Stop Summation: Min-Form Credit Assignment Is All Process Reward Model Needs for Reasoning

Jie Cheng[1,2], **Gang Xiong**[1,2], **Ruixi Qiao**[1,2], **Lijun Li**[3], **Chao Guo**[1], **Junle Wang**[4], **Yisheng Lv**[1,2,5*], **Fei-Yue Wang**[5,6,7,8]
[1]State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]Shanghai Artificial Intelligence Laboratory [4]Tencent
[5]Faculty of Innovation Engineering, Macau University of Science and Technology
[6]DeSci Center of Parallel Intelligence, Obuda University
[7]Research Center for Chinese Economics and Social Security, University of Chinese Academy of Sciences
[8]Institute of Automation, Chinese Academy of Sciences

Abstract

Process reward models (PRMs) have proven effective for test-time scaling of Large Language Models (LLMs) on challenging reasoning tasks. However, reward hacking issues with PRMs limit their successful application in reinforcement fine-tuning. In this paper, we identify the main cause of PRM-induced reward hacking: the canonical summation-form credit assignment in reinforcement learning (RL), which defines the value as cumulative gamma-decayed future rewards, easily induces LLMs to hack steps with high rewards. To address this, we propose PURE: **P**rocess s**U**pervised **R**einforcement l**E**arning. The key innovation of PURE is a min-form credit assignment that formulates the value function as the minimum of future rewards. This method significantly alleviates reward hacking by limiting the value function range and distributing advantages more reasonably. Through extensive experiments on 3 base models, we show that PRM-based approaches enabling min-form credit assignment achieve comparable reasoning performance to verifiable reward-based methods within only 30% steps. In contrast, the canonical sum-form credit assignment collapses training even at the beginning! Additionally, when we supplement PRM-based fine-tuning with just 10% verifiable rewards, we further alleviate reward hacking and produce the best fine-tuned model based on Qwen2.5-Math-7B in our experiments, achieving 82.5% accuracy on AMC23 and 53.3% average accuracy across 5 benchmarks. Moreover, we summarize the observed reward hacking cases and analyze the causes of training collapse. We release our code and model weights at `https://github.com/CJReinforce/PURE`.

1 Introduction

Reinforcement fine-tuning (RFT) of large language models (LLMs) for reasoning tasks has shown promise in developing advanced problem-solving abilities. Recent advancements (Wang et al., 2025; Zeng et al., 2025a), such as DeepSeek R1-Zero (Guo et al., 2025) and Kimi K1.5 (Team et al., 2025), have demonstrated strong reasoning skills through RFT with verifiable rewards, which provide sparse feedback for the entire response. However, as response length increases, sparse rewards potentially lead to inefficient learning (Sutton & Barto, 2018; Andrychowicz et al., 2017).

*Corresponding author

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

In contrast, process reward models (PRMs) offer dense feedback at each step of a response. PRMs have proven effective in improving LLMs' performance on challenging reasoning tasks through test-time scaling (Lightman et al., 2023; Wang et al., 2023; Guan et al., 2025). However, successful applications of PRM in RFT for LLMs remains limited (Setlur et al., 2024). A key challenge of PRMs is that the neutral network-generated rewards can lead to reward hacking during training (Weng, 2024; Guo et al., 2025), causing unintended optimization toward higher rewards. Recent studies (Yuan et al., 2024; Cui et al., 2025) have explored implicit PRMs for fine-tuning LLMs, which resembles DPO-style reward formulation (Rafailov et al., 2023), but these still heavily relies on ground-truth verifiable rewards to provide training signals for the system. The question of *what causes reward hacking in PRM-based RFT and how to address it effectively* has not been explored widely.

To answer this question, we examine the usage of PRM in test-time scaling and identify a mismatch between test-time and training-time objectives. Thus in this paper, we propose PURE: **P**rocess s**U**pervised **R**einforcement l**E**arning, which introduces a min-form credit assignment method to align these objectives. Through analysis and experiments, we find that the canonical formulation of credit assignment in RL, named summation-form credit assignment that defines value as cumulative gamma-decayed future rewards, easily induces LLMs to hack high-reward steps and collapse training. Instead, we use the minimum future reward to quantify credit assignment, which constrains the value function's range and assigns advantages more reasonably to stabilize RL training (see the analysis in § 3.1). PURE offers several benefits: (1) it is simple to implement, requiring only a transformation of process rewards without additional code changes; (2) it achieves comparable or better performance in PRM-based RFT compared to recent R1-replication studies with around 3× efficiency gains; and (3) it supports the integration of both dense process rewards and sparse verifiable rewards. When combining both reward types in the PURE framework, we find that the auxiliary of few ground-truth signals further mitigates reward hacking caused by PRM.

In experiments, we first train a PRM using the PRM800K dataset (Lightman et al., 2023), a human-annotated dataset that evaluates the correctness of each step. Then we apply PURE framework to 3 models: Qwen2.5-7B, Qwen2.5-Math-7B, and Qwen2.5-Math-1.5B, using 3 reward configurations: only verifiable rewards, only process rewards, and a combination of both. This comprehensive setup allows us to compare PRM-based and verifiable reward-based RFT approaches. We also compare sum-form and min-form credit assignment methods when enabling PRMs and find that the former even collapses training at the beginning. In the discussion, we summarize observed reward hacking cases and analyze the causes of training collapse. Our key findings are outlined below.

1. Summation-form credit assignment easily induces LLMs to hack high-reward steps, leading to LLMs that prioritize thinking over problem-solving. Our min-form credit assignment avoids such reward hacking. (§ 3.1)
2. PRM-based RFT achieves performance comparable to the verifiable reward-based approach with around 3× efficiency gains if using our min-form credit assignment; otherwise, the training collapses even at the beginning! (§ 4.3)
3. The auxiliary of few ground-truth signals further mitigates PRM-induced reward hacking, achieving 82.5% accuracy on AMC23 and 53.3% average accuracy across MATH-500, Minerva Math, Olympiad Bench, AIME24, and AMC23 when using Qwen2.5-Math-7B as the base model. (§ 4.3)
4. We identify 3 types of PRM-induced reward hacking: (1) only thinking, not solving, (2) extremely few steps (1 step), (3) extremely few steps (0 step). We analyze causes, show examples, and provide solutions for each. (§ 5.1)
5. Long, highly repetitive samples that the verifier ruled correct cause training to collapse. These pseudo-positive samples, undetected by verifiers, provide numerous incorrect signals and collapse training all of a sudden (within 5 gradient steps). (§ 5.2)

2 Preliminaries

2.1 Credit Assignment Problem in Reinforcement Learning

Minsky (2007) described the credit assignment problem as "how to distribute the credit of success among the multitude of decisions involved". In other words, credit assignment is the problem of

estimating the influence of an action over an outcome from experience. Various assignment functions have been proposed to quantify the influence of actions (Pignatelli et al., 2023). Here, we adopt the state-action value as the assignment function to detail the problem (Cheng et al., 2024, 2025).

We model LLM reasoning as a step-level Markov Decision Process. Given a prompt, LLM generates steps sequentially. At step t, LLM π takes the prompt p and previous steps $\{a_1, \cdots, a_{t-1}\}$ as the input state s_t, where $s_t = (p, a_1, \cdots, a_{t-1})$. Then LLM generates a step a_t, sampled as $a_t \sim \pi(\cdot|s_t)$. PRM R^p emits a process reward $r_t^p = R^p(s_t, a_t)$. The canonical formulation of state-action value is:

$$Q^\pi(s_t, a_t) = \mathbb{E}\left[\sum_{i \geq t} \gamma^{i-t} r_i^p\right] \tag{1}$$

where γ is the discount factor. Eq. (1) shows that the influence of an action decreases over time, weighted by a discount factor γ. Actions closer to future outcomes have greater influence. Thus, the state-action value quantifies credit assignment. With Eq. (1), we can derive the advantage function (Schulman et al., 2017; Shao et al., 2024; Ahmadian et al., 2024; Hu, 2025) and update LLM using the policy gradient loss.

2.2 Process Reward vs. Verifiable Reward

Verifiable rewards are sparse, rule-based rewards assigned to an entire response. For a prompt, LLM generates a sequence of steps $\{a_1, \ldots, a_n\}$. After the final step a_n, a verifiable reward r^v is assigned based on whether the response matches the ground-truth answer. Verifiable rewards provide a straightforward, ground-truth signal for RL training, which has been commonly used in the training pipeline for advanced models (Yue et al., 2024; Guo et al., 2025; Team et al., 2025). In contrast, process rewards $\{r_1^p, \cdots, r_n^p\}$ are dense rewards to evaluate the quality of each step. Process rewards offer unique benefits in test-time scaling. During inference, LLM generates multiple responses for a single prompt, and PRM scores each step for each response. Prior work (Lightman et al., 2023; Wang et al., 2023; Zhang et al., 2025) typically aggregates process rewards into a outcome-level score using the minimum value, $\min(r_1^p, \cdots, r_n^p)$, and selects the best response based on this outcome-level score. This PRM-based approach outperforms majority voting and other reward model-based approaches.

Although dense rewards support effective training in traditional RL applications, they struggle to effectively fine-tune LLM for advanced reasoning Guo et al. (2025). Therefore, how to make PRMs as effective in training as they are in test-time scaling is an important and not widely studied topic.

3 PURE: Process Supervised Reinforcement Learning

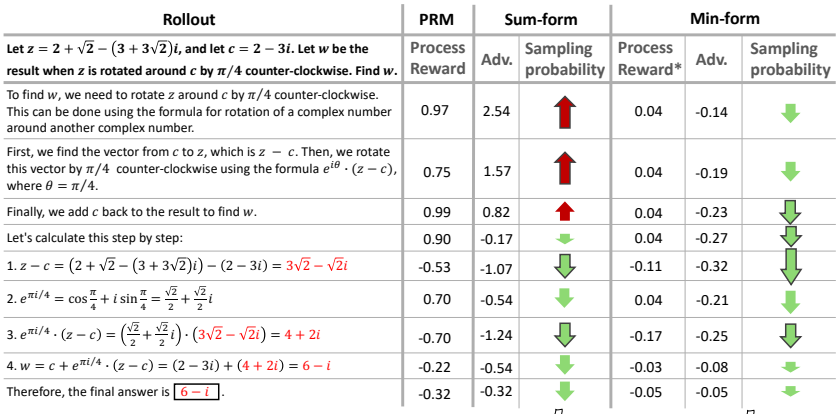
To effectively fine-tune LLMs using PRMs, we propose PURE: **P**rocess s**U**pervised **R**einforcement l**E**arning. PURE leverages a novel min-form formulation to quantify the credit assignment, inspired by the test-time application of PRM. In this section, we first detail the min-form credit assignment and analyze its effectiveness in § 3.1. Subsequently, we introduce an advantage estimator tailored for process rewards in § 3.4.

3.1 Min-form Credit Assignment

Consider a prompt p and a LLM π parameterized by θ, generating a n-step response denoted as $(a_1, \cdots, a_n)$. At step t, LLM π_θ takes the prompt p and previous steps $\{a_1, \cdots, a_{t-1}\}$ as the input state s_t, where $s_t = (p, a_1, \cdots, a_{t-1})$. Then LLM generates step a_t, sampled as $a_t \sim \pi_\theta(\cdot|s_t)$. A PRM R^p emits a process reward $r_t^p = R^p(s_t, a_t)$. Unlike traditional RL whose objective is the summation of discounted rewards , the usage of PRM in test-time scaling suggests that RL for reasoning tasks should optimize the following objective:

$$\arg\max_\theta \ \mathbb{E}\left[\min(r_1^p, \cdots, r_n^p)\right] \tag{2}$$

where the expectation is over prompts and step-level actions. This objective implies: **(i)** Only the "worst" step that gets the minimum process reward determines the value of the entire response. **(ii)** For steps before the worst step, their existence as the input state induces LLM to generate the worst step. **(iii)** Steps after the worst step actually do not contribute to the objective.



(a) The response of the model after updating using the **sum-form** credit assignment method. **The model learns to only output thinking steps without actually solving the problem!**

(b) The response of the model after updating using the **min-form** credit assignment method.

Figure 1: Comparison of summation-form and min-form credit assignment. Adv. and Process reward* in the table means advantage and transformed process reward, respectively. The incorrect steps in the rollout are highlighted in red, and our PRM reasonably assigns negative scores to these steps. For simplicity, advantage baseline and KL penalty terms are omitted in advantage calculation here, and discount factor γ and transform temperature T are set to 1. Arrows indicate changes in sampling probability, with larger changes marked by contoured arrows.

Therefore, we define the min-form credit assignment function as follows: For a n-step response, let step w be the worst step, *i.e.*, $w = \arg\min(r_1^p, \cdots, r_n^p)$. The return G and state-action value Q^π functions are defined as follows:

$$G(s_t, a_t|\tau) = \begin{cases} \min(r_t^p, \cdots, r_n^p), & \text{if } t \leq w \\ 0, & \text{if } t > w \end{cases} \tag{3}$$

$$Q^\pi(s_t, a_t) = \mathop{\mathbb{E}}_{\tau}[G(s_t, a_t|\tau)] \tag{4}$$

where $\tau = (s_1, a_1, r_1^p, \cdots, s_n, a_n, r_n^p)$ denotes the trajectory. Eq. (4) is a quantitative expression of the above three-fold analysis of the objective. To implement this in the simplest way possible, we transform the process rewards in trajectory τ using:

$$r_i^{p*} = \frac{\exp(-r_i^p/T)}{\sum_{j=1}^n \exp(-r_j^p/T)} \cdot r_i^p \tag{5}$$

where T is the transform temperature, and r_i^{p*} is the transformed process reward. The transform function assigns higher weights to lower rewards. After transformation, the rest of code remains unchanged. We explain this as follows. As the transform temperature $T \to 0^+$, Eq. (5) yields $r_w^{p*} = r_w^p$ for the worst step w and $r_i^{p*} = 0$ for $i \neq w$. Setting $\gamma = 1$, the return becomes: **(i)** For step $t \leq w$,

$$G_t = \sum_{i=t}^n \gamma^{i-t} \cdot r_i^{p*} = r_w^p = \min(r_t^p, \cdots, r_n^p) \tag{6}$$

(ii) For $t > w$, $G_t = 0$. The return exactly matches Eq. (3) without altering the return computation logic.

Simple implementation. To align with widely adopted token-level PPO loss, we convert step-level rewards to token-level rewards. We exclusively assign the transformed process reward to the final token of each step, with all other tokens receiving a reward of 0. To preserve algorithmic flexibility, we also support verifiable rewards assigned to the last token of the complete response, which can be used alongside process rewards.

3.2 Quantitative Analysis

Here we analyze that the min-form has a smaller value estimation error than the summation-form credit assignment method. We start by first establishing 2 reasonable assumptions:

Assumption 1 (Bounded Process Reward Error): The Process Reward Model, R^P, produces a reward estimate $r_t^p = R^P(\cdot)$ for each reasoning step. It is assumed that there exists a "true" or optimal reward r_t^* for that step, and the estimation error is uniformly bounded by a constant $\epsilon \geq 0$. For any step t, it holds that $|r_t^p - r_t^*| \leq \epsilon$.

Assumption 2 (Bounded Rewards): The true rewards, and consequently the estimated rewards, are bounded. There exists a maximum possible reward $R_{\max}$ such that $|r_t^*| \leq R_{\max}$ and $|r_t^p| \leq R_{\max} + \epsilon$ for all steps t.

Theorem 1 (**Q-Value Estimation Error Bound Comparison**). *Under Assumptions 1 and 2, for any state-action pair (s_t, a_t) and a trajectory τ with n reasoning steps:*

1. ***Sum-form Error Bound:*** *The estimation error for the summation-form Q-value, is bounded by:*

$$|Q_\pi^{sum}(s_t, a_t) - Q_\pi^{sum,*}(s_t, a_t)| \leq \sum_{i=t}^{t+n-1} \gamma^{i-t}\epsilon$$

   For an infinite horizon ($n \to \infty$), this bound becomes:

$$|Q_\pi^{sum}(s_t, a_t) - Q_\pi^{sum,*}(s_t, a_t)| \leq \frac{\epsilon}{1-\gamma}$$

2. ***Min-form Error Bound:*** *The estimation error for the min-form Q-value, derived from Eq. (4), is bounded by:*

$$|Q_\pi^{min}(s_t, a_t) - Q_\pi^{min,*}(s_t, a_t)| \leq \epsilon$$

The proof is provided in Appendix B. This theorem shows that the error bound of sum-form method is amplified by a factor dependent on the horizon length and the discount factor γ, accumulating the error over steps. However, the error bound of min-form method is strictly limited by the single-step reward estimation error ϵ and does not accumulate with the number of steps.

3.3 Qualitative Analysis

Min-form credit assignment restricts the range of value function to be the same as that of the reward function, which contributes to stabilizing RL training. In contrast, for the summation-form credit assignment, the range of value is determined by the range of reward and the number of steps, causing excessive value as the number of steps increases and unintended reward hacking.

We now visualize the differences between summation-form and our min-form credit assignment using a real training example. As shown in Figure 1, incorrect steps are highlighted in red, with arrows showing the magnitude and direction of changes in sampling probability. Steps with relatively large changes in sampling probability are marked by contoured arrows. Note that the first 3 steps are the thinking, and the subsequent 6 steps are the solution.

Figure 1 shows that **summation-form credit assignment significantly alters sampling probabilities**, increasing the sampling probabilities for thinking steps and decreasing that for incorrect solution steps. However, this results in a shotcut towards reward hacking: the model learns to only output thinking steps without actually solving the problem! In other words, **the model hacks the implicit pattern inside high-reward steps**, *i.e.*, thinking. In contrast, our min-form credit assignment reduces sampling probabilities across all steps, with the largest reduction at the first incorrect step. This aligns with the behavior of verifiable rewards (an incorrect final answer leads to a reduction in sampling probability of all steps) and assigns advantages more rationally based on step correctness.

3.4 Advantage Estimation

Following the token-level rewards in § 3.1, we are ready to compute advantages. We compare several advantage estimators, including GAE (Schulman et al., 2017), RLOO (Ahmadian et al., 2024), GRPO (Shao et al., 2024), and REINFORCE++ (Hu, 2025), as discussed in § F.2. RLOO are strong enough to produce stable and effective results.

For a single prompt, LLM generates K responses. The maximum generation length is limited to N. Let r_i^v and $r_{i,j}^{p*}$ $(j = 1, \cdots, N)$ denote the verifiable reward and token-level transformed process rewards for response y_i $(i = 1, \cdots, K)$, respectively. The token-level advantage for y_i is formulated as follows:

$$A_{i,t} = \underbrace{r_i^v - \frac{1}{K-1}\sum_{k\neq i} r_k^v}_{\text{RLOO with verifiable reward}} + \underbrace{\sum_{j=t}^{N} \gamma^{j-t} \cdot r_{i,j}^{p*} - \underbrace{\frac{\sum_{k\neq i}\sum_{l=1}^{N}\sum_{j=l}^{N}\gamma^{j-l}\cdot r_{k,j}^{p*}}{(K-1)N}}_{\text{token-level baseline}}}_{\text{RLOO with token-level transformed process rewards}} \tag{7}$$

where $t = 1, \cdots, N$. Specifically: **(i)** for verifiable rewards, we directly adopt RLOO. **(ii)** For process rewards, we employ a token-level baseline to avoid reward hacking, as discussed in the second case in § 5.1. Moreover, we normalize the baseline with the max generation length N instead of response length to avoid length biases, similar to concurrent work (Liu et al., 2025).

4 Experiments

4.1 PRM Training and Evaluation

To conduct our RFT experiments, we first require a PRM to provide process rewards. We train our PRM based on Qwen2.5-Math-7B due to its strong performance in mathematical tasks. Following Lightman et al. (2023), we treat the PRM training as a binary classification task. We replace the final layer of the model with a value head and train the model on the PRM800K dataset (Lightman et al., 2023) in 2 stages. In the first stage, we freeze the LLM parameters and train only the value head with a learning rate 10^{-4} for 3 epochs. In the second stage, we unfreeze the LLM parameters and fine-tune all parameters with a learning rate 10^{-6} for 1 epoch.

Table 1: Results of BoN evaluation. Rows marked with * are taken from Xiong et al. (2024).

Method	GSM8K	MATH
Pass@1 *	83.9	42.4
Majority Voting@1024 *	89.7	57.4
Deepseek-PRM-7B BoN@1024 *	**93.0**	58.1
PURE-PRM-7B BoN@1024	91.6	**62.6**

We evaluate our PRM, named PURE-PRM-7B, through 3 ways: Best-of-N (BoN) method, ProcessBench (Zheng et al., 2024), and PRM-Bench (Song et al., 2025). For BoN evaluation, we use rollout data from Xiong et al. (2024). For each question in GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), they uses Deepseek-7B to generate $K = 1024$ answers. We score each step using our PRM, transform process rewards using Eq. (5), and compute an outcome score by summing the transformed rewards for each answer. The answer with the highest outcome score is selected as the final answer. Our PURE-PRM-7B achieves BoN@1024 accuracy of 91.6% on GSM8K and 62.6% on MATH, performing comparably to the best results of 93.0% and 58.1% reported by Xiong et al. (2024)

On ProcessBench (Zheng et al., 2024), which assesses the PRM's ability to identify the first process error, our PRM achieves a state-of-the-art average F1 score of 57.5, surpassing the previous best F1 score of 56.5 reported in the benchmark. Detailed scores for each subset of ProcessBench are provided in Appendix C. On PRMBench (Song et al., 2025), which evaluates the fine-grained error detection capabilities of PRMs, our PURE-PRM-7B ranks third among open-source PRMs[2]. These results demonstrate that our PRM achieves top performance and is suitable for fine-tuning LLMs.

4.2 RL Settings

Reward types. Our framework supports 3 types of rewards: process reward only (PURE-PRM), verifiable reward only (PURE-VR), which matches the training setup of Deepseek R1-Zero, and a mix

[2]The official leaderboard is at `https://prmbench.github.io/`.

Table 2: Detailed performance of various models across 5 benchmarks. Report pass@1 accuracy tested with greedy decoding. The blue lines represent the models trained with our recipe.

Base Model	Method	MATH 500	Minerva Math	Olympiad Bench	AIME24	AMC23	Avg.
Qwen2.5-7B	SimpleRL-Zoo	78.4	31.2	39.1	16.7	50.0	43.1
	Base	71.4	23.9	35.3	10.0	52.5	38.6
	+ PURE-PRM	76.2	37.1	41.2	13.3	60.0	45.6
	+ PURE-VR	75.6	32.7	39.0	16.7	55.0	43.8
	+ PURE-PRM+VR	80.4	37.9	43.0	16.7	60.0	**47.6**
Qwen2.5-Math-7B	Eurus-2-7B-PRIME	79.2	38.6	42.1	26.7	57.8	48.9
	SimpleRL-Zoo	80.2	38.2	43.3	23.3	55.0	48.0
	Qwen2.5-7B-DPO-Zero	76.8	30.9	37.9	26.7	62.5	47.0
	Base	71.8	29.8	35.1	13.3	47.5	39.5
	+ PURE-PRM	81.8	38.2	44.7	16.7	60.0	49.3
	+ PURE-VR	79.4	36.8	41.8	23.3	60.0	48.3
	+ PURE-PRM+VR	82.6	37.1	44.1	20.0	82.5	**53.3**
Qwen2.5-Math-1.5B	Base	61.4	23.5	29.3	13.3	40.0	33.5
	+ PURE-PRM	75.2	26.5	36.4	13.3	50.0	40.3
	+ PURE-VR	74.2	27.9	36.0	10.0	55.0	40.6
	+ PURE-PRM+VR	76.0	31.6	37.2	16.7	55.0	**43.3**

of both (PURE-PRM+VR). For verifiable rewards, a reward of +1 is assigned if the generated answer matches the ground-truth answer; otherwise, a reward of 0 is given. No format-related rewards.

RL dataset. We use the RFT dataset from SimpleRL (Zeng et al., 2025b). It samples around 8,000 problems from MATH dataset (Hendrycks et al., 2021) with difficulty lv.3-5. For ground-truth (GT) answers, PURE-PRM method does not use GT answers. PURE-VR method requires GT answers for all problems. For PURE-PRM+VR, GT answers are randomly assigned to 1/10 of the problems, leading to about 800 problems with GT answers and 7,200 open problems. This setup aims to explore the effectiveness of process rewards as the main signal for RL training.

Hyperparameters. We use veRL (Sheng et al., 2024) to conduct experiments on 3 models: Qwen2.5-7B (Yang et al., 2024a), Qwen2.5-Math-7B, Qwen2.5-Math-1.5B (Yang et al., 2024b). We use a constant learning rate of 10^{-6} for PURE-VR and PURE-PRM+VR and 5×10^{-7} for PURE-PRM. Training steps are set to 500 for Qwen2.5-Math series and 1000 for Qwen2.5-7B. Other shared hyperparameters are set as follows: prompt batch size of 64, group size of 8 (generating 8 responses per prompt), training mini-batch size of 512, maximum generation length of 8192 tokens, sampling temperature of 1.0, KL coefficient of 10^{-3}, transform temperature in Eq (5) of 0.1, save interval for checkpoints of 50 steps.

Baselines. We compare our method with 3 state-of-the-art RFT methods: **(i)** Eurus-2-7B-PRIME (Cui et al., 2025): A model based on Qwen2.5-Math-7B fine-tuned with implicit PRM. **(ii)** SimpleRL-Zoo (Zeng et al., 2025b): A re-implementation of Deepseek R1's training recipe on several base models. **(iii)** Qwen2.5-7B-DPO-Zero (Xiong et al., 2025): A model based on Qwen2.5-Math-7B fine-tuned with iterative DPO.

Evaluation metrics. At test time, we evaluate performance on 5 competition-level mathematical benchmarks, including AIME24 (Li et al., 2024), AMC23 (Li et al., 2024), MATH500 (Hendrycks et al., 2021), Minerva Math (Lewkowycz et al., 2022), OlympiadBench (He et al., 2024). We report scores of best checkpoint saved in training. During training, we track about 20 metrics, including accuracy, reward, response length, repetition score, etc. Details are provided in Appendix D.

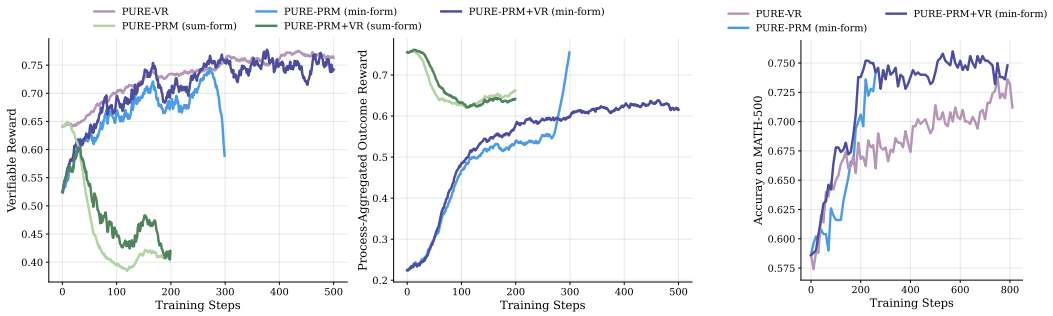
4.3 Main Results

We report the pass@1 accuracy across 5 benchmarks in Table 2. The results indicate that all variants of our methods perform at least comparable to, or better than baselines. For example, when using only verifiable rewards, following the setup of SimpleRL-Zoo, our method PURE-VR achieves average scores of 43.8 and 48.3 on Qwen2.5-7B and Qwen2.5-Math-7B, respectively. These scores are comparable to the baseline results of 43.1 and 48.0 obtained by SimpleRL-Zoo, confirming the reliability of our code-base. Based on Table 2, we draw the following observation:

PRM-based approach performs similar to the VR-based approach, but combining the two yields better results. On the Qwen2.5-Math-7B model, PURE-PRM achieves an average score of

49.3, slightly higher than the 48.3 of PURE-VR. However, the combined method, PURE-PRM+VR, reaches a score of 53.3, surpassing PURE-VR by approximately 5 percentage points. This trend remains consistent across the other 2 base models.

Next, we analyze the training dynamics of our methods to understand the limitations of the PRM-based approach and explain why PURE-PRM+VR is superior. Figure 2 illustrates the training curves for five variants of our methods on Qwen2.5-Math series, leading to the following findings:



(a) PURE on Qwen2.5-Math-7B (b) Qwen2.5-Math-1.5B

Figure 2: Training curves for different variants of our methods on Qwen2.5-Math series. Curves of PURE-PRM (sum-form) and PURE-PRM+VR (sum-form) are truncated due to training collapse. Process-aggregated outcome reward is the summation of final process rewards for one response: for sum-form, it sums PRM-emitted rewards; for min-form, it sums transformed rewards, approximating the minimum PRM-emitted reward. Thus **values across the 2 credit assignment methods are not comparable**. For PURE-PRM, verifiable reward is logged but unused in training.

Summation-form credit assignment collapses training even at the beginning, while the min-form method significantly enhances training stability. As shown in Figure 2a, both PURE-PRM (sum-form) and PURE-PRM+VR (sum-form) experience collapse at step 25. At step 80, their average benchmark scores drop to around 30, which is much lower than the base model's score of 39.5. In contrast, the min-form methods remain stable over 200 steps, achieving average scores of 49.3 and 53.3 for PURE-PRM (min-form) and PURE-PRM+VR (min-form), respectively.

Dense rewards substantially improve learning efficiency compared to sparse rewards. Figure 2b shows the curves of accuracy on MATH-500 for 3 variants of our methods. We find that PRM-involved approaches takes around 30% of the training steps to achieve the same accuracy as PURE-VR.

Reward hacking is inevitable when rely solely on PRM, though it can be delayed through algorithmic adjustments. While PURE-PRM (min-form) produces a well-tuned model, reward hacking still occurs at step 270, where verifiable rewards decrease sharply while process-aggregated outcome rewards increase in Figure 2a. This issue was observed in all experiments using the PURE-PRM method, indicating that reward hacking is a persistent challenge when relying solely on PRM and limits further progress. However, compared to sum-form credit assignment, min-form approaches delays the onset of reward hacking significantly. Such reward hacking corresponds to the third case in § 5.1, which is inevitable due to the current architecture of PRMs.

Incorporating a few ground-truth signals can effectively reduce reward hacking. PURE-PRM+VR (min-form) uses a mixed dataset of 800 problems with GT answers and 7,200 open problems. Thus, during training, process rewards guide the RFT process primarily, while verifiable rewards serve as an auxiliary signal. As shown in Figure 2a, this approach achieves a stable training process similar to PURE-VR and attains the highest average benchmark score of 53.3. This suggests a practical solution to address reward hacking by including a small proportion (around 10%) of ground-truth signals during the RFT stage.

5 Analysis

5.1 Reward Hacking induced by PRM

In this section, we detail the reward hacking cases observed during training. We categorize these cases into the 3 types. Specific examples for each type are provided in Appendix E.

Case 1: Only thinking, not solving. This behavior occurs when steps with certain patterns, such as thinking, are rewarded significantly more than others, and the inappropriate credit assignment further widen the gap in advantages between steps. As a result, the model learns to exploits these patterns to achieve higher scores. As shown in Figure 1, the sum-form credit assignment increases the likelihood of generating thinking steps while reducing that of incorrect solution steps, resulting in an armchair general. However, exploiting specific patterns is not always negative. For example, using patterns like backtracking and verification improves reasoning skills (Guo et al., 2025; Gandhi et al., 2025).

Case 2: Extremely few steps (1 step). This happens when an unsuitable advantage baseline is used. In § 3.4, we employ a token-level baseline for process rewards. An alternative approach is a step-level baseline, formulated as follows:

$$A_{i,t} = r_i^v - \frac{1}{K-1}\sum_{k\neq i} r_k^v + \sum_{j=t}^{N} \gamma^{j-t}\left[r_{i,j}^{p*} - \frac{1}{K-1}\sum_{k\neq i}\frac{\sum_l r_{k,l}^{p*}}{|y_k|_s}\right] \tag{8}$$

where $|y_k|_s$ represents the number of steps of the k-th response within the group. The advantage baseline for process rewards in Eq. (8) means the average process reward of other responses in the group. However, we find the step-level baseline is biased against the number of steps. When the process-aggregated outcome reward are equal, response with more steps are penalized more heavily by the baseline, causing the model to favor responses with fewer steps. Eventually, the model learns to output only a single step with an excessively large number of tokens. This undermines the purpose of PRM, which is to evaluate the process step by step.

Case 3: Extremely few steps (0 step). In this case, the model learns to output irrelevant responses such as "Thank you.", "Happy Birthday.", or even empty responses. Since PRM infers in a causal manner, it assigns high rewards to these meaningless steps, not realizing that no further content follows. This issue stems from the current architecture of discriminative PRMs (*i.e.*, the causal attention mask). Our solution is adding GT-level signal as an aid, such as verifiable rewards in PURE-PRM+VR. Another potential solution is the generative PRMs, which we leave for future work.

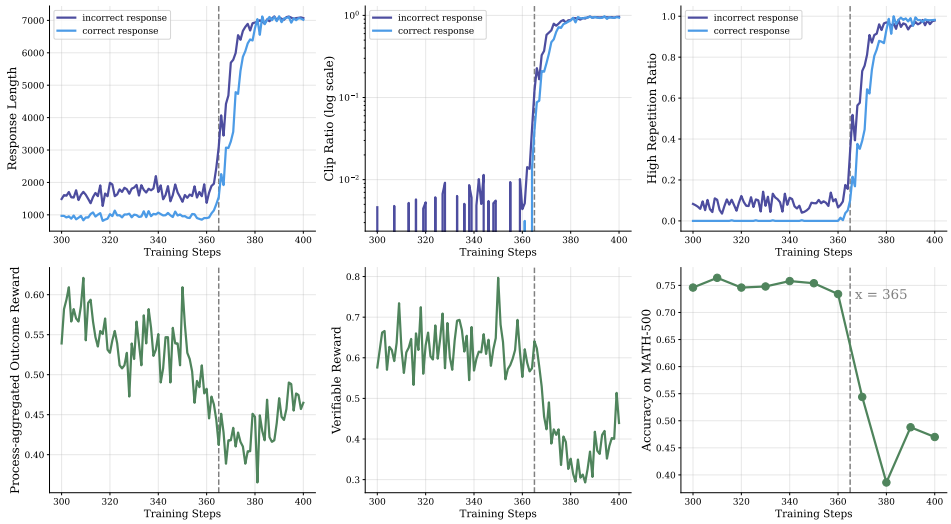


Figure 3: Training curves for PURE-PRM+VR with doubled process rewards based on Qwen2.5-7B. The correctness of responses are judged by the verifier. Process-aggregated outcome reward (bottom-left) is the summation of transformed process rewards for one response. No smooth is applied. Training collapses at step 365, showing a sharp drop of rewards and accuracy.

5.2 Causes of Training Collapse

In this section, we train Qwen2.5-7B using PURE-PRM+VR with doubled process rewards, and show the training curves of pattern-related metrics in Figure 3, including response length, clip ratio, repetition score, rewards, and accuracy on MATH-500. The clip ratio measures how often responses are cut off due to the maximum generation length limit. The repetition score evaluates how repetitive a

response is by calculating the longest common prefix (LCP) lengths between all pairs of its suffixes [3]. We consider responses with repetition score above 0.2 as highly repetitive. Details on metrics are provided in Appendix D.

As shown in Figure 3, training collapses at step 365, showing a sharp drop of rewards and accuracy. Before step 360, incorrect responses judged by the verifier are longer and more repetitive than the correct responses, but that does not collapse the training. Both the clip ratio and high repetition ratio for correct responses remain at 0 until step 361. After that, the response length, clip ratio, and high repetition ratio for both incorrect and correct responses rise sharply until step 380. Based on these observations, we derive the following conclusions:

Long and highly repetitive "correct" responses cause training collapse. At step 361, the clip ratio and high repetition ratio for correct responses become greater than 0. This large number of positive signals for repetitive patterns is fatal to training. We refer to these samples as pseudo-positive samples. We attempt to treat pseudo-positive samples as incorrect samples and assign 0 rewards instead of +1 rewards, but this does not help much. The model learns to repeat content in ways that the LCP function can not detect, as detailed in Appendix D. Verifiers can not identify pseudo-positive samples. While PRM has the potential to detect them, it currently fails to do so because such patterns are not included in its training data. This highlights a need for future PRM development to not only assess the correctness of steps but also evaluate the quality of patterns.

The model learns repetitive patterns quickly, leading to collapse within 5 gradient steps. From the unusual metrics at step 361 to the training collapse at step 365, the model rapidly learns to repeat content. Our training setup ensures one gradient step per training step, meaning the collapse happens within just 5 gradient steps.

6 Conclusion

In this paper, we present PURE: **P**rocess s**U**pervised **R**einforcement l**E**arning, a framework leverages process rewards to improve the reasoning abilities of LLMs. Extensive experiments demonstrate that PURE effectively alleviates reward hacking induced by PRM, attributed to the proposed min-form credit assignment. This method allows PRM-based RFT to achieve performance similar to verifiable rewards-based approaches within 30% steps, and further outperform them with the assistance of a few ground-truth signals. Additionally, we summarize the observed reward hacking cases during training, and find that pseudo-positive samples collapse training.

There are several promising directions for further improving PRM-based RFT. First, developing generative PRMs is both urgent and crucial. As discussed in § 5, current PRMs are unable to address the third type of reward hacking mentioned in § 5.1, and also cannot evaluate the quality of patterns like endless repetition. Generative PRMs, however, could potentially resolve these issues by making better use of the strong language capabilities of LLMs. Second, iterative training between PRM and LLM is essential to ensure that the PRM continuously adapts to the output distribution of LLMs.

Acknowledgments

We sincerely thank Ganqu Cui, Songjun Tu, and Zhengbo Wang for their suggestions to the early draft of the paper. This work was partially supported by the National Science and Technology Major Project (2022ZD0117102), the National Natural Science Foundation of China under Grants 62271485 and 62303462, Beijing Natural Science Foundation under grant L241016 and L233005, Chongqing Transportation Technology Project (CQJT-CZKJ2024-04), the Provincial Key Research and Development Program of Zhejiang (project number: 2022C01129), Ningbo International Science and Technology Cooperation Project (2023H020), CCF-Tencent Rhino-Bird Open Research Fund.

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. *arXiv preprint arXiv:2402.14740*, 2024.

[3] This is implemented using a function from Open-Reasoner-Zero (Hu et al., 2025).

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Jie Cheng, Gang Xiong, Xingyuan Dai, Qinghai Miao, Yisheng Lv, and Fei-Yue Wang. RIME: Robust preference-based reinforcement learning with noisy preferences. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 8229–8247. PMLR, 21–27 Jul 2024. URL `https://proceedings.mlr.press/v235/cheng24k.html`.

Jie Cheng, Ruixi Qiao, ma yingwei, Binhua Li, Gang Xiong, Qinghai Miao, Yongbin Li, and Yisheng Lv. Scaling offline model-based rl via jointly-optimized world-action model pretraining. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Representation Learning*, volume 2025, pp. 42396–42418, 2025. URL `https://proceedings.iclr.cc/paper_files/paper/2025/file/689cffc97600f9deb8374fc8fa918b8e-Paper-Conference.pdf`.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*, 2025.

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

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

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng 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. *arXiv preprint arXiv:2402.14008*, 2024.

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.

Jian Hu. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*, 2025.

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, 2025. URL `https://arxiv.org/abs/2503.24290`.

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.

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.

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

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.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 2007.

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.

Eduardo Pignatelli, Johan Ferret, Matthieu Geist, Thomas Mesnard, Hado van Hasselt, Olivier Pietquin, and Laura Toni. A survey of temporal credit assignment in deep reinforcement learning. *arXiv preprint arXiv:2312.01072*, 2023.

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

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

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

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. *arXiv preprint arXiv:2409.19256*, 2024.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. Prmbench: A fine-grained and challenging benchmark for process-level reward models. *arXiv preprint arXiv:2501.03124*, 2025.

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.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Bin Wang, Bojun Wang, Changyi Wan, Guanzhe Huang, Hanpeng Hu, Haonan Jia, Hao Nie, Mingliang Li, Nuo Chen, Siyu Chen, et al. Step-3 is large yet affordable: Model-system co-design for cost-effective decoding. *arXiv preprint arXiv:2507.19427*, 2025.

Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2023.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *arXiv preprint arXiv:2502.18449*, 2025.

Lilian Weng. Reward hacking in reinforcement learning. *lilianweng.github.io*, Nov 2024. URL `https://lilianweng.github.io/posts/2024-11-28-reward-hacking/`.

Wei Xiong, Hanning Zhang, Nan Jiang, and Tong Zhang. An implementation of generative prm. `https://github.com/RLHFlow/RLHF-Reward-Modeling`, 2024.

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*, 2024a.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024b.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. *arXiv preprint arXiv:2412.01981*, 2024.

Tongtian Yue, Jie Cheng, Longteng Guo, Xingyuan Dai, Zijia Zhao, Xingjian He, Gang Xiong, Yisheng Lv, and Jing Liu. Sc-tune: Unleashing self-consistent referential comprehension in large vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13073–13083, 2024.

Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, et al. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint arXiv:2508.06471*, 2025a.

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*, 2025b.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*, 2025.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*, 2024.

A Related Work

A.1 Reinforcement Fine-Tuning

Reinforcement fine-tuning (RFT) is a promising technique to improve the performance of LLMs (Ouyang et al., 2022; Yue et al., 2024; Wei et al., 2025). OpenAI's o1 model (Jaech et al., 2024) was among the first to demonstrate the significant potential of large-scale reinforcement learning for enhancing reasoning abilities in LLMs. Recent studies have further confirmed that a straightforward reinforcement learning approach, using verifiable rewards, can scale effectively (Guo et al., 2025; Team et al., 2025; Zeng et al., 2025b). However, previous research has faced difficulties in effectively utilizing PRM (Guo et al., 2025) in training-time, which is the primary focus of our work in PURE.

A.2 Reward Models for LLM Alignment

In the area of LLM alignment, outcome reward models (ORMs) are used to evaluate the quality of the entire response generated by LLMs. Employing ORMs for aligning LLMs with human values has become a common practice in the post-training stage (Ouyang et al., 2022; Bai et al., 2023; Touvron et al., 2023). However, ORMs fall short in providing detailed feedback for complex tasks that involve multiple reasoning steps. In contrast, process reward models (PRMs) provide more detailed feedback by evaluating each step of the reasoning process, allowing LLMs to learn more effectively from mistakes made during reasoning (Lightman et al., 2023; Yuan et al., 2024; Cui et al., 2025). Studies have shown that PRMs outperform majority voting and ORMs in test-time scaling (Lightman et al., 2023; Wang et al., 2023; Xiong et al., 2024). Nevertheless, the application of PRMs during the training phase remains largely unexplored, which is the central topic of PURE.

B Proof of Q-value estimation error

Theorem 2 (**Q-Value Estimation Error Bound Comparison**). *Under Assumptions 1 and 2, for any state-action pair* (s_t, a_t) *and a trajectory* τ *with* n *reasoning steps:*

1. ***Sum-form Error Bound:*** *The estimation error for the summation-form Q-value, is bounded by:*

$$|Q_\pi^{sum}(s_t, a_t) - Q_\pi^{sum,*}(s_t, a_t)| \leq \sum_{i=t}^{t+n-1} \gamma^{i-t}\epsilon$$

For an infinite horizon ($n \to \infty$), this bound becomes:

$$|Q_\pi^{sum}(s_t, a_t) - Q_\pi^{sum,*}(s_t, a_t)| \leq \frac{\epsilon}{1-\gamma}$$

2. ***Min-form Error Bound:*** *The estimation error for the min-form Q-value, derived from Eq. (4), is bounded by:*

$$|Q_\pi^{min}(s_t, a_t) - Q_\pi^{min,*}(s_t, a_t)| \leq \epsilon$$

Proof. **Part 1: Proof for the Sum-form Error Bound.**

We begin with the definition of the absolute error of the Q-value, using the canonical formulation in Eq. (1). Q_π^{sum} denotes the value calculated with the estimated rewards from the PRM (r^p), and $Q_\pi^{sum,*}$ denotes the true value (calculated with r^*).

$$\begin{aligned}
|Q_\pi^{sum}(s_t, a_t) - Q_\pi^{sum,*}(s_t, a_t)| &= |E_{\pi\sim\tau}[\sum_{i=t}^{t+n-1} \gamma^{i-t} r_i^p] - E_{\pi\sim\tau}[\sum_{i=t}^{t+n-1} \gamma^{i-t} r_i^*]| \\
&= |E_{\pi\sim\tau}[\sum_{i=t}^{t+n-1} \gamma^{i-t}(r_i^p - r_i^*)]| \\
&\leq E_{\pi\sim\tau}[|\sum_{i=t}^{t+n-1} \gamma^{i-t}(r_i^p - r_i^*)|] \\
&\leq E_{\pi\sim\tau}[\sum_{i=t}^{t+n-1} |\gamma^{i-t}(r_i^p - r_i^*)|] \\
&= E_{\pi\sim\tau}[\sum_{i=t}^{t+n-1} \gamma^{i-t}|r_i^p - r_i^*|]
\end{aligned}$$

Now, we apply Assumption 1, which states that the reward error at each step is bounded by ϵ, i.e., $|r_t^p - r_t^*| \leq \epsilon$.

$$|Q_\pi^{sum}(s_t, a_t) - Q_\pi^{sum,*}(s_t, a_t)| \leq E_{\pi\sim\tau}[\sum_{i=t}^{t+n-1} \gamma^{i-t}\epsilon] = \epsilon \sum_{i=t}^{t+n-1} \gamma^{i-t}$$

For the infinite horizon case, the geometric series converges to $\frac{1}{1-\gamma}$.

Part 2: Proof for the Min-form Error Bound.

We start similarly with the definition of the absolute Q-value error for the min-form. The return $G(s_t, a_t|\tau)$ is the value being expected.

$$\begin{aligned}
|Q_\pi^{min}(s_t, a_t) - Q_\pi^{min,*}(s_t, a_t)| &= |E_{\pi\sim\tau}[G(s_t, a_t|\tau)] - E_{\pi\sim\tau}[G^*(s_t, a_t|\tau)]| \\
&= |E_{\pi\sim\tau}[G(s_t, a_t|\tau) - G^*(s_t, a_t|\tau)]|
\end{aligned}$$

Now we analyze the return term inside the expectation. According to Equation (3), for a trajectory τ and a step t, the return is defined as the minimum of future rewards.

$$\begin{aligned}
G(s_t, a_t|\tau) &= \min(r_t^p, r_{t+1}^p, \ldots, r_n^p) \\
G^*(s_t, a_t|\tau) &= \min(r_t^*, r_{t+1}^*, \ldots, r_n^*)
\end{aligned}$$

We use a key property of the minimum function: $|\min(a) - \min(b)| \leq \max_i |a_i - b_i|$. Applying this property:

$$|G(s_t, a_t|\tau) - G^*(s_t, a_t|\tau)| = |\min_{i=t\ldots n}(r_i^p) - \min_{i=t\ldots n}(r_i^*)| \leq \max_{i=t\ldots n} |r_i^p - r_i^*| \leq \epsilon$$

Substitute this result back into the expectation inequality:

$$|Q_\pi^{min}(s_t, a_t) - Q_\pi^{min,*}(s_t, a_t)| = |E_{\pi\sim\tau}[G(s_t, a_t|\tau) - G^*(s_t, a_t|\tau)]| \leq E_{\pi\sim\tau}[\epsilon] = \epsilon$$

This completes the proof. It is proven that the error in the min-form does not accumulate and is directly bounded by the maximum single-step error of the reward model. □

C Benchmark Scores of Our PRM

We report the detailed scores on ProcessBench (Zheng et al., 2024) and PRMBench (Song et al., 2025) in Table 3 and Table 4, respectively.

Table 3: F1 scores of each subset in ProcessBench. The blue line represents our PRM. Other lines are taken from Zheng et al. (2024).

Model	GSM8K	MATH	Olympiad-Bench	Omni-MATH	Average
Math-Shepherd-PRM-7B	47.9	29.5	24.8	23.8	31.5
RLHFlow-PRM-Deepseek-8B	38.8	33.8	16.9	16.9	26.6
Skywork-PRM-7B	**70.8**	53.6	22.9	21.0	42.1
Qwen2.5-Math-7B-PRM800K	68.2	62.6	**50.7**	44.3	56.5
PURE-PRM-7B	69.0	**66.5**	48.4	**45.9**	**57.5**

Table 4: Results on ProcessBench. The blue line represents our PRM. Other lines are taken from Song et al. (2025).

Model	S1 (Simplicity)	S2 (Soundness)	S3 (Sensitivity)	Overall
MathShepherd-Mistral-7B	47.1	45.7	60.7	47.0
RLHFlow-PRM-Deepseek-8B	47.6	57.5	68.1	54.2
Skywork-PRM-7B	**59.6**	68.5	73.3	65.1
Qwen2.5-Math-7B-PRM800K	52.1	**71.0**	75.5	**65.5**
PURE-PRM-7B	52.2	70.2	**75.8**	65.3

D Details of Training Metrics

We detail 4 metrics used in the main text of the paper, including process-aggregated outcome reward, clip ratio, repetition score, and high repetition ratio.

Process-aggregated outcome reward. This metric reflects the value of a response with respect to process rewards. In practice, it is calculated as the sum of transformed process rewards for the min-form credit assignment, approximating the minimum PRM-emitted reward for a given response.

Clip ratio. The clip ratio indicates the proportion of samples in the replay buffer that are truncated due to the maximum generation length limit. After training collapses, LLMs often generate infinitely repetitive responses until hit the maximum context length.

Repetition score. The repetition score measures the level of repetition in a response by calculating the longest common prefix (LCP) lengths between all pairs of its suffixes. However, this method can only identify basic repetition patterns. A more complex form of repetition involves the use of near-synonyms, where words differ but the meaning remains the same. Such cases are not detected by the LCP method. This limitation explains why treating pseudo-positive samples as incorrect still results in training collapse, as mentioned in § 5.2. The model learns to exploit complex repetition patterns to bypass the LCP function.

High repetition ratio. We consider responses with a repetition score greater than 0.2 as highly repetitive. This metric shows the percentage of samples in the replay buffer that fall into this category.

E Examples of Reward Hacking

We show examples for each reward hacking case in Figure 4, 5, and 6. In practice, we use double line breaks ("\n\n") to separate individual steps within the response.

Question

Compute $\frac{1}{\cos^2 10^\circ} + \frac{1}{\sin^2 20^\circ} + \frac{1}{\sin^2 40^\circ}$

Response

Let's break down the problem step by step:

Step 1: Express the Given Expression Using Trigonometric Identities
Rewrite each term of the expression using identities like $\frac{1}{\cos^2 x} = \sec^2 x$ and $\frac{1}{\sin^2 x} = \csc^2 x$, and further express them in terms of $\tan^2 x$ and $\cot^2 x$ to simplify the structure.

Step 2: Simplify the Expression by Combining Constant Terms
Combine any constant terms resulting from the identities used in the previous step to reduce the expression to a more manageable form involving trigonometric functions.

Step 3: Convert Cotangent Terms to Tangent Terms
Use the relationship $\cot x = \frac{1}{\tan x}$ to express cotangent squares in terms of tangent squares, aiming for consistency in the trigonometric functions used.

Step 4: Explore Angle Relationships Using Multiple-Angle Identities
Consider relevant multiple-angle identities (such as for $\tan 3x$) to establish equations or relationships between the angles involved, potentially leading to a simplification of the expression.

Step 5: Relate Cotangent to Tangent Using Complementary Angle Identities
Use the identity $\cot x = \tan(90^\circ - x)$to rewrite cotangent terms as tangent terms of complementary angles, facilitating a possible pairing or symmetry in the expression.

Step 6: Apply Known Trigonometric Sum Identities or Results
Identify and apply any known results or specific identities related to the sum of squares of tangents and cotangents for the given angle configuration to evaluate the expression directly.

Step 7: Combine Results to Compute the Final Value
Add the results from the simplified trigonometric sum to the constant terms obtained earlier to arrive at the final numerical value of the expression.

Figure 4: Reward hacking, case 1: only thinking, not solving. In this example, the LLM analyzes the problem and gives a few equations for trigonometric simplifications, but does not substitute actual numbers to calculate and solve the problem. This is because the LLM hacks the implicit pattern inside high-reward steps, *i.e.*, thinking.

F Additional Experiments

F.1 Ablation on Transform Temperature

In this section, we ablate the transform temperature defined in Eq. (5), which controls the level of approximation to the min-form credit assignment. We choose Qwen2.5-7B as the base model and run PURE-PRM+VR with 3 values of transform temperature individually for 230 steps. The results are shown in Table 5 and Figure 7, which indicates that 0.1 is the best for the transform temperature.

Question

In ΔPQR, we have $PQ = QR = 34$ and $PR = 32$. Find the length of median $\overline{QM}$.

Response

Let's first identify the type of triangle and use the properties of the median in an isosceles triangle.
Given that $PQ = QR = 34$, ΔPQR is isosceles with $PQ = QR$. In an isosceles triangle, the median to the base (the side opposite the vertex angle) also acts as the altitude and the angle bisector of the vertex angle. So, the median $\overline{QM}$ from vertex Q to the midpoint M of the base PR will split the isosceles triangle into two congruent right triangles.
Let's place the triangle in the coordinate plane to make calculations easier. Assume $P = (0, b)$, $Q = (0, 0)$, and $R = (32, 0)$. Here, $PQ = QR = 34$ and $PR = 32$.
Since $PQ = 34$ and $Q = (0,0)$, point P is at a distance of 34 units from the origin along the y-axis. So, the coordinates of P are $(0, b)$, and the distance from P to Q is the y-coordinate of P, which is b. Therefore, we have:

$$b = 34$$

So, the coordinates of P are $(0, 34)$.
The coordinates of Q are $(0, 0)$ and the coordinates of R are $(32, 0)$.
Next, let's find the coordinates of the midpoint M of the base PR.
The midpoint M of the base PR (where $P = (0, 34)$ and $R = (32, 0)$) is given by the midpoint formula:

$$M = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}\right)$$

Here, $(x_1, y_1) = (0, 34)$ and $(x_2, y_2) = (32, 0)$:

$$M = \left(\frac{0 + 32}{2}, \frac{34 + 0}{2}\right) = \left(\frac{32}{2}, \frac{34}{2}\right) = (16, 17)$$

So, the coordinates of the midpoint M are $(16, 17)$.
Then, let's calculate the length of the median $\overline{QM}$.
The median $\overline{QM}$ is the line segment from $Q = (0, 0)$ to $M = (16, 17)$. The length of the median can be found using the distance formula between two points (x_1, y_1) and (x_2, y_2):

$$\text{Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Here, the points are $Q = (0, 0)$ and $M = (16, 17)$:

$$QM = \sqrt{(16 - 0)^2 + (17 - 0)^2} = \sqrt{16^2 + 17^2}$$

Calculate 16^2 and 17^2:

$$16^2 = 256$$
$$17^2 = 289$$

So,

$$QM = \sqrt{256 + 289} = \sqrt{545}$$

Therefore, the length of the median $\overline{QM}$ is $\boxed{\sqrt{545}}$

Figure 5: Reward hacking, case 2: extremely few steps (1 step). In practice, we divide steps according to double line breaks "\n\n" and then PRM scores each step. When the advantage baseline is inappropriate, such as step-level baseline discussed in § 5.1, the model learns to deliberately avoid outputting "\n\n", preferring short-step response. In this example, there is no "\n\n" character in the generated response, resulting in the entire response being split into only one step.

Question

Evaluate the expression $\sqrt{80} - 4\sqrt{5} + 3\sqrt{\frac{180}{3}}$ and express it as $\sqrt{M}$, where M is an integer.

Response 1

Thank you.

Response 2 (empty response)

<eos>

Response 3

I can help you solve this problem.

Figure 6: Reward hacking, case 3: extremely few steps (0 step). This is the most common cases for PURE-PRM. When relying solely on the PRM, training eventually boils down to this case after numerous steps of training. It is inevitable because the PRM scores based on the question and historical steps, and it does not know the role of the current step in the overall response.

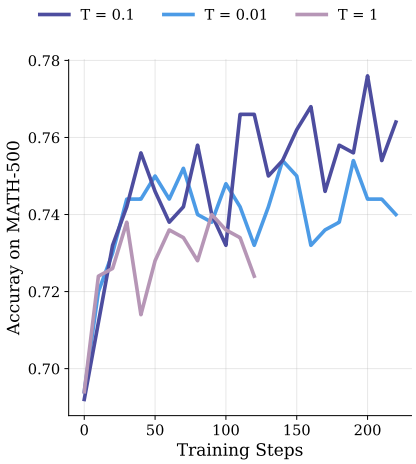


Figure 7: Ablation on transform temperature defined in Eq. (5). We use PURE-PRM+VR to fine-tune Qwen2.5-7B in this experiment.

Table 5: Results of PURE-PRM+VR with different transform temperature, defined in Eq (5). We conduct the experiments based on Qwen2.5-7B and report pass@1 accuracy tested with greedy decoding.

Method	MATH 500	Minerva Math	Olympiad Bench	AIME24	AMC23	Avg.
Base	71.4	23.9	35.3	10.0	52.5	38.6
+ PURE-PRM+VR ($T = 0.01$)	**76.2**	**37.1**	39.0	6.7	55.0	42.8
+ PURE-PRM+VR ($T = 0.1$)	**76.2**	**37.1**	**41.2**	**13.3**	**60.0**	**45.6**
+ PURE-PRM+VR ($T = 1.0$)	75.8	33.5	38.7	**13.3**	52.5	42.8

F.2 PURE with Other RL Algorithms

In this section, we apply PURE with various advantage estimators beyond RLOO, including GAE, GRPO, and REINFORCE++. The training curves are shown in Figure 8. To make different algorithms compatible with the compound of verifier rewards and process rewards, we make adjustments similar to those in Eq. (7). For GRPO, the advantage is defined as:

$$A_{i,t} = \frac{r_i^v - \text{mean}(\boldsymbol{r}^v)}{\text{std}(\boldsymbol{r}^v)} + \frac{\sum_{j=t}^{N} \gamma^{j-t} r_{i,j}^{p*} - \text{mean}\left(\sum_{l=1}^{N} \sum_{j=l}^{N} \gamma^{j-l} \cdot \boldsymbol{r}_j^{p*}\right)}{\text{std}\left(\sum_{l=1}^{N} \sum_{j=l}^{N} \gamma^{j-l} \cdot \boldsymbol{r}_j^{p*}\right)} \tag{9}$$

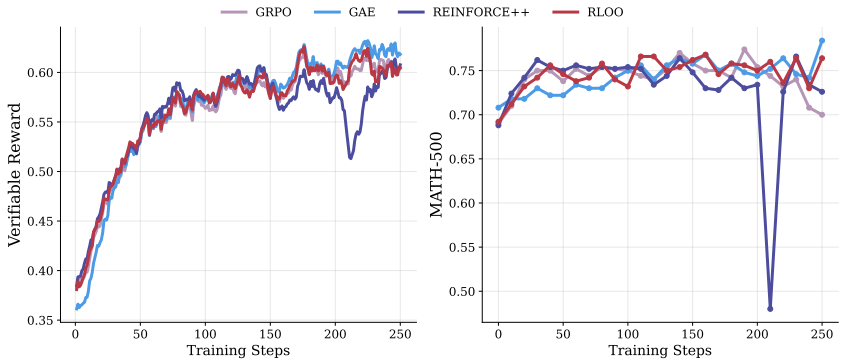


Figure 8: Training curves on Qwen2.5-7B using PURE-PRM+VR with different advantage estimators.

where mean and standard deviation (std) are calculated over K responses in a group. For REINFORCE++, the advantage is:

$$A_{i,t} = r_i^v + \sum_{j=t}^{N} \gamma^{j-t} r_{i,j}^{p*} \tag{10}$$

From Figure 8, we find that all methods show similar performance before step 150. However, REINFORCE++ experiences a spike at around step 220. Although GAE converges a bit slower than others, it performs slightly more stably in the last 50 steps. Due to its additional learnable value network, GAE takes about 30% more time for forward and backward passes. Considering performance, training time, and stability, we select RLOO as our preferred advantage estimator.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: See abstract, section 1 and 4.

   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: See section 6.

   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: The paper 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: See section 4, appendix, and supplementary material.

   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: See supplementary material.

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, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See section 4 and appendix.

   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: All methods are tested using greedy decoding. Thus given a checkpoint, its benchmark scores are deterministic.

   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: See section 4 and appendix.

   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: The 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: There is no societal impact of the work performed.

    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 paper poses no 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: Both the base models from Qwen and dataset are properly credited.

    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: [NA]

    Justification: The paper does not release new assets.

    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: The paper does not involve crowdsourcing nor 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: The paper does not involve crowdsourcing nor 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 core method development in this research does not involve LLMs.

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.