

# AgentV-RL: Scaling Reward Modeling with Agentic Verifier

Jiazheng Zhang<sup>1\*</sup>, Ziche Fu<sup>1\*</sup>, Zhiheng Xi<sup>1\*</sup>, Wenqing Jing<sup>1</sup>, Mingxu Chai<sup>1</sup>, Wei He<sup>1</sup>, Guoqiang Zhang<sup>1</sup>, Chenghao Fan<sup>2</sup>, Chenxin An<sup>3</sup>, Wenxiang Chen<sup>1</sup>, Zhicheng Liu<sup>4</sup>, Haojie Pan<sup>4</sup>, Dingwei Zhu<sup>1</sup>, Tao Gui<sup>5,6†</sup>, Qi Zhang<sup>5,6</sup>, Xuanjing Huang<sup>5,6</sup>

<sup>1</sup>College of Computer Science and Artificial Intelligence, Fudan University

<sup>2</sup>Huazhong University of Science and Technology <sup>3</sup>The University of Hong Kong

<sup>4</sup>ByteDance Seed <sup>5</sup>Institute of Trustworthy Embodied AI, Fudan University

<sup>6</sup>Shanghai Key Laboratory of Multimodal Embodied AI

jzzhang24@m.fudan.edu.cn, tgui@fudan.edu.cn

## Abstract

Verifiers have been demonstrated to enhance LLM reasoning via test-time scaling (TTS). Yet, they face significant challenges in complex domains. Error propagation from incorrect intermediate reasoning can lead to false positives for seemingly plausible solutions, while lacking external grounding makes verifiers unreliable on computation or knowledge-intensive tasks. To address these challenges, we propose **Agentic Verifier**, a framework that transforms reward modeling into a multi-turn, tool-augmented deliberative process. We introduce complementary forward and backward agents: one traces solutions from premises to conclusions, while the other re-checks conclusions against their underlying premises. This bidirectional process enables a comprehensive, reliable, and interpretable assessment of solutions. To facilitate practical deployment, we propose **AgentV-RL**. Through proactive exploration and reinforcement learning, the verifier autonomously interleaves tool-use with internal reasoning. Extensive experiments show that Agentic Verifier yields consistent performance gains under both parallel and sequential TTS. Notably, our 4B variant surpasses state-of-the-art ORMs by 25.2%, positioning it as a promising paradigm for agentic reward modeling. Our code is available at [GitHub](#).

## 1 Introduction

Recent milestones on the International Mathematics Olympiad by OpenAI and Google (OpenAI, 2025; DeepMind, 2025) highlight the rapid ascent of reasoning models like Gemini-3 and DeepSeek-Math-V2 (Shao et al., 2025). To push the capability boundaries of LLM, scaling inference-time compute has emerged as a prevalent trend (Muenighoff et al., 2025a; Huang and Yang, 2025; Chen et al., 2025a). Whether through parallel methods

\* Equal contribution.

† Corresponding author.

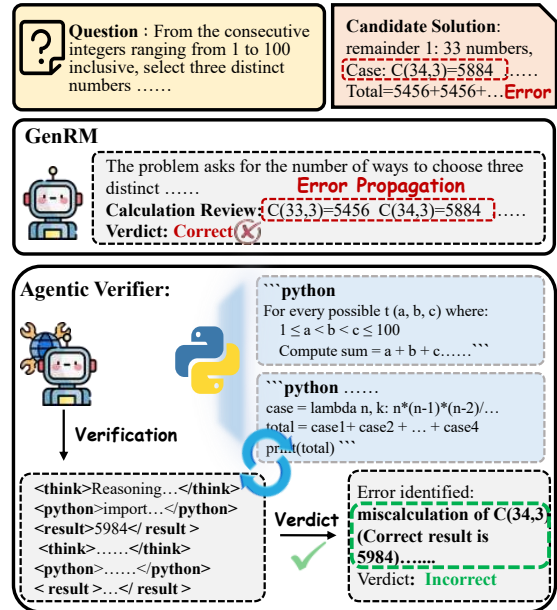


Figure 1: Agentic Verifier v.s. GenRM: while GenRM suffers from error propagation and is misled by incorrect solutions, Agentic Verifier ensures a rigorous review with external grounding.

(e.g., Best-of- $N$ ) or sequential refinement, the efficacy of Test-Time Scaling (TTS) is fundamentally dependent on the Reward Model, i.e., verifier, serving as the critical compass for guiding the search process and discerning solution quality.

Existing reward models, represented by outcome-level RM (ORM, Zhong et al., 2025; Wang et al., 2024a; Zhang et al., 2025a) and process-level RM (PRM, Zhang et al., 2025c; Chen et al., 2025c; Lightman et al., 2024; Cui et al., 2025), only output scalar values while lacking interpretability. Though recent efforts utilize the next-token prediction objective for training Generative Reward Model (GenRM, Mahan et al., 2024a; Zhang et al., 2025b; Chen et al., 2025d; Liu et al., 2025b), this line of work typically employs single-turn reasoning to assess the candidate solution. Specifically, this prevalent paradigm suffers from critical limitations in Figure 1:

(1) **Error propagation**, since LLM are mostly trained on correct or near-correct solutions, they struggle to obtain correct verdict conditioned on flawed solution, easily misled by superficially plausible but incorrect answers (Zhang et al., 2025b). (2) **External grounding**, verifiers often struggle with computation-intensive or knowledge-heavy domains. Without integration with symbolic solvers or external tools (Dong et al., 2025a; Feng et al., 2025; Dong et al., 2025b), they are prone to hallucinations that render unstable performance. These limitations highlight the trend for a paradigm shift towards agentic reward modeling through multi-turn reasoning integrated with external tools.

To bridge this gap, we propose **Agentic Verifier**, a multi-agent framework that emulates rigorous human-like checking. Inspired by mathematical proof strategies, Agentic Verifier coordinates two specialized agents: the forward agent and the backward agent. The forward agent, responsible for sufficiency checking by tracing the logical flow from premises to conclusion, and the backward agent performs necessity checking by validating that the whether the solution is grounded in the problem constraints. Both agents are empowered with multi-turn reasoning and tool-augmented verification capabilities: they can iteratively decompose complex solutions into verifiable sub-steps, invoke external tools such as code interpreters for numerical calculation. Together, this collaborative mechanism allows a comprehensive review that ensures a reliable verdict, proactively identifying immediate flaws or unwarranted content.

To address the challenges inherent in training multi-agent systems, we introduce **AgentV-RL** to distill this multi-agent capacity into a single model. This recipe comprises an end-to-end synthetic data engine that automatically generates verification trajectories and conducts quality control, and a two-stage training schema to unlock the reasoning potential. By automatically constructing verification trajectories that encompass a broad spectrum of logical and computational challenges, this synthetic engine not only alleviates data scarcity but also ensures comprehensive coverage of difficult reasoning patterns. Building on this foundation, our two-stage training recipe is designed to empower the verifier multi-turn, long-horizon, and tool-integrated reasoning by rejection sampling fine-tuning followed by RL.

Finally, we conduct thorough experiments to re-

view the effectiveness of Agentic Verifier under extensive settings. For parallel TTS, i.e., BoN (Best-of-N), our Agentic Verifier outperforms proprietary reasoning models and off-the-shelf BT-RMs on mathematics. Remarkably, our 4B variant consistently outperforms INF-ORM-Llama3.1-70B (Minghao Yang, 2024), an outcome reward model with ten times more parameters. For sequential TTS, Agentic Verifier serves as an effective critique model and provides desirable feedback to help correct the actor’s flaws. In-depth analysis in Sec. 4 confirms its efficacy and improves actor performance. Overall, our main contributions are:

- We offer an agentic paradigm for reward modeling, **Agentic Verifier**, orchestrating two specialized agents to proactively identify flaws in seemingly plausible solutions.
- We introduce **AgentV-RL**, a scalable recipe that distills the capabilities of multi-agents into a single LLM, empowering the verifier with multi-turn, long-horizon, and tool-integrated reasoning.
- Empirical experiments demonstrate the efficacy of the proposed method. Notably, our 4B variant outperforms the state-of-the-art ORM and attains up to 25.2% improvements.

## 2 Related Work

**Reward Model (RM).** RMs play a pivotal role in aligning large language models (LLMs) with human preferences. Traditional outcome-level RMs, assign scalar rewards to the complete response based on preference rankings (Zhong et al., 2025; Wang et al., 2024a). To address the limitations of sparse supervision, PRMs provide dense signals by supervising intermediate steps (Zhang et al., 2025c; Chen et al., 2025c; Lightman et al., 2024; Cui et al., 2025). Recent work has explored generative reward models (GenRMs), which reformulate reward modeling as next-token prediction that generates natural-language feedback (Mahan et al., 2024b; Chen et al., 2025b; Zhang et al., 2024; Li et al., 2025). Building on this paradigm, rubric-based GenRMs dynamically construct task-specific rubrics and reason about evaluation criteria (Guo et al., 2025a; Liu et al., 2025c; Chen et al., 2025d; Guo et al., 2025b). In parallel, several approaches augment RMs with tool under LLM-as-Judge framework (Li et al., 2024; Xu et al., 2025; Peng et al., 2025). However, existing methods

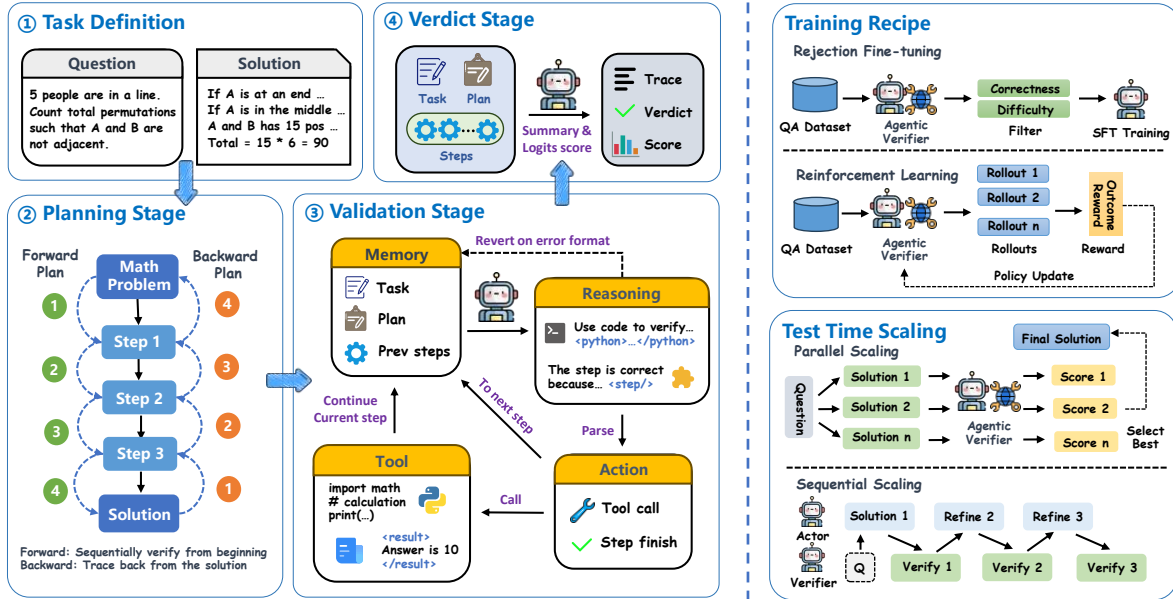


Figure 2: **Overview of Agentic Verifier’s architecture.** Agentic Verifier coordinates forward and backward agents with multi-turn reasoning and tool-augmented verification for reliable validation.

either do not tightly integrate tool execution into the reasoning process or fail to provide point-wise feedback required for test-time scaling (TTS). In contrast, our work reformulates verification as an agentic, multi-turn process, enabling test-time exploration and reliable assessment.

**Test-Time Scaling and Verifiers.** Recent work has shown that scaling inference-time compute can substantially improve LLM reasoning, and test-time scaling (TTS) has emerged as a general paradigm for both parallel selection and sequential refinement (Muennighoff et al., 2025b). Within this setting, critique-based methods use auxiliary models to guide actor correction and self-improvement at test time (Xi et al., 2024a), while more recent studies show that reward models and process verifiers themselves can also benefit from additional inference-time computation (Liu et al., 2025d; Khalifa et al., 2026). Our work is closely related to this line, but differs in that we cast verification as a bidirectional, multi-turn, tool-augmented process, enabling both sufficiency and necessity checking under parallel and sequential TTS.

### 3 Method

In this paper, we focus on the interactive TTS behavior between the actor and verifier on mathematics. The actor is involved in solving problems, while the verifier provides supervisory feedback on the generated solution chains.

**Parallel Scaling.** Best-of-N (BoN) has emerged as a prevalent parallel sampling strategy that leverages a verifier to select high-quality solutions (Gui et al., 2024; Kang et al., 2025). Specifically, for a given input  $x$ , the actor samples  $k$  candidate solutions, denoted as  $\{y^{(j)}\}_{j=1}^k$ . Subsequently, the verifier  $\pi_\psi$  assesses these candidates and generates a verifying rationale  $f$  to evaluate their correctness. The solution with the highest confidence score in  $\pi_\psi$  is then selected. This confidence score is defined as the likelihood of the `[True]` token within the verifying critique  $f$ , calculated as:

$$l(x, y^{(j)}) = \pi_\psi(\overline{\text{True}} \mid x, y^{(j)}, f^{(j)}, \mathbf{I})$$

$$f^{(j)} \sim \pi_\psi(x, y^{(j)}, \mathbf{I}), \quad (1)$$

where  $\mathbf{I}$  is the instruction: Is the solving process correct? (`[True]` / `[False]`).

**Sequential Scaling.** Given query  $x$  and an initial solution  $y_0$ , the verifier analyzes the solving process and provides supervisory critique, i.e.,  $f_0 \sim \pi_\psi(x, y_0)$ . The actor then receives this feedback and refines its solution as  $y_1$ . This refinement cycle can be repeated for multiple rounds until a correct answer is obtained or stopping criterion, formally expressed as

$$y_t \sim \pi_\theta(x, y_0, f_0, \dots, y_{t-1}, f_{t-1}). \quad (2)$$

#### 3.1 Agentic Verifier

Prevalent GenRMs suffer from error propagation and attention drift due to the single-turn reasoning

shown in Figure 1, which limits their efficacy for reliable verification. As depicted in Figure 2, we propose **Agentic Verifier** coordinating complementary forward and backward agents that trace solutions from premises to conclusions and re-examine conclusions against their premises. Both agents are accommodated with multi-turn reasoning and tool-augmented verification: they can iteratively decompose complex solutions into verifiable sub-steps, and invoke external tools like code for numerical calculation. We provide comprehensive details in Appendix B.2.

### 3.2 Task Definition

Formally, given  $\mathcal{D} = \{(x, g)\}$ , where  $x$  and  $g$  is the question and ground truth. The actor  $\pi_\theta$  generates the solution  $y \sim \pi_\theta(\cdot|x)$  autoregressively, which consists of the step-by-step CoT rationale and the final answer  $\tilde{y}$ . Following Xi et al. (2024b); Zhang et al. (2025b), we investigate two popular Test-Time Scaling paradigms: Parallel Scaling and Sequential Scaling.

**Forward Agent.** Starting from the problem premises, forward agent sequentially traces the solution path to review whether each step in the solution is correct, and validate whether the preceding step constitutes a sufficient condition for the subsequent derivation. To inspire the agent’s capabilities, we adopt a "Plan-Validate-Verdict" strategy in Figure 3:

- **Planning.** This phase aims at decomposing the overcomplicated reasoning solution into a sequence of atomic, verifiable sub-steps  $\Pi = \{v_1, v_2, \dots, v_n\}$ . The agent formulates an explicit plan specifying the logical flow for the subsequent stage. This plan includes specific checkpoints and necessary calculations, establishing a structured template for the whole verification.
- **Validation.** After receiving the plan  $\Pi$ , the agent examines the correctness of each atomic step  $v_i$ , ensuring the logical sufficiency between  $v_{i-1}$  and  $v_i$ . The agent is empowered to invoke Python interpreter and incorporate the execution results into the reasoning chain. Following Yao et al., involve multiple rounds of Thought-Action-Observation:

$$\mathcal{H} = (s_0, a_0, o_0, s_1, \dots, s_t, a_t, o_t) \quad (3)$$

where the agent articulates thoughts  $s$ , performs actions  $a$  (Python), and receives observations  $o$

as feedback from the interpreter. Action segments invoking python are closed by `<python>` and `</python>` more details are provided in Appendix B.2.3.

- **Verdict.** In the final stage, the agent transitions from a local, step-wise analysis to a global, holistic assessment. It aggregates the evidence collected during the previous phase to issue a definitive judgment on the solution’s correctness. The output is a strict binary classification (`[True]` / `[False]`) enclosed by `<answer>` and `</answer>` we adopt the logits of `[True]` serving as the confidence signal for the verdict.

**Backward Agent.** The backward agent is designed to identify errors the forward agent may overlook, such as solutions that superficially logically sound but fail to satisfy problem constraints or omit necessary proof. It verifies the necessity of a solution by reasoning in reverse, from the final answer back to the problem statement. Following the "Plan-Validate-Verdict" pipeline, the backward agent decomposes the solution into backward-checkable steps, systematically checks whether all required conditions are fulfilled. We aggregate the results from forward and backward agents for a bidirectional and reliable assessment in Appendix C.3.

### 3.3 AgentV-RL

The bidirectional design detailed above provides an effective multi-agent framework for identifying flaws in the candidate solution. Furthermore, to distill the capability of the multi-agent into a single LLM, a dedicated training recipe is necessary. In this section, we propose **AgentV-RL**, a scalable recipe comprising a synthetic data engine and a two-stage training schema.

#### 3.3.1 Synthetic Trajectories Sampling

To bootstrap the verifier in performing tool-integrated verification, we design a synthetic data pipeline to obtain multi-turn high-quality trajectories. To achieve this, we curate the data by filtering high-quality public datasets, including Polaris (An et al., 2025), DeepScaleR-40K (Luo et al., 2025), and AReaL-boba-106k (Mei et al., 2025). For the diversity of synthetic data, we sample  $k$  solutions for each question (setting  $k = 8$ ) and filter out questions (where  $k$  solutions are all correct/incorrect). This yields an initial dataset containing solution candidates,  $\mathcal{D}_{init} = \{x, y, l\}$ , where

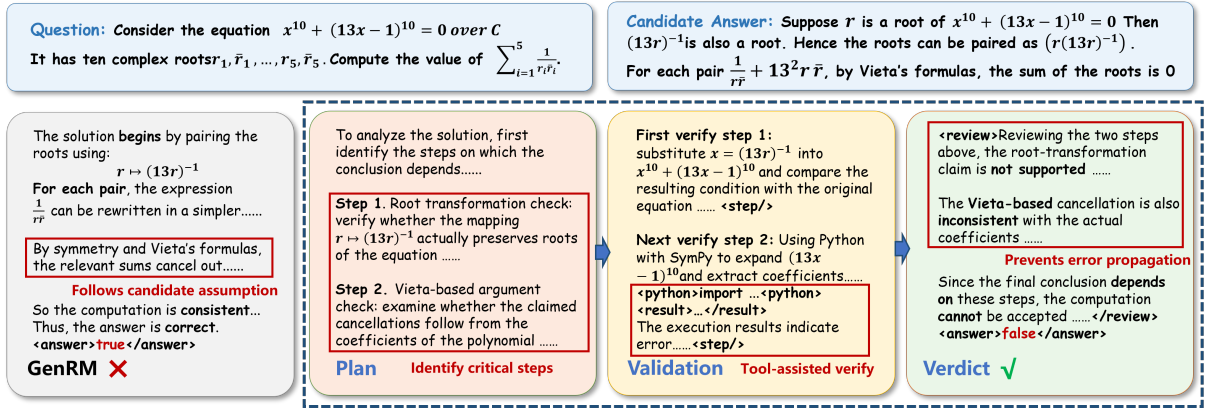


Figure 3: **Case study comparing GenRM and Agentic Verifier.** The example highlights the error propagation of GenRM, while our method obtains the correct verdict. More examples are provided in Appendix D.

$l \in \{\text{True}/\text{False}\}$  indicates the correctness of the solution  $y$ .

Based on  $\mathcal{D}_{init}$ , we employ the LLM to automate the generation of tool-augmented verification trajectories. Specifically, LLM role plays either the forward or backward Agent, generating verification trajectory  $\mathcal{H}$  that concludes with a final verdict  $\tilde{l}$ . Besides, we retain only those trajectories where the predicted verdict  $\tilde{l}$  matches the ground truth  $l$  for quality controlling. This results in the final dataset  $\mathcal{D}_{sft}$ , which comprises both positive and negative verification examples:

$$\mathcal{D}_{sft} = \{(x, y^+, \mathcal{H}^+)\} \cup \{(x, y^-, \mathcal{H}^-)\}, \quad (4)$$

where the former term denotes the positive rollout  $\mathcal{H}^+$  verified as **True**, and the latter is negative rollout  $\mathcal{H}^-$  validated as **False**.

### 3.3.2 Rejection Fine-tuning

With the synthetic dataset  $\mathcal{D}_{sft}$ , we conduct supervised fine-tuning (SFT) to endow the verifier with agentic proficiency. This stage is designed to align the model's behavior with the multi-turn, decision-making processes.

The core objective is to encourage the verifier to reproduce stepwise reasoning and effective tool interaction. Formally, for each data point  $(x, y, \mathcal{H})$ , where  $x$  is the problem,  $y$  is the candidate solution, and the annotated verification trajectory  $\mathcal{H} = (\tau_0, \tau_1, \dots, \tau_n)$ , where each  $\tau_i \in \{s_i, a_i, o_i\}$ , we optimize the following loss:

$$\mathcal{L} = -\mathbb{E}_{\tau \sim \mathcal{H}} \left[ \sum_{i=1}^{|\mathcal{H}|} \mathbb{I}[\tau_i \neq o_i] \cdot \log \pi_{\theta}(\tau_i | \mathcal{H}_{<i}) \right] \quad (5)$$

### 3.3.3 Reinforcement Learning

To further unlock the reasoning potential of the Agentic Verifier and incentivize autonomous exploration, we integrate Group Relative Policy Optimization (GRPO) within our recipe. Following the SFT phase, this stage aims to optimize the reasoning patterns to achieve multi-turn, long-horizon, and tool-integrated reasoning. Specifically, we sample a group of candidate trajectories from verifier  $\pi_{\psi}$  for each question-solution pair,

$$\{\mathcal{H}_i\}_{i=1}^G \sim \pi_{\psi}(\cdot | x, y, \mathbf{I}), \quad (6)$$

where  $\mathbf{I}$  represents the specific instruction prompt. In our experiments, we employ a mixed sampling strategy where  $\pi_{\psi}$  acts as either forward agent or backward agent, allowing balanced optimization for dual-perspectives.

## 4 Experiments

**Verifiable Reward.** We apply a binary outcome-level reward by comparing the consistency between the predicted verdict  $\tilde{l}$  and ground truth  $l$ ,

$$r(\mathcal{H}) = \begin{cases} 1, & \text{if } \tilde{l} = l \\ -1, & \text{otherwise} \end{cases} \quad (7)$$

Following Yu et al. (2025), we dynamically filter those zero reward-variance groups of rollouts, which is too easy or hard for policy model (predominantly all +1 or all -1). Then derive the normalized advantages  $\hat{A}_{i,t}$  within each group and update the

Models	MATH500			GSM8K			Gaokao2023			AIME24		
BoN	@32	@64	@128	@32	@64	@128	@32	@64	@128	@32	@64	@128
<i>Text Reasoning LLM</i>												
Qwen2.5-7B-Instruct	53.0	50.4	51.0	85.5	85.7	83.9	40.8	42.3	41.3	30.0	<u>26.7</u>	<u>33.3</u>
Llama3.1-8B-Instruct	46.7	48.0	45.2	81.5	80.1	81.7	34.5	35.6	35.6	30.0	<b>30.0</b>	30.0
Qwen3-4B-Think	<b>70.0</b>	<b>72.6</b>	<b>72.4</b>	<b>92.3</b>	<b>91.2</b>	<b>92.2</b>	<b>52.2</b>	<b>51.4</b>	<b>51.9</b>	<b>40.0</b>	<b>30.0</b>	<b>36.7</b>
DS-Distill-14B	<u>66.2</u>	<u>69.6</u>	<u>69.4</u>	<u>90.1</u>	<u>88.9</u>	<u>89.8</u>	<u>48.3</u>	<u>49.1</u>	<u>48.6</u>	<u>33.3</u>	<u>26.7</u>	<u>36.7</u>
Mistral-Small-24B-Instruct	52.2	51.0	51.8	85.4	85.5	85.9	39.0	38.4	40.0	30.0	<b>30.0</b>	<b>36.7</b>
<i>Outcome-level RM</i>												
GRM-Gemma-2B	45.6	48.8	46.6	77.9	75.5	73.4	33.5	31.9	33.2	33.3	26.7	30.0
Skywork-V2-Llama-8B	<u>54.4</u>	<u>55.2</u>	<u>53.8</u>	87.5	87.3	87.6	<b>41.6</b>	37.4	39.7	30.0	<u>33.3</u>	<u>36.7</u>
InternLM2-20B-RM	54.0	52.2	53.6	<u>89.5</u>	<u>89.6</u>	<u>89.8</u>	<b>41.6</b>	<b>42.9</b>	<u>43.9</u>	<u>36.7</u>	<b>36.7</b>	40.0
INF-ORM-Llama3.1-70B	<b>54.6</b>	<b>56.4</b>	<b>55.4</b>	<b>91.2</b>	<b>90.8</b>	<b>91.5</b>	<u>40.8</u>	<u>42.6</u>	<b>44.4</b>	<b>40.0</b>	<b>36.7</b>	<b>40.0</b>
Starling-RM-34B	50.0	52.0	50.8	88.4	87.6	86.3	40.5	39.7	39.0	26.7	33.0	<u>36.7</u>
<i>Process-level RM</i>												
Qwen2.5-Math-PRM-7B	<u>65.2</u>	<b>69.4</b>	<b>70.2</b>	<b>94.5</b>	<b>94.7</b>	<b>95.4</b>	<b>50.9</b>	<b>51.4</b>	<b>54.3</b>	<b>43.3</b>	<b>43.3</b>	<b>46.7</b>
Math-Shepherd-Mistral-7B-PRM	<b>66.6</b>	<u>66.6</u>	<u>66.6</u>	90.0	89.2	89.6	43.1	41.4	39.3	<b>30.0</b>	36.7	30.0
EurusPRM-Stage1	38.8	36.0	33.8	70.0	66.3	62.5	33.2	29.1	26.8	20.0	23.3	26.7
EurusPRM-Stage2	36.0	31.8	31.0	66.0	62.1	57.5	29.1	26.5	25.7	16.7	20.0	23.3
Skywork-PRM	61.6	66.2	65.6	<u>92.5</u>	<u>92.7</u>	<u>93.4</u>	<u>46.0</u>	<u>47.0</u>	<u>46.8</u>	23.3	<b>40.0</b>	<b>33.3</b>
<i>Ours</i>												
Agentic-Verifier-Llama3-8B	59.6	60.8	60.6	90.8	91.2	90.7	45.5	47.3	47.0	30.0	33.3	40.0
Agentic-Verifier-Qwen3-4B	<b>73.8</b>	<b>76.2</b>	<b>79.0</b>	<b>93.0</b>	<b>92.6</b>	<b>93.3</b>	<b>54.5</b>	<b>55.1</b>	<b>57.4</b>	<b>46.7</b>	<b>50.0</b>	<b>53.3</b>

Table 1: **Performance (%) of Best-of-N sampling on GSM8K, MATH500, Gaokao2023, and AIME24.** Within each category, the best results are in bold and the second-best results are underlined.

verifier with the objective:

$$\mathcal{J}_{\text{GRPO}}(\psi) = \mathbb{E}_{(x,y) \sim \mathcal{D}, \{\mathcal{H}_i\}_{i=1}^G \sim \pi_{\psi_{\text{old}}}} \frac{1}{G} \sum_{i=1}^G \frac{1}{|\mathcal{H}_i|} \sum_{t=1}^{|\mathcal{H}_i|} \min(r_{i,t}(\psi) \hat{A}_{i,t}, \text{clip}(r_{i,t}(\psi), 1 - \epsilon_{\text{low}}, 1 + \epsilon_{\text{high}}) \hat{A}_{i,t}) - \beta D_{\text{KL}}(\pi_{\psi} \parallel \pi_{\text{ref}}) \quad (8)$$

where  $r_{i,t}(\psi) = \frac{\pi_{\psi}(\mathcal{H}_i|x,y,\mathcal{H}_{i,<t})}{\pi_{\psi_{\text{old}}}(\mathcal{H}_i|x,y,\mathcal{H}_{i,<t})}$  is the importance sampling ratio. To mitigate the risk of memorizing environment observations, we exclude the compiler execution from the loss calculation.

## 4.1 Setup

**Task.** We evaluate the proposed method from two popular TTS paradigms: parallel scaling and sequential scaling. The former focuses on BoN, while the latter is verifier-revision iteratively. We conduct experiments on widely used reasoning benchmarks: GSM8K (Cobbe et al., 2021), AIME24 (AIME, 2024), MATH500 (Lightman et al., 2024) and Gaokao2023 (Liao et al., 2024).

**Implementation Details.** We utilize Qwen3-4B (Yang et al., 2025a) as our base model. In the SFT phase, the verifier is trained on 15K synthesized samples constructed in Section 3. The model

is subsequently further optimized using GRPO on an additional 50K samples. We provide implementation illustrations in Appendix C.4.

**Evaluation Setting.** For parallel scaling, we report the accuracy under different sampling rollouts  $N$ , where the verifier scores  $N$  candidate solutions and selects the highest-scoring one as the final output. For sequential scaling, we report the average accuracy at each refinement iteration, together with  $\Delta_{\uparrow}$  and  $\Delta_{\downarrow}$ , which measure the rates of correction and degradation relative to the previous iteration. To ensure strict comparability, all verifier variants are evaluated on the same fixed candidate pool and the same initial solution for each problem instance. Additional evaluation details and baseline descriptions are provided in the Appendix C.1 and C.2.

## 4.2 Main Result

**Agentic Verifier improves Best-of-N sampling performance.** Table 1 reports BoN performance across state-of-the-art competitors. We make several key observations: (1) Agentic Verifier delivers consistently strong results across all benchmarks, establishing new state-of-the-art performance. Notably, Agentic-Verifier-Qwen3-4B achieves the highest accuracy on MATH500 (up to 79.0%), surpassing the previous best outcome-level RM,

Verifier	MATH500			GSM8K			Gaokao2023			AIME24		
Metrics	Acc	$\Delta_{\uparrow}$	$\Delta_{\downarrow}$	Acc	$\Delta_{\uparrow}$	$\Delta_{\downarrow}$	Acc	$\Delta_{\uparrow}$	$\Delta_{\downarrow}$	Acc	$\Delta_{\uparrow}$	$\Delta_{\downarrow}$
<i>Turn 1</i>												
Qwen2.5-7B-Instruct	60.4	14.0	0.6	85.9	5.9	0.6	47.0	12.7	2.9	13.3	0.0	0.0
Llama3.1-8B-Instruct	54.6	11.0	3.4	84.1	4.6	1.1	44.9	10.7	2.9	13.3	0.0	0.0
Qwen3-4B-Think	80.0	33.8	0.8	90.9	11.1	0.9	64.9	31.2	3.4	16.7	3.3	0.0
DS-Distill-14B	<u>83.0</u>	37.4	1.4	<u>92.0</u>	12.1	0.8	<u>67.0</u>	32.5	2.6	<u>23.3</u>	10.0	0.0
Mistral-Small-24B-Instruct	61.4	15.6	1.2	88.4	8.5	0.8	49.1	15.6	3.6	13.3	0.0	0.0
Agentic-Verifier-Qwen3-4B	<b>84.2</b>	41.6	0.6	<b>94.6</b>	14.5	0.5	<b>75.6</b>	40.3	1.8	<b>40.0</b>	26.7	3.3
<i>Turn 2</i>												
Qwen2.5-7B-Instruct	64.8	6.2	1.8	87.4	3.3	1.9	49.9	5.2	2.3	6.7	0.0	6.7
Llama3.1-8B-Instruct	58.4	6.0	2.2	82.8	3.3	4.6	48.1	5.2	2.1	10.0	0.0	3.3
Qwen3-4B-Think	84.6	6.0	1.4	92.2	2.4	1.1	69.1	5.5	1.3	23.3	6.7	3.3
DS-Distill-14B	<u>87.4</u>	5.8	1.4	<u>92.7</u>	2.4	1.7	<u>70.1</u>	4.7	1.6	<u>26.7</u>	3.3	0.0
Mistral-Small-24B-Instruct	66.4	5.8	0.8	90.0	2.7	1.1	52.7	4.9	1.3	13.3	0.0	0.0
Agentic-Verifier-Qwen3-4B	<b>89.2</b>	2.4	1.2	<b>94.1</b>	0.4	0.9	<b>76.6</b>	3.4	2.3	<b>33.3</b>	0.0	6.7
<i>Turn 3</i>												
Qwen2.5-7B-Instruct	66.0	4.2	3.0	87.5	2.96	2.8	52.2	3.64	1.3	10.0	6.7	3.3
Llama3.1-8B-Instruct	58.8	4.0	3.6	82.8	3.3	3.3	47.0	1.3	2.3	6.0	0.0	3.00
Qwen3-4B-Think	84.8	2.4	2.2	92.0	1.2	1.4	69.6	2.3	1.6	26.0	6.7	3.0
DS-Distill-14B	<u>85.8</u>	2.2	3.8	<u>93.4</u>	1.8	1.1	<u>73.0</u>	4.4	1.6	<u>33.0</u>	6.7	0.0
Mistral-Small-24B-Instruct	67.8	4.4	3.0	90.6	2.65	2.05	52.2	2.08	2.60	13.0	0.00	0.00
Agentic-Verifier-Qwen3-4B	<b>89.8</b>	1.8	1.2	<b>94.1</b>	1.0	1.0	<b>76.4</b>	2.6	2.9	<b>33.0</b>	3.3	3.3

Table 2: **Performance comparison against different verifiers for iterative refinement.** *Acc* denotes the average accuracy at each refinement iteration, together with  $\Delta_{\uparrow}$  and  $\Delta_{\downarrow}$ , which measure the rates of correction and degradation relative to the previous iteration. The best result is in **bold**, and the second-best is underlined.

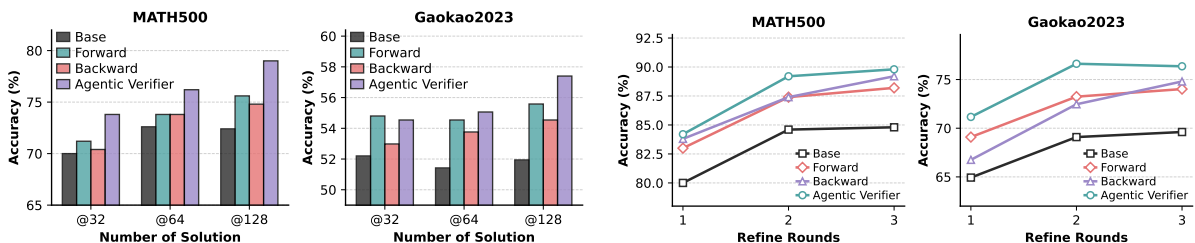


Figure 4: **Ablation study of different variants on Best-of-N sampling (left) and verifier revision (right).** Forward-only and backward-only variants are both competitive, while the full design performs best.

Skywork-V2-Llama-8B, by a substantial margin of 25.2 percentage points. Similar trends are observed for GSM8K, Gaokao2023, and AIME24, with our model maintaining significant leads. (2) Importantly, as the number of BoN samples  $N$  increases (from 32 to 128), Agentic Verifier’s performance continues to improve steadily, particularly on challenging benchmarks like AIME24. On AIME24, the 4B variant reaches 53.3% accuracy at  $N = 128$ , while all baselines—including larger models—remain significantly lower. (3) Our variants deliver a substantial improvement compared to the base model, e.g. Qwen3-4B and Llama-3.1-8B. The substantial gains stem from the bidirectional verification and seamless tool integration in the

proposed framework, enabling it to scrutinize reasoning chains and catch subtle errors—especially for challenge benchmarks like AIME24.

**Agentic Verifier provides effective guidance for iterative refinement.** To assess the helpfulness of the verifiers’ feedback, we conduct a thorough analysis of critique-revision strategies with different verifiers. Based on the results in Table 2, we can conclude: (1) In the 1-st turn, the actor achieves 94.6% accuracy on GSM8K and 84.2% on MATH500 with the critique from Agentic-Verifier-Qwen3-4B. As indicated by the up to 41.6%  $\Delta_{\uparrow}$  (percentage of wrong solutions corrected), while maintaining a minimal rate 0.6% of incorrectly

revised solutions ( $\Delta_{\downarrow}$ ). Agentic Verifier delivers high-quality critiques that assist the actor in correcting erroneous solutions. (2) Compared to other verifiers, Agentic Verifier achieves faster convergence: significant performance boosts are observed in the first one or two refinement rounds. The performance remains stable or only slightly declines in later iterations, avoiding the degradation encountered like DS-Distill-14B. This robustness can be attributed to its agentic multi-turn reasoning and tool-augmented verification mechanisms, which allow it to maintain rigorous oversight throughout the refinement process.

### 4.3 Analysis

#### 4.3.1 Ablation Study

**Effectiveness of Bidirectional Components.** Recall that Agentic Verifier comprises two specialized agents: a forward agent that traces the logical flow from premises to conclusion within the candidate solution, and a backward agent that verifies necessity by reasoning from the final conclusion back to the original query. To assess the individual contributions of each agent, we conduct an ablation study comparing two variants on both BoN sampling and verifier-revision, as presented in Fig. 4: (a) forward agent only, and (b) backward agent only. Both variants achieve competitive results; however, Agentic Verifier, which integrates both agents, achieves the state-of-the-art performance. This demonstrates that our bidirectional design objectives are synergistic, leading to improved generalization and reliable verification.

**Effectiveness of Tool Usage.** We observe that even the variant without tool already significantly outperforms the base model, indicating that the Agentic Verifier itself contributes substantially to performance gains, beyond what is achieved by tool alone. When the tool is incorporated, performance improves further. Notably, tool usage remains moderate and stable in practice. For a detailed analysis, including ablations and execution statistics, please refer to Appendix C.5.

**Training Recipe.** We explore the key ingredients in our training recipe. Through a series of controllable studies, we quantitatively evaluate the effect of our main design choices by comparing the following configurations: Train-free, SFT, and SFT+RL. Figure 5 presents the results of our experiments and more details about settings can be

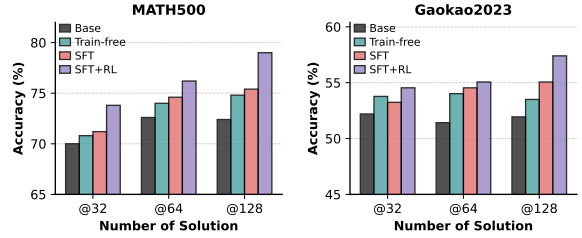


Figure 5: **Controllable study of different training design choices on BoN performance.** The SFT+RL design achieves the strongest results.

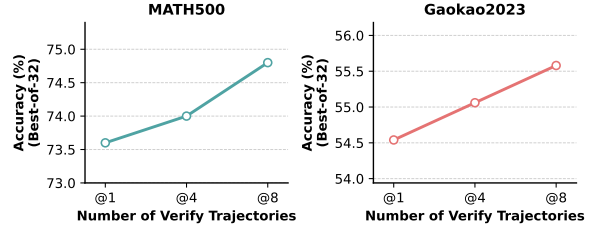


Figure 6: **Scaling inference-time compute for verification.** Sampling multiple verification trajectories improves BoN accuracy.

found in Appendix C.4. We observe that the Train-free variant already achieves relatively competitive results, outperforming the base model by up to 2.6 points on Gaokao2023. Furthermore, the SFT variant demonstrates additional performance gains, confirming the effectiveness of our synthetic data engine. Finally, the introduction of RL further unlocks the model’s reasoning potential by enabling direct interaction with the environment and optimizing behavioral patterns.

#### 4.3.2 Scaling Effect

**Scaling Inference-time Compute.** Scaling inference-time compute within Agentic Verifier can be achieved by sampling multiple verification trajectories and aggregating their scores, as defined in Equation 9. Figure 6 demonstrates that BoN performance scales positively with the allocation of additional inference-time compute.

**Scaling Model Size.** We also examine the effect of scaling model size under train-free setting. As shown in Table 3, larger models consistently achieve higher accuracy across benchmarks. Scaling from 0.6B to 1.7B parameters yields an improvement of 5.2 points on Gaokao2023, with further gains observed at Qwen3-4B. These findings demonstrate that scaling up model capacity continually benefits from our agentic framework.

Model	MATH500			Gaokao2023		
	@32	@64	@128	@32	@64	@128
0.6B	59.8	63.2	62.2	43.9	44.9	43.1
1.7B	64.8	67.4	68.0	46.5	49.4	48.3
4B	<b>73.8</b>	<b>76.2</b>	<b>79.0</b>	<b>54.5</b>	<b>55.1</b>	<b>57.4</b>

Table 3: **Scaling model size of Agentic Verifier with the Qwen3 series.** Larger models consistently benefit from the proposed agentic verification framework.

Model	LCB	HotpotQA
Qwen3-4B	57.14	40.00
Qwen2.5-7B-Instruct	58.29	40.67
DS-Distill-14B	64.57	52.05
Mistral-Small-24B-Instruct	50.30	63.00
Agentic-Verifier-Qwen3-4B	<b>70.86</b>	<b>66.00</b>

Table 4: **Generalization Performance** on LiveCodeBench and HotpotQA.

### 4.3.3 Generalization Performance

We further evaluate Agentic Verifier on LiveCodeBench (Jain et al., 2024) and HotpotQA (Yang et al., 2018) to test its versatility on competitive code and multihop QA. Agentic-Verifier-Qwen3-4B achieves superior performance on both benchmarks (Table 4), confirming that our bidirectional, tool-augmented paradigm generalizes to broader reasoning domains. See Appendix C.6 for experimental details.

### 4.3.4 Latency Analysis

**Computational Overhead.** As detailed in Table 5, Agentic Verifier achieves the highest accuracy at the expense of increased tokens and latency. This overhead stems from its multi-turn deliberative process, yet the Forward and Backward variants provide scalable trade-offs for resource-constrained settings. Benchmark results are averaged over trajectories using batch size of 128 on A100 GPU.

## 5 Conclusion

This paper introduces Agentic Verifier, an agentic reward modeling paradigm that leverages multi-turn, tool-augmented verification. By orchestrating forward and backward agents, our approach enables comprehensive and interpretable solution checking. Extensive experiments show that Agentic Verifier achieves substantial improvements in both parallel and sequential TTS, outperforming SOTA ORM and PRM. These results highlight Agentic Verifier as a promising direction for advancing agentic reward modeling. We hope our

Model	Tokens	Rounds	Tool Calls	Time (s)
Base	2560	1.0	0.0	119.0
Forward	4114	5.7	0.9	159.1
Backward	4235	5.6	0.7	164.3
Agentic-Verifier	8349	11.3	1.6	323.4

Table 5: **Computational cost and latency analysis.** All metrics are averaged over verification trajectories, and latency is measured on a single A100 GPU using vLLM with batch size 128.

work provides meaningful insights for the development of reward models and inspires future research on autonomous verification and reasoning.

## 6 Limitations

Though Agentic Verifier displays state-of-the-art effectiveness, several limitations should be noted. First, the reliance on synthetic, tool-augmented data may not fully represent the variety of real-world reasoning problems, which impede generalization. Second, the multi-turn process increases computational cost, presenting challenges for real-time or resource-limited deployment. Finally, the framework’s performance is tied to the coverage and reliability of external tools, which can be a bottleneck for certain tasks. Future work will focus on optimizing efficiency, broadening tool integration, and improving adaptability to a wider range of tasks.

## 7 Ethics Statement

This paper introduces an agentic reward modeling paradigm designed to enhance the reliability of automatic solution verification. The methods and findings presented are intended solely for research purposes; Our study does not pertain to malicious applications, unintended uses, or issues related to fairness, privacy, security, crowdsourcing, or human subject research.

## Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by National Key R&D Program of China No.2025ZD0215702, National Natural Science Foundation of China (No. 162576106, 62476061, 62376061).

## References

AIME. 2024. [Aime2024](#).

- Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. 2025. [Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models](#).
- Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiaoyi Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, and 81 others. 2024. [Internlm2 technical report](#). *Preprint*, arXiv:2403.17297.
- Luoxin Chen, Jinming Gu, Liankai Huang, Wenhao Huang, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Kaijing Ma, and 1 others. 2025a. [Seed-prover: Deep and broad reasoning for automated theorem proving](#). *arXiv preprint arXiv:2507.23726*.
- Nuo Chen, Zhiyuan Hu, Qingyun Zou, Jiaying Wu, Qian Wang, Bryan Hooi, and Bingsheng He. 2025b. [Judgelrm: Large reasoning models as a judge](#). *arXiv preprint arXiv:2504.00050*.
- Wenxiang Chen, Wei He, Zhiheng Xi, Honglin Guo, Boyang Hong, Jiazheng Zhang, Nijun Li, Tao Gui, Yun Li, Qi Zhang, and 1 others. 2025c. [Better process supervision with bi-directional rewarding signals](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 14471–14485.
- Xiuxi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, Hanghang Tong, and Heng Ji. 2025d. [RM-R1: reward modeling as reasoning](#). *CoRR*, abs/2505.02387.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *CoRR*, abs/2110.14168.
- Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, and 6 others. 2025. [Process reinforcement through implicit rewards](#). *Preprint*, arXiv:2502.01456.
- Google DeepMind. 2025. [Advanced version of gemini with deep think officially achieves gold-medal standard at the international mathematical olympiad](#).
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). *Preprint*, arXiv:2501.12948.
- Guanting Dong, Licheng Bao, Zhongyuan Wang, Kangzhi Zhao, Xiaoxi Li, Jiajie Jin, Jinghan Yang, Hangyu Mao, Fuzheng Zhang, Kun Gai, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. 2025a. [Agentic entropy-balanced policy optimization](#). *CoRR*, abs/2510.14545.
- Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. 2025b. [Toolstar: Empowering llm-brained multi-tool reasoner via reinforcement learning](#). *CoRR*, abs/2505.16410.
- Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. [Retool: Reinforcement learning for strategic tool use in llms](#). *CoRR*, abs/2504.11536.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Lin Gui, Cristina Garbacea, and Victor Veitch. 2024. [Bonbon alignment for large language models and the sweetness of best-of-n sampling](#). In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.
- Honglin Guo, Kai Lv, Qipeng Guo, Tianyi Liang, Zhiheng Xi, Demin Song, Qiuyinzhe Zhang, Yu Sun, Kai Chen, Xipeng Qiu, and 1 others. 2025a. [Critiq: Mining data quality criteria from human preferences](#). *arXiv preprint arXiv:2502.19279*.
- Jiaxin Guo, Zewen Chi, Li Dong, Qingxiu Dong, Xun Wu, Shaohan Huang, and Furu Wei. 2025b. [Reward reasoning model](#). *arXiv preprint arXiv:2505.14674*.
- Jujie He, Tianwen Wei, Rui Yan, Jiakai Liu, Chaojie Wang, Yimeng Gan, Shiwen Tu, Chris Yuhao Liu, Liang Zeng, Xiaokun Wang, Boyang Wang, Yongcong Li, Fuxiang Zhang, Jiacheng Xu, Bo An, Yang Liu, and Yuhui Zhou. 2024. [Skywork-o1 open series](#).
- Yichen Huang and Lin F Yang. 2025. [Gemini 2.5 pro capable of winning gold at imo 2025](#). *arXiv preprint arXiv:2507.15855*, 7:7.
- Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. [Livecodebench: Holistic and contamination free evaluation of large language models for code](#). *arXiv preprint arXiv:2403.07974*.
- Zhewei Kang, Xuandong Zhao, and Dawn Song. 2025. [Scalable best-of-n selection for large language models via self-certainty](#). *CoRR*, abs/2502.18581.

- Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. 2026. [Process reward models that think](#). *Trans. Mach. Learn. Res.*, 2026.
- Lei Li, Yekun Chai, Shuohuan Wang, Yu Sun, Hao Tian, Ningyu Zhang, and Hua Wu. 2024. [Tool-augmented reward modeling](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Yi-Chen Li, Tian Xu, Yang Yu, Xuqin Zhang, Xiong-Hui Chen, Zhongxiang Ling, Ningjing Chao, Lei Yuan, and Zhi-Hua Zhou. 2025. Generalist reward models: Found inside large language models. *arXiv preprint arXiv:2506.23235*.
- Minpeng Liao, Chengxi Li, Wei Luo, Jing Wu, and Kai Fan. 2024. [MARIO: math reasoning with code interpreter output - A reproducible pipeline](#). In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 905–924. Association for Computational Linguistics.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. [Let’s verify step by step](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiakai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, Yang Liu, and Yahui Zhou. 2025a. Skywork-reward-v2: Scaling preference data curation via human-ai synergy. *arXiv preprint arXiv:2507.01352*.
- Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. 2025b. [Inference-time scaling for generalist reward modeling](#). *CoRR*, abs/2504.02495.
- Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. 2025c. Inference-time scaling for generalist reward modeling. *arXiv preprint arXiv:2504.02495*.
- Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. 2025d. [Inference-time scaling for generalist reward modeling](#). *CoRR*, abs/2504.02495.
- Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. [Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl](#). Notion Blog.
- Dakota Mahan, Duy Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. 2024a. [Generative reward models](#). *CoRR*, abs/2410.12832.
- Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. 2024b. Generative reward models. *arXiv preprint arXiv:2410.12832*.
- Zhiyu Mei, Wei Fu, Kaiwei Li, Guangju Wang, Huan Chen Zhang, and Yi Wu. 2025. Real: Efficient rlhf training of large language models with parameter reallocation. In *Proceedings of the Eighth Conference on Machine Learning and Systems, MLSys 2025, Santa Clara, CA, USA, May 12-15, 2025*. mlsys.org.
- Xiaoyu Tan Minghao Yang, Chao Qu. 2024. [Inf-orm-llama3.1-70b](#).
- Mistral AI. 2025. [Mistral small 3](#).
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel J. Candès, and Tatsunori Hashimoto. 2025a. [s1: Simple test-time scaling](#). *CoRR*, abs/2501.19393.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel J. Candès, and Tatsunori Hashimoto. 2025b. [s1: Simple test-time scaling](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 20275–20321. Association for Computational Linguistics.
- OpenAI. 2025. [Openai achieved gold medal-level performance on the 2025 international mathematical olympiad](#).
- Hao Peng, Yunjia Qi, Xiaozhi Wang, Zijun Yao, Bin Xu, Lei Hou, and Juanzi Li. 2025. Agentic reward modeling: Integrating human preferences with verifiable correctness signals for reliable reward systems. *arXiv preprint arXiv:2502.19328*.
- Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jingren Zhou, and 25 others. 2025. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.
- Zhihong Shao, Yuxiang Luo, Chengda Lu, ZZ Ren, Jiewen Hu, Tian Ye, Zhibin Gou, Shirong Ma, and Xiaokang Zhang. 2025. Deepseekmath-v2: Towards self-verifiable mathematical reasoning. *arXiv preprint arXiv:2511.22570*.
- Binghai Wang, Rui Zheng, Lu Chen, Yan Liu, Shihan Dou, Caishuang Huang, Wei Shen, Senjie Jin, Enyu Zhou, Chenyu Shi, and 1 others. 2024a. Secrets of rlhf in large language models part ii: Reward modeling. *arXiv preprint arXiv:2401.06080*.

- Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. [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)*, ACL 2024, Bangkok, Thailand, August 11-16, 2024, pages 9426–9439. Association for Computational Linguistics.
- Zhiheng Xi, Dingwen Yang, Jixuan Huang, Jiafu Tang, Guanyu Li, Yiwen Ding, Wei He, Boyang Hong, Shihan Dou, Wenyu Zhan, Xiao Wang, Rui Zheng, Tao Ji, Xiaowei Shi, Yitao Zhai, Rongxiang Weng, Jingang Wang, Xunliang Cai, Tao Gui, and 5 others. 2024a. [Enhancing LLM reasoning via critique models with test-time and training-time supervision](#). *CoRR*, abs/2411.16579.
- Zhiheng Xi, Dingwen Yang, Jixuan Huang, Jiafu Tang, Guanyu Li, Yiwen Ding, Wei He, Boyang Hong, Shihan Dou, Wenyu Zhan, Xiao Wang, Rui Zheng, Tao Ji, Xiaowei Shi, Yitao Zhai, Rongxiang Weng, Jingang Wang, Xunliang Cai, Tao Gui, and 5 others. 2024b. [Enhancing LLM reasoning via critique models with test-time and training-time supervision](#). *CoRR*, abs/2411.16579.
- Ran Xu, Jingjing Chen, Jiayu Ye, Yu Wu, Jun Yan, Carl Yang, and Hongkun Yu. 2025. Incentivizing agentic reasoning in llm judges via tool-integrated reinforcement learning. *arXiv preprint arXiv:2510.23038*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 40 others. 2025a. [Qwen3 technical report](#). *CoRR*, abs/2505.09388.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025b. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.
- Rui Yang, Ruomeng Ding, Yong Lin, Huan Zhang, and Tong Zhang. 2024. Regularizing hidden states enables learning generalizable reward model for llms. *arXiv preprint arXiv:2406.10216*.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. [React: Synergizing reasoning and acting in language models](#). In *The eleventh international conference on learning representations*.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, and 16 others. 2025. [DAPO: an open-source LLM reinforcement learning system at scale](#). *CoRR*, abs/2503.14476.
- Lifan Yuan, Wendi Li, Huayu Chen, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. 2025. [Free process rewards without process labels](#). In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net.
- Jiazheng Zhang, Wenqing Jing, Zizhuo Zhang, Zhiheng Xi, Shihan Dou, Rongxiang Weng, Jiahuan Li, Jingang Wang, Mingxu Chai, Shibo Hong, Tao Gui, and Qi Zhang. 2025a. [Two minds better than one: Collaborative reward modeling for LLM alignment](#). *CoRR*, abs/2505.10597.
- Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2024. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*.
- Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2025b. [Generative verifiers: Reward modeling as next-token prediction](#). In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.
- Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025c. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.
- Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025d. [The lessons of developing process reward models in mathematical reasoning](#). In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 10495–10516. Association for Computational Linguistics.
- Jialun Zhong, Wei Shen, Yanzeng Li, Songyang Gao, Hua Lu, Yicheng Chen, Yang Zhang, Wei Zhou, Jinjie Gu, and Lei Zou. 2025. A comprehensive survey of reward models: Taxonomy, applications, challenges, and future. *arXiv preprint arXiv:2504.12328*.
- Banghua Zhu, Evan Frick, Tianhao Wu, Hanlin Zhu, and Jiantao Jiao. 2023. Starling-7b: Improving llm helpfulness & harmlessness with rlaiif.

## A Use Of AI Assistants

LLMs are utilized to polish the article to improve the reading experience.

## B Details of Evaluation

### B.1 Text Reasoning Prompt

For reference, we include two commonly used prompting strategies as baselines. The Best-of-N verifier evaluates each candidate solution independently in a single turn and outputs a boolean correctness judgment. The Refine-style verifier re-evaluates an initial judgment by providing feedback conditioned on the original problem and solution.

#### B.1.1 Best-of-N

##### System

You are a teacher. Your task is to grade the solution, verifying correctness. Use Expected Answer to find any erroneous step in the Solution.

## GOAL

Given a problem and a solution, you must:

- 1) write a concise verification text that inspects the given solution step-by-step (do not re-solve unless a single local derivation is trivially needed);
- 2) decide whether the solution correctly solves the problem;
- 3) output a boolean verdict.

You must prioritize checking the original solution via paper checks: legality of algebraic steps, substitution mentally for proposed roots, domain and edge cases, theorem prerequisites, and consistency of the final statement with intermediate steps. Do NOT produce a fresh full solution when the provided reasoning is wrong or incomplete.

## ALLOWED TAGS

- \* `<rubric>...</rubric>` - required at the start; list 2 - 4 decisive axes you'll check.
- \* `<think>...</think>` - private reasoning exactly once; each update must add evidence or tighten the verdict (state micro-goal, the two most decisive axes, a compact known/unknown ledger, then choose the smallest next step: conclude or one precise mental check).
- \* `<verify>...</verify>` - public verification text, focused on checking the given reasoning.
- \* `<answer>...</answer>` - final boolean verdict (true/false) exactly once.

##### User

```
### Problem ###
```

```
{problem}
```

```
### Solution ###
```

```
{solution}
```

```
### Verification Task Reminder ###
```

- Start with `<rubric>...</rubric>`.
- Include exactly `<think>...</think>`.
- Then verify the solution step by step and output the final verdict:`<verify>...</verify>`  
`><answer>true|false</answer>`

#### B.1.2 Refine

##### System

You are a Verifier agent responsible for performing a verification of a math problem's solution. Your mission is to determine whether the given solution is correct.

##### User

Read the following carefully and think critically:

```
### Problem ###
```

```
{problem}
```

```
### Solution ###
```

```
{solution}
```

You are required to verify the solution carefully and show your reasoning process. At the end, provide your final judgment in a single line using:

`<answer>true</answer>` - if the reasoning confirms that the conclusion logically holds.

`<answer>false</answer>` - if there exists any missing justification, invalid reversal, or unsupported dependency.

#### B.2 Agentic Verifier Prompt

For reference, we report the prompt templates used in our proposed multi-stage verification framework. The framework decomposes verification into three stages, namely Plan, Validation, and Verdict. We consider two planning variants: a forward verifier, which plans verification steps by reasoning from the original problem, and a backward verifier, which plans by reasoning backward from the final

conclusion. Both variants share the same step-wise Validation and Verdict prompts. Intermediate user prompts that advance the verifier between stages are automatically inserted by the system during execution.

### B.2.1 System Prompt

#### Forward

You are a Verifier agent responsible for performing a multi-turn verification of a math problem's solution.

Your mission is to determine whether the given solution is correct.

You must complete the task strictly in three distinct stages, executed sequentially and independently.

## Stage A: Task Analysis \& Extraction

Analyze the original question and its provided solution. Decompose the verification into smaller, checkable steps.

## Stage B: Solution Analysis \& Judgment

Perform the actual verification one step at a time, following the plan you created in Stage A.

## Stage C: Final Review \& Verdict

After all Stage B steps are complete, review the entire reasoning process and produce a final boolean verdict.

In every turn, you must include your reasoning process inside a <think>...</think> block before presenting your results.

#### Backward

You are a Verifier agent responsible for performing a multi-turn verification of a math problem's solution. Your mission is to determine whether the given solution is correct.

Unlike a forward, step-by-step verifier, your reasoning proceeds in reverse: you begin from the final result or conclusion presented in the solution and trace backward to determine whether it can be rigorously justified by the given question and established facts.

You must complete the task strictly in three distinct stages, executed sequentially and independently.

## Stage A: Task Analysis \& Extraction

Analyze the original question and its provided solution. Decompose the verification into smaller, checkable backward steps.

## Stage B: Solution Analysis \& Judgment

Perform the actual verification one step at a time, following the plan you created in Stage A.

## Stage C: Final Review \& Verdict

After all Stage B steps are complete, review the entire reasoning process and produce a final boolean verdict.

In every turn, you must include your reasoning process inside a <think>...</think> block before presenting your results.

### B.2.2 Stage A: Planning Prompts

#### Forward

You are now entering \*\*Stage A: Task Analysis \& Extraction\*\*.

### Problem ###

{problem}

### Solution ###

{solution}

In this stage, you are limited to analysis and planning only. You must NOT verify, compute, or decide correctness during Stage A.

Your objectives:

- \* Break down the original question into its essential components.

- \* Decompose the provided solution into its main steps.

- \* Design a sequence of verification steps for Stage B.

- Each verification step must describe \* what\* to check later, not perform the check itself.

- Focus on logical reasoning, consistency, assumptions, and calculations that need to be validated in Stage B.

After you list all the planned verification steps, stop your generation immediately to indicate that Stage A is complete.

#### Backward

You are now entering **Stage A: Backward Task Analysis \& Extraction**.

### Problem ###

{problem}

### Solution ###

{solution}

In this stage, you are limited to analysis and planning only.

You must NOT perform explicit verification, detailed calculations, or decide correctness during Stage A.

Your objectives:

- \* Identify the final conclusion or result asserted by the solution.

- \* Work backward from this conclusion to determine which intermediate statements or transformations must hold.

- \* Break down the original question into its essential components that support these backward dependencies.

- \* Decompose the provided solution into its main steps in a way that makes backward checking possible.

- \* Design a sequence of verification steps for Stage B.

- Each verification step must describe what backward relationship needs to be checked later, not perform the check itself.

- Focus on logical implications, hidden assumptions, reversibility of operations, and calculations that need to be validated in Stage B.

After you list all the planned verification steps, stop your generation immediately to indicate that Stage A is complete.

### B.2.3 Stage B: Validation Prompt

You are now entering **Stage B: Solution Analysis \& Judgment**.

In this stage, you will conduct a multi-turn verification of the steps designed in Stage A

. Each verification step may require one or more turns to complete.

**Turn Logic:**

- A single verification step may take multiple turns.

- Each turn must end with one of the following:

- \* `<python>...</python>` - when you need to perform a computation before continuing the same step in the next turn.

- \* `<step/>` - when the current step has been fully verified and you are ready to move to the next one.

- \* `<end_of_analysis/>` - when all steps are verified or a clear error has been found.

**Instructions for each turn:**

1. Restate the step you are verifying and reason carefully about it.

2. Check whether this step is logically consistent with all previously verified steps

.

3. If a calculation or detailed check is needed, output a `<python>...</python>` block after reasoning and stop your response.

The system will execute it and return the results for you to continue in the next turn.

**Rules for `<python>` blocks:**

- Use only necessary imports and `print()` statements for outputs.

- Do not use OS commands, file I/O, `input()`, or networking.

- A `<python>` block always marks the end of the current turn.

- Python tool can only be invoked at most three times across stage B.

### B.2.4 Stage C: Verdict Prompt

Now you are required to conduct stage C.

Given all prior analyses, provide your final review and boolean verdict.

Requirements:

- Review all previous verification steps and summarize why each step was correct or incorrect in `<review>...</review>`.

- If all previous steps were confirmed to be correct, output `<answer>>true</answer>`.

- If any step contained errors, or if you identify new inconsistencies at this stage, output `<answer>>false</answer>`.

## C Experimental Details

### C.1 Evaluation Protocol Details

To ensure strict comparability across settings, all candidate solutions in both BoN and sequential refinement are generated by the same fixed actor, Qwen2.5-7B-Instruct, using identical sampling hyperparameters (temperature = 1.0, top- $k$  = 50, and maximum completion length = 4096). For each problem, we sample 128 rollouts once and reuse this shared candidate pool across all verifier variants. In BoN, every verifier evaluates exactly the same 128 rollouts. In sequential refinement, the initial solution is sampled from this same pool and kept fixed across all refinement variants.

### C.2 Baselines.

We compare against three families of baselines that are commonly used in solution verification and preference modeling: (i) *text-reasoning* judges that directly generate a judgment from raw text, (ii) *outcome reward models* (ORMs) that score entire solutions, and (iii) *process reward models* (PRMs) that provide step-level supervision.

**Text-Reasoning LLMs.** These baselines use an instruction-tuned model as a text-only judge to produce a free-form verification and/or a final correctness decision without any explicit training. We instantiate this family with the following models: Qwen2.5-7B-Instruct (Qwen et al., 2025), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen3-4B (Yang et al., 2025b), DeepSeek-R1-Distill-Qwen-14B (DeepSeek-AI et al., 2025) and Mistral-Small-24B-Instruct (Mistral AI, 2025).

**Outcome Reward Models (ORM).** ORM as-sign a scalar score to a complete solution (or a response) to reflect its overall quality/correctness under preference supervision. We evaluate the following baselines: GRM-Gemma-2B (Yang et al., 2024), Skywork-V2-Llama-8B (Liu et al., 2025a), InternLM2-20B-RM (Cai et al., 2024), INF-ORM-Llama3.1-70B (Minghao Yang, 2024) and Starling-RM-34B (Zhu et al., 2023).

**Process Reward Models (PRM).** PRMs provide step-level feedback along a reasoning trajectory, enabling process supervision beyond outcome-only scoring. We include state-of-the-art PRMs: Qwen2.5-Math-PRM-7B (Zhang et al., 2025d), Math-Shepherd-Mistral-7B-PRM (Wang

et al., 2024b), EurusPRM (Yuan et al., 2025) and Skywork-PRM (He et al., 2024).

### C.3 Aggregation for Forward & Backward Agents

This section clarifies how verification results of forward and backward agents are aggregated in different experimental settings.

**BoN.** We average the confidence score from the forward and backward traces to obtain the aggregated score. Candidate solutions are then ranked according to the aggregated score, and the highest-scoring solution is selected. This score-based aggregation is sufficient for selection and introduces no additional modeling complexity.

**Verifier Revision.** In the multi-round verifier-revision, the forward and backward verification traces each produce a correctness judgment together with natural-language feedback. We adopt a conservative aggregation rule: a solution is considered correct only if both verification traces judge it as correct. If either verification trace identifies an error, the solution is regarded as incorrect and refinement is triggered. In this case, we invoke the verifier once more to generate targeted modification suggestions. The resulting feedback is then used as the supervisory signal for actor LLM.

### C.4 Experiment Setup

This section summarizes the experimental training setup used in our study, including data construction, supervised fine-tuning, and reinforcement learning.

**Data Construction.** We construct our training data through a two-stage synthesis and filtering pipeline. Qwen2.5-7B-Instruct is first used as a generator to sample question-answer pairs from diverse mathematical benchmarks. We then employ Qwen3-4B as a vanilla verifier and perform a Best-of-8 verification for each sample, using the number of correct candidates as a coarse difficulty signal. Based on this signal, the data are grouped by difficulty and used to support subsequent SFT and RL training.

**Supervised Fine-Tuning.** All models are fine-tuned using a learning rate of  $5 \times 10^{-6}$  with a batch size of 128 for a single epoch. The maximum sequence length is set to 21,600 tokens. During training, user inputs and tool outputs are masked and excluded from loss computation. SFT is con-

Model	@32	@64	@128
Base	70.60	72.60	72.40
Agentic Verifier w.o. Tool	73.20	76.00	77.40
Agentic Verifier w. Tool	<b>73.80</b>	<b>76.20</b>	<b>79.00</b>

Table 6: **Tool grounding on Math500.** BoN accuracy under different sampling budgets.

Model	@32	@64	@128
Base	52.20	51.42	51.94
Agentic Verifier w.o. Tool	51.95	54.52	55.06
Agentic Verifier w. Tool	<b>54.54</b>	<b>55.06</b>	<b>57.40</b>

Table 7: **Tool grounding on Gaokao2023.** BoN accuracy under different sampling budgets.

ducted on 16 NVIDIA A100 GPUs (80GB) with bfloat16 precision enabled.

**Reinforcement Learning.** For reinforcement learning, we adopt Group Relative Policy Optimization (GRPO). The actor model is trained with a learning rate of  $1 \times 10^{-6}$ , a per-device batch size of 1, and gradient accumulation over 64 steps. For each query, 8 candidate responses are sampled. Each rollout uses a maximum completion length of 4,096 tokens per turn, with a sampling temperature of 1.0, and the total rollout length is capped at 21,600 tokens. The same masking strategy as in SFT is applied, excluding user inputs and tool outputs from loss computation. The number of tool invocations is limited to at most three per rollout. Reinforcement learning is conducted on 32 NVIDIA A100 GPUs (80GB) with bfloat16 precision enabled.

**Scaling Inference-time Compute for Verifier.** For the same question  $x$  and candidate answer  $y$ , the verifier samples  $k$  verification trajectories and aggregating their scores, as

$$l_{\text{Agg@K}} = \frac{1}{k} \sum_{i=1}^k \pi_{\psi}(\boxed{\text{True}} \mid x, y, f_i, \mathbf{I})$$

$$f_i \sim \pi_{\psi}(\cdot \mid x, y, \mathbf{I}) \quad (9)$$

### C.5 Additional Ablation on Tool Usage

We provide additional analysis of tool usage in Agentic Verifier. Table 6 and Table 7 compares the Base model, Agentic Verifier without tools, and Agentic Verifier with Python tools. The results show that the without-tool variant already substantially improves over the Base model, indicating that the gains do not come solely from Python execution, but also from the structured verification pro-

Metric	Correct verdict	Incorrect verdict
Mean calls	1.60	1.34
Median calls	1.0	0.0
Success	88.2%	87.3%
Python Error	9.6%	10.6%
Quota Exceeded	2.1%	2.0%
Timeout	< 0.1%	< 0.1%

Table 8: **Tool execution statistics.** We compare tool-usage patterns between samples with correct and incorrect final verifier verdicts.

cess itself. Adding the Python tool improves performance on both MATH500 and Gaokao2023, showing that external grounding provides additional benefits on top of the verifier logic.

Table 8 further summarizes tool execution behavior during inference. We cap tool usage at 3 Python calls per sample. A successful tool call is defined as one that finishes normally and returns an executable result, i.e., without Python/runtime errors, timeout, or being blocked by the call budget. Overall, tool usage remains moderate and stable. Tool usage is slightly more frequent for correct verdicts, while the execution outcome breakdown remains broadly similar across the two groups. This suggests that correct verification tends to benefit from somewhat more tool grounding.

### C.6 Experimental Details for Generalization Beyond Math

This section summarizes the experimental setup for the generalization analysis beyond mathematical reasoning.

**Benchmarks.** We use two complementary benchmarks: LiveCodeBench for code generation and HotpotQA for multi-hop question answering.

**Tool Interfaces.** For LiveCodeBench, verification is grounded by Python execution. For HotpotQA, verification is grounded by web-search-based retrieval.

**Actor and Rollouts.** We use Qwen3-8B as the actor to sample rollouts for both benchmarks. For HotpotQA, 150 randomly selected queries are used.

**Evaluation Protocol.** Different from the main TTS experiments, this setting isolates verifier capability rather than evaluating the full test-time scaling pipeline. We report evaluation accuracy on sampled rollouts, i.e., whether the verifier correctly judges a sampled rollout. At inference time, the verifier does not observe the ground-truth answer

and only inspects the sampled rollout to produce a verdict.

### C.7 Data Contamination Check

We additionally examine whether the training sources overlap with the evaluation benchmarks. Based on the documented provenance of the training datasets, we do not find evidence that the evaluation benchmarks are included in training. Specifically, all three data sources trace their competition data only up to AIME 2023, thus excluding AIME 2024. GSM8K and Gaokao2023 are not included in any declared source dataset. While MATH appears in Light-R1 (a component of AReaL-boba-106k), it explicitly removes MATH500 from training. Therefore, we do not find direct overlap between the reported evaluation benchmarks and the training data used in this work.

## D Case Study

This section presents representative case studies to qualitatively analyze the behavior of different verification methods.

### D.1 Generative Verifier

#### Question

Two runners,  $A$  and  $B$ , start at a point  $O$  on a linear track and run in the same direction. Runner  $B$  runs three times as fast as runner  $A$ . An observer stands at point  $P$  such that  $\overline{OP}$  is perpendicular to the track. Find the maximum value of  $\angle APB$ , in degrees.

#### Candidate Solution

The solution assumes runner  $A$  and runner  $B$  move at speeds 1 and 3 units per second, respectively. Let  $d$  denote the distance from point  $P$  to the starting point  $O$ . After  $t$  seconds, runner  $A$  has traveled  $t$  units and runner  $B$  has traveled  $3t$  units. The solution models the distances from  $P$  to  $A$  and  $B$  as  $d - t$  and  $3d - 3t$ , and applies the Pythagorean theorem to derive

$$d^2 = (d - t)^2 + (3d - 3t)^2.$$

Solving this equation yields  $d = t$ , and the solution concludes that  $\angle APB = 90^\circ$ .

Note that the verifier’s original reasoning is lengthy and difficult to follow. For clarity, we pro-

vide a concise summary of the core reasoning ideas, while retaining the key computational steps used by the verifier.

#### Generative Verifier

<think> To verify the solution, proceed step by step by examining the geometric modeling.

First, model the geometry explicitly. Align the track with the  $x$ -axis and place point  $P$  at  $(0, d)$ . After  $t$  seconds, runner  $A$  and runner  $B$  are located at  $(t, 0)$  and  $(3t, 0)$ , respectively.

Next, compute the distances from point  $P$  to the runners. These distances follow directly from the coordinate setup:

$$PA = \sqrt{t^2 + d^2},$$

$$PB = \sqrt{(3t)^2 + d^2}.$$

Then, compare these expressions with the distances assumed in the solution. The solution instead uses  $d - t$  and  $3d - 3t$ , which would only be valid if the runners were moving directly toward point  $P$ . This does not match the given motion, where both runners move along the track away from the starting point  $O$ .

Next, examine the equation derived in the solution. Using the assumed distances, the solution proposes:

$$d^2 = (d - t)^2 + (3d - 3t)^2.$$

With the correct distance expressions, this equation cannot be obtained from a valid application of the Pythagorean theorem to points  $A$ ,  $B$ , and  $P$ .

Finally, check the implication of the condition  $d = t$  obtained in the solution. Under this condition, both runners coincide at point  $O$ , causing points  $A$  and  $B$  to overlap. In this degenerate configuration, the angle  $\angle APB$  is not defined, rather than being  $90^\circ$ . </think>

<verify> The solution incorrectly models the distances from point  $P$  to the runners. With the track aligned to the  $x$ -axis and  $P$  at  $(0, d)$ , the correct distances are

$$PA = \sqrt{t^2 + d^2}, \quad PB = \sqrt{(3t)^2 + d^2}.$$

Replacing these with  $d - t$  and  $3d - 3t$  leads to an invalid application of the Pythagorean theorem and produces a degenerate configuration in which the angle  $\angle APB$  is not well-defined. Therefore, the angle maximization argument in the solution is incorrect. </verify>

<answer> **false** </answer>

## D.2 Forward Verifier

The forward verifier performs multi-turn verification by sequentially inspecting the candidate solution and identifying potential issues at each step. Below we present the intermediate outputs produced during this process. Note that the original thinking traces are often lengthy. For clarity, we summarize the core reasoning ideas while explicitly retaining the key computations whenever they are involved.

### Question

The equation

$$x^{10} + (13x - 1)^{10} = 0$$

has ten complex roots

$r_1, \bar{r}_1, r_2, \bar{r}_2, r_3, \bar{r}_3, r_4, \bar{r}_4, r_5, \bar{r}_5$ , where the bar denotes complex conjugation. Find the value of

$$\frac{1}{r_1 \bar{r}_1} + \frac{1}{r_2 \bar{r}_2} + \frac{1}{r_3 \bar{r}_3} + \frac{1}{r_4 \bar{r}_4} + \frac{1}{r_5 \bar{r}_5}.$$

### Candidate Solution

Note that if  $r$  is a root of  $x^{10} + (13x - 1)^{10} = 0$ , then  $(13r)^{-1}$  is also a root. Hence, the roots can be paired as  $(r_k, (13r_k)^{-1})$  for  $k = 1, \dots, 5$ .

For each such pair, note that

$$((13r_k)^{-1} (13\bar{r}_k)^{-1})^{-1} = 13^2 r_k \bar{r}_k.$$

Hence the paired contribution is

$$\frac{1}{r_k \bar{r}_k} + 13^2 r_k \bar{r}_k.$$

Summing over all five pairs yields

$$\sum_{i=1}^5 \frac{1}{r_i \bar{r}_i} = \sum_{k=1}^5 \left( (r_k \bar{r}_k)^{-1} + 13^2 r_k \bar{r}_k \right).$$

By Vieta's formulas, the sum of the roots is 0, which implies that the sum of their squares is also 0. Consequently,

$$\sum_{k=1}^5 r_k \bar{r}_k = 0,$$

and therefore

$$\sum_{i=1}^5 \frac{1}{r_i \bar{r}_i} = 0.$$

The answer is  $\boxed{0}$ .

## Forward Verifier — Stage A: Plan

<think> To start Stage A, first identify what the problem asks and what the candidate solution assumes.

The problem gives a degree-10 polynomial equation

$$x^{10} + (13x - 1)^{10} = 0$$

with roots occurring in complex conjugate pairs. The target is

$$\sum_{i=1}^5 \frac{1}{r_i \bar{r}_i},$$

which is the sum of reciprocals of squared moduli. This quantity is typically approached via algebraic identities about roots, rather than solving roots explicitly.

The candidate solution relies on two main structural moves: (1) a claimed root transformation: if  $r$  is a root then  $\frac{1}{13r}$  is also a root, and uses it to pair roots and rewrite the target sum; (2) an application of Vieta's formulas to assert certain symmetric sums are zero, which then leads to a denominator claimed to be zero.

Since later steps depend on these claims, Stage B should verify them in the same order: validate the claimed root mapping first, then inspect how the pairing is used, then check the Vieta-based statements that produce the zero denominator. </think>

Stage B will follow these checkable steps: Step 1. Root transformation check: Test whether "if  $r$  is a root, then  $\frac{1}{13r}$  is also a root" holds by direct substitution.

Step 2. Pairing-and-rewrite dependency check: Trace how the solution uses the claimed mapping to pair roots and rewrite the target expression, and identify which later equalities depend on this pairing.

Step 3. Vieta/coefficient consistency check: Verify whether the solution's Vieta-based claims (e.g., "sum of roots is 0") match the actual polynomial coefficients.

Step 4. Zero-denominator justification check: Examine the step that introduces a denominator asserted to be 0, and determine whether that "0" follows from valid identities or from earlier invalid assumptions.

**Stage A complete.**

### Forward Verifier — Stage B: Validation

#### Step 1

<think> Start with the claimed root transformation. The solution asserts:

If  $r$  is a root of

$$x^{10} + (13x - 1)^{10} = 0,$$

then  $s = \frac{1}{13r}$  is also a root.

To test this claim, substitute  $s = \frac{1}{13r}$  into the polynomial.

Compute:

$$s^{10} = \left(\frac{1}{13r}\right)^{10} = \frac{1}{13^{10}r^{10}}.$$

Also,

$$13s - 1 = 13 \cdot \frac{1}{13r} - 1 = \frac{1}{r} - 1 = \frac{1 - r}{r},$$

so

$$(13s - 1)^{10} = \left(\frac{1 - r}{r}\right)^{10} = \frac{(1 - r)^{10}}{r^{10}}.$$

Thus,

$$s^{10} + (13s - 1)^{10} = \frac{1 + 13^{10}(1 - r)^{10}}{13^{10}r^{10}}.$$

For  $s$  to be a root, this expression must equal 0, which requires

$$1 + 13^{10}(1 - r)^{10} = 0$$

$$(1 - r)^{10} = -\frac{1}{13^{10}}.$$

However, the original root condition is

$$r^{10} + (13r - 1)^{10} = 0$$

$$(13r - 1)^{10} = -r^{10},$$

which is not the same constraint as  $(1 - r)^{10} = -\frac{1}{13^{10}}$ . So the substitution does not show that  $s = \frac{1}{13r}$  must satisfy the original equation. </think>

The direct substitution yields

$$s^{10} + (13s - 1)^{10} = \frac{1 + 13^{10}(1 - r)^{10}}{13^{10}r^{10}},$$

so  $s$  is a root only if  $(1 - r)^{10} = -\frac{1}{13^{10}}$ , which does not follow from  $r^{10} + (13r - 1)^{10} = 0$ . The claimed " $r \mapsto \frac{1}{13r}$ " root transformation is therefore not justified by the solution's argument. <step/>

### Forward Verifier — Stage B: Validation

#### Step 2

<think> Next, inspect how the solution uses the claimed mapping. It pairs terms as

$$\frac{1}{r\bar{r}} + \frac{1}{\frac{1}{13r} \cdot \frac{1}{13\bar{r}}} = \frac{1}{r\bar{r}} + \frac{13^2}{r\bar{r}}.$$

Algebraically, the simplification

$$\frac{1}{\frac{1}{13r} \cdot \frac{1}{13\bar{r}}} = \frac{13^2}{r\bar{r}}$$

is fine, but this step implicitly assumes that both  $r$  and  $\frac{1}{13r}$  are among the roots being summed over. Therefore, even if the algebra inside the term is correct, the pairing is only valid if the root-mapping claim is correct and the roots can indeed be partitioned into such pairs.

Since the previous step did not establish the root transformation, the correctness of the pairing—and hence the rewritten form of the target sum—depends on an unverified assumption. </think>

The rewrite

$$\frac{1}{r\bar{r}} + \frac{1}{\frac{1}{13r} \cdot \frac{1}{13\bar{r}}} = \frac{1}{r\bar{r}} + \frac{13^2}{r\bar{r}}$$

is algebraically valid, but the use of this identity inside the target sum relies on the

unverified claim that  $\frac{1}{13r}$  is also a root paired with  $r$ . Thus, the subsequent reduction of the five-term sum to a smaller grouped sum is not justified without establishing the root mapping. <step/>

### Forward Verifier — Stage B: Validation

#### Step 3

<think> Now inspect the Vieta-based claim. The solution states:

“By Vieta’s formulas, the sum of the roots is 0, so the sum of the squares of the roots is 0 as well.”

To check the “sum of roots is 0” statement, consider the polynomial

$$P(x) = x^{10} + (13x - 1)^{10}.$$

The sum of roots equals

$$-\frac{[x^9] P(x)}{[x^{10}] P(x)}.$$

Compute the leading coefficient:

$$[x^{10}] P(x) = 1 + 13^{10}.$$

Compute the  $x^9$  coefficient: the term  $x^{10}$  contributes 0 to  $x^9$ . From  $(13x - 1)^{10}$ , the  $x^9$  term comes from choosing 9 factors of  $13x$  and one factor of  $(-1)$ :

$$[x^9] (13x - 1)^{10} = \binom{10}{9} 13^9 (-1)^1.$$

Thus,

$$[x^9] P(x) = -10 \cdot 13^9.$$

Therefore the sum of roots is

$$-\frac{-10 \cdot 13^9}{1 + 13^{10}} = \frac{10 \cdot 13^9}{1 + 13^{10}},$$

which is not 0. So the solution’s Vieta statement “sum of roots is 0” does not match the polynomial coefficients. </think>

For  $P(x) = x^{10} + (13x - 1)^{10}$ ,

$$[x^{10}] P(x) = 1 + 13^{10},$$

$$[x^9] P(x) = -10 \cdot 13^9,$$

so the sum of roots is

$$-\frac{[x^9] P(x)}{[x^{10}] P(x)} = \frac{10 \cdot 13^9}{1 + 13^{10}} \neq 0.$$

Hence the solution’s claim that the sum of the roots is 0 is inconsistent with Vieta’s formulas for this polynomial. <step/>

### Forward Verifier — Stage B: Validation

#### Step 4

<think> Finally, examine the step where the solution forms a fraction with a denominator it asserts equals 0, and concludes the entire expression equals 0.

That step depends on two separate requirements: (1) the earlier pairing/rewrite must be valid (which depended on the root transformation); (2) the denominator being set to 0 must follow from correct identities about the roots of the polynomial.

From the previous turn, the claim “sum of roots is 0” is not supported by the actual coefficients. So any subsequent inference that a symmetric sum vanishes (and in particular a denominator equals 0) lacks justification from Vieta’s formulas as applied to this polynomial.

Moreover, even conceptually,  $r\bar{r} = |r|^2$  is nonnegative real for any complex root  $r \neq 0$ , so a sum of several such terms being exactly 0 would force each term to be 0, which would require  $r = 0$ . This would need to be consistent with the original equation, and cannot be assumed without verification. </think>

The “division by 0” step relies on unsupported claims. The polynomial coefficients do not support the statement that the sum of roots is 0, so the subsequent denominator being asserted as 0 is not justified by Vieta’s formulas. In addition, since  $r\bar{r} = |r|^2 \geq 0$  for each nonzero root, forcing a sum of such terms to be 0 would require each  $|r|^2 = 0$ , i.e.,  $r = 0$ , which must be checked against the original equation rather than assumed. <end\_of\_analysis/>

### Forward Verifier — Stage C: Verdict

<review> The candidate solution is invalid for two independent reasons.

(1) The solution assumes a root-closure property: if  $r$  is a root, then  $\frac{1}{13r}$  is also a root. Direct substitution shows that  $s = \frac{1}{13r}$  satisfies the equation only under an additional constraint on  $r$ , which is not implied by  $r^{10} + (13r - 1)^{10} = 0$ . Therefore the proposed root pairing is not justified.

(2) The solution incorrectly applies Vieta's formulas by claiming the sum of the roots is 0. For  $P(x) = x^{10} + (13x - 1)^{10}$ , the coefficients imply the sum of roots is  $\frac{10 \cdot 13^9}{1 + 13^{10}} \neq 0$ . As a result, later deductions (including the step that introduces a denominator asserted to be 0) do not follow from valid polynomial identities.

Since the key structural claims fail, the final conclusion 0 is not supported.   
 </review>  
 <answer> **false** </answer>

### D.3 Backward Verifier

#### Question

The product of  $3t^2 + 5t + a$  and  $4t^2 + bt - 2$  is  $12t^4 + 26t^3 - 8t^2 - 16t + 6$ . What is  $a + b$ ?

#### Candidate Solution

We can expand the product of the two polynomials to get Let

$$E(t) = (3t^2 + 5t + a)(4t^2 + bt - 2).$$

Then

$$\begin{aligned} E(t) &= 12t^4 + (3b + 20)t^3 \\ &\quad + (4a + 5b - 10)t^2 \\ &\quad + (-2a - 4b + 10)t - 2a. \end{aligned}$$

Since this is equal to  $12t^4 + 26t^3 - 8t^2 - 16t + 6$ , we can equate the coefficients of corresponding powers of  $t$  to get the following system of equations:

$$\begin{aligned} 3b + 20 &= 26, \\ 4a + 5b - 10 &= -8, \\ -2a - 4b + 10 &= -16, \\ -2a &= 6. \end{aligned}$$

From the last equation, we have  $a = -3$ . Substituting this into the third equation, we get  $-2(-3) - 4b + 10 = -16$ , which simplifies to  $-4b = -22$  or  $b = \frac{11}{2}$ . Therefore,

$$a + b = -3 + \frac{11}{2} = \frac{5}{2}.$$

The answer is:  $\frac{5}{2}$

#### Backward Verifier — Stage A: Plan

<think> The candidate solution claims the final value  $a + b = \frac{5}{2}$ . For this to be justified, several dependencies must hold in reverse order.

If the final value is correct, then the solution must have obtained correct values of  $a$  and  $b$ . Those values must satisfy a valid coefficient-matching system. And that system must come from a correct expansion of  $(3t^2 + 5t + a)(4t^2 + bt - 2)$ .

So the backward checks are: (1) expansion correctness; (2) coefficient equations correctness; (3) consistency of the reported  $(a, b)$  with the verified equations. </think>

#### Backward verification plan

- Step 1: Verify the polynomial expansion the solution relies on (especially  $t^2$  and  $t$  coefficients).
- Step 2: Verify the coefficient-matching system derived from that expansion matches  $12t^4 + 26t^3 - 8t^2 - 16t + 6$ .
- Step 3: Check whether the reported  $a = -3$  and  $b = \frac{11}{2}$  satisfy the verified system, and whether they imply  $a + b = \frac{5}{2}$ .

#### Backward Verifier — Stage B: Validation

##### Step 1

<think> To support the final claim, the expansion

$$(3t^2 + 5t + a)(4t^2 + bt - 2)$$

used by the solution must be algebraically correct.

I first re-expand the product in a structured way to see what coefficients must appear.

Compute key contributions:

$$3t^2 \cdot (4t^2 + bt - 2) = 12t^4 + 3bt^3 - 6t^2$$

$$5t \cdot (4t^2 + bt - 2) = 20t^3 + 5bt^2 - 10t$$

$$a \cdot (4t^2 + bt - 2) = 4at^2 + abt - 2a$$

Summing like terms gives:

$$\begin{aligned} &12t^4 + (3b + 20)t^3 \\ &\quad + (4a + 5b - 6)t^2 \\ &\quad + (ab - 10)t \\ &\quad - 2a. \end{aligned}$$

The solution instead uses  $(4a + 5b - 10)t^2$  and  $(-2a - 4b + 10)t$ , so I want a quick tool check to confirm the correct symbolic expansion. </think>

```
<python>
from sympy import symbols, expand
t = symbols('t')
a, b = symbols('a b')
poly = (3*t**2 + 5*t + a) * (4*t**2 + b*t - 2)
print(expand(poly))
</python>
```

#### Tool Result

```
[Stdout] a*b*t + 4*a*t**2 - 2*a + 3*b*t**3
+ 5*b*t**2 + 12*t**4 + 20*t**3 - 6*t**2 -
10*t
```

### Backward Verifier — Stage B: Validation

#### Step 2

<think> The tool output confirms the correct expansion is

$$\begin{aligned} &12t^4 + (3b + 20)t^3 \\ &\quad + (4a + 5b - 6)t^2 \\ &\quad + (ab - 10)t - 2a. \end{aligned}$$

Now I compare this with the candidate's claimed expansion:

$$\begin{aligned} &12t^4 + (3b + 20)t^3 \\ &\quad + (4a + 5b - 10)t^2 \\ &\quad + (-2a - 4b + 10)t - 2a. \end{aligned}$$

Two mismatches appear immediately:

- (i)  $(4a + 5b - 6) \neq (4a + 5b - 10)$ .
- (ii)  $(ab - 10) \neq (-2a - 4b + 10)$ .

Since the coefficient expressions differ, the coefficient-matching system written in the solution cannot be derived from the true product, so the backward dependency from

the final value is broken at the equation-derivation stage. </think>

<step/>

#### Step 3

<think> Even if the solution's later arithmetic were internally consistent, the reported parameters must satisfy the *verified* coefficient equations from the correct expansion when matched to  $12t^4 + 26t^3 - 8t^2 - 16t + 6$ .

The solution reports  $a = -3$  (from  $-2a = 6$ ), and  $b = \frac{11}{2}$ . I check whether  $b = \frac{11}{2}$  is compatible with  $3b + 20 = 26$ :

$$3 \cdot \frac{11}{2} + 20 = \frac{33}{2} + 20 = \frac{73}{2} \neq 26,$$

so the reported  $b$  already fails the verified system.

Therefore the final claim  $a + b = \frac{5}{2}$  is not supported by the correct algebraic conditions. </think>

<end\_of\_analysis/>

### Backward Verifier — Stage C: Verdict

<review> Step 1 verified the correct expansion of  $(3t^2 + 5t + a)(4t^2 + bt - 2)$  by structured term collection and a single symbolic tool check. The correct coefficients are  $(3b + 20)$  for  $t^3$ ,  $(4a + 5b - 6)$  for  $t^2$ , and  $(ab - 10)$  for  $t$ .

Step 2 showed the candidate solution uses incorrect expressions for the  $t^2$  and  $t$  coefficients, so the coefficient-matching system it writes does not follow from the actual product.

Step 3 checked consistency of the reported values with the verified system and found the claimed  $b = \frac{11}{2}$  contradicts  $3b + 20 = 26$  immediately. Thus the final claim  $a + b = \frac{5}{2}$  is not justified.

Overall, the backward dependencies required for the conclusion fail due to an incorrect expansion and inconsistent parameter values. </review>

<answer>false</answer>