Trust, But Verify: A Self-Verification Approach to Reinforcement Learning with Verifiable Rewards

Xiaoyuan Liu[1,2*] **Tian Liang**[2] **Zhiwei He**[2,3] **Jiahao Xu**[2] **Wenxuan Wang**[4]
Pinjia He[1†] **Zhaopeng Tu**[2†] **Haitao Mi**[2] **Dong Yu**[2]

[1]School of Data Science, The Chinese University of Hong Kong, Shenzhen
[2]Tencent [3]Shanghai Jiao Tong University [4]Renmin University of China

Abstract

Large Language Models (LLMs) show great promise in complex reasoning, with Reinforcement Learning with Verifiable Rewards (RLVR) being a key enhancement strategy. However, a prevalent issue is "superficial self-reflection", where models fail to robustly verify their own outputs. We introduce RISE (**R**einforcing Reason**i**ng with **S**elf-V**e**rification), a novel online RL framework designed to tackle this. RISE explicitly and simultaneously trains an LLM to improve both its problem-solving and self-verification abilities within a single, integrated RL process. The core mechanism involves leveraging verifiable rewards from an outcome verifier to provide on-the-fly feedback for both solution generation and self-verification tasks. In each iteration, the model generates solutions, then critiques its own on-policy generated solutions, with both trajectories contributing to the policy update. Extensive experiments on diverse mathematical reasoning benchmarks show that RISE consistently improves model's problem-solving accuracy while concurrently fostering strong self-verification skills. Our analyses highlight the advantages of online verification and the benefits of increased verification compute. Additionally, RISE models exhibit more frequent and accurate self-verification behaviors during reasoning. These advantages reinforce RISE as a flexible and effective path towards developing more robust and self-aware reasoners.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable potential in complex reasoning tasks. A promising avenue for further enhancing these capabilities is Reinforcement Learning (RL), particularly methods that utilize verifiable rewards (RLVR) from outcome verifiers [Gao et al., 2024, Guo et al., 2025, Lambert et al., 2024, Yue et al., 2025]. This paradigm, often applied to domains like mathematics where solution correctness can be programmatically evaluated, enabling models to improve through direct feedback on their generated solutions.

However, even with outcome-based RLVR, models may still learn to generate spurious reasoning without truly understanding the underlying logical process or developing robust self-assessment skills. This can lead to "superficial self-reflection" [Liu et al., 2025], where models struggle to reliably identify errors in their own reasoning and verify the correctness of their outputs, ultimately resulting in flawed solutions and suboptimal performance. While some approaches explicitly incorporate self-critique [Xi et al., 2024, Xie et al., 2025] to provide additional signals, the process of learning to solve problems and learning to verify solutions are often decoupled or lack direct, contemporaneous feedback for the verification skill itself within the RL loop.

* Work was done when Xiaoyuan and Zhiwei were interning at Tencent.
† Correspondence to: Pinjia He <hepinjia@cuhk.edu.cn> and Zhaopeng Tu <zptu@tencent.com>.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

To address this limitation and foster more robust reasoning, we introduce **RISE** (**R**einforcing Reasoning with **S**elf-V**e**rification) as a novel online reinforcement learning framework. RISE is designed to explicitly and simultaneously train an LLM to improve both its problem-solving ability and its capacity to verify its own generated solutions within a single, integrated RL process. The key idea is to leverage the verifiable reward signal from a rule-based outcome verifier not only to guide the generation of correct solutions but also to align the model's self-verification ability on-the-fly.

In the RISE framework, during each training iteration, the model first generates solutions for a batch of problems. Subsequently, using these on-policy generated solutions and the original problems, verification problems are constructed with a predefined template, prompting the model to critique its own solution and provide a score. The same outcome verifier used to assess problem solutions also provides ground-truth supervision for the verification task, based on an exact match between the predicted verification score and the ground-truth solution score. Both the problem-solving trajectories and the self-verification trajectories, along with their respective verifiable rewards, are then combined to update the model's parameters using a unified RL objective. This tight coupling enables the model to learn not only to solve problems, but also to critique and verify its own outputs, fostering a more dynamic and grounded self-improvement loop.

In our experiments, we implement and evaluate RISE using the Proximal Policy Optimization (PPO) algorithm, applying it to the 1.5B, 3B, and 7B base models from the Qwen2.5 series. Compared to a Zero-RL baseline, which incorporates only problem-solving supervision, RISE consistently improves reasoning accuracy and achieves up to a $2.8\times$ increase in verification accuracy on challenging mathematical benchmarks. Moreover, RISE outperforms instruction-tuned models across both tasks. For instance, RISE-3B improves the reasoning accuracy by 3.7 points and self-verification accuracy by 33.4 points compared with Qwen2.5-3B-Instruct.

We also find that this enhanced self-verification ability contributes to improved test-time performance. Specifically, RISE-3B and RISE-7B outperform standard majority voting by +0.2% and +1.9%, respectively, under a $k{=}4$ inference budget. Further analysis reveals that RISE enhances the internal reasoning process by encouraging more frequent and effective verification behaviors. Finally, our ablations demonstrate that online verification is crucial to the success of RISE.

Our main contributions are as follows:

- We introduce **RISE** (**R**einforcing Reason**i**ng with **S**elf-V**e**rification), a novel online reinforcement learning framework that explicitly and simultaneously trains LLMs to improve both problem-solving and self-verification capabilities within a single, integrated RL process, leveraging verifiable rewards for both tasks on-the-fly.
- We demonstrate, through extensive experiments on challenging mathematical reasoning benchmarks using a PPO-based implementation, that RISE significantly boosts problem-solving performance while instilling robust self-verification skills in the LLM.
- We provide comprehensive analyses elucidating the critical role of RISE's online verification mechanism, the benefits of scaling verification training compute, and how the developed self-verification capability contributes to more accurate and reliable solution generation.

2 Related Work

RLVR for LLM Reasoning In the literature, reinforcement learning has been widely used to align language models with human preferences, typically through reward models or pairwise preference comparisons [Christiano et al., 2017, Ouyang et al., 2022, Rafailov et al., 2023]. More Recently, Reinforcement Learning with Verifiable Rewards (RLVR) has emerged as a powerful approach for improving the reasoning capabilities of LLMs in domains such as mathematics and programming [Jaech et al., 2024, Guo et al., 2025]. Using only outcome rewards, recent work has demonstrated the scalability of RL algorithms for LLM reasoning [Guo et al., 2025, Team et al., 2025, Zeng et al., 2025, Hu et al., 2025]. However, leveraging verifiable rewards not only for reasoning supervision but also as a direct training signal for self-verification remains underexplored, which is the main focus of RISE.

Learning to Solve and Verify Solution generation and verification are two foundational capabilities of LLMs [Huang et al., 2024, Song et al., 2024], echoing the classic P versus NP dichotomy in computer science [Wikipedia contributors, 2025]. In the context of LLM reasoning, previous work has focused on teaching models either to solve problems [Guo et al., 2025, Zelikman et al., 2022], to verify solutions [Wang et al., 2024, Lightman et al., 2023, Shi and Jin, 2025, Zhang et al., 2024], or to leverage the verification capability to perform self-improvement [Yuan et al., 2024, Xiong et al., 2025] and self-calibration [Huang et al., 2025]. More recently, Lin et al. [2025] introduced a self-play framework that trains LLMs to generate code and corresponding test cases through two-stage training, and Ma et al. [2025] proposed training methods that teach LLMs to self-verify and self-correct based on deliberately constructed trajectories. In contrast, we introduce an online RL framework that explicitly leverages verifiable reward signals to jointly align the model's problem-solving and self-verification abilities in a unified training process.

3 Reinforcement Learning Preliminaries

Policy Gradient Methods The goal of RL is to learn a policy that maximizes the expected cumulative reward (namely return), denoted as the performance measure J. Policy gradient methods learn a parameterized policy that can select actions to maximize J without consulting other value functions. Grounded by the *policy gradient theorem* [Sutton and Barto, 2018], the optimization is performed as gradient ascent based on the gradient of $J(\theta)$ with respect to the policy parameter θ.

A large language model is naturally a parametrized policy π_θ. The state at time t, denoted as s_t, is the concatenation of the prompt $\mathbf{x}$ and the response $\mathbf{y}_{<t}$ generated so far, while the action a_t is the next token y_t. T refers to total timestamps (response length + 1). Thus, the gradient can be expressed as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D},\,\mathbf{y}\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(y_t \mid \mathbf{x}, \mathbf{y}_{<t})\, A_t\right].$$

The core part of this method is the advantage function A_t, which determines the extent to increase or decrease the probability of selecting this action (token) in the given state. In practice, the advantage function is implemented as cumulative discounted rewards subtracting an optional baseline, representing how much better an action is compared to the alternatives:

$$A_t = \sum_{t=t_0}^{T} \gamma^{t-t_0} r_t - b(s_{t_0}), \tag{1}$$

where $\gamma \in [0,1]$ is the discount factor for the future rewards and $r_t = R(s_t, a_t, s_{t+1})$ is the reward from the environment at time t. Different implementations of the baseline formulate multiple variants of policy gradient methods, including using learned state-value functions (e.g., REINFORCE [Williams, 1992], Actor-Critic [Barto et al., 1983]), group-level reward means (e.g., GRPO [Shao et al., 2024]), and leave-one-out (e.g. RLOO [Ahmadian et al., 2024]).

Proximal Policy Optimization Proximal Policy Optimization (PPO) [Schulman et al., 2017] is a popular algorithm of Actor-Critic method, which incorporates a critic model ϕ to help estimate advantage for training the actor model θ (i.e., policy). One major improvement of PPO is penalizing excessive policy updates and thereby maintaining training stability. In practice, the objective of the actor model is defined as follows:

$$\mathcal{J}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right) - \beta KL\left(\pi_\theta || \pi_{ref}\right)\right], \tag{2}$$

where $r_t(\theta) = \frac{\pi_\theta(y_t|x,y_{<t})}{\pi_{\theta_{\text{old}}}(y_t|x,y_{<t})}$. Clip() and KL() are two techniques used for limiting update magnitudes. With Generalized Advantage Estimation (GAE) [Schulman et al., 2015], the advantage is estimated as a λ-weighted sum of step-emporal-Difference (TD) errors:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t-1}\delta_{T-1}, \tag{3}$$
$$\text{where } \delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t).$$

T denotes response length with token indexes from 0 to $T-1$. $V_\phi(s_t)$ is the value predicted by the critic model ϕ at state s_t, r_t is the scalar reward from the environment at time t, and $\lambda \in [0,1]$ is the GAE parameter that trades off between bias and variance. In practice, we set $\lambda = \gamma = 1$, thus

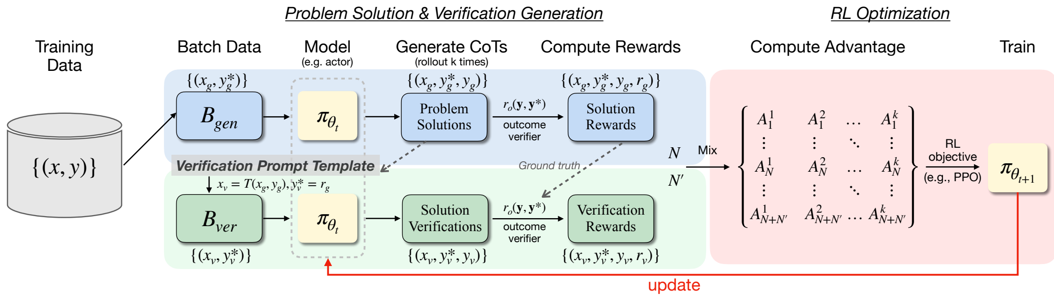


Figure 1: Illustration of RISE, which consists of two stages: (i) *Problem Solution & Verification Generation*: problems from the training batch are used to generate chain-of-thought solutions from the model. Problems and model solutions are then formatted as verification prompts to generate verifications of the solutions. (ii) *RL Optimization*: the original generation data and their verification are mixed as the new batch, and the model is optimized based on the RL objective.

making the per-token loss averaged over the full response length T. By design, $r_t = 0$ for $t < T - 1$, and $r_t = r$ for $t = T - 1$ (i.e., outcome reward). After we update the actor model, the critic model should also be updated for accurate value estimations. In practice, we use Mean Squared Error (MSE) to measure the prediction loss and perform the update:

$$\mathcal{J}(\phi) = \mathbb{E}_t\left[\max\left((V_\phi(s_t) - V_t^{targ})^2,\ \left(\text{clip}(V_\phi(s_t), V_{\phi_{\text{old}}}(s_t) - \epsilon, V_{\phi_{\text{old}}}(s_t) + \epsilon) - V_t^{targ}\right)^2\right)\right], \\ \text{where } V_t^{targ} = V_{\phi_{\text{old}}}(s_t) + \hat{A}_t. \tag{4}$$

Verifiable Reward Unlike the rewards from conventional reward models which are continuous values denoting the goodness of the response, verifiable rewards are usually discrete signals representing the correctness of the final result [Lambert et al., 2024, Guo et al., 2025]. Given the prompt $\mathbf{x}$ and the complete response $\mathbf{y}$ from the LLM π_θ, a classic verifiable reward is defined as a binary value produced by a deterministic outcome verifier OV: $r = OV(\mathbf{x}, \mathbf{y}) \in \{0, 1\}$, where $r = 1$ if and only if the final answer is exactly correct (e.g., the numeric result is mathematically equivalent to the ground truth answer) and $r = 0$ otherwise. In practice, an auxiliary format reward can be included to encourage the model to present its answer in a prescribed style.

4 Methodology: Reinforcing Reasoning with Self-Verification (RISE)

To address the challenge of superficial self-reflection, we propose RISE for self-improving reasoners, which is a scalable online RL method with explicit verification objective. **The key idea of RISE is the use of the verifiable reward signal from the rule-based outcome verifier to align the model's verification ability on-the-fly.** This enables us to teach the model to verify its own response at the same time it solves the problem, as depicted in Figure 1 and Algorithm 1.

4.1 Online Reasoning and Verification

Problem Solution Generation Given an initial model π_θ and a training set $D = \{(\mathbf{x_i}, \mathbf{y}_i^*)\}$ consisting of problems $\mathbf{x}_i$, and their corresponding ground-truth answers $\mathbf{y}_i^*$, we begin each RL iteration by sampling a data batch. At iteration t, the model first generates k solutions for each problem in the batch, each comprising a chain-of-thought reasoning followed by a final answer.

Next, the reward is computed for each generated response. Following prior RLVR approaches, we define a rule-based outcome verifier (OV) that incorporates both answer and format correctness:

$$r_o(\mathbf{y}, \mathbf{y}^*) = \begin{cases} 1, & \text{boxed and matched} \\ -0.5, & \text{boxed but not matched} \\ -1, & \text{unboxed} \end{cases} \tag{5}$$

Here "matched" means the final answer in the generated solution $\mathbf{y}$ is mathematically identical to the provided ground truth $\mathbf{y}^*$, and "boxed" means the final answer in $\mathbf{y}$ is wrapped in the `\boxed{}`.

Algorithm 1 RISE (PPO)

Input Language model $\pi_{\theta_{\text{init}}}$; outcome verifier OV; dataset $\mathcal{D}$; rollout number K; generation batch size $\mathcal{B}_g$, verification batch size $\mathcal{B}_v$; verification prompt template $\mathcal{T}$; total iteration N.

Initialize: actor $\pi_\theta \leftarrow \pi_{\theta_{\text{init}}}$, old-actor $\pi_{\theta_{\text{old}}}$, critic π_ϕ, reference π_{ref}

for iteration $= 1$ **to** N **do**

Sample $\mathcal{B}_g$ samples for generation $\mathcal{P}_g = \{(\mathbf{x}_i, \mathbf{y}_i^*)\}_{i=1}^{\mathcal{B}_g} \sim \mathcal{D}$

Get generation batch: ▷ Generate solutions

$\mathcal{G} \leftarrow \left\{(\mathbf{x}_i, \mathbf{y}_i^{(k)}, r_{\text{ov}}(\mathbf{y}_i^{(k)}, \mathbf{y}_i^*)) \;\middle|\; \mathbf{y}_i^{(k)} \sim \pi_\theta(\cdot|\mathbf{x}_i),\, i \leq \mathcal{B}_g,\, k \leq K\right\}$

Select $\mathcal{B}_v$ triples $\mathcal{P}' = \{(\mathbf{x}_i, \mathbf{y}_i, r_i)\}_{i=1}^{\mathcal{B}_v} \subseteq \mathcal{G}$ for verification

$\mathcal{P}_v \leftarrow \left\{\left(\mathcal{T}(\mathbf{x}, \mathbf{y}),\, r\right) \mid (\mathbf{x}, \mathbf{y}, r) \in \mathcal{P}'\right\}$ // each element is a new prob-ans tuple $(\mathbf{x}, \mathbf{y}^*)$

Get verification batch: ▷ Verify generations

$\mathcal{V} \leftarrow \left\{(\mathbf{x}_j, \mathbf{y}_j^{(k)}, r_{\text{ov}}(\mathbf{y}_j^{(k)}, \mathbf{y}_j^*)) \;\middle|\; \mathbf{y}_j^{(k)} \sim \pi_\theta(\cdot|\mathbf{x}_j),\, j \leq \mathcal{B}_v,\, k \leq K\right\}$

Get complete training batch $\mathcal{B} \leftarrow \mathcal{G} \cup \mathcal{V}$

Estimate advantages $\hat{A}$ using Eq. (3) ▷ Joint optimization

Update critic π_ϕ by critic loss in Eq. (4)

Update actor π_θ by actor loss in Eq. (2); update $\theta_{\text{old}} \leftarrow \theta$

end for

Output Optimized actor model π_θ

This produces the generation batch $\mathcal{G} = (\mathbf{x}, \mathbf{y}, r)$, where each element includes the input problem, a model-generated solution, and its associated reward.

Online Solution Verification To construct verification data, we apply a predefined prompt template (see Figure 8) to $\mathcal{G}$, formatting the problem-solution pair into a new verification prompt $\mathbf{x}_{\text{ver}}$ that explicitly states the verification criteria and asks the model to critique the provided solution and assign a score. Since the criteria specified in the prompt are exactly the rules employed by the outcome verifier, the original reward r from the generation phase is reused as the ground-truth score for the verification task. Thus, for each triple $(\mathbf{x}, \mathbf{y}, r) \in \mathcal{G}$, we construct the verification data as $(\mathbf{x}_{\text{ver}} = \mathcal{T}(\mathbf{x}, \mathbf{y}),\ \mathbf{y}_{\text{ver}}^* = r)$. In practice, the amount of verification data is controlled by the verification batch size, allowing a flexible balance between problem-solving and solution verification.

For each verification prompt, the model generates K responses, each containing a natural language critique and a final score. These responses are evaluated using the same OV criteria as Eq. (5). Concretely, the verification reward is determined by comparing the score extracted from the model's verification response with the reward assigned by the rule-based outcome verifier for the same judged solution. A reward of $+1.0$ is given when the model correctly predicts the score in the specified format (e.g., `\boxed{1}`); a reward of -0.5 is assigned when the predicted score is incorrect but the format is valid; and a reward of -1.0 is applied when the response is in an invalid format (missing the `\boxed{}` wrapper). This process yields the verification batch $\mathcal{V} = \{(\mathbf{x}, \mathbf{y}, r)\}$, maintaining the same structure as the problem-solution batch.

4.2 RL Integration

The preceding *Online Reasoning and Verification* stage is architecturally agnostic to the choice of the underlying policy-gradient algorithm; its only algorithm-specific interface is the advantage estimator $\hat{A}$ used in the policy update. In our formulation, advantage values are computed from a concatenated mini-batch $\mathcal{B} = \mathcal{G} \cup \mathcal{V}$, encompassing samples from both the reasoning and verification tasks. Since every sample in $\mathcal{B}$ is annotated with a scalar reward and the action log-probability under the current policy π_θ, any estimator that maps a sequence of state–action–reward tuples to an advantage can be incorporated without further structural change.

For our main experiments with PPO (see Algorithm 1), we apply GAE (Eq. 3) independently to each trajectory. The generation and verification trajectories are jointly processed within the same stochastic gradient descent (SGD) step, enabling the actor to be optimized with respect to both types of data. Meanwhile, the shared critic learns a unified value function across tasks. PPO's clipping mechanism further ensures that updates remain stable within a consistent trust region.

Table 1: Detailed results of RISE and other baseline methods on various math benchmarks. Zero-RL models are trained under the same setting as RISE, but without the verification objective.

	Reasoning						***Self-Verification***					
Model	**MATH**	**AIME**	**AMC**	**Mine.**	**Olym.**	**Avg.**	**MATH**	**AIME**	**AMC**	**Mine.**	**Olym.**	**Avg.**
GPT-4o	79.0	13.3	55.0	50.0	42.5	48.0	83.4	33.3	67.5	50.4	54.4	57.8
Qwen2.5-1.5B												
Base	2.0	0.0	1.9	0.8	0.6	1.1	19.4	21.9	22.7	15.9	21.1	20.2
Instruct	37.5	0.8	19.4	8.3	11.7	15.5	48.8	22.1	36.5	36.9	29.6	34.8
SFT	10.1	0.0	4.1	1.8	2.0	3.6	19.0	5.8	12.3	10.5	10.9	11.7
Zero-RL	**55.3**	2.1	25.9	**17.4**	19.5	24.0	54.1	5.0	30.7	21.0	23.0	26.8
RISE	54.6	**2.9**	**27.5**	17.2	**19.8**	**24.4**	**75.9**	**85.0**	**70.6**	**66.0**	**74.9**	**74.5**
Qwen2.5-3B												
Base	32.7	1.3	15.3	10.3	10.7	14.1	39.5	13.6	22.5	29.9	21.2	25.3
Instruct	61.0	3.8	34.1	25.6	24.6	29.8	65.6	21.0	45.5	37.6	35.0	40.9
SFT	14.4	0.4	5.3	2.9	2.8	5.2	21.5	2.1	10.9	17.9	13.2	13.1
Zero-RL	64.2	6.7	37.5	**27.4**	**26.6**	32.5	64.9	13.0	39.7	30.3	31.2	35.8
RISE	**64.3**	**7.9**	**42.5**	26.2	**26.6**	**33.5**	**81.0**	**86.3**	**74.4**	**56.1**	**73.6**	**74.3**
Qwen2.5-7B												
Base	38.3	2.1	21.9	11.9	13.2	17.5	58.4	45.9	51.5	48.4	48.4	50.5
Instruct	73.8	10.0	50.6	**35.9**	35.8	41.2	77.2	26.3	57.0	40.2	45.2	49.2
SFT	28.7	0.8	13.8	6.2	7.2	11.3	40.5	36.6	47.4	39.2	36.1	40.0
Zero-RL	74.5	12.1	51.3	34.2	**36.7**	41.7	75.9	21.7	56.5	37.3	41.6	46.6
RISE	**74.8**	**12.5**	**55.9**	34.6	**36.7**	**42.9**	**83.8**	**75.0**	**72.5**	**48.6**	**65.9**	**69.2**

5 Experiment

5.1 Experiment Setup

Dataset We follow the previous study [Zeng et al., 2025] to utilize MATH-Hard (Level 3–5) [Hendrycks et al., 2021] as our training set, which in total comprising 8,523 problems. This training set is used for all SFT baselines, Zero-RL baselines, and RISE models.

Models We conduct our main experiments on three Qwen2.5 models [Yang et al., 2024] with different sizes (i.e., 1.5B, 3B, and 7B) for their strong reasoning capabilities. The RL training of our models is based on the verl [Sheng et al., 2025] framework with a train batch size of 1024 and a mini-batch size of 128. We follow [Zeng et al., 2025] by setting the sampling temperature to 1.0 and rollout 8 responses for each problem. The RISE models have a default verification batch size 128. We set the RL configurations same for RISE models and Zero-RL models, ensuring a fair comparison. Additionally, we include the RL experiments on Qwen3 models in Appendix H.1, where we observe larger performance gains, further demonstrating the effectiveness of RISE on newer models.

Benchmarks We evaluate model performance on standard mathematical reasoning benchmarks: MATH500 [Hendrycks et al., 2021, Lightman et al., 2023], Minerva Math [Lewkowycz et al., 2022], OlympiadBench [He et al., 2024], and competition-level benchmarks AIME 2024 and AMC 2023. Following [Zeng et al., 2025], we generate 8 responses per problem using a sampling temperature of 1.0, and report Pass@1 accuracy [Chen et al., 2021] as the evaluation metric. Solution correctness is based on exact match of the final answer, and verification correctness depends on the agreement between the predicted verification score and the score from the outcome verifier.

5.2 Experimental Results

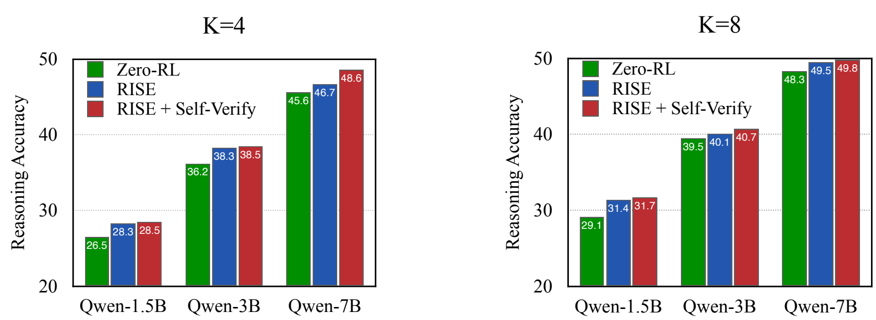
Table 1 presents the results of RISE across model sizes and benchmarks.

RISE significantly enhances self-verification capabilities while improving reasoning performance. RISE models consistently outperform their Zero-RL counterparts across both reasoning and self-verification tasks on all model sizes. The improvement in self-verification is particularly dramatic: RISE-1.5B achieves 74.5% average verification accuracy compared to just 26.8% for Zero-RL, representing a 47.7 percentage point improvement. This demonstrates that our integrated approach successfully develops robust self-verification skills while simultaneously enhancing problem-solving capabilities. Notably, **the verification improvements are particularly pronounced on the most challenging benchmarks** like AIME24 and OlympiadBench, suggesting that RISE enables models to better recognize their limitations and errors on difficult problems.

Scaling model size improves reasoning performance while maintaining strong verification capabilities. Scaling model size from 1.5B to 7B parameters consistently enhances reasoning performance across all benchmarks. Interestingly, the verification performance of RISE models remains consistently high across model sizes, with all models achieving over 69% average accuracy. The ability to maintain strong verification capabilities while scaling reasoning performance aligns with our contribution of developing a framework that simultaneously improves both critical capabilities.

RISE models outperform standard SFT and base models by a substantial margin. The results clearly demonstrate that RISE models substantially outperform their SFT and base model counterparts. For instance, RISE-7B achieves 42.9% average reasoning accuracy compared to just 11.3% for SFT-7B and 17.5% for the base model. This substantial improvement demonstrates the effectiveness of our integrated, online learning approach built upon state-of-the-art RLVR paradigms.

5.3 Test-Time Scaling with Self-verification



(a) k=4 (b) k=8

Figure 2: Test-time scaling performance across different sampling budgets ("k").

To further evaluate the benefits of the enhanced self-verification capabilities developed by RISE, we investigate its impact at test-time using self-consistency majority voting ("maj@k") [Wang et al., 2023] and verification-weighted majority voting. In the latter, following [Wang et al., 2024], the model's self-generated verification scores for each candidate solution are used to weight its contribution in the majority vote. The results, presented in Figure 2, compare RISE models against Zero-RL models across different sampling budgets ("k=4" and "k=8"). With the lightweight verification cost (Appendix G.2), the experiments maintain fairness in terms of total sampling cost.

RISE consistently improves test-time scaling performance with self-verification and majority voting. RISE models outperform their Zero-RL counterparts when employing test-time strategies such as majority voting and verification-weighted selection. Across model sizes and sampling budgets, RISE achieves higher average accuracy, with the largest relative gains observed when self-verification scores are used to re-rank majority votes. For example, RISE-7B achieves an average score of 49.8% with $k = 8$ + self-verify, surpassing Zero-RL's 48.3% under the same conditions. This consistent improvement confirms the effectiveness of integrating self-verification in both training and inference.

Verification-weighted voting delivers further accuracy gains. Incorporating self-verification scores as weights in the voting process leads to additional accuracy improvements for all RISE models. For instance, RISE-3B and RISE-7B models see improvements of +0.2% and +1.9% over standard majority voting at the $k = 4$ budget, respectively. These results indicate that the self-verification policy learned by RISE provides meaningful confidence signals for answer calibration.

5.4 Comparison with Off-the-shelf Verifiers

We further compare the verification accuracy between our RISE models as self-verifiers and off-the-shelf verifiers, including a discriminative verifier (Math-Shepherd-7B [Wang et al., 2024]) and a generative verifier (GPT-4o [OpenAI, 2024]). Specifically, we use the verification prompt in Figure 8 for both RISE models and GPT-4o and adhere to the original logic for Math-Shepherd to verify the generated solutions. The results of RISE-1.5B, 3B and 7B are presented in Figure 3, which show that RISE models consistently outperform existing outcome verifiers in judge their solutions' correctness. This serves as a great advantage for the model to further improve its test-time performance, by leveraging the self-verification signal either externally or internally. Detailed results and evaluation implementation can be found in Appendix D.

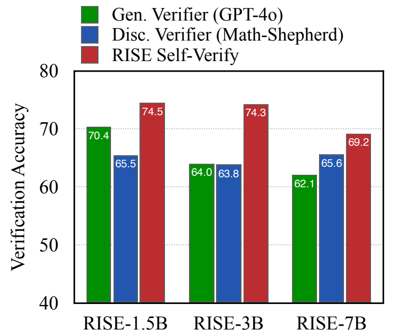


Figure 3: Comparisons between RISE (self-verify) and off-the-shelf verifiers.

5.5 Analysis

In this section, we provide some insights into how RISE improves performance.

RISE demonstrates robust and simultaneous learning of problem-solving and self-verification, with self-verification skills developing notably faster across different model scales. The learning curves, illustrated by the reward trends in Figure 4, reveal a consistent and steady improvement in both reasoning (problem-solving) and self-verification rewards throughout the RL training process for all evaluated models. This uniform positive progression across varying model sizes highlights the robustness of the RISE framework in co-training these two abilities, a core contribution of our work. A key observation is that the self-verification reward generally exhibits a more rapid increase and reaches a higher relative level compared to the problem-solving reward within the same training period. This aligns with the "Generation-Verification Gap" posited by Song et al. [2024], suggesting that models might acquire verification capabilities more readily than problem-solving abilities.

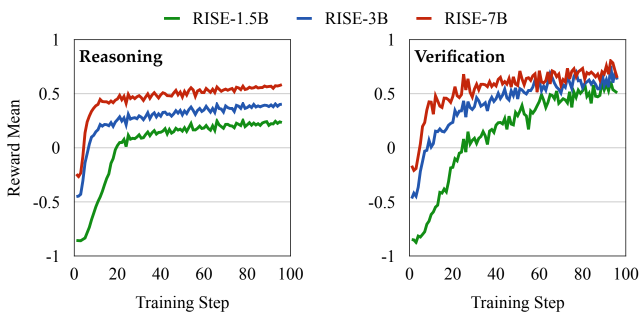


Figure 4: Reasoning and verification reward at train time.

Impact of Verification Compute In the main experiment, we trained our RISE models with a verification batch size of 128, which is 12.5% of the generation batch 1024. We further explore the model performance by scaling up the verification data batch, i.e., the train-time compute, up to 100% of the generation batch. In practice, we choose the percentages S of $\{0, 12.5\%, 25\%, 50\%, 100\%\}$ and perform experiment on our RISE models. The results are shown in Figure 5. The problem-solving performance first increases, and then slightly decreases, and finally increases across the benchmark, maintaining at a high level. Furthermore, the verification performance keeps scaling with more training compute, indicating the robustness of scalability of our RISE method.

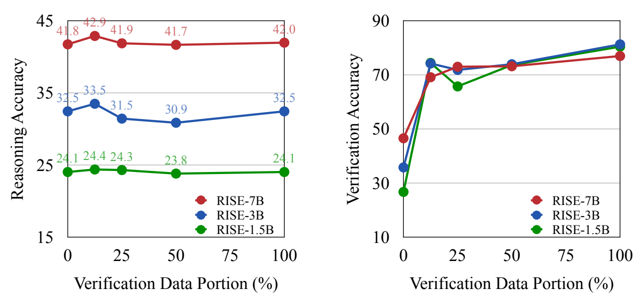


Figure 5: Impact of verification data ratio.

Online and Offline Verification We validate the effectiveness of online verification by comparing it to a offline variant, where the verification data are collected from a distant policy and directly added to the training set. In practice, we select the policy at step 96 (final step) of the Zero-RL model and use its generated solutions to construct offline verification set. In the experiment, we keep the portion of verification data and the training batch size same to eliminate other influence factors, making the only changing variable the source of the verification data. Figure 6 shows the results. While the problem-solving performance of offline verification models are on par with the online ones, they have a significant drop in terms of self-verification accuracy, which indicates the importance of online verification designed in our RISE method.

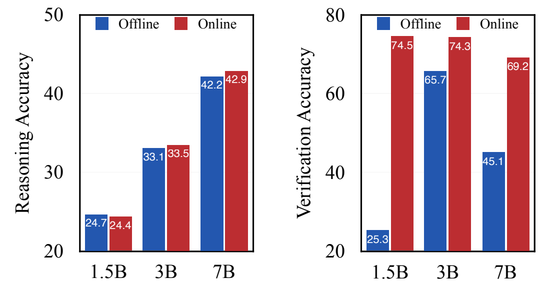


Figure 6: Comparisons between online and offline verification.

Enhanced Verification for Reasoning Besides leveraging the self-verification ability externally during the test-time as in § 5.3, such ability is also internalized by the model to enhance its reasoning generation process. To analysis this effect from the quantitative perspective, we measure the *Verification Frequency* and *Self-Verified Reasoning Accuracy* in model's problem-solving process. Inspired by [Yeo et al., 2025], we use a set of verification keywords to select the responses containing self-verification behaviors, namely {"verify", "verifying", "recheck", "validate", "re-evaluate"}. Figure 7 (left) presents the first result, where the proportion of responses containing explicit verification phrases consistently rises after RISE training. The increase is modest for the 1.5 B model (+0.05 absolute) but becomes more pronounced as model scale grows (+1.09 for 3 B and +1.05 for 7 B). Since the decoding hyper-parameters are fixed, the higher verification frequency arises from policy updates that jointly optimize for both solution and verification rewards, an effect enabled by our first contribution, which integrates verification into a unified RL training loop rather than treating it as a post-hoc heuristic. This mechanism allows models to check their intermediate solutions before reaching the final conclusion in more cases.

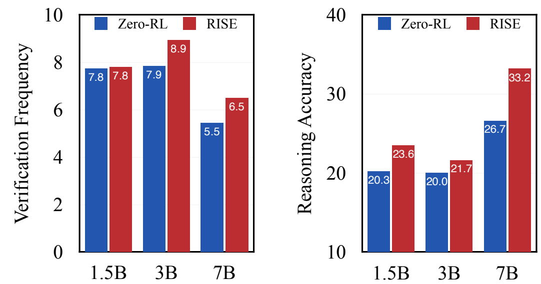


Figure 7: Verification frequency (left panel) and its effectiveness in reasoning (right panel).

Table 2: Examples of verification behaviors of different methods on Qwen2.5-7B: RISE can incur more effectiveness self-verification in the reasoning.

Problem	What is the least positive integer multiple of 30 that can be written with only the digits 0 and 2?
Zero-RL	To find the least positive integer multiple of 30 that can be written with only the digits 0 and 2, we need to consider the properties of multiples of 30 . . . Since 222 is divisible by both 2 and 3, it is a multiple of 30. **Let's verify:** 222 is even, so it is divisible by 2. The sum of the digits of 222 is $2+2+2=6$, which is divisible by 3. Thus, 222 is indeed a multiple of 30. Therefore, the least positive integer multiple of 30 that can be written with only the digits 0 and 2 is $\boxed{222}$.
RISE	To find the least positive integer multiple of 30 that can be written with only the digits 0 and 2, we need to . . . Therefore, the smallest number N is 2220. **To verify:** 2220 ends in 0, so it is a multiple of 10. The sum of the digits of 2220 is $2+2+2+0=6$, which is a multiple of 3. Since 2220 satisfies both conditions, it is the least positive integer multiple of 30 that can be written with only the digits 0 and 2. Thus, the final answer is $\boxed{2220}$.

Simply asking the model to "verify" is not sufficient; the verification needs to be correct. In Figure 7 (right), we measure the accuracy of problems on which the model self-verifies. RISE yields clear improvements over Zero-RL for self-verified reasoning accuracy at every scale: +3.3% (1.5 B), +1.7% (3 B), and a striking +6.5% (7 B). These gains show that RISE's online verifier reward shapes the policy toward not only producing more verifications, but also ones that align with ground truth.

The case in Table 2 illustrates this distinction. Zero-RL "verifies" 222 by merely restating divisibility rules, overlooking the necessity of a trailing zero for multiples of 30. RISE, in contrast, recomputes both the units-digit constraint and the digit-sum divisibility test, ultimately validating the answer of 2220. Such structured, multi-step verification reflects a genuinely internalized skill and explains the quantitative trend that higher verification frequency correlates with higher reasoning accuracy.

6 Conclusion

In this work, we introduced RISE, a novel online reinforcement learning framework that integrates problem-solving with explicit self-verification training for LLMs. By leveraging verifiable rewards for both generation and verification tasks within a unified RL objective, RISE aims to overcome superficial self-reflection and foster more robust reasoning capabilities. Our experiments, primarily using PPO with Qwen2.5 models on diverse mathematical reasoning benchmarks, demonstrate that RISE significantly improves problem-solving accuracy while concurrently developing strong self-verification skills. Notably, RISE models learn to verify their own solutions more effectively than off-the-shelf verifiers and further benefit from leveraging this capability at test time. Together, RISE provides a promising direction for building more reliable and self-aware LLM reasoners, adaptable to various policy-gradient algorithms and extendable to other domains with verifiable rewards.

Acknowledgment

This paper was supported by the Guangdong Basic and Applied Basic Research Foundation (No. 2024A1515010145) and the Shenzhen Science and Technology Program (Shenzhen Key Laboratory Grant No. ZDSYS20230626091302006).

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12248–12267, 2024.

Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983. doi: 10.1109/TSMC.1983.6313077.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL `https://arxiv.org/abs/2107.03374`.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Jiaxuan Gao, Shusheng Xu, Wenjie Ye, Weilin Liu, Chuyi He, Wei Fu, Zhiyu Mei, Guangju Wang, and Yi Wu. On designing effective rl reward at training time for llm reasoning. *arXiv preprint arXiv:2410.15115*, 2024.

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

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, 2024.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, et al. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 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.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025.

Audrey Huang, Adam Block, Dylan J Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, Jordan T Ash, and Akshay Krishnamurthy. Self-improvement in language models: The sharpening mechanism. *arXiv preprint arXiv:2412.01951*, 2024.

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. Efficient test-time scaling via self-calibration. *arXiv preprint arXiv:2503.00031*, 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.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626, 2023.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

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.

Zi Lin, Sheng Shen, Jingbo Shang, Jason Weston, and Yixin Nie. Learning to solve and verify: A self-play framework for code and test generation. *arXiv preprint arXiv:2502.14948*, 2025.

Zichen Liu, Changyu Chen, Wenjun Li, Tianyu Pang, Chao Du, and Min Lin. There may not be aha moment in r1-zero-like training — a pilot study. `https://oatllm.notion.site/oat-zero`, 2025. Notion Blog.

Ruotian Ma, Peisong Wang, Cheng Liu, Xingyan Liu, Jiaqi Chen, Bang Zhang, Xin Zhou, Nan Du, and Jia Li. S^2r: Teaching llms to self-verify and self-correct via reinforcement learning. *arXiv preprint arXiv:2502.12853*, 2025.

OpenAI. Gpt-4o, 2024. URL `https://openai.com/index/hello-gpt-4o/`. Accessed: 2025-09-18.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

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

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

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297, 2025.

Wenlei Shi and Xing Jin. Heimdall: test-time scaling on the generative verification. *arXiv preprint arXiv:2504.10337*, 2025.

Yuda Song, Hanlin Zhang, Carson Eisenach, Sham Kakade, Dean Foster, and Udaya Ghai. Mind the gap: Examining the self-improvement capabilities of large language models. *arXiv preprint arXiv:2412.02674*, 2024.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*, 2024.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning, 2020. URL `https://github.com/huggingface/trl`.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=1PL1NIMMrw`.

Wikipedia contributors. P versus np problem — Wikipedia, the free encyclopedia, 2025. URL `https://en.wikipedia.org/w/index.php?title=P_versus_NP_problem&oldid=1287287306`. [Online; accessed 13-May-2025].

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Zhiheng Xi, Dingwen Yang, Jixuan Huang, Jiafu Tang, Guanyu Li, Yiwen Ding, Wei He, Boyang Hong, Shihan Do, Wenyu Zhan, et al. Enhancing llm reasoning via critique models with test-time and training-time supervision. *arXiv preprint arXiv:2411.16579*, 2024.

Zhihui Xie, Jie Chen, Liyu Chen, Weichao Mao, Jingjing Xu, and Lingpeng Kong. Teaching language models to critique via reinforcement learning. *arXiv preprint arXiv:2502.03492*, 2025.

Wei Xiong, Hanning Zhang, Chenlu Ye, Lichang Chen, Nan Jiang, and Tong Zhang. Self-rewarding correction for mathematical reasoning. *arXiv preprint arXiv:2502.19613*, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 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.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*, 2025.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-rewarding language models. In *Forty-first International Conference on Machine Learning*, 2024.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?, 2025. URL `https://arxiv.org/abs/2504.13837`.

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

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The contributions of this paper, including the proposed method, performance results, ablation studies, and further analysis, are clearly outlined in the abstract and the introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: Due to space limit, we discuss the limitations of this work and future work in detail in Appendix A.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: This is an empirical paper which does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Experiment details are provided in Section 5.1, Appendix C, and Appendix D. Prompt templates are provided in Appendix B.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code and documentations are provided in the supplementary materials to facilitate the reproduction of experiment results.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Experiment setting and details are provided in Section 5.1, Appendix C and D. Prompt templates are provided in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Following common practice in the RL post-training literature, we do not report error bars due to the high computational cost of running multiple trials. Additionally, we follow established conventions by reporting the pass ratio, which inherently averages over multiple generations.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

Justification: We provide a detailed description of the experimental environment in Appendix F.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: We have ensured that the research conducted in our paper complies with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: The paper does not identify any immediate or direct societal risks or benefits, as the proposed method primarily introduces a training framework for reasoning tasks such as mathematics. We emphasize that future works building upon our method should be developed and applied responsibly.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The work in this paper does not involve such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: License details are provided in Appendix E. We have fully complied with the licensing terms and usage policies for all data, code, and models used in our experiments.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: For the new assets introduced in this paper, detailed documentation is provided alongside the assets in the supplementary materials.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.

- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The LLM is used only for writing and does not impact the core methodology, scientific rigorousness, or originality of the research in this paper.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Limitations

On experiments: First, after training the LLM to self-verify its outputs, it may act as a pseudo–rule-based verifier to further guide RL training without relying on ground-truth labels. This opens the possibility of self-improvement on unlabeled data, which we leave for future work as it lies beyond the main scope of this study. Second, the training data is exclusively drawn from math reasoning tasks. Given the observed generalization performance, we expect the effectiveness of RISE to persist when trained on other domains with verifiable rewards. Third, the metrics adopted for evaluating verification effectiveness have certain limitations, as they do not capture the necessity of the emerged verification behaviors. In this work, we use final reasoning accuracy as an indicator of goodness, while future work could develop more fine-grained criteria that better quantify the necessity such as partial trajectory correctness or model uncertainty.

On algorithm: RISE trains the LLM as a generative verifier that produces natural language critiques, which has shown benefits for both reasoning and test-time scaling. An alternative design is to co-train a discriminative verifier with a separate classification head. While it remains unclear how RISE would perform in that setting, we believe this does not affect our main contributions which demonstrate the effectiveness of generative verification in improving problem-solving capabilities.

B Prompt Templates

Prompt Template

Below you are presented with a question and a tentative response. Your task is to evaluate and assign a rating to the response based on the following clear criteria:

Rating Criteria:

1. Missing final answer enclosed in \\boxed{} at the end: assign \\boxed{-1}.
2. Correct response with the final answer enclosed in \\boxed{} at the end: assign \\boxed{1}.
3. Incorrect response with the final answer enclosed in \\boxed at the end: assign \\boxed{-0.5}.

Question Begin
{**Question**}
Question End

Response Begin
{**Response**}
Response End

Briefly summarize your analysis, then clearly state your final rating value enclosed in \\boxed{} at the end.

Figure 8: Verification prompt used in the experiment.

Prompt Template

<|im_start|>system
Please reason step by step, and put your final answer within \\boxed{}.<|im_end|>
<|im_start|>user
{**Input**}<|im_end|>
<|im_start|>assistant

Figure 9: Prompt template used in the training and evaluation.

Prompt Template

<|im_start|>system
You are a helpful assistant.<|im_end|>
<|im_start|>user
{**Input**} Please reason step by step, and put your final answer within \\boxed{}.<|im_end|>
<|im_start|>assistant

Figure 10: Prompt template used for Qwen base model evaluation.

C Training Details

During RL training, we set the actor's clipping ratio to 0.2 and disable the KL penalty loss. The critic uses a clipping range of 0.5. The learning rates are fixed at 5×10^{-7} for the actor and 9×10^{-6} for the critic. The KL divergence coefficient is set to 1×10^{-2}. We limit the maximum response length to 3000 tokens, which already results in a negligible clip ratio. The full dataset is trained for 12 epochs. This configuration is shared across both the Zero-RL and RISE models.

For the SFT baseline models, we use a batch size of 32 and apply a cosine learning rate scheduler with a learning rate of 2×10^{-5} and a warm-up ratio of 1×10^{-3}. The dataset is trained for 3 epochs.

D Evaluation Details

D.1 Verification Evaluation with Other Verifiers

To evaluate the verification accuracy of RISE and GPT-4o (prompted as a verifier), we extract the final verification score from each response and normalize it to either +1 (predicted correct) or 0 (predicted incorrect). The normalization is defined as:

$$s_{\text{normalized}} = \begin{cases} 1, & s = 1 \\ 0, & \text{otherwise} \end{cases}$$

which aligns with the criteria used by the rule-based outcome verifier. For the Math-Shepherd model, which outputs a continuous score in the range $[0, 1]$ (with 0 indicating the solution/step is predicted to be incorrect and 1 indicating correct), we apply a threshold of 0.5 for normalization:

$$s_{\text{normalized}} = \begin{cases} 1, & s > 0.5 \\ 0, & \text{otherwise} \end{cases}$$

After normalization, we compute verification accuracy by directly comparing the predicted scores against those returned by the outcome verifier.

D.2 Weighted Majority Voting with Self-Verification

In § 5.3, we explore the combination of self-consistency and self-verification in test time following [Wang et al., 2024]. In practice, we initially classify solutions into distinct groups according to their final answers. Following that, we extract and normalize the self-verification scores and normalize them as +1 (correct) and 0 (incorrect) as in D.1. Since our score are binary and could lead to an unexpected zero sum, we integrate Laplace smoothing for computing the mean score for the answer. Formally, the final selected answer based on N candidate solutions is:

$$a_{\text{maj@N+self-verify}} = \text{argmax}_a \underbrace{\sum_{i=1}^{N} \mathbb{I}(a_i = a)}_{\text{frequency}} \cdot \underbrace{\frac{\alpha + \sum_{i=1}^{N} S(p, s_i)}{N + \alpha d}}_{\text{smoothed mean score}} . \tag{6}$$

where $S(p, S_i)$ is the score of the i-th solution assigned by the self-verification. In practice, we set $\alpha = 2$ and $d = 2$ empirically, suggesting a prior belief of a 0.5 average score.

E Licenses

Datasets and Benchmarks. The training dataset is derived from MATH (MIT License). We evaluate on five benchmarks: MATH 500 (MIT License), AIME 2024 (CC0: Public Domain), AMC 2023 (Apache License 2.0), Minerva Math (license not found), and Olympiad Bench (MIT License).

Framework. RL training is based on verl v0.2 (Apache-2.0 license), and SFT training is based on trl [von Werra et al., 2020] v0.14.0 (Apache-2.0 license). Evaluation is performed using vllm framework [Kwon et al., 2023] v0.7.2 (Apache-2.0 License) and the script is based on OpenMathInstruct-2 [Toshniwal et al., 2024].

Models. We train our models based on the Qwen2.5 series. Specifically, Qwen2.5-1.5B[3] and Qwen2.5-7B[4] are released under the Apache License 2.0, while Qwen2.5-3B[5] is released under a custom Qwen Research license. We also compare against Math-Shepherd[6] model (license not found), and GPT-4o (accessed via OpenAI API, governed by OpenAI Terms of Use[7]).

F Computing Resources

All experiments were conducted on machines with AMD EPYC 9K84 96-core CPUs and 8 NVIDIA H20 GPUs. Our code is primarily based on Python 3.12.2 and PyTorch 2.5.1. RL training took approximately 1 day for Qwen-1.5B and Qwen-3B, and 2 days for Qwen-7B. The SFT baseline training required about 2 hours per model, and evaluation took roughly 1 hour per model. In total, the experiments consumed approximately 700 GPU hours. Including preliminary and failed runs, the overall project required more compute.

G Detailed Experiment Results

G.1 Detailed Comparison with off-the-shelf verifiers

In § 5.4, we report the average verification accuracy across the five benchmarks. Here, we present the detailed verification accuracy comparison between RISE models, Math-Shepherd, and GPT-4o on each evaluation benchmark.

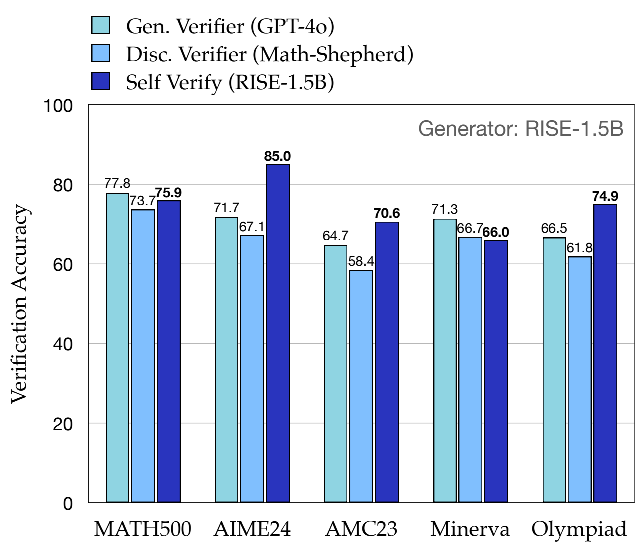


Figure 11: Detailed comparisons of verification accuracy between RISE-1.5B and other verifiers.

[3] `https://huggingface.co/Qwen/Qwen2.5-1.5B`
[4] `https://huggingface.co/Qwen/Qwen2.5-7B`
[5] `https://huggingface.co/Qwen/Qwen2.5-3B`
[6] `https://huggingface.co/peiyi9979/math-shepherd-mistral-7b-prm`
[7] `https://openai.com/policies/row-terms-of-use/`

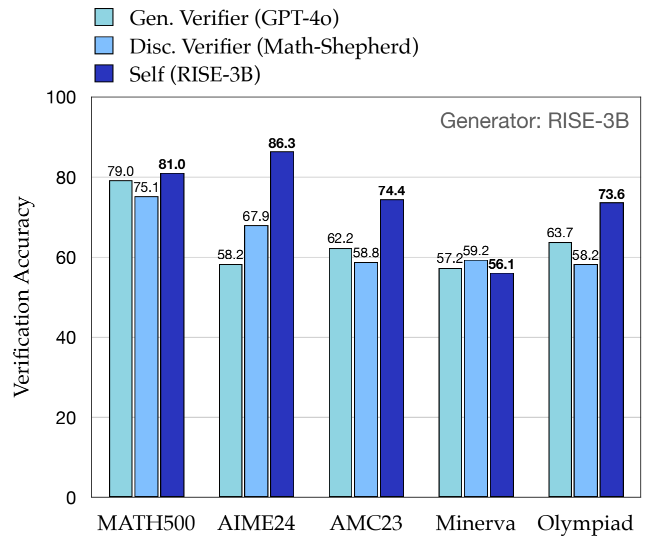


Figure 12: Detailed comparisons of verification accuracy between RISE-3B and other verifiers.

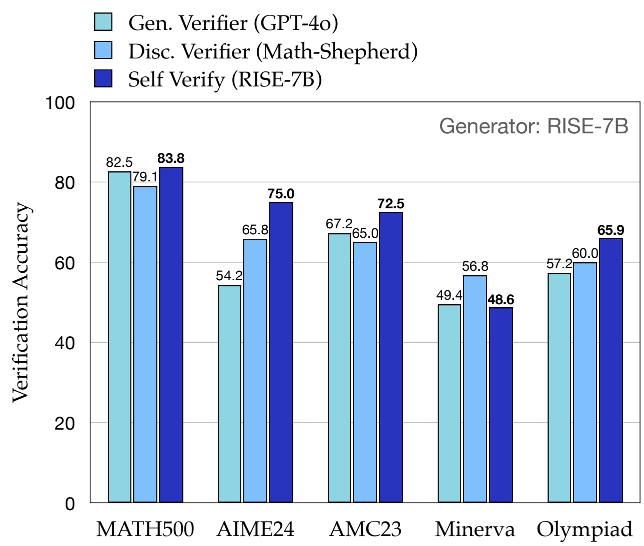


Figure 13: Detailed comparisons of verification accuracy between RISE-7B and other verifiers.

G.2 Verification Cost

Table 3: Average reasoning and verification token usage across RISE models.

Model	Reason.	Veri.	Ratio (V/R)
RISE-1.5B	676	13	**0.02**
RISE-3B	686	94	**0.14**
RISE-7B	693	52	**0.08**

To assess the verification cost, we compute the average verification token usage across the five evaluated benchmarks, as summarized in Table 3. In general, the cost of solution verification is lightweight compared to the problem-solving process, with ratios ranging from 0.02 to 0.14. The verification responses optionally highlight the critical issues and provide the final score succinctly. This property allows us to incorporate the verification outputs from RISE models at test time (e.g., through weighted majority voting) without introducing significant computational overhead, thereby maintaining fairness in comparison to baseline budgets and preserving the robustness of RISE under realistic evaluation settings. We also provide the results with adjusted sampling budgets in Table 4 for reference, where RISE continues to improve upon vanilla majority voting and outperforms the Zero-RL baselines in most cases.

Table 4: Test-time scaling (maj@k) performance under adjusted sampling budgets. Zero-RL and RISE results are measured under baseline budget; SV denotes Self-verify.

Model	RISE + SV Budget	Baseline Budget	Zero-RL	RISE	RISE + SV
$k = 4$					
Qwen2.5-1.5B	4.08 sols	4 sols	26.5	28.3	**28.5**
Qwen2.5-3B	4.52 sols	5 sols	37.3	**38.6**	38.5
Qwen2.5-7B	4.32 sols	4 sols	45.6	46.7	**48.6**
$k = 8$					
Qwen2.5-1.5B	8.16 sols	8 sols	29.1	31.4	**31.7**
Qwen2.5-3B	9.12 sols	9 sols	39.1	39.2	**40.7**
Qwen2.5-7B	8.64 sols	9 sols	49.1	48.2	**49.8**

G.3 Detailed Analysis for Enhanced Verification

In Figure 7, we report the average verification frequency and accuracy of self-verified solutions on the five benchmarks. Here, we present the fine-grained results between RISE models and Zero-RL baseline on each evaluation benchmark.

Table 5: Performance comparison between RISE models and Zero-RL models on verification frequency and effectiveness for the generation.

Method	*Verification Frequency*					
	MATH	**AIME**	**AMC**	**Minerva**	**Olympiad**	**Avg.**
Qwen2.5-1.5B-Zero-RL	6.45	6.67	**7.81**	2.25	**15.59**	7.75
RISE-1.5B	**7.10**	**8.75**	5.31	**2.53**	15.31	**7.80**
Qwen2.5-3B-Zero-RL	**4.90**	8.33	14.29	2.99	8.72	7.85
RISE-3B	4.63	**9.17**	**18.18**	**3.08**	**9.67**	**8.94**
Qwen2.5-7B-Zero-RL	5.30	5.00	7.19	1.56	8.19	5.45
RISE-7B	**6.08**	**7.92**	**8.13**	**1.79**	**8.57**	**6.50**
	Self-Verified Reasoning Accuracy					
Qwen2.5-1.5B-Zero-RL	37.21	0.00	24.00	**24.49**	**15.59**	20.26
RISE-1.5B	**38.73**	**4.76**	**35.29**	23.64	15.31	**23.55**
Qwen2.5-3B-Zero-RL	**45.92**	0.00	14.29	20.00	19.96	20.03
RISE-3B	43.78	**4.55**	**18.18**	**22.39**	**19.54**	**21.69**
Qwen2.5-7B-Zero-RL	62.74	0.00	8.70	**35.29**	26.70	26.68
RISE-7B	**65.43**	**5.26**	**38.46**	28.21	**28.73**	**33.22**

Table 6: Reflection Keywords Rate between RISE models and Zero-RL models.

Method	*Verification Frequency in Generation*					
	MATH	**AIME**	**AMC**	**Minerva**	**Olympiad**	**Avg.**
Qwen2.5-1.5B-Zero-RL	0.16	0.40	0.26	**0.16**	0.29	0.25
RISE-1.5B	**0.19**	**0.45**	**0.29**	**0.16**	**0.32**	**0.28**
Qwen2.5-3B-Zero-RL	0.14	0.40	**0.24**	0.11	0.27	0.23
RISE-3B	**0.16**	**0.45**	0.20	**0.13**	**0.29**	**0.25**
Qwen2.5-7B-Zero-RL	0.13	0.38	0.23	0.08	0.23	0.21
RISE-7B	**0.14**	**0.50**	**0.29**	**0.10**	**0.27**	**0.26**

G.4 Reflection Keywords Analysis

Following Yeo et al. [2025], we track the self-reflection keywords {“wait”, “however”, “alternatively”, “retry”, “recheck”} to quantitatively measure the general reflection behaviors beyond the self-verification among the model problem-solving responses. In practice, we sum the total word counts for the keyword set and normalize it by the dataset size.

The results in Table 6 show that our RISE model constantly have a higher level of reflection-related behaviors than the Zero-RL models, indicating the positive effect of self-verification training.

H Generalization Results

H.1 Generalization on Qwen3 Models

Table 7: Performance comparison of RISE and baseline methods for Qwen3 models.

	Reasoning						***Self-Verification***					
Model	**MATH**	**AIME**	**AMC**	**Mine.**	**Olym.**	**Avg.**	**MATH**	**AIME**	**AMC**	**Mine.**	**Olym.**	**Avg.**
Qwen3-4B-Base												
Base	39.4	6.3	24.1	12.6	17.8	20.0	60.9	72.6	61.9	**59.4**	63.8	63.7
Zero-RL	73.7	**13.3**	45.9	29.5	37.2	39.9	73.7	39.9	52.9	37.9	47.8	50.4
RISE	**77.8**	12.9	**52.8**	**43.4**	**40.6**	**45.5**	**87.4**	**79.6**	**70.9**	50.8	**68.0**	**71.3**
Qwen3-8B-Base												
Base	42.5	8.3	28.4	15.4	18.4	22.6	67.0	65.5	64.4	**62.9**	62.0	64.4
Zero-RL	77.6	13.8	58.1	37.7	41.6	45.7	79.7	54.1	68.8	46.9	56.9	61.3
RISE	**83.0**	**21.3**	**59.4**	**48.4**	**44.4**	**51.3**	**91.8**	**85.4**	**87.4**	53.4	**72.2**	**78.1**

To further demonstrate the effectiveness of RISE, we conduct a new set of experiments on the latest Qwen3 models (Apache 2.0 License). As the mathematical reasoning capability of Qwen3 has been shown to be substantially higher than that of Qwen2.5 [Yang et al., 2025], we construct a more challenging training set to better match their ability. Specifically, we downsample the DeepMath-103K [He et al., 2025] dataset (MIT License) by difficulty level to obtain a 10K subset. Using this data, we train both Qwen3-4B-Base and Qwen3-8B-Base models with the Zero-RL (vanilla PPO) baseline and our RISE, under identical configurations except for the inclusion of the verification objective. For RISE models, we set the verification batch size to 128 by default, consistent with the Qwen2.5 experiments. After training, we follow the same evaluation protocols as in the main experiment to assess model performance on both problem-solving and solution verification tasks. The results, summarized in Table 7, show that RISE achieves substantial improvements in reasoning accuracy over the Zero-RL baseline, with average gains of +5.6% and +5.6% for the 4B and 8B models, respectively. Verification accuracy also improves markedly (+20.9% for the 4B model and +16.8% for the 8B model). Together with our main experiments on Qwen2.5, these results demonstrate the generalizability and effectiveness of RISE across different model scales and families.

H.2 Generalization on Diverse Tasks

Table 8: Performance of RISE and baseline models on diverse tasks.

Model	**MMLU-Pro**	**GPQA**	**HumanEval**	**Veri. Acc. (Avg)**
Qwen2.5-1.5B-Zero-RL	19.7	20.0	39.9	21.4
RISE-1.5B	**21.3**	**23.0**	**44.7**	**67.7**
Qwen2.5-7B-Zero-RL	46.3	27.8	63.7	49.5
RISE-7B	**47.4**	**28.3**	**64.1**	**62.1**

To evaluate the cross-domain generalization of RISE beyond mathematical reasoning, we further conduct a zero-shot transfer study on RISE models. Concretely, we directly test the math-tuned models on diverse tasks, including general knowledge (MMLU-Pro), science (GPQA-Diamond), and code generation (HumanEval). The results in Table 8 show that RISE-1.5B and RISE-7B consistently outperform their Zero-RL baselines in both reasoning and verification accuracy across these out-of-distribution domains. These findings indicate that the self-verification ability learned by RISE is robust and transferable, providing benefits even in tasks unseen during training. Future work could try applying RISE to broader domains where verifiable reward signals are available.

H.3 Comparison to Separate Training of Reasoning and Verification

While our main focus is to improve the reasoning capability of a single LLM rather than building a multi-agent system with specialized components, we provide pilot results on separately fine-tuned solver–verifier models using the medium-sized Qwen2.5-3B. Specifically, we use the same MATH-Hard dataset and finetune an additional verifier model on the Zero-RL model's (i.e., solver) solutions, formatted as verification problems. As shown in Table 9, the results averaged over five benchmarks indicate that RISE achieves higher overall reasoning performance, despite being slightly less verification-accurate than the separately trained verifier. Unlike this separate training scheme, RISE improves model performance holistically and scales more efficiently, requiring significantly less training and inference cost.

Table 9: Reasoning and verification accuracy for Qwen2.5-3B Finetuned Models.

Model	Reason.	Veri.
Base	14.1	25.3
Zero-RL (Solver) + Zero-RL (Verifier)	32.5	**78.3**
RISE	**33.5**	74.3

H.4 Verification on External Solution Set

Although we mainly use the verification ability of RISE models for self-improvement purposes, we construct a test set with external solutions to address potential concerns about verification generalizability. Specifically, we use GPT-4o's solutions on the five evaluated benchmarks and ask the models to verify their correctness. The results are shown in Table 10, which indicate that RISE models still maintain a high level of verification accuracy on the external solutions. This suggests that the verifier learned by RISE generalizes well beyond its own generation style and has acquired a robust notion of correctness.

Table 10: Verification performance of RISE models and other verifiers on the external solution set.

Verifier Model	Verification Accuracy
RISE-1.5B	67.9
RISE-3B	74.4
RISE-7B	70.7
GPT-4o	57.8
Math-Shepherd	58.9

I Cases Study

Case Study

Analysis:
The response provided contains several errors and incorrect calculations. Here's a step-by-step breakdown of the issues:
1. **Triangle Type Identification**: The response incorrectly identifies the triangle as a right triangle based on the Pythagorean theorem, but it does not correctly apply the theorem to confirm the right angle.
2. **Angle Bisector Theorem Application**: The Angle Bisector Theorem is applied incorrectly, leading to an incorrect ratio for BD and DC.
3. **Length Calculation of AD**: The length of the angle bisector AD is calculated incorrectly using the formula for the length of the angle bisector in a right triangle.
4. **Power of a Point Theorem**: The application of the Power of a Point theorem is incorrect, leading to an incorrect value for AP.

Final Rating:
\\boxed{-0.5}

Figure 14: Example verification returned by RISE-7B on AIME 24.