectory diversity collapses (Appendix D.3). Because their labels are reliable,

exploration can be safely restored through an additional incentive, which motivates the length-diversity reward. Medium-confidence samples (66.07% correct) retain a moderate advantage, and verification brings no gain over majority voting in this regime (Appendix D.2), so we leave them unchanged.

In short, the limitation of directly transferring Pass@k optimization to label-free TTRL lies in the uniform utilization of samples across confidence regions: samples from different regions differ in both exploration value and label reliability, yet they are optimized in the same way. This leads to two mismatches. In the low-confidence region, samples provide the strongest Pass@k exploration signal, but their unreliable majority labels can direct this exploration toward incorrect trajectories. In the high-confidence region, samples provide reliable supervision, but their Pass@k exploration value is nearly exhausted, leaving little incentive to discover alternative trajectories. More importantly, each region lacks what the other offers, yet the two remain uncoupled: the reliable supervision of high-confidence samples could train a verifier that redirects exploration on low-confidence samples, but Pass@k TTA does not exploit this relationship.

4 Proposed Method

Overview. TTRL-CoCoV uses a single shared-weight model π_θ as both generator and verifier, and addresses the mismatches identified in Section 3 by conditioning the operation applied to each sample on its confidence region. In the low-confidence region, samples are relabeled through confidence-conditioned verification, which replaces unreliable majority labels with verified candidates and skips samples for which no candidate can be verified. In the high-confidence region, samples receive a length-diversity reward to restore exploration under reliable pseudo-labels. Medium-confidence samples retain their majority labels, since verification offers little gain over majority voting in this regime.

The two confidence regions form a feedback loop: high-confidence pseudo-labels provide reliable supervision for the verifier, the resulting verifier refines the low-confidence pseudo-labels used for Pass@k optimization, and the improved generator in turn produces more reliable high-confidence pseudo-labels. Because both roles are learned by the same model, this loop allows generation and verification to co-evolve within a single label-free training process. Since false acceptance can introduce an incorrect pseudo-label into generator optimization whereas false rejection merely discards a sample, we train the verifier with an asymmetric reward that penalizes false acceptance more heavily. Majority voting, Top-K candidate extraction, repeated verification sampling, and the Pass@k advantage follow TTRL (Zuo et al., 2025) and Pass@k Training (Chen et al., 2025b), while Pass@k-to-Pass@1 annealing is applied as a separate subsequent stage.

4.1 Overall Pipeline

Given a question x, the model π_θ first operates in generation mode to sample N reasoning trajectories and extract Top-K candidate answer set. We define the majority confidence as the proportion of the majority answer among these N trajectories, $C_{maj} = \max_a \sum_{i=1}^{N} \mathbb{I}[a_i = a]/N$. Based on C_{maj} and the preset thresholds (τ_{low}, τ_{high}), each sample is classified into one of three training optimization zones (detailed in Section 4.2). If the zone triggers the verification mechanism, the model switches to the verifier role and generates m independent verification trajectories for each Top-K candidate answer, thereby determining the final pseudo-label y^*_{pseudo} and its corresponding confidence weight $\mathcal{C}_{maj}$. Finally, the overall optimization objective is a dynamically weighted sum of the first-stage generation policy loss $\mathcal{L}_{first}$ and the second-stage verification policy loss $\mathcal{L}_{second}$:

$$\mathcal{L}_{total} = \mathcal{L}_{first} + \mathbb{I}_{verify} \cdot \mathcal{C}_{maj} \cdot \left(\frac{1}{K} \sum_{j=1}^{K} \mathcal{L}^{(j)}_{second} \right). \tag{1}$$

where $\mathbb{I}_{verify} \in \{0, 1\}$ is an indicator controlling whether the verification stage is activated, and j indexes the j-th candidate answer. $\mathcal{C}_{maj}$ serves as a dynamic weight to prevent low-consistency, high-noise samples from dominating the overall gradient. Averaging the verification loss over K candidate answers further stabilizes the direction of the verifier update. Algorithm 1 outlines the complete training procedure.

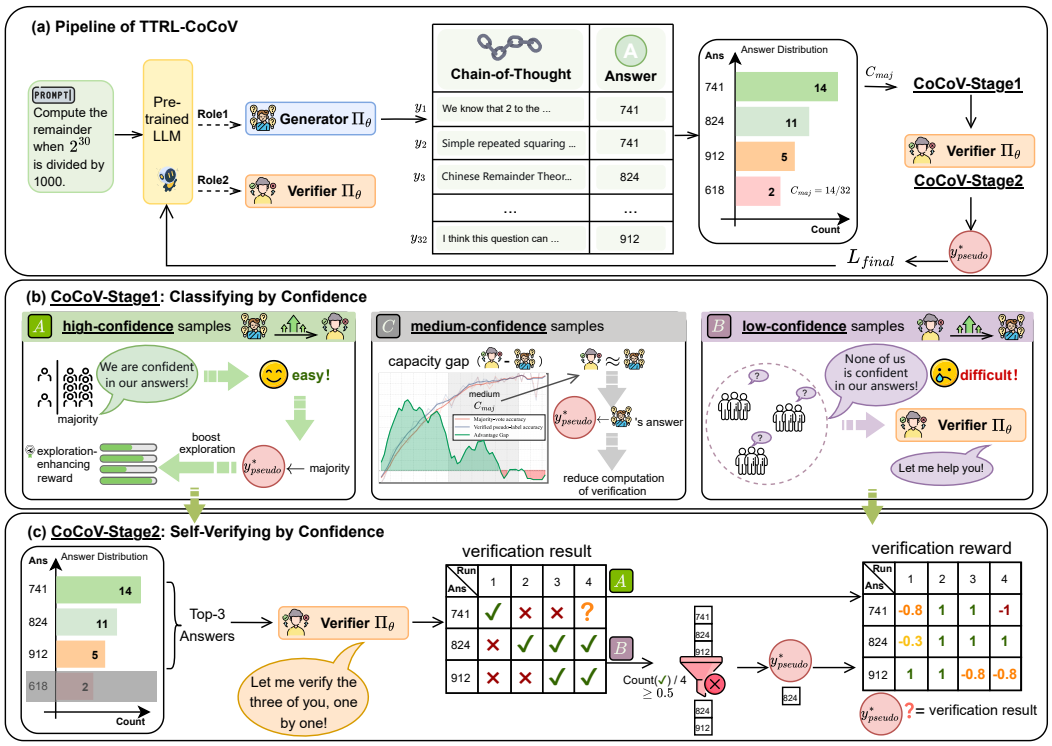


Figure 2: **(a) Overview of TTRL-CoCoV**, which employs a shared-weight model (π_θ) as both generator and verifier to sample trajectories and establish answer consensus. **(b) CoCoV-Stage 1 (Classifying by Confidence):** Based on consensus confidence, high-confidence samples receive an exploration-enhancing reward and activate the verifier for training; low-confidence samples wait for explicit verification; medium-confidence samples bypass the verification stage and update generator with majority-labels. **(c) CoCoV-Stage 2 (Self-Verifying by Confidence):** For high-confidence samples, majority-labels are used to update the verifier; for low-confidence samples, the verifier selects refined pseudo-labels to updates both the generator and the verifier.

4.2 TTRL-CoCoV Stage1: Conditioning By Confidence

Based on the confidence metric C_{maj} which defines three regimes that differ in both pseudo-label reliability and exploration value, each sample is classified into one of three regions.

4.2.1 Region A: High-Confidence Exploration Zone

When $C_{maj} \geq \tau_{high}$, the majority answer is reliable enough to be the pseudo-label y^*_{pseudo}. However, the standard Pass@k signal is almost exhausted because most trajectories agree. We therefore use these samples as clean, label-free supervision for the verifier and add a bounded length-diversity reward to recover alternative trajectories. Verification in this region is not used to correct the label, but to improve the verifier that will later filter low-confidence candidates.

Verifier Optimization In this region, the strong consensus makes the majority answer a reliable pseudo label. We therefore activate the verifier not to revise this label, but to learn from it. Independent verification trajectories are rewarded according to whether their judgments agree with the majority answer. High confidence samples thus bootstrap the verifier through reliable self generated supervision without external labels.

Exploration-Enhance Reward High-confidence samples provide reliable pseudo-labels, making them suitable for both verifier supervision and generator optimization. However, their strong consensus also indicates that the sampled trajectories have become highly concentrated, while the standard Pass@k advantage is nearly exhausted. Consequently, directly optimizing these samples provides

little incentive to discover alternative valid reasoning trajectories. To preserve solution coverage in this regime, we introduce a bounded length-diversity reward:

$$R_{div}(y_i) = \lambda \cdot \min\left(\frac{|l_i - \mu_L|}{\sigma_L + \epsilon},\, C_{max}\right). \tag{2}$$

where l_i is the token length of the current trajectory, μ_L and σ_L are the mean and standard deviation of positive sample lengths in the current batch. We set $\lambda = 0.05$ and $C_{max} = 2$. The final advantage is computed as:

$$\tilde{A}_i = A_i^{pass@k} + \mathbb{I}[r_i = 1] \cdot R_{div}(y_i). \tag{3}$$

To prevent positive advantages from becoming negative after normalization under extreme sample distributions, we apply an advantage-clipping mechanism:

$$A_i^{\text{final}} = \begin{cases} \max\left(\tilde{A}_i,\, \text{Norm}(\tilde{A}_i)\right), & r_i = 1. \\ \text{Norm}(\tilde{A}_i), & r_i = 0. \end{cases} \tag{4}$$

where $\text{Norm}(\cdot)$ denotes within-group mean-variance normalization.

4.2.2 Region B: Low-Confidence Verification Zone

When $C_{maj} < \tau_{low}$, these samples carry the greatest exploration value under Pass@k optimization, but their majority labels are the least reliable. Directly using the majority answer would therefore direct the strongest exploration signal toward incorrect trajectories and reinforce existing errors. To resolve this mismatch, we use the verifier to screen the Top-K candidate answers and identify candidates with sufficient support. If one or more candidates are retained, the candidate with the strongest first-stage consensus is selected as y^*_{pseudo} for joint generator and verifier updates. If no candidate can be reliably verified, we skip the sample to prevent an uncertain pseudo label from determining the exploration direction.

4.2.3 Region C: Medium-Confidence Transition Zone

When $\tau_{low} \leq C_{maj} < \tau_{high}$, majority voting and verification have comparable accuracy, while the sample provides neither clean verifier supervision nor a reliable basis for aggressive exploration. We therefore use the majority answer with the standard Pass@k update and disable verification. This preserves the useful middle regime while avoiding unnecessary inference and extra noise.

4.3 TTRL-CoCoV Stage2: Self-Verifying based on Confidence Conditioning

When self-verification is triggered, the model π_θ verifies the extracted Top-K candidate answers $\{o_k\}_{k=1}^K$ to assess the validity of each and filter out obviously incorrect ones. To quantify the verification results, we introduce the Verification Pass Rate (VPR). For a candidate answer a_k, the model generates m verification trajectories, and its VPR is defined as the proportion of trajectories that output True: $\text{VPR}(o_k) = \frac{1}{m}\sum_{j=1}^{m} \mathbb{I}[\text{verify}^{(j)}(o_k) = \text{True}]$. The computation of the verifier reward $R^{(j,k)}_{\text{second}}$ depends heavily on the classification region, as the source of the supervisory pseudo-label differs across regions:

Region A: Since y_{maj} is highly reliable, it is directly used as the pseudo-label ($y^*_{pseudo} = y_{maj}$), and the verifier's predictions are compared against this pseudo-label to compute rewards.
Region B: To filter the severe pseudo-label noise inherent in uncertain regimes, only candidate answers with $\text{VPR}(o_k) > 0.5$ are admitted into the trusted set $\mathcal{A}_{true}$. We then select the answer with the highest first-stage consensus from $\mathcal{A}_{true}$ as the pseudo-label. The verifier's reward is computed based on whether its prediction matches this dynamically mined label. If $\mathcal{A}_{true} = \emptyset$, we assume that none of the Top-K candidates is likely to be correct and skip the update for this sample.

When computing the verifier's policy loss $\mathcal{L}_{second}$, we adopt the principle of "lenient to false negatives while strict with false positives". The primary function of the verifier is not to directly select the correct answer, but to intercept potentially erroneous reasoning that may hide behind high confidence. We design an asymmetric soft reward matrix for verification trajectories (the precise reward formulation is detailed in Appendix B.4). By assigning a heavier penalty to false positives (FP), the verifier is

shaped into a more stringent screening mechanism. The above rewards are normalized at the prompt level to obtain the final verifier loss $\mathcal{L}_{second}$, which is added to the first-stage policy loss $\mathcal{L}_{first}$ to update θ, forming a closed training loop of generator exploration and verifier filtering.

5 EXPERIMENTS

Models For our main experiments, we evaluate the effectiveness of our method using Qwen3-4B-Base and Qwen3-8B-Base (Yang et al., 2025) as the foundational models. To further assess scalability and generalizability across varying model capacities and architectures, we conduct auxiliary experiments and ablation studies on a diverse suite of models. These include lightweight variants (Qwen3-0.6B and 1.7B Base models (Yang et al., 2025)), a domain-specific model (Qwen2.5-Math-7B (Yang et al., 2024)), and OctoThinker-8B-Hybrid-Base (Wang et al., 2025b), which is continually pre-trained from Llama3.1-8B (Grattafiori et al., 2024).

Datasets For label-free test-time adaptation, we use the **unlabeled** DAPO-14k-MATH corpus, an English subset derived from DAPO-Math-17k (Yu et al., 2025a), comprising 14,000 deduplicated and standardized math reasoning prompts with varying difficulty levels. During adaptation, only the problem prompts are provided to the model; **no ground-truth answers, human annotations, or annotated reasoning trajectories are used for either generator or verifier updates**. All supervision signals are constructed solely from model-generated trajectories. We evaluate the resulting checkpoint, without further adaptation, on six held-out benchmarks: four mathematical reasoning benchmarks, MATH-500 (Hendrycks et al., 2021), AIME24, AIME25, and AMC (Li et al., 2024), and two benchmarks for broader capabilities, GPQA-Diamond (Rein et al., 2023) and the DAPO-14k-MATH-test set (Yu et al., 2025a). No prompts or ground-truth answers from these evaluation benchmarks are accessed during adaptation.

Baselines We compare TTRL-CoCoV with four representative label-free test-time reinforcement learning methods. (1) **TTRL** (Zuo et al., 2025) constructs pseudo-labels through majority voting over self-sampled trajectories and uses them as reward signals for policy optimization. (2) **SCRL** (Yan et al., 2026) selectively constructs positive or negative pseudo-labels according to generation consensus and uncertainty. (3) **Co-Rewarding-III** (Zhang et al., 2025c) improves reward reliability through multi-view cross-validation and an Exponential Moving Average (EMA) teacher. (4) **CoVerRL** (Pan et al., 2026) jointly optimizes generation and verification within a shared-weight generator–verifier co-evolution framework.

Detailed experiment setup are in Appendix C.3.

5.1 EXPERIMENTAL RESULTS

TTRL-CoCoV consistently improves both single-sample accuracy and multi-sample generation coverage. As shown in Table 1, standard TTRL substantially improves Pass@1 on several benchmarks but often sacrifices Pass@16, indicating a contraction of the model's exploratory space. For example, on AIME25 with Qwen3-4B, Pass@16 drops from 32.50% for the base model to 14.40% after TTRL training. In contrast, TTRL-CoCoV reverses this degradation while simultaneously improving Pass@1. Averaged across all six benchmarks, our method improves over TTRL by 10.42 and 20.25 points in Pass@1 and Pass@16, respectively, on Qwen3-4B, with corresponding gains of 6.58 and 14.13 points on Qwen3-8B.

TTRL-CoCoV jointly improves generator and verifier capabilities during training. Tracking the verifier's internal dynamics in Fig. 3, we observe a steady increase in verification accuracy together with a sustained decline in verification errors. As the generator produces increasingly informative trajectories, they provide supervision for improving the verifier; the improved verifier, in turn, filters unreliable pseudo-labels more effectively for subsequent generator updates. This closed feedback loop enables the two roles to co-evolve through TTRL-CoCoV training.

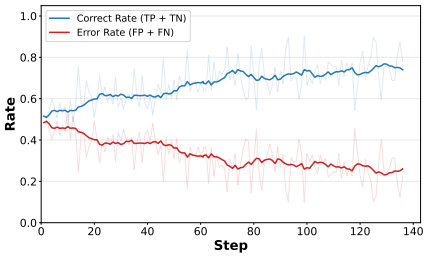


Figure 3: Training dynamics of Verifier: validation correct rate increases while error rate declines.

Table 1: Main experimental results on six reasoning benchmarks. Each cell reports *pass@1 / pass@16*, averaged over four seeds (standard deviations are reported in Table 5). **Ours** refers to TTRL-CoCoV. Δ rows show gains of Ours over TTRL: **absolute** (top) and *relative %* (bottom). †Uses labeled reward data; all other methods are *label-free*.

Method	AIME24	MATH500	AIME25	AMC	GPQA	DAPO	Avg
Qwen3-4B-Base							
Base	8.02 / 26.70	67.20 / 91.75	6.82 / 32.50	40.10 / 76.51	34.70 / 90.00	28.90 / 69.00	30.96 / 64.41
GRPO†	22.20 / 45.60	81.40 / 92.70	21.10 / 38.90	59.00 / 81.10	51.80 / 78.80	45.70 / 82.40	46.87 / 69.92
TTRL	8.90 / 21.10	72.80 / 86.10	3.30 / 14.40	41.80 / 63.50	36.00 / 58.00	30.60 / 65.30	32.23 / 51.40
SCRL	11.30 / 22.50	76.00 / 88.30	9.70 / 30.83	43.90 / 71.40	35.50 / 79.31	34.40 / 59.59	35.13 / 58.66
Co-Rewarding-III	16.10 / 25.80	77.60 / 86.60	8.30 / 20.00	49.70 / 65.70	35.49 / 67.42	39.90 / 59.20	37.85 / 54.12
CoVerRL	10.73 / 26.70	75.31 / 90.80	8.20 / 34.20	46.40 / 75.00	35.23 / 83.90	33.70 / 66.80	34.93 / 62.90
Ours	**20.40 / 44.20**	**81.40 / 94.40**	**17.90 / 45.00**	**53.50 / 81.90**	**38.20 / 86.40**	**44.50 / 78.00**	**42.65 / 71.65**
Δ (Ours – TTRL)	**+11.5 / +23.1** *+129% / +109%*	**+8.6 / +8.3** *+12% / +10%*	**+14.6 / +30.6** *+442% / +213%*	**+11.7 / +18.4** *+28% / +29%*	**+2.2 / +28.4** *+6% / +49%*	**+13.9 / +12.7** *+45% / +19%*	**+10.42 / +20.25** *+32% / +39%*
Qwen3-8B-Base							
Base	10.70 / 36.70	63.83 / 91.90	8.00 / 25.83	38.50 / 78.61	35.22 / 87.60	29.33 / 69.95	30.93 / 65.10
GRPO†	26.60 / 48.33	85.70 / 94.60	21.90 / 44.20	63.54 / 85.24	48.00 / 91.50	55.33 / 83.00	50.18 / 74.48
TTRL	14.11 / 29.20	76.84 / 87.95	10.70 / 23.33	50.55 / 72.60	39.21 / 80.60	38.70 / 66.45	38.35 / 60.02
SCRL	14.70 / 33.33	77.82 / 91.80	13.90 / 28.33	50.23 / 81.33	37.90 / 86.70	38.12 / 72.25	38.78 / 65.62
Co-Rewarding-III	15.50 / 31.70	80.15 / 89.25	12.34 / 23.33	55.90 / 72.30	41.10 / 76.02	44.65 / 65.85	41.61 / 59.74
CoVerRL	16.00 / 38.33	78.30 / 92.30	12.40 / 28.33	50.60 / 78.92	38.70 / 85.03	39.10 / 71.60	39.18 / 65.75
Ours	**23.00 / 51.70**	**82.62 / 95.40**	**17.30 / 42.50**	**57.74 / 86.45**	**43.65 / 91.00**	**45.30 / 77.85**	**44.94 / 74.15**
Δ (Ours – TTRL)	**+8.9 / +22.5** *+63% / +77%*	**+5.8 / +7.5** *+8% / +8%*	**+6.6 / +19.2** *+62% / +82%*	**+7.2 / +13.9** *+14% / +19%*	**+4.4 / +10.4** *+11% / +13%*	**+6.6 / +11.4** *+17% / +17%*	**+6.58 / +14.13** *+17% / +24%*

TTRL-CoCoV substantially narrows the gap to fully supervised GRPO. Despite using no ground-truth labels during adaptation, TTRL-CoCoV approaches the performance of GRPO trained with labeled rewards. On Qwen3-4B, its average Pass@16 reaches 71.65%, exceeding the fully supervised GRPO reference of 69.92%. TTRL-CoCoV further surpasses GRPO on several individual benchmarks, including AIME25 in Pass@16 (45.00 vs. 38.90), while matching its Pass@1 on MATH500 (81.40). These results demonstrate that reliable self-generated supervision can recover a large fraction of the gains otherwise obtained from labeled rewards.

Policy annealing further improves Pass@1. We further examine whether the exploration preserved by TTRL-CoCoV can support subsequent exploitation. Following the Pass@k-to-Pass@1 annealing schedule of prior Pass@k Training (Chen et al., 2025b), we remove R_{div} and shift optimization toward Pass@1. As shown in Table 2, the annealed variant consistently improves over TTRL-CoCoV on all three evaluated benchmarks. This result shows that preserving exploration during TTRL provides a strong basis for subsequent Pass@1 optimization. We report annealing as an optional post-training extension rather than part of the core TTRL-CoCoV framework.

Table 2: Pass@1 after policy annealing. * denotes TTRL-CoCoV followed by a Pass@1 annealing stage. Δ is relative to Ours.

Method	AIME24	MATH500	AMC	Avg
Ours	20.4	81.4	53.5	51.8
Ours*	**23.5**	**82.8**	**54.9**	**53.7**
Δ (Ours* - Ours)	**+3.1**	**+1.4**	**+1.4**	**+1.9**

6 Analysis and Discussions

We next examine the three key effects targeted by TTRL-CoCoV: improving the reliability of self-generated supervision, sustaining generator–verifier co-evolution, and preserving exploration across confidence regimes. We then examine whether the resulting gains can be explained by additional computation.

Q1: How reliable is confidence-conditioned self-supervision? To examine the quality of the self-generated training signals, we track reward accuracy and pseudo-label accuracy throughout training (Fig. 4, left and middle). While standard TTRL suffers from substantial reward degradation in later

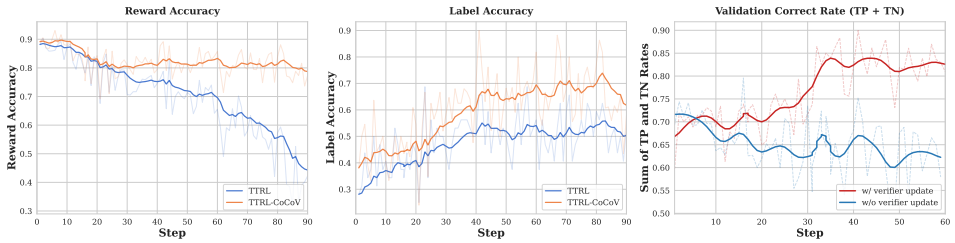


Figure 4: Training and internal verification dynamics of TTRL-CoCoV. **(Left & Middle):** Reward Accuracy and Label Accuracy. While standard TTRL suffers from late-stage pseudo-label collapse and confirmation bias, TTRL-CoCoV maintains highly stable reward accuracy (> 0.8) and smoothly increasing label accuracy. **(Right):** Validation Correction Rate demonstrating co-evolution. Under joint updates, the correct verification rate steadily climbs, whereas a static verifier suffers from capability mismatch and degrades.

training, TTRL-CoCoV maintains reward accuracy above 0.8 and steadily improves pseudo-label quality. Moreover, pseudo-label correctness increases sharply with majority confidence, reaching 97.07% in the high-confidence region on MATH500. These results show that confidence-conditioned routing separates supervision of different reliability levels and prevents unreliable pseudo-labels from dominating policy optimization. Detailed analyses are provided in Appendix F.1.

Q2: How does the co-evolution mechanism in TTRL-CoCoV affect training? We test whether the reliable supervision provided by high-confidence samples can continuously improve the verifier as the generator evolves. We compare TTRL-CoCoV with a frozen-verifier variant and analyze their internal verification dynamics (Fig. 4, right). Without co-evolution, the static verifier becomes increasingly mismatched with the evolving generator and its verification accuracy progressively degrades. Joint updates eliminate this mismatch: high-confidence generator outputs continuously refine the verifier, while the improved verifier provides cleaner supervision for subsequent generator updates. Freezing the verifier causes substantial drops in both Pass@1 and Pass@16, demonstrating that generator–verifier co-evolution is a key component of TTRL-CoCoV. Detailed results are provided in Appendix F.2.

Q3: Does TTRL-CoCoV preserve exploration? We evaluate exploration through embedding-based semantic diversity on MATH500, together with the corresponding change in response length. Compared with TTRL, TTRL-CoCoV generates longer responses while substantially improving semantic diversity, both over all trajectories and over correct trajectories only. The improvement among correct trajectories shows that the increased response length is accompanied by richer reasoning variation rather than merely producing longer outputs. These results demonstrate that TTRL-CoCoV maintains exploration even in regimes where the original Pass@k signal becomes weak. Detailed analyses are provided in Appendix F.3.

Q4: Are the gains of TTRL-CoCoV simply due to increased computation? TTRL-CoCoV introduces additional computation through its verification stage. To isolate this factor, we construct a FLOPs-matched TTRL baseline by allocating a larger rollout budget to standard TTRL. Despite consuming slightly more FLOPs than TTRL-CoCoV, the matched baseline still underperforms our method by 4.92 Pass@1 and 13.94 Pass@16 points on average. This control rules out additional computation alone as the source of the improvement and shows that confidence-conditioned verification uses the extra computation substantially more effectively than simply increasing the sampling budget. Detailed computational profiling is provided in Appendix F.4.

7 Conclusion

We showed that directly transferring Pass@k Training to label-free TTRL utilizes samples uniformly across confidence regions, although exploration value and label reliability move in opposite directions along majority confidence. TTRL-CoCoV addresses this by verifying low-confidence labels, restoring exploration on high-confidence samples, and coupling the two regions through a co-evolving verifier. It reverses the Pass@k degradation of standard TTRL while also improving Pass@1, suggesting that label-free RL benefits from treating samples according to the reliability of their self-generated supervision. Since our routing uses consensus as a proxy for correctness, adaptive confidence thresholds and process-level verification are promising directions for future work.

AI USE STATEMENT

In this work, we used generative AI tools to assist with writing, including editing parts of the manuscript and improving readability and grammar. Generative AI tools were not used for formulating research ideas, designing algorithms, deriving mathematical formulations, or executing experiments. All text, code, and experimental results have been thoroughly reviewed and verified by the authors, who take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

To ensure full reproducibility of our empirical results, complete experimental details—including model architectures, hyperparameter configurations, routing thresholds (τ_{low}, τ_{high}), sampling temperatures, and prompt templates—are fully specified in Section 5 and Appendix C. Furthermore, an anonymized code repository providing the complete training pipeline, execution scripts, and evaluation routines for TTRL-CoCoV is publicly accessible at `https://anonymous.4open.science/r/TTRL-CoCoV-C5BC`.

ETHICS STATEMENT

This work investigates test-time reinforcement learning to enhance mathematical and scientific reasoning capabilities in large language models using open, standard academic benchmarks (e.g., MATH500, AIME, AMC). It does not involve human subjects, personal sensitive data, or harmful real-world applications, and strictly adheres to the ICLR Code of Ethics.

REFERENCES

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning. *arXiv preprint arXiv:2412.09078*, 2024.

Jiaqi Chen, Bang Zhang, Ruotian Ma, Peisong Wang, Xiaodan Liang, Zhaopeng Tu, Xiaolong Li, and Kwan-Yee K Wong. Spc: Evolving self-play critic via adversarial games for llm reasoning. *arXiv preprint arXiv:2504.19162*, 2025a.

Yuxin Chen, Yu Wang, Yi Zhang, Ziang Ye, Zhengzhou Cai, Yaorui Shi, Qi Gu, Hui Su, Xunliang Cai, Xiang Wang, et al. Learning to self-verify makes language models better reasoners. *arXiv preprint arXiv:2602.07594*, 2026.

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. Pass@ k training for adaptively balancing exploration and exploitation of large reasoning models. *arXiv preprint arXiv:2508.10751*, 2025b.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-zero: Self-evolving reasoning llm from zero data. *arXiv preprint arXiv:2508.05004*, 2025.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Joshua Kazdan, Rylan Schaeffer, Youssef Allouah, Colin Sullivan, Kyssen Yu, Noam Levi, and Sanmi Koyejo. Efficient prediction of pass@k scaling in large language models. *arXiv preprint arXiv:2510.05197*, 2025.

Ly Tran Ho Khanh, Dongxuan Zhu, Man-Chung Yue, and Viet Anh Nguyen. Test-time diverse reasoning by riemannian activation steering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 31429–31437, 2026.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024.

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

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The twelfth international conference on learning representations*, 2023.

Jia Liu, ChangYi He, YingQiao Lin, MingMin Yang, FeiYang Shen, and ShaoGuo Liu. Ettrl: Balancing exploration and exploitation in llm test-time reinforcement learning via entropy mechanism. *arXiv preprint arXiv:2508.11356*, 2025.

Teng Pan, Yuchen Yan, Zixuan Wang, Ruiqing Zhang, Gaiyang Han, Wanqi Zhang, Weiming Lu, Jun Xiao, and Yongliang Shen. Coverrl: Breaking the consensus trap in label-free reasoning via generator-verifier co-evolution. *arXiv preprint arXiv:2603.17775*, 2026.

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. *arXiv preprint arXiv:2311.12022*, 2023.

Sheikh Shafayat, Fahim Tajwar, Ruslan Salakhutdinov, Jeff Schneider, and Andrea Zanette. Can large reasoning models self-train? *arXiv preprint arXiv:2505.21444*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yuda Song, Julia Kempe, and Remi Munos. Outcome-based exploration for llm reasoning. *arXiv preprint arXiv:2509.06941*, 2025.

Christian Walder and Deep Karkhanis. Pass@ k policy optimization: Solving harder reinforcement learning problems. *arXiv preprint arXiv:2505.15201*, 2025.

Zhongwei Wan, Yun Shen, Zhihao Dou, Donghao Zhou, Yu Zhang, Xin Wang, Hui Shen, Jing Xiong, Chaofan Tao, Zixuan Zhong, et al. Dsdr: Dual-scale diversity regularization for exploration in llm reasoning. *arXiv preprint arXiv:2602.19895*, 2026.

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.

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. Octothinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint arXiv:2506.20512*, 2025b.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 2550–2575, 2023.

Jianghao Wu, Yasmeen George, Jin Ye, Yicheng Wu, Daniel F Schmidt, and Jianfei Cai. Spine: Token-selective test-time reinforcement learning with entropy-band regularization. *arXiv preprint arXiv:2511.17938*, 2025.

Dong Yan, Jian Liang, Yanbo Wang, Shuo Lu, Ran He, and Tieniu Tan. What if consensus lies? selective-complementary reinforcement learning at test time. *arXiv preprint arXiv:2603.19880*, 2026.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

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*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale, 2025. *URL https://arxiv. org/abs/2503.14476*, 1:2, 2025a.

Zhaoning Yu, Will Su, Leitian Tao, Haozhu Wang, Aashu Singh, Hanchao Yu, Jianyu Wang, Hongyang Gao, Weizhe Yuan, Jason Weston, et al. Restrain: From spurious votes to signals–self-driven rl with self-penalization. *arXiv preprint arXiv:2510.02172*, 2025b.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Fuxiang Zhang, Jiacheng Xu, Chaojie Wang, Ce Cui, Yang Liu, and Bo An. Incentivizing llms to self-verify their answers. *arXiv preprint arXiv:2506.01369*, 2025a.

Zhengxin Zhang, Chengyu Huang, Aochong Oliver Li, and Claire Cardie. Better llm reasoning via dual-play. *arXiv preprint arXiv:2511.11881*, 2025b.

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*, 2025c.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025.

Yujun Zhou, Zhenwen Liang, Haolin Liu, Wenhao Yu, Kishan Panaganti, Linfeng Song, Dian Yu, Xiangliang Zhang, Haitao Mi, and Dong Yu. Evolving language models without labels: Majority drives selection, novelty promotes variation. *arXiv preprint arXiv:2509.15194*, 2025.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, et al. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

A Related Work

Label-free test-time reinforcement learning. STaR (Zelikman et al., 2022) and SRLMs (Yuan et al., 2024) established the foundation of annotation-free self-improvement. Building on this, TTRL (Zuo et al., 2025) formalized majority-vote consensus over self-sampled rollouts as a general unsupervised fine-tuning paradigm. Subsequent works further strengthened this framework along the dimension of label reliability: SRT (Shafayat et al., 2025) via consistency-based filtering, DARE (Du et al., 2026) via distributional reward estimation, RESTRAIN (Yu et al., 2025b) via self-penalization, SCRL (Yan et al., 2026) via complementary positive-negative labeling, and Co-rewarding (Zhang et al., 2025c) via cross-view supervision. Our work complements this line by introducing confidence stratification to jointly address pseudo-label quality and trajectory diversity within a unified framework.

Synergistic generation and verification. A related line of research uses the asymmetry that verification is easier than generation to construct reliable training signals without external annotation. Early verifier-based methods ((Lightman et al., 2023),(Weng et al., 2023)) relied on dedicated reward models, while more recent work unifies generation and verification within a single model through self-play, adversarial games, or cooperative RL objectives ((Zhao et al., 2025),(Huang et al., 2025),(Chen et al., 2025a),(Zhang et al., 2025b)). Self-Harmony (Wang et al., 2025a), Incentivizing LLMs to Self-Verify (Zhang et al., 2025a), and Learning to Self-Verify (Chen et al., 2026) further investigate conditions under which self-verification yields reliable signals, focusing on training stability, distribution alignment, and over-verification suppression respectively. CoVerRL (Pan et al., 2026) represents a particularly relevant instantiation, bootstrapping both roles through shared weights in a co-evolutionary TTRL loop. Our work extends this direction by introducing confidence-conditioned verification allocation.

Exploration and diversity in LLM reasoning. In standard training-time RL, methods such as Outcome-based Exploration (Song et al., 2025), PKPO (Walder & Karkhanis, 2025), Pass@k Training (Chen et al., 2025b), and DSDR (Wan et al., 2026) promote diverse solution strategies with access to ground-truth labels. At pure inference time, FoT (Bi et al., 2024) and SPREAD (Khanh et al., 2026) improve output diversity without parameter updates. In the label-free test-time RL setting, SPINE (Wu et al., 2025) and Evol-RL (Zhou et al., 2025) explore diversity-aware objectives, representing early steps toward sustained exploration under unsupervised conditions. Our work builds on this direction by explicitly studying diversity collapse as a distinct failure mode and proposing a confidence-conditioned remedy.

B METHOD DETAILS

B.1 ALGORITHMIC PSEUDOCODE OF TTRL-COCOV

This section presents the algorithmic pseudocode for the TTRL-CoCoV training loop, detailing the conditional verification process and trajectory generation based on majority confidence.

Algorithm 1 TTRL-CoCoV Training Loop

Require: Pretrained model π_θ, dataset $\mathcal{D}$, thresholds τ_{high}, τ_{low}, candidate size K, verification attempts m

for each prompt x in $\mathcal{D}$ **do**
  Generate N reasoning trajectories $\{y_1, \ldots, y_N\} \sim \pi_\theta(\cdot|x)$
  Compute per-prompt majority confidence C_{maj} and extract Top-K candidates $\{o_k\}_{k=1}^{K}$
  if $C_{maj} \geq \tau_{high}$ **then** ▷ High-Confidence
    $\mathbb{I}_{verify} \leftarrow 1$; $y^* \leftarrow y_{maj}$
    Calculate $\mathcal{L}_{first}$ with Length-Diversity Reward R_{div}
  else if $\tau_{low} \leq C_{maj} < \tau_{high}$ **then** ▷ Mid-Confidence
    $\mathbb{I}_{verify} \leftarrow 0$; $y^* \leftarrow y_{maj}$
    Calculate baseline $\mathcal{L}_{first}$ (No verification)
  else ▷ Low-Confidence
    $\mathbb{I}_{verify} \leftarrow 1$
  end if
  if $\mathbb{I}_{verify} = 1$ **then**
    Generate m verification trajectories per candidate
    Compute $\mathrm{VPR}(o_k)$ for Top-K candidates
    if $C_{maj} < \tau_{low}$ **then** ▷ Relabel using verification
      $\mathcal{A}_{true} \leftarrow \{o_k \mid \mathrm{VPR}(o_k) > 0.5\}$
      if $\mathcal{A}_{true} \neq \emptyset$ **then**
        $y^* \leftarrow$ candidate in $\mathcal{A}_{true}$ with highest first-stage consensus
        Calculate $\mathcal{L}_{first}$ using y^*
      else
        $\mathcal{L}_{total} \leftarrow 0$; **Continue** ▷ Skip update
      end if
    end if
    Compute $\mathcal{L}_{second}$ using Asymmetric Rewards matrix
  end if
  $\mathcal{L}_{total} \leftarrow \mathcal{L}_{first} + \mathbb{I}_{verify} \cdot C_{maj} \cdot \mathrm{Mean}(\mathcal{L}_{second})$
  Update parameters θ using gradients $\nabla_\theta \mathcal{L}_{total}$
end for

B.2 MATHEMATICAL FORMULATION OF GRPO

In the section 2, we briefly introduce Group Relative Policy Optimization (GRPO) (Shao et al., 2024). Here, we provide the complete mathematical formulations. For a given input $\boldsymbol{x}$, the policy samples N responses $\{\boldsymbol{y}_i\}_{i=1}^{N}$. Each response receives a reward R_i. The sequence-level advantage $\hat{A}_i$ is estimated by normalizing the rewards within the sampled group:

$$\hat{A}_i = \frac{R_i - \text{mean}(\{R_j\}_{j=1}^{N})}{\text{std}(\{R_j\}_{j=1}^{N})}.$$

Using this normalized advantage, GRPO optimizes the policy π_θ via the standard clipped surrogate objective, formulated as:

$$\mathcal{J}(\theta) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D},\boldsymbol{y}\sim\pi_{\theta_{\text{old}}}}\left[\frac{1}{|\boldsymbol{y}|}\sum_{t=1}^{|\boldsymbol{y}|}\min\left(\frac{\pi_\theta(y_t \mid \boldsymbol{x}, \boldsymbol{y}_{<t})}{\pi_{\theta_{\text{old}}}(y_t \mid \boldsymbol{x}, \boldsymbol{y}_{<t})}\hat{A}_i,\right.\right.$$
$$\left.\left.\text{clip}\left(\frac{\pi_\theta(y_t \mid \boldsymbol{x}, \boldsymbol{y}_{<t})}{\pi_{\theta_{\text{old}}}(y_t \mid \boldsymbol{x}, \boldsymbol{y}_{<t})}, 1-\epsilon, 1+\epsilon\right)\hat{A}_i\right)\right]$$

This formulation allows the model to optimize its reasoning policy efficiently using internal relative rankings without relying on an external, independently trained value network.

B.3 Analytical Advantage for Pass@k Training

In Section 2, we refer to the analytical advantage $A^{pass@k}$ derived for Pass@k Training (Chen et al., 2025b). For completeness, we provide the exact closed-form formulations here. Given a group of N responses containing N_{neg} incorrect samples, the expected group reward is $\bar{R}^{\text{group}} = 1 - \binom{N_{\text{neg}}}{k} / \binom{N}{k}$ and its standard deviation is $\sigma^{\text{group}} = \sqrt{\bar{R}^{\text{group}}(1 - \bar{R}^{\text{group}})}$. The closed-form advantages for positive and negative responses are defined as:

$$\hat{A}_{\text{pos}} = \frac{1 - \bar{R}^{\text{group}}}{\sigma^{\text{group}}}, \qquad \hat{A}_{\text{neg}} = \left(1 - \bar{R}^{\text{group}} - \frac{\binom{N_{\text{neg}}-1}{k-1}}{\binom{N-1}{k-1}}\right) \cdot (\sigma^{\text{group}})^{-1}. \tag{5}$$

These state-action advantages serve as the foundational $A^{pass@k}$ terms used during policy optimization.

B.4 Asymmetric Soft Reward Matrix for Verification

In Section 4, we introduce an asymmetric soft reward matrix to shape the verifier into a stringent screening mechanism. Let $R_{\text{second}}^{(j,k)}$ denote the reward for the k-th verification trajectory evaluating the j-th candidate answer. The exact reward formulation is defined as follows:

$$R_{\text{second}}^{(j,k)} = \begin{cases} -1.0, & \text{Format error.} \\ +1.0, & \text{True Positive / True Negative.} \\ -0.3, & \text{False Negative.} \\ -0.8, & \text{False Positive.} \end{cases}$$

This design embodies the principle of being "lenient to false negatives while strict with false positives". By assigning a -0.8 penalty to False Positives versus a milder -0.3 penalty to False Negatives, we explicitly penalize the verifier more heavily for endorsing incorrect reasoning paths than for cautiously rejecting potentially correct ones.

B.5 SELF VERIFICATION PROMPT

Box: Self Verification Prompt for Mathematical Hypothesis Testing

Problem:
{problem}

[Hypothesis to Test]
A previous attempt at this problem resulted in the following answer:
{candidate_answer}

[Task]
Act as a rigorous mathematical reviewer. Treat the previous answer ({candidate_answer}) as a given hypothesis. Plug this answer BACK into the original problem conditions. Perform a rigorous backward-substitution to check if it satisfies all constraints or if it leads to a mathematical contradiction.
You MUST conclude your verification by explicitly stating either "Verification Result: True" (if the hypothesis perfectly satisfies all conditions) or "Verification Result: False" (if it leads to any contradiction).
You MUST strictly use the following XML format for your response:

<reverse_verification>
(Your step-by-step backward substitution checking if {candidate_answer} contradicts the problem conditions)
Verification Result: [True/False]
</reverse_verification>

C EXPERIMENTAL DETAILS

C.1 RL HYPERPARAMETERS

The specific hyperparameters utilized during the reinforcement learning phase of TTRL-CoCoV are outlined in Table 3. This includes generator, verifier, and PPO-specific configurations.

Table 3: TTRL-CoCoV Training Settings

Method	Hyperparameters
Generator	$n_{\text{vote}} = 64$ $n_{\text{samples_per_prompt}} = 32$ Top-p = 1.0 Training Temperature = 1.0 $K_{pass} = 4$
Verifier	Temperature: $T_{high} = 1.0$, $T_{low} = 0.6$ $\tau_{high} = 0.6$, $\tau_{low} = 0.4$ Top-K candidates: $K_{high} = 3$, $K_{low} = 5$ Top-p = 0.85 $n_{\text{verification_samples}} = 8$
Length Diversity	$\lambda_{\text{div}} = 0.05$, $C_{max} = 2$ $\tau_{high} = 0.6$
PPO Trainer	Learning Rate = 5×10^{-7} $\gamma = 1.0$, $\lambda = 1.0$ Use KL Loss = True KL Coefficient = 0.001
Batch Sizes	train_batch_size = 32 rollout_batch_size = 32 mini_batch_size = 1 micro_batch_size = 8
Lengths	Prompt Max Length = 1024 Generate Max Length = 4096 Verify Max Length = 2048
Training Schedule	Epochs = 1
Evaluation	$n_{\text{samples}} = 32$ Temperature $T = 0.6$ Top-$p = 0.95$, Top-$K = 20$ Metrics: Pass@1, Pass@16

C.2 HARDWARE SETUP

Unless otherwise specified, all experiments, including the reinforcement learning training phase and subsequent evaluations of the TTRL-CoCoV framework, were conducted on a compute node equipped with 4 $\times$ NVIDIA H100 GPUs. The computational profiling and FLOPs-matched comparison reported in Appendix F.4 were conducted on a node equipped with 4 $\times$ NVIDIA RTX PRO 6000 GPUs.

C.3 METRICS

This section provides details regarding our experimental configurations to ensure full reproducibility. Specifically, we outline the exact sampling procedures used for our metrics.

Metrics For each problem, we generate 32 reasoning trajectories with temperature 0.6, top-$p = 0.95$, and top-$k = 20$. We then evaluate each checkpoint with four fixed seeds. For every seed, we draw 16 trajectories without replacement from the 32 generated trajectories and compute

- **pass@1**: the fraction of correct answers among the 16 selected trajectories, averaged over problems;
- **pass@16**: the fraction of problems for which at least one of the 16 selected trajectories is correct.

D SUPPLEMENTARY DIAGNOSIS

D.1 DETAILED EMPIRICAL ANALYSIS OF NAIVE PASS@K ADAPTATION

To empirically demonstrate the failure of naively adapting the Pass@k objective in label-free settings, we evaluated a straightforward adaptation (denoted as Pass@k TTA), which treats non-majority candidate responses as N_{neg}, using the Qwen3-4B-Base model across the AIME25, AMC and GPQA benchmarks. Main results are present in Table 4. Standard TTRL often improves Pass@1 but contracts Pass@16, most severely on AIME25 ($32.50 \rightarrow 14.40$) and GPQA ($90.00 \rightarrow 58.00$). Pass@k TTA further improves Pass@1, yet restores only part of the lost coverage: 24.60 on AIME25, 72.10 on AMC, and 75.00 on GPQA, all below the base model. Without a ground-truth oracle, rewarding disagreement with the majority pseudo-label remains compromised by label noise and cannot reliably sustain the exploratory space required for complex reasoning.

Table 4: Pass@1 / Pass@16 of naive Pass@k TTA on Qwen3-4B-Base. TTRL contracts coverage; Pass@k TTA raises Pass@1 but does not restore Base Pass@16.

Method	AIME25	AMC	GPQA
Qwen3-4B-Base	6.82 / 32.50	40.10 / 76.51	34.70 / 90.00
TTRL	3.30 / 14.40	41.80 / 63.50	36.00 / 58.00
Pass@k TTA	8.50 / 24.60	47.10 / 72.10	36.10 / 75.00

D.2 VERIFICATION ADVANTAGE ACROSS CONFIDENCE LEVELS

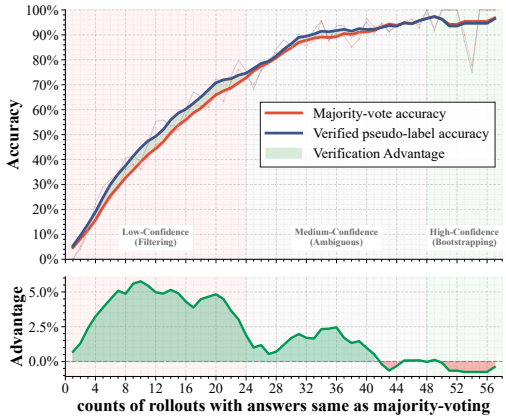
To determine where self-verification provides useful supervision beyond majority voting, we compare majority-vote pseudo-label accuracy with the accuracy obtained after self-verification across confidence levels. For this diagnostic, we sample $N = 64$ generator rollouts per problem and report majority confidence as the number of rollouts that agree with the majority answer.

As shown in Fig. 5a, the benefit of verification is strongly confidence-dependent. In the low-confidence region, verification provides the largest improvement over majority voting, making it particularly valuable for correcting the unreliable pseudo-labels that receive the strongest Pass@k exploration signal. As confidence increases, this advantage gradually diminishes. In the medium-confidence region, verification provides little additional gain, while in the high-confidence region majority voting is already sufficiently reliable and verification no longer improves pseudo-label accuracy.

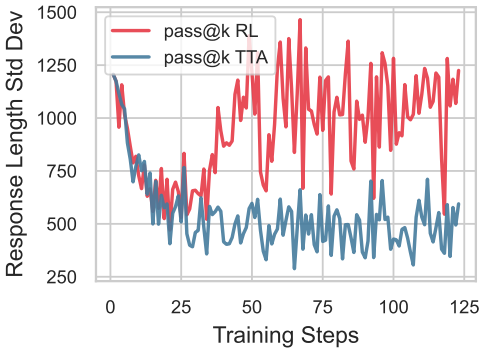
This confidence-dependent behavior directly motivates our routing strategy: low-confidence samples invoke verification before Pass@k optimization, medium-confidence samples retain the majority label without additional verification, and high-confidence samples are instead used to supervise the verifier.

D.3 EXPLORATION DYNAMICS UNDER PASS@K TTA

We further examine how the exploratory behavior of Pass@k optimization changes when ground-truth labels are replaced by majority-vote pseudo-labels. We compare the response-length standard deviation throughout training between supervised Pass@k RL and Pass@k TTA.



(a) Verification advantage across confidence levels.



(b) Response-length standard deviation during training.

Figure 5: (a) Self-verification provides the largest improvement over majority voting in the low-confidence region, offers little additional gain in the medium-confidence region, and provides no further benefit at high confidence. (b) Supervised Pass@k RL and label-free Pass@k TTA begin with similar response-length variance, but the variance under Pass@k TTA progressively contracts, whereas Pass@k RL with ground-truth labels maintains a broader trajectory distribution.

As shown in Fig. 5b, the two settings exhibit similar response-length variance at the beginning of training, but diverge substantially as optimization proceeds. Supervised Pass@k RL maintains a broad response-length distribution, whereas the variance under Pass@k TTA progressively contracts to a much narrower range.

This result complements the analysis in Section 3. As consensus increases, the Pass@k advantage becomes increasingly small and eventually vanishes, leaving little incentive to maintain alternative trajectories. In the label-free setting, minority trajectories cannot recover this signal because they are treated as incorrect whenever they disagree with the majority pseudo-label. The observed contraction therefore provides empirical evidence that directly transferring Pass@k Training to TTRL does not sustain exploration, motivating an additional exploration incentive in the reliable high-confidence regime.

E Additional Experimental Results

E.1 Full Main Results with Standard Deviations

Table 5 reports the complete results of Table 1, including the standard deviation of each method over four evaluation seeds. The standard deviations of TTRL-CoCoV are comparable to or smaller than those of the baselines on most benchmarks, indicating that its improvements are consistent across seeds. Variance is highest on AIME24 and AIME25 for all methods, as these benchmarks contain only 30 problems each.

E.2 Generalization and Transferability

We evaluate the transferability of TTRL-CoCoV along three complementary dimensions: adaptation corpus, model scale, and backbone architecture.

E.2.1 Transfer Across Adaptation Corpora

To test whether the method is tied to DAPO-14k-MATH, we directly transfer the complete default configuration to ACCReason without retuning any confidence threshold, verifier budget, or diversity-reward hyperparameter.

As shown in Table 6, TTRL-CoCoV consistently outperforms TTRL across all five downstream benchmarks, yielding average gains of 5.31 Pass@1 and 11.42 Pass@16 points. The unchanged

Table 5: Full results on six reasoning benchmarks with standard deviations. Each cell reports *pass@1 / pass@16* as mean ± standard deviation over four evaluation seeds. **Ours** refers to TTRL-CoCoV. †Uses labeled reward data; all other methods are *label-free*.

Method	AIME24	MATH500	AIME25	AMC	GPQA	DAPO
Qwen3-4B-Base						
Base	8.02±0.60 / 26.70±2.40	67.20±0.20 / 91.75±0.52	6.82±0.80 / 32.50±1.44	40.10±0.60 / 76.51±3.13	34.70±0.50 / 90.00±1.21	28.90±0.20 / 69.00±0.60
GRPO†	22.20±1.60 / 45.60±1.60	81.40±0.30 / 92.70±0.20	21.10±4.20 / 38.90±4.20	59.00±1.70 / 81.10±0.60	51.80±1.10 / 78.80±0.20	45.70±1.30 / 82.40±0.60
TTRL	8.90±1.60 / 21.10±1.60	72.80±0.70 / 86.10±0.30	3.30±2.70 / 14.40±1.60	41.80±0.60 / 63.50±3.00	36.00±0.80 / 58.00±1.30	30.60±0.70 / 65.30±0.60
SCRL	11.30±0.40 / 22.50±1.44	76.00±0.10 / 88.30±0.30	9.70±0.80 / 30.83±2.80	43.90±0.30 / 71.40±1.31	35.50±0.44 / 79.31±1.70	34.40±0.14 / 59.59±0.40
Co-Rewarding-III	16.10±0.70 / 25.80±1.44	77.60±0.10 / 86.60±0.10	8.30±0.61 / 20.00±0.00	49.70±0.54 / 65.70±2.00	35.49±0.24 / 67.42±0.80	39.90±0.15 / 59.20±0.50
CoVerRL	10.73±0.40 / 26.70±2.40	75.31±0.23 / 90.80±0.14	8.20±0.60 / 34.20±4.33	46.40±0.42 / 75.00±1.60	35.23±0.44 / 83.90±1.41	33.70±0.14 / 66.80±0.50
Ours	**20.40±0.65 / 44.20±1.44**	**81.40±0.21 / 94.40±0.30**	**17.90±1.10 / 45.00±1.70**	**53.50±0.60 / 81.90±1.50**	**38.20±0.61 / 86.40±1.21**	**44.50±1.10 / 78.00±0.70**
Qwen3-8B-Base						
Base	10.70±0.31 / 36.70±4.10	63.83±0.14 / 91.90±0.50	8.00±1.00 / 25.83±4.33	38.50±0.60 / 78.61±1.60	35.22±0.35 / 87.60±1.50	29.33±0.30 / 69.95±0.40
GRPO†	26.60±1.00 / 48.33±1.70	85.70±0.20 / 94.60±0.14	21.90±1.00 / 44.20±1.44	63.54±0.30 / 85.24±1.00	48.00±0.40 / 91.50±1.30	55.33±0.24 / 83.00±0.62
TTRL	14.11±0.50 / 29.20±1.44	76.84±0.20 / 87.95±0.40	10.70±0.31 / 23.33±2.40	50.55±0.50 / 72.60±1.31	39.21±0.15 / 80.60±0.80	38.70±0.20 / 66.45±0.50
SCRL	14.70±0.70 / 33.33±0.00	77.82±0.15 / 91.80±0.50	13.90±0.31 / 28.33±1.70	50.23±0.35 / 81.33±1.04	37.90±0.70 / 86.70±1.70	38.12±0.10 / 72.25±0.90
Co-Rewarding-III	15.50±0.34 / 31.70±1.70	80.15±0.13 / 89.25±0.22	12.34±0.40 / 23.33±2.40	55.90±0.23 / 72.30±1.20	41.10±0.12 / 76.02±0.80	44.65±0.04 / 65.85±0.73
CoVerRL	16.00±1.23 / 38.33±1.70	78.30±0.10 / 92.30±0.40	12.40±0.35 / 28.33±2.90	50.60±0.44 / 78.92±2.00	38.70±0.20 / 85.03±0.44	39.10±0.20 / 71.60±0.42
Ours	**23.00±0.40 / 51.70±1.70**	**82.62±0.23 / 95.40±0.20**	**17.30±0.53 / 42.50±1.44**	**57.74±0.72 / 86.45±0.52**	**43.65±0.33 / 91.00±0.60**	**45.30±0.21 / 77.85±0.73**

Table 6: Transfer to ACCReason as the adaptation corpus without hyperparameter retuning. Both methods are adapted on ACCReason. Each cell reports Pass@1 / Pass@16.

Method	AIME24	MATH500	AIME25	AMC	GPQA
TTRL	10.7/21.7	75.1/87.3	6.51/29.2	45.7/67.8	36.3/78.3
TTRL-CoCoV	**19.4/43.3**	**79.5/94.0**	**14.4/35.8**	**49.7/81.9**	**37.9/86.4**

configuration therefore transfers effectively to a different adaptation corpus, showing that the gains are not specific to DAPO-14k-MATH.

E.2.2 Scalability Across Model Sizes

We next evaluate Qwen3 models ranging from 0.6B to 8B parameters. As shown in Fig. 6, TTRL-CoCoV improves Pass@1 across all evaluated model sizes, while its Pass@k gains become substantially larger at the 4B and 8B scales. The smaller models exhibit mixed Pass@k behavior, indicating that effective generator–verifier co-evolution benefits from sufficient underlying reasoning and verification capacity.

E.2.3 Generalization Across Model Architectures

We further evaluate TTRL-CoCoV on Qwen3-8B-Base, OctoThinker-8B-Hybrid-Base, and Qwen2.5-Math-7B. As shown in Fig. 7, TTRL-CoCoV improves Pass@1 across all three backbones, demonstrating that the framework is not specific to the Qwen3 architecture. The gains are largest on challenging mathematical benchmarks for models with stronger mathematical priors, while improvements on broader-domain tasks such as GPQA are smaller. These results show that TTRL-CoCoV transfers across model families, while the achievable gain remains closely tied to the underlying reasoning and verification capability of the base model.

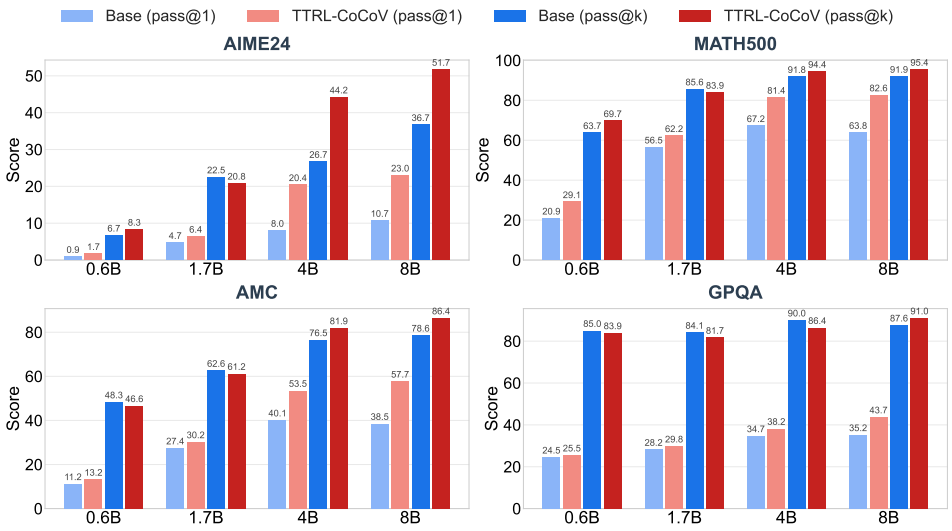


Figure 6: Scalability of TTRL-CoCoV across model sizes. TTRL-CoCoV yields consistent Pass@1 improvements across the evaluated scales. Pass@k gains become more pronounced for 4B and 8B models, while performance on smaller models is mixed on several benchmarks.

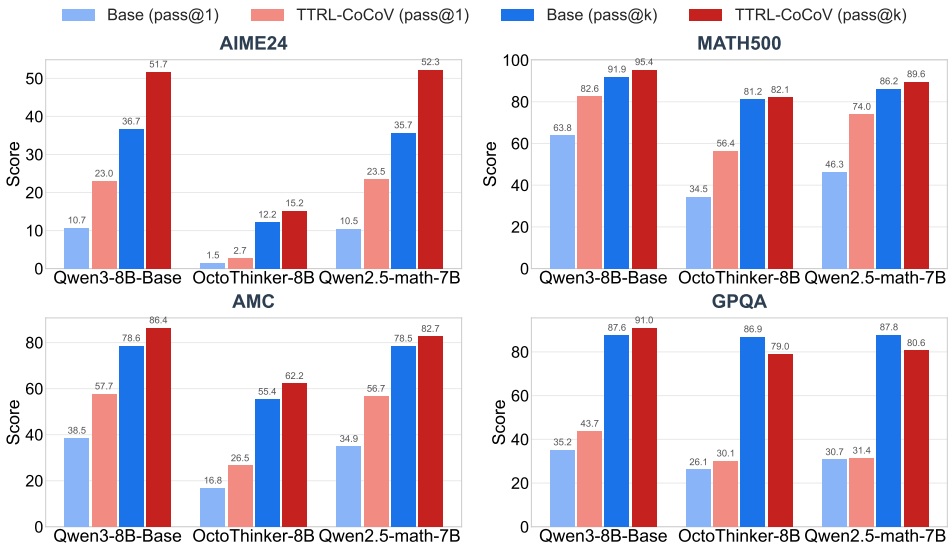


Figure 7: Generalization of TTRL-CoCoV across different model architectures at the 7B/8B scale. We compare Qwen3-8B-Base, OctoThinker-8B-Hybrid-Base, and Qwen2.5-Math-7B on AIME24, MATH500, AMC, and GPQA.

F Analysis of TTRL-CoCoV

F.1 Detailed Reliability Analysis of Confidence-Conditioned Self-Supervision

This section provides detailed evidence for the reliability of confidence-conditioned self-supervision. We analyze both the quality of the resulting training signals and the relationship between majority confidence and pseudo-label correctness.

Training-time reward and pseudo-label reliability. Figure 4 shows that standard TTRL becomes increasingly vulnerable to confirmation bias during training: reward accuracy deteriorates as incorrect self-generated labels are repeatedly reinforced. TTRL-CoCoV breaks this cycle. Its reward accuracy remains around 0.8 throughout the later stages of training, while pseudo-label accuracy continues to increase. Confidence-conditioned verification therefore maintains substantially cleaner supervision over the course of adaptation.

Reliability of majority-defined confidence. We next directly measure how pseudo-label correctness varies with majority confidence on MATH500. Ground-truth answers are used only for this post-hoc analysis and are never exposed during adaptation. For each problem, we sample 32 trajectories and partition samples according to the same confidence boundaries used by TTRL-CoCoV: $C_{\mathrm{maj}} < 0.4$ (low), $0.4 \leq C_{\mathrm{maj}} < 0.6$ (medium), and $C_{\mathrm{maj}} \geq 0.6$ (high).

Table 7: Pseudo-label correctness across confidence regions on MATH500. Confidence is defined by the frequency of the majority answer among 32 sampled trajectories.

Method	Low		Medium		High	
	#Samples	Acc.	#Samples	Acc.	#Samples	Acc.
Base	86	33.72%	56	66.07%	358	95.25%
TTRL	28	32.14%	54	29.63%	418	84.45%
TTRL-CoCoV	49	34.69%	41	53.66%	410	97.07%

As shown in Table 7, pseudo-label accuracy under TTRL-CoCoV rises from 34.69% in the low-confidence region to 53.66% in the medium-confidence region and 97.07% in the high-confidence region. This strong monotonic relationship shows that confidence-conditioned routing separates supervision of different reliability levels.

Standard TTRL exhibits the opposite failure mode in the high-confidence region: its pseudo-label accuracy falls from 95.25% for the base model to 84.45%. Repeated self-reinforcement can therefore convert incorrect solutions into false consensus. TTRL-CoCoV raises high-confidence accuracy to 97.07%, showing that confidence-conditioned verification prevents unreliable pseudo-labels from dominating policy optimization.

F.2 Detailed Impact of Generator-Verifier Co-evolution

We test whether the reliable supervision from high-confidence samples can continuously improve the verifier as the generator evolves. We isolate this co-evolution by freezing the verifier while keeping the remaining training pipeline unchanged.

As shown in Fig. 8, freezing the verifier causes a substantial degradation across downstream benchmarks. On AIME25, Pass@1 drops from 17.9% to 9.8%, while Pass@16 falls from 45.0% to 28.8%. The verifier must therefore evolve together with the generator to remain effective as the policy distribution changes.

The internal dynamics in Fig. 9 explain this gap. With a frozen verifier, verification error continuously increases as the generator moves away from the distribution on which the verifier was initially calibrated. Under joint updates, Region A supplies high-quality supervision for verifier learning, while filtered Region-B samples further refine its discrimination boundary. As a result, both validation error and format error decrease sharply during training. These results directly validate co-evolution:

high-confidence supervision keeps the verifier aligned with the generator, and the updated verifier in turn maintains reliable verification throughout adaptation.

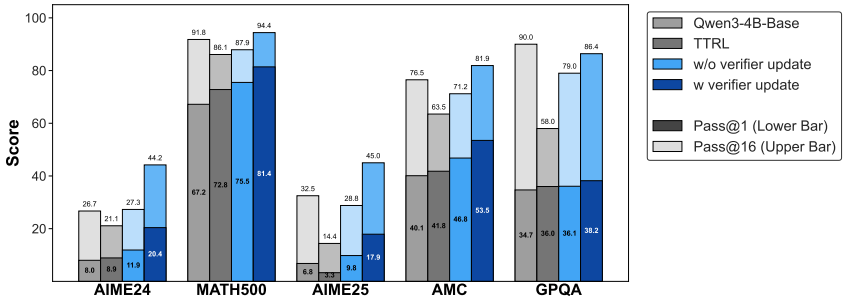


Figure 8: Impact of verifier co-evolution on downstream task performance. Freezing the verifier (w/o verifier update) consistently degrades performance across all five benchmarks, while joint generator-verifier updates (w/ verifier update) achieve the best results.

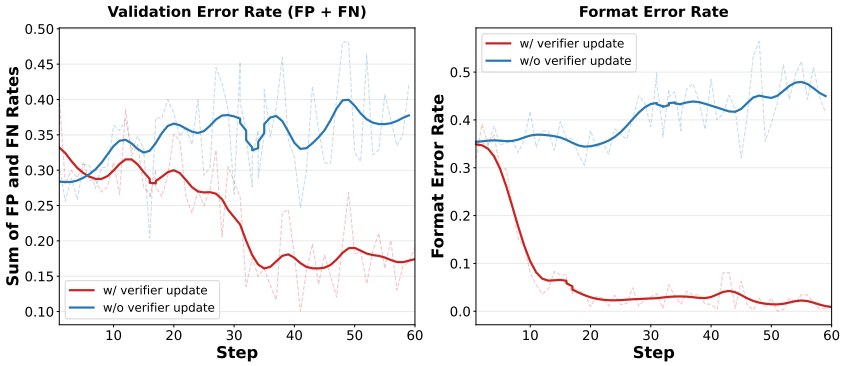


Figure 9: Detailed internal verification error dynamics. Under joint updates (w/ verifier update), both the validation error rate (FP+FN) and format error rate drop significantly. Conversely, a static verifier (w/o verifier update) suffers from capability mismatch, leading to an escalating validation error rate.

F.3 Detailed Analysis of Exploration Diversity

To complement the response-length analysis, we evaluate embedding-based semantic diversity on MATH500. For each problem, we sample 32 trajectories under the same evaluation protocol as the main experiments. Each trajectory is encoded with Qwen3-Embedding-4B, and semantic diversity is defined as the mean pairwise cosine distance among the resulting embeddings. We report this quantity over all 32 trajectories as well as over the subset of correct trajectories.

Table 8: Response length and embedding-based semantic diversity on MATH500. Higher semantic-diversity values indicate larger variation among sampled reasoning trajectories.

Method	Semantic Diversity (All)	Semantic Diversity (Correct)	Mean Tokens
Base	0.0868	0.0665	701.60
TTRL	0.0233	0.0196	534.99
TTRL-CoCoV	0.0373	0.0333	1208.24

Standard TTRL substantially contracts both response length and semantic diversity relative to the base model. TTRL-CoCoV reverses this trend: the mean response length increases from 534.99 to 1208.24 tokens, while semantic diversity simultaneously rises from 0.0233 to 0.0373 over all trajectories and from 0.0196 to 0.0333 over correct trajectories.

The semantic-diversity gain remains when only correct responses are considered. This shows that the increased response length is not merely verbosity or additional erroneous generation; it is accompanied by a broader set of successful reasoning trajectories. Together, these results demonstrate that TTRL-CoCoV maintains exploration even in regimes where the original Pass@k signal becomes weak, rather than simply producing longer responses.

F.4 Computational Cost and Compute-Matched Comparison

TTRL-CoCoV introduces additional computation because samples routed to the verifier require extra verification trajectories. We quantify this overhead and then construct a FLOPs-matched TTRL control to determine whether additional computation alone explains the performance gain.

Runtime and token overhead. All methods in this profiling experiment are measured on the same node equipped with four NVIDIA RTX PRO 6000 GPUs under the same training batch configuration.

Table 9: Per-step computational profile. The profiling experiment is conducted under identical hardware and batch settings for all compared methods.

Method	Policy/Vote Tokens	Verification Tokens	Total Generated	Update Tokens	GPU-h/Step	Time/Step	FLOPs/Step
TTRL	2.72M	0	2.72M	1.52M	0.3736	336.28s	78.722 PF
TTRL (FLOPs-matched)	5.17M	0	5.17M	2.14M	1.1463	1031.64s	145.040 PF
TTRL-CoCoV	2.79M	0.75M	3.54M	1.94M	0.6895	620.54s	133.464 PF

Compared with standard TTRL, TTRL-CoCoV generates approximately $1.30\times$ more tokens and increases the per-step FLOPs from 78.722 PF to 133.464 PF. This overhead primarily comes from verifier generation and the corresponding verifier-side policy updates.

FLOPs-matched TTRL control. We then allocate an even larger compute budget to standard TTRL by increasing the rollout budget used for majority-label estimation. The resulting control consumes 145.040 PF per step, exceeding TTRL-CoCoV by approximately 8.7%.

Table 10: Comparison with the FLOPs-matched TTRL control. Each cell reports Pass@1 / Pass@16.

Method	AIME24	MATH500	AIME25	AMC	GPQA	Avg.
TTRL (FLOPs-matched)	11.1/28.9	76.3/88.4	12.2/26.7	48.6/72.7	38.6/65.5	37.36/56.44
TTRL-CoCoV	20.4/44.2	81.4/94.4	17.9/45.0	53.5/81.9	38.2/86.4	42.28/70.38

Despite this larger compute budget, the FLOPs-matched TTRL baseline remains 4.92 Pass@1 and 13.94 Pass@16 points below TTRL-CoCoV on average and loses 9 of the 10 benchmark–metric comparisons. Simply increasing the number of voting samples therefore cannot reproduce the gains of TTRL-CoCoV. The substantially larger gap in Pass@16 further shows that confidence-conditioned verification is particularly effective at preserving multi-sample solution coverage.

F.5 Ablation Study

We conduct controlled ablations on Qwen3-4B-Base to isolate the contributions of verifier-based filtering, generator–verifier co-evolution, the length-diversity reward, and the asymmetric verifier reward.

Verifier-based filtering. Removing verifier-based filtering while retaining confidence routing and R_{div} reduces the five-benchmark average by 6.34 Pass@1 and 10.62 Pass@16 points. Confidence partitioning alone is therefore insufficient: explicitly filtering low-confidence pseudo-labels is essential for preventing noisy supervision from entering policy optimization.

Generator–verifier co-evolution. Freezing the verifier reduces the five-benchmark average from 42.28/70.38 to 36.02/58.84 in Pass@1/Pass@16. This large degradation confirms that verifier filtering

Table 11: Component ablation on Qwen3-4B-Base. Each cell reports Pass@1 / Pass@16.

Method	AIME24	MATH500	AIME25	AMC	GPQA	Avg.
Without verifier	9.1/20.9	76.1/89.9	12.0/30.3	46.2/75.6	36.3/82.1	35.94/59.76
Frozen verifier	11.9/27.3	75.5/87.9	9.8/28.8	46.8/71.2	36.1/79.0	36.02/58.84
Without R_{div}	13.7/28.6	76.9/89.8	13.5/32.4	45.4/72.2	35.3/77.1	36.96/60.02
TTRL-CoCoV	**20.4/44.2**	**81.4/94.4**	**17.9/45.0**	**53.5/81.9**	**38.2/86.4**	**42.28/70.38**

cannot remain static while the generator distribution changes. Co-evolution continuously recalibrates the verifier and preserves the quality of its supervision throughout training.

Length-diversity reward. Removing R_{div} reduces the five-benchmark average to 36.96/60.02, with a particularly large drop in Pass@16. This result confirms the role of R_{div} in preserving generation coverage rather than merely improving single-sample accuracy. The length and semantic-diversity analyses in Appendix F.3 provide direct trajectory-level evidence for this effect.

Together, these ablations demonstrate that the three major components solve complementary failure modes: verifier gating removes noisy low-confidence supervision, co-evolution keeps verification aligned with the evolving generator, and R_{div} restores exploration pressure in high-confidence groups.

Asymmetric verification reward. We finally compare the asymmetric verifier reward with a symmetric alternative. As shown in Fig. 10, the symmetric reward maintains a substantially higher false-positive ratio, meaning that incorrect candidates are more frequently admitted by the verifier. Assigning a larger penalty to false positives imposes a stricter screening criterion and reduces the FP ratio to approximately 0.6. This experiment validates the asymmetric reward design: false acceptance is more harmful than false rejection because an accepted incorrect candidate can directly become a pseudo-label and corrupt the generator update.

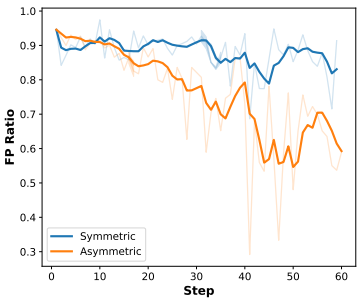


Figure 10: FP ratio dynamics under symmetric vs. asymmetric reward strategies.

F.6 Hyperparameter Sensitivity

We evaluate the sensitivity of TTRL-CoCoV to the major routing, verification, and exploration hyperparameters through one-factor studies, varying one parameter at a time while keeping all remaining settings fixed. The default configuration is marked by †.

Confidence-routing thresholds. The settings $(0.2, 0.5)$, $(0.3, 0.6)$, and the default $(0.4, 0.6)$ achieve comparable performance, demonstrating that TTRL-CoCoV is not tied to a single threshold pair. Performance drops when both thresholds are shifted upward: more medium-confidence samples are unnecessarily routed to verification, while fewer high-confidence samples remain available to supervise the verifier and receive exploration rewards. The default $(0.4, 0.6)$ provides a strong balance between pseudo-label reliability and region coverage.

Top-K candidate size. TTRL-CoCoV performs consistently with small and moderate candidate sets, while increasing both candidate sizes to seven causes a substantial degradation. A larger candidate pool improves answer coverage but simultaneously increases verifier-side sampling and optimization pressure. We therefore use $(K_{\text{high}}, K_{\text{low}}) = (3, 5)$ as an effective balance between candidate coverage and computational cost.

Number of verifier trajectories. Using 4, 8, or 12 verification trajectories yields comparable overall performance, showing that the method is not sensitive to the exact verification budget within

Table 12: Sensitivity of confidence-routing and verification parameters. Each cell reports Pass@1 / Pass@16.

Parameter	Setting	AIME24	MATH500	AIME25	AMC	GPQA
Thresholds	$(0.2, 0.5)$	21.8/45.8	80.9/94.1	16.5/39.2	52.5/82.5	39.5/88.3
	$(0.3, 0.6)$	20.1/43.3	80.8/93.8	20.9/46.7	52.4/83.7	39.4/87.7
	$(0.4, 0.7)$	20.31/35.0	78.7/92.75	14.1/40.0	46.84/78.92	38.0/84.4
	$(0.5, 0.8)$	14.11/35.0	78.0/93.65	13.54/39.2	47.53/76.81	38.3/87.82
	$(0.4, 0.6)^\dagger$	20.4/44.2	81.4/94.4	17.9/45.0	53.5/81.9	38.2/86.4
Top-K	$(3, 3)$	20.8/41.7	81.2/93.5	17.0/41.7	53.8/86.75	38.1/87.6
	$(5, 5)$	20.6/47.5	81.3/94.0	20.1/45.0	51.6/83.73	39.2/88.32
	$(7, 7)$	12.0/26.7	76.0/89.1	10.4/31.7	46.6/74.1	36.9/85.3
	$(3, 5)^\dagger$	20.4/44.2	81.4/94.4	17.9/45.0	53.5/81.9	38.2/86.4
Verifier trajectories	4	17.8/41.1	80.7/93.3	15.6/44.4	54.2/84.3	45.0/77.3
	12	22.2/46.7	80.3/94.1	22.2/48.9	50.2/85.1	43.7/77.3
	$8^\dagger$	20.4/44.2	81.4/94.4	17.9/45.0	53.5/81.9	38.2/86.4
VPR threshold	0.375	20.1/37.7	79.1/92.5	15.7/36.5	51.5/82.7	38.3/82.4
	0.625	13.0/25.0	74.1/87.6	10.5/28.3	45.4/72.6	37.5/85.7
	$0.5^\dagger$	20.4/44.2	81.4/94.4	17.9/45.0	53.5/81.9	38.2/86.4

this range. Additional trajectories provide more stable verification decisions but incur proportionally greater inference cost; we therefore use eight trajectories as the default.

Verification pass-rate threshold. The default VPR threshold of 0.5 achieves the strongest overall performance. Lowering the threshold admits candidates with weaker verifier support, while raising it to 0.625 removes too many potentially useful low-confidence candidates. The results expose a clear precision–coverage trade-off and support majority verification as the natural operating point.

Table 13: Sensitivity of the length-diversity reward parameters. Each cell reports Pass@1 / Pass@16.

Parameter	Setting	AIME24	MATH500	AIME25	AMC	GPQA
λ_{div}	0.03	16.7/36.7	77.9/93.3	16.7/35.6	50.2/76.3	40.4/73.8
	0.07	17.1/36.7	79.2/93.5	13.8/35.8	49.2/84.3	38.4/84.3
	$0.05^\dagger$	20.4/44.2	81.4/94.4	17.9/45.0	53.5/81.9	38.2/86.4
$C_{\max}$	1.75	13.54/31.7	76.8/91.8	10.9/35.83	46.0/77.41	37.3/85.3
	2.25	19.2/38.3	79.6/94.0	14.6/41.7	50.7/83.1	37.5/90.4
	$2.0^\dagger$	20.4/44.2	81.4/94.4	17.9/45.0	53.5/81.9	38.2/86.4

Length-diversity coefficient. The default $\lambda_{\text{div}} = 0.05$ provides the best balance among the tested values. A smaller coefficient provides insufficient exploration pressure, whereas a larger value places excessive emphasis on length variation relative to the underlying Pass@k objective.

Length-deviation cap. The default $C_{\max} = 2.0$ achieves the strongest overall performance among the tested settings. Tightening the cap to 1.75 substantially weakens the diversity bonus, while increasing it to 2.25 provides no consistent additional gain. We therefore use $C_{\max} = 2.0$ throughout the main experiments.

Overall, TTRL-CoCoV remains effective across a practical range of routing, verification, and diversity settings. Together with the ACCReason transfer results, these experiments demonstrate that the performance gains do not depend on narrowly tuned hyperparameters or a single adaptation corpus.

G Limitations and Future Work

Ceiling of Co-evolution. We begin by discussing the performance upper bound of the generator-verifier dynamic. In TTRL-CoCoV, the verifier enhances its discriminative ability using high-confidence consensus from mastered problems, which in turn empowers it to securely mine and learn from filtered samples in uncertain regimes. However, if the underlying language model possesses weak foundational reasoning capabilities, the high-confidence region will be extremely sparse, directly compromising the reliability of the subsequent low-confidence filtering. In such cases, the entire co-evolutionary cycle is starved of high-fidelity training signals, inherently restricting the framework's ability to guide the generator. This observation aligns with the consensus that while label-free RL effectively unlocks a model's existing potential, the ultimate performance ceiling is still fundamentally constrained by the base model's inherent capacity.

Limitation of Outcome-Oriented Verification. We then consider the scope of the verification mechanism. To manage computational overhead and focus on final utility, our verifier primarily evaluates the extracted candidate answers rather than the full reasoning rollouts. For tasks with deterministic numerical answers, this approach is highly efficient. However, in scenarios where the correctness of the logical process is paramount, such as complex mathematical proofs, outcome-oriented verification might overlook spurious reasoning steps that coincidentally yield the correct answer. To address this limitation, future work could focus on utilizing rollout-level self-evaluation mechanisms to validate the integrity of the intermediate reasoning process.

Failure Cases of Consensus-Based Confidence. Our routing strategy uses majority consensus C_{maj} as an empirical estimate of sample reliability. Although our analyses show a strong positive relationship between consensus and pseudo-label correctness, consensus is not equivalent to correctness. In particular, the model may exhibit *false consensus*, where multiple trajectories repeatedly converge to the same systematic error, resulting in a high-confidence but incorrect pseudo-label. Conversely, semantically equivalent reasoning paths may produce superficially different final answers or answer formats, artificially lowering consensus despite the presence of valid solutions. Moreover, the distribution of C_{maj} can vary with model capability, sampling configuration, and training stage. Therefore, the current confidence score should be interpreted as a practical routing statistic rather than a calibrated estimate of correctness.

Adaptive Confidence Routing. The current framework uses globally fixed confidence thresholds $(\tau_{\mathrm{low}}, \tau_{\mathrm{high}})$ throughout training. While our sensitivity and cross-corpus experiments demonstrate that the method operates effectively within a practical range of threshold values, a fixed partition may not be optimal across different model capacities, data distributions, or stages of self-evolution. An important direction for future work is therefore to replace static thresholds with adaptive routing. For example, the thresholds could be dynamically adjusted according to recent verifier reliability, confidence-region occupancy, or the evolving distribution of model consensus, allowing verification computation to be allocated according to the model's current learning state rather than a predefined global partition.