# Efficient Process Reward Modeling via Contrastive Mutual Information

**Nakyung Lee**[1], **Sangwoo Hong**[2]*, **Jungwoo Lee**[1]*,

[1]Department of Electrical and Computer Engineering, Seoul National University,
[2]Department of Computer Science and Engineering, Konkuk University,

**Correspondence:** swhong06@konkuk.ac.kr, junglee@snu.ac.kr

## Abstract

Recent research has devoted considerable effort to verifying the intermediate reasoning steps of chain-of-thought (CoT) trajectories using process reward models (PRMs) and other verifier models. However, training a PRM typically requires human annotators to assign reward scores to each reasoning step, which is both costly and time-consuming. Existing automated approaches, such as Monte Carlo (MC) estimation, also demand substantial computational resources due to repeated LLM rollouts. To overcome these limitations, we propose **contrastive pointwise mutual information (CPMI)**, a novel automatic reward labeling method that leverages the model's internal probability to infer step-level supervision while significantly reducing the computational burden of annotating dataset. CPMI quantifies how much a reasoning step increases the **mutual information** between the step and the correct target answer relative to hard-negative alternatives. This **contrastive** signal serves as a proxy for the step's contribution to the final solution and yields a reliable reward. The experimental results show that CPMI-based labeling reduces dataset construction time by **84%** and token generation by **98%** compared to MC estimation, while achieving higher accuracy on process-level evaluations and mathematical reasoning benchmarks. The code is available at https://github.com/nakyungLee20/CPMI.

## 1 Introduction

Recent studies have demonstrated that large language models (LLMs) exhibit remarkable reasoning abilities, particularly when paired with inference-time techniques such as chain-of-thought prompting (Wei et al., 2022) and self-consistency (Wang et al., 2022). These advances have motivated growing interest in PRMs, which assess the correctness of intermediate reasoning steps, rather than relying solely on outcome reward models (ORMs) that evaluate final answers. However, constructing high-quality process-level supervision data sets for PRM training remains a challenge. Annotating fine-grained step rewards typically requires human experts or high-performing LLMs such as GPT-4, which is both labor-intensive and costly. To address these issues, prior works have explored automated labeling via MC estimation (Wang et al., 2024; Zhang et al., 2025), where multiple reasoning trajectories are sampled and rewards are assigned based on the proportion of trajectories that ultimately yield the correct answer. While this approach eliminates the need for human annotation, it still entails substantial computational overhead, as numerous rollouts are required to obtain low-variance and reliable reward signals. This challenge has motivated the development of frameworks that leverage a model's inherent knowledge to reduce such overhead (Jin et al., 2025).

In this paper, we propose a novel approach for efficient and automatic step-level rewards labeling by leveraging a model's internal certainty through likelihood estimates. We hypothesize that modern pretrained LLMs already encode substantial mathematical knowledge. Under this assumption, we posit that quantifying *contrastive* changes in the predicted probability of correct versus incorrect answers at each reasoning step can provide a meaningful reward signal. This formulation eliminates the need for laborious human annotation and the costly rollouts required by MC estimation. Using this reward design, we curate **CPMI-80k**, a step-level supervision dataset for training verifier models derived from Math-Shepherd (Wang et al., 2024). We extract an $80k$ subset of question–solution pairs, automatically annotate scalar rewards for each step along the solution paths.

Empirically, we show that using CPMI-based labeling can assign reliable reward signals that effectively distinguish correct from incorrect reasoning

*Corresponding authors: Sangwoo Hong; Jungwoo Lee



*Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 37603–37617

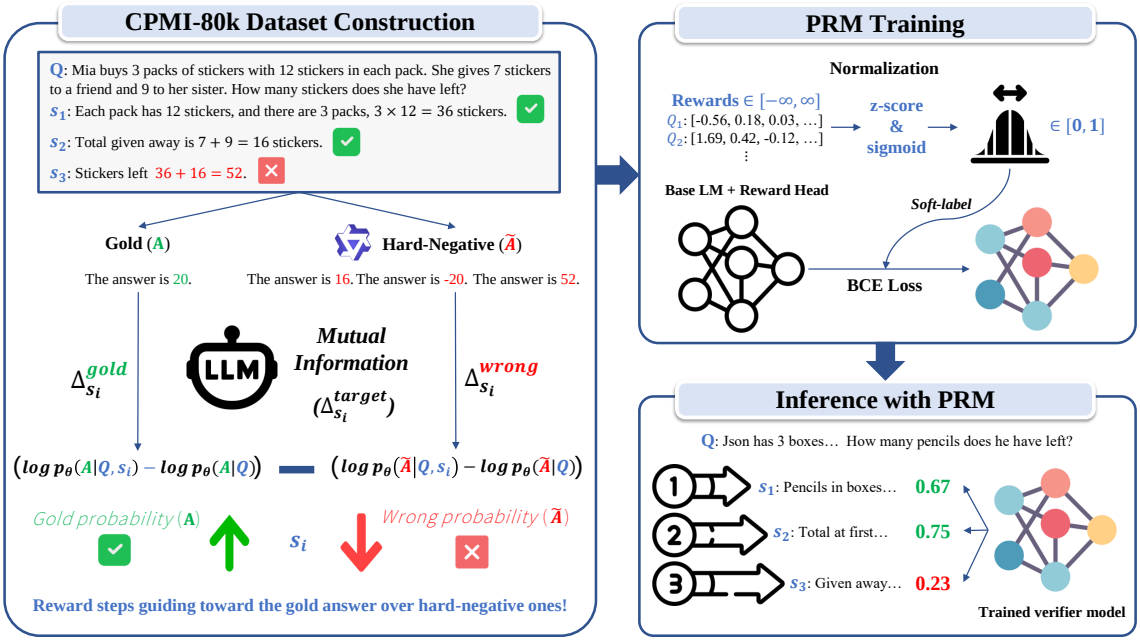


Figure 1: Main framework of our reward modeling and PRM training. We sample both gold and hard-negative answer and compute the logit differences between the with-step and without-step through a simple forward pass for each answer. The CPMI reward for each step is defined as the difference between these logit gaps, quantifying how much the step shifts the model's belief toward the gold answer while diverging from the wrong answer. After normalizing the CPMI-based rewards and using them as soft labels, we train the PRM with a binary cross-entropy objective. The trained PRM is subsequently used during inference to evaluate or rerank reasoning trajectories.

steps. Moreover, it encourages the model to internalize coherent reasoning trajectories during inference, leading to improved performance on mathematical reasoning benchmarks. Compared to MC, our method further reduces the time and computational cost of process-level reward generation by more than 80%, demonstrating its efficiency and practical scalability.

### Contributions

- We propose **contrastive pointwise mutual information (CPMI)**, a new automatic step-level reward labeling approach that removes the need for human annotation and substantially reduces the supervision cost without requiring multiple rollouts.
- We construct **CPMI-80k**, an automatically labeled step-level reward dataset using our CPMI framework, which provides informative scalar signals for process-level reasoning.
- Through extensive experiments, we show that PRMs trained with CPMI-based labels outperform MC-based approaches on process-level evaluation benchmarks, and further improve mathematical reasoning performance when combined with Best-of-$N$ inference.

## 2 Related works

We review prior work along two aspects, (1) reward design for PRMs, and (2) applying trained PRMs to improve mathematical reasoning at inference time.

**Reward design for training PRMs** A prevalent strategy for constructing step-level rewards is MC estimation, which samples multiple trajectories and attributes reward by the fraction that ultimately reaches a correct answer (Wang et al., 2024). Building on this foundation, several studies have proposed improved labeling or filtering mechanisms. For instance, Jeong et al. (2025) proposes evaluation frameworks to assess the accuracy of individual reasoning steps, and Setlur et al. (2024) reformulates step rewards as the difference between MC estimates before and after a step, analogous to the advantage function in reinforcement learning. Meanwhile, Sun et al. (2025) and Xiong et al. (2025) refine the data construction process through efficient annotation and stepwise verification. To further improve label quality, Zhang et al. (2025) adopts consensus filtering using high-performing LLMs, while Chen et al. (2024) introduces a Monte Carlo Tree Search algorithm to annotate step-level rewards without manual supervision. However, all

of the above methods still depend heavily on repeated MC rollouts, leading to substantial generation time and computational cost. In contrast, our method requires no MC supervision or uses it only once for initialization and instead relies on the model's internal log-probabilities, which can be obtained efficiently with a single forward pass.

**PRMs for Mathematical Reasoning** Recently, extensive works have focused on improving mathematical reasoning via inference-time strategies such as CoT-style prompting and program-aided computation (Chen et al., 2022; Gao et al., 2023; Zhou et al., 2023). When integrating PRMs at inference, one can re-rank or weight candidate traces (e.g., PRM-weighted majority voting or search) (Zhang et al., 2025; Sun et al., 2025) or inject PRM signals as dense rewards for RL-style training (Setlur et al., 2024; Cheng et al., 2025). Our work falls in the inference-time control regime. We score candidate trajectories with a trained PRM and select the highest average PRM-weighted trajectory as the final solution.

## 3 Preliminary

**Notations.** We consider a policy model $\pi_\theta$ as a pretrained LLM that generates step-by-step solution trajectories for a given question $q$. Each reasoning step is denoted by $s_i$, representing the $i$-th step in the full solution sequence. The concatenation of all steps up to step $i$ forms a partial solution, denoted as $s_{\le i}$. We denote the correct answer set as $A$, and an incorrect answer as $\tilde{A}$.

**Monte Carlo Estimation.** Given a partial solution $s_{\le i}$, Wang et al. (2024) estimate the step reward using MC rollouts by computing the probability of eventually reaching the correct answer $A$, as defined in Eq. (1).

$$r_{\text{MC}}^{(i)} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{1}\Big\{\text{Ans}\big(\tau^{(t)} \mid s_{\le i}\big) = A\Big\} \qquad (1)$$

$T$ denotes the number of rollout trajectories. Each continuation trajectory $\tau^{(t)} \sim \pi(\cdot \mid s_{\le i})$ is sampled from the policy conditioned on the prefix $s_{\le i}$, $\text{Ans}(\cdot)$ maps a completed trajectory to its final answer, and $\mathbf{1}\{\cdot\}$ is the indicator function.

**Process Advantage Verifiers (PAV).** Setlur et al. (2024) introduces an *effective reward* that measures the incremental progress of reasoning between consecutive steps. At each step $i$, PAV combines the MC reward $r_{\text{MC}}^{(i,\theta)}$, obtained from the current policy model $\pi_\theta$, with a *process advantage* term that evaluates the improvement made by a reference policy $\pi_\mu$ relative to the previous step $(i-1)$.

$$r_{\text{PAV}}^{(i)} = r_{\text{MC}}^{(i,\theta)} + \alpha\big(r_{\text{MC}}^{(i,\mu)} - r_{\text{MC}}^{(i-1,\mu)}\big) \qquad (2)$$

This formulation encourages the model to favor reasoning trajectories that exhibit consistent stepwise improvement throughout the intermediate reasoning process.

Using these MC and PAV formulations as Monte Carlo–based baseline methods, we compare them against our proposed reward design.

## 4 CPMI: Contrastive Pointwise Mutual Information-based Reward

The approach of quantifying a step's contribution through **pointwise mutual information (PMI)** is motivated by a temporal-difference (TD-$\lambda$) perspective (Sutton, 1988) in reinforcement learning. Unlike standard mutual information which averages over all possible outcomes, PMI utilizes log-probabilities to measure the information gain between specific instances, in our case, how a particular step $s_i$ shifts the likelihood of a specific target $A$ over the prior. This provides a useful analogy for understanding the trade-off between rollout-based and bootstrap-based estimation. Viewed through this TD($\lambda$) lens, the MC reward corresponds to the $\lambda \to 1$ regime, where long-horizon rollouts serve as full-return targets, whereas the proposed CPMI reward operates in the $\lambda \to 0$ regime, performing a one-step bootstrap that captures the model's local belief shift. Thus, to ensure efficiency, we aim to train the verifier relying on bootstrapping (TD-0).

### 4.1 CPMI Reward

We introduce a **contrastive** property that explicitly examines: *"How much does a step $s_i$ shift the model's belief toward the gold answer space and away from the negative answer space?"*. Intuitively, an effective step should simultaneously increase the likelihood of producing the correct answer while decreasing likelihood of incorrect ones. By combining temporal bootstrapping and contrastive reasoning perspectives, our formulation enables the reward model to capture fine-grained, discriminative signals that reward genuine reasoning progress and penalize misleading intermediate steps.

Building on the idea above, we define the CPMI step reward for a given prompt and step $i$, where $M$

denotes the number of incorrect answer alternatives, as shown in Eq. (3).

$$r^{i}_{\text{CPMI}} = \underbrace{[\log p_\theta(A \mid q, s_i) - \log p_\theta(A \mid q)]}_{\textbf{likelihood for gold answer}} - \frac{1}{M}\sum_{m=1}^{M} \underbrace{[\log p_\theta(\tilde{A} \mid q, s_i) - \log p_\theta(\tilde{A} \mid q)]}_{\textbf{shift toward incorrect answer}} \tag{3}$$

The first term quantifies how much the step $s_i$ increases the model's likelihood of producing the gold answer, while the second term measures its relative shift toward incorrect alternatives. To reduce prompt-specific variance, we further average the reward across multiple prompt templates $k \in K$ as shown in Eq. (4).

$$\hat{r^{i}}_{\text{CPMI}} = \frac{1}{K}\sum_{k=1}^{K} r^{i,k}_{\text{CPMI}} \tag{4}$$

This formulation provides an efficient, single-forward-pass estimation of step-level rewards, avoiding the costly MC rollouts.

### 4.2 Theoretical Derivation

For each prompt template $k$, we consider the answer distributions $P_k(\cdot) = p_\theta(\cdot \mid q_k, s_i)$ and $Q_k(\cdot) = p_\theta(\cdot \mid q_k)$, where $q_k$ denotes the instantiated template for the given problem. Our key observation is that the CPMI reward can be interpreted as an approximation to the *Jeffreys divergence* between these two conditional distributions, which can be expressed as

$$J(P_k, Q_k) = \mathrm{KL}(P_k \| Q_k) + \mathrm{KL}(Q_k \| P_k). \tag{5}$$

Unlike standard KL divergence, this symmetric measure rigorously penalizes discrepancies in both directions. Consequently, it yields a highly sensitive reward signal that intensifies specifically when the distributions occupy distinct regions of the answer space, thereby encouraging the model to clearly distinguish between correct and incorrect reasoning paths. We provide empirical evidence for this behavior in Section 6.1, where we demonstrate that the contrastive signal achieves superior discriminative power compared to non-contrastive counterparts.

In our formulation, positive samples are drawn from $A \sim P_k(A) = p_\theta(A \mid q_k, s_i)$ and negative samples from $\tilde{A} \sim Q_k(\tilde{A}) = p_\theta(\tilde{A} \mid q_k)$. Since mathematical reasoning tasks in GSM8K and MATH typically feature a unique ground-truth answer, the empirical target distribution $T$ can be modeled as a near-deterministic delta distribution. This property justifies the approximation $T \approx P_k$, so that expectations with respect to $T$ can be estimated using the single sample. Moreover, our controlled sampling setup ensures that the samples from $P_k$ and $Q_k$ satisfy the IID assumption with respect to their underlying model distributions. All samples are generated independently via ancestral decoding, and both $P_k$ and $Q_k$ are computed using the same model and tokenizer under identical temperature and vocabulary settings. This alignment guarantees that the two conditional distributions are normalized over the same support and assign probability mass to a shared answer space, enabling a valid contrastive comparison.

Under this assumption, the CPMI reward in Eq. (3) can be rewritten as

$$\begin{aligned} r^{i,k}_{\text{CPMI}} &\approx \mathbb{E}_{A\sim P_k}\big[\log \frac{P_k}{Q_k}\big] - \mathbb{E}_{\tilde{A}\sim Q_k}\big[\log \frac{P_k}{Q_k}\big] \\ &= \mathrm{KL}(P_k \| Q_k) + \mathrm{KL}(Q_k \| P_k) = J(P_k, Q_k), \end{aligned} \tag{6}$$

showing that CPMI provides a sample-based approximation to the Jeffreys divergence.

From a reward perspective, this derivation implies that CPMI explicitly favors steps that induce a large, asymmetric shift between the pre-step and post-step answer distributions, rather than merely increasing the likelihood of the correct answer in isolation. Specifically, the step that shift probability mass toward the correct answer and suppress hard-negative answers yield higher CPMI scores. In contrast, uninformative or redundant steps that result in negligible changes to $P_k$ and $Q_k$ are penalized.

### 4.3 CPMI-80k Dataset Construction

We now construct the **CPMI-80k** dataset by automatically labeling step-level rewards on reasoning trajectories from the MATH-SHEPHERD dataset.[1] The original corpus contains model-generated reasoning trajectories, where each problem is accompanied by a sequence of step-by-step solutions and a binary correctness label. Since MATH-SHEPHERD lacks explicit gold answers, we align each problem with reference solutions from GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021b) to recover

[1] Available at zhuzilin/Math-Shepherd.

ground-truth answers. Each generated solution is segmented into reasoning steps and assigned dense scalar rewards using our CPMI-based labeling method, implemented with the Qwen3-8B-Base (Team, 2025) model. The resulting dataset contains 80k step-annotated trajectories, supporting efficient training and evaluation of PRMs with fine-grained reward supervision. Statistics for **CPMI-80k** are summarized in Appendix Table 6.

## 5 Experimental Settings

In this section, we explain the practical implementation details of our CPMI-based reward labeling, focusing on the balance between accuracy and computational efficiency, as well as the training configuration used in subsequent experiments. Further practical details, including reward length normalization and prompt diversification with multiple templates are deferred to Appendix A.

### 5.1 Contrastive Sample Generation

Constructing high-quality hard negatives is crucial for stable contrastive estimation. We combine model-based sampling with lightweight heuristic perturbations to produce plausible yet incorrect answer candidates that reflect realistic reasoning errors. Specifically, we obtain negative samples $\tilde{A}$ from the model itself, following $\tilde{A} \sim p_\theta(\tilde{A} \mid q)$ for theoretical consistency discussed in Section 4.2. We sample $M = 4$ wrong candidates from the model, but if the model keeps generating correct answers on easy problems, we apply heuristic edits such as operator substitution or sign inversion to generate semantically coherent but intentionally incorrect alternatives. Details of the ablation study on different choices of $M$ are provided in Appendix E.1.

### 5.2 Design Choice for Balancing Efficiency

We empirically observe that CPMI rewards at the initial reasoning steps tend to be noisy, as the model is far from the answer space and exhibits high uncertainty in early reasoning stages. To stabilize this effect and enhance the overall accuracy of CPMI labeling, we introduce a hybrid reward formulation that merges CPMI with MC rewards, which we refer to as CPMI-MERGE. This hybrid design leverages the complementary strengths of both signals, as discussed in Section 4 through the lens of TD-$\lambda$. While MC rewards, though sparse and computationally expensive, capture global information by evaluating full rollouts. On the other hand, CPMI rewards provide dense, bootstrapped feedback that refines local belief updates toward the correct answer. As a result, the hybrid design achieves a more balanced bias–variance trade-off between accuracy and efficiency.

Implementation details of different merging variants are presented in Appendix E.2. Empirically, we report the results of merging MC and CPMI rewards at step index 1, which provides comparable or even improved performance over MC-only labeling while significantly reducing the computational cost of reward construction. This configuration demonstrates that a modest degree of early-stage CPMI integration is sufficient to preserve the global supervision of MC rewards while achieving a favorable balance between performance and efficiency.

### 5.3 PRM Training

We train the PRM using Qwen3-4B-Base (Team, 2025) as both the policy model during inference and the PRM backbone. A two-layer linear reward head with a non-linear activation is attached to the backbone to output scalar type step rewards. The PRM is optimized using a binary cross-entropy (BCE) loss, as shown in Eq. (7), where $p_i$ denotes the predicted probability for step $i$, and $\tilde{r}_i$ represents the CPMI-based soft supervision target.

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N}\Big[\tilde{r}_i \log p_i + (1-\tilde{r}_i)\log(1-p_i)\Big] \quad (7)$$

Since CPMI scores theoretically range over $\mathbb{R}$ (empirically between $-3$ and 3), we apply robust z-score normalization to constrain the target values within $[0, 1]$ before training as

$$\hat{r}_i = \frac{r_i - \text{median}(r)}{\text{MAD}(r) + \epsilon}, \quad \tilde{r}_i = \sigma(\hat{r}_i), \quad (8)$$

where $\text{MAD}(r)$ denotes the median absolute deviation, and $\sigma(\cdot)$ is the sigmoid function that maps normalized rewards into the $[0, 1]$ scale.

These training configurations closely follow the setup introduced in MATH-SHEPHERD (Wang et al., 2024) to ensure fair comparison with prior work. This alignment enables consistent evaluation of reward labeling effects across both CPMI-based and MC-based estimation methods.

**Baseline Comparison.** For a fair comparison, we directly implement both $r_{\text{MC}}$ (Eq. (1)) and $r_{\text{PAV}}$ (Eq. (2)) using the same backbone model employed for CPMI labeling (Qwen3-8B-Base (Team, 2025)), and we generate 8 MC rollouts on the same set of

| Reward Type | Model Quality | | | | Computation Cost (Ratio) | | RelEff (×) | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | AUC | PB | PRMB | MATH | Time | Token | AUC | PB | PRMB | MATH |
| MC | 0.759 | 27.7 | 38.8 | 45.4 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| PAV | 0.757 | 36.6 | 49.6 | 47.2 | 1.17 | 2.38 | 0.85 | 1.12 | 1.09 | 0.89 |
| CPMI | 0.765 | 34.6 | 58.8 | 48.2 | 0.16 (↓84%) | 0.02 (↓98%) | **6.34** | **7.85** | **9.53** | **6.68** |
| CPMI_Merge | **0.766** | **36.8** | **60.7** | **49.4** | 0.30 (↓70%) | 0.18 (↓82%) | 3.38 | 4.45 | 5.24 | 3.64 |

Table 1: Efficiency comparison of CPMI-based reward methods with MC. Computation costs (Time, Generated Tokens) are accumulated over a subset of 10K samples. **RelEff** quantifies the quality–cost trade-off relative to the baseline, as illustrated in Eq. 9. Best results are highlighted in **bold**, and second-best results are underlined.

80k question–solution pairs. For $r_{\text{PAV}}$, we additionally load the Qwen2-1.5B-Math model as a prover model, performing 4 rollouts to compute the advantage term, which entails a much higher computational burden than MC methods.

To further isolate the contribution of CPMI labeling, we introduce a control variant, RAND_MERGE, which preserves the same structure of CPMI_Merge but replaces CPMI rewards with uniform random values for all steps after the first. This design allows us to disentangle the specific effect of CPMI from other factors in the merging procedure.

### 5.4 Evaluation

We assess the efficiency of our proposed reward methods using a *relative efficiency* metric that quantifies the trade-off between model quality and computational cost. Given a quality measure $Q$ and a cost measure $C$, the relative efficiency of a reward method $m$ with respect to a baseline $b$ is defined as

$$\text{RelEff}(m \mid b) = \frac{Q_m / C_m}{Q_b / C_b}, \tag{9}$$

where values greater than 1 indicate higher cost-effectiveness. In our experiments, $Q$ corresponds to one of AUC, ProcessBench F1, overall PRMScore of PRMBench or MATH500 accuracy, while $C$ denotes labeling time.

We first evaluate the PRM performance on the PROCESSBENCH benchmark (Zheng et al., 2025) (PB), which measures a model's ability to detect the first incorrect step within a multi-step reasoning trajectory. To obtain a more fine-grained assessment of step-level abilities, we additionally evaluate on PRMBENCH (Song et al., 2025) (PRMB), which categorizes reasoning steps into nine dimensions such as consistency and redundancy, enabling a deeper analysis of PRM behavior. For both benchmarks, thresholds are selected based on the best F1 score on each dataset.

| Setting | AUC | PB | PRMB | Math | MMLU |
|---|---|---|---|---|---|
| **Gold_only** | 0.419 | 15.53 | 41.85 | 41.6 | 71.52 |
| **Neg_only** | 0.725 | 24.65 | 57.42 | 40.8 | 68.92 |
| **Both** | **0.765** | **39.48** | **60.04** | **50.4** | **75.67** |

Table 2: Comparison of different reward signal configurations. **Gold_only** uses only the correct-answer term, **Neg_only** uses only the contrastive negative term, and **Both** combines both terms as in Eq. 3. **AUC** measures the step-level discriminative ability in a threshold-free manner. **PB** denotes the average F1 score, **PRMB** denotes the overall PRMScore, and **MATH** refers to the MATH500 BoN@8 accuracy.

For mathematical reasoning evaluation, we consider two in-domain datasets, GSM8K and MATH500, and two out-of-domain datasets, MMLU-STEM (Hendrycks et al., 2021a) and Omni-MATH (Gao et al., 2024). To further examine the PRM's effectiveness in ranking and selecting high-quality reasoning trajectories, we employ a best-of-N (BoN) test-time scaling strategy. For each problem, the model generates $N$ candidate solution traces, the PRM assigns step scores, and the trajectory with the highest average reward is chosen as the final output, as illustrated in Figure 1.

## 6 Results

The main result is reported in Table. 1. We first examine the effect of our CPMI reward design, particularly focusing on the role of contrastive targets, in Section 6.1. We then evaluate the overall efficiency and effectiveness of our approach in Sections 6.2–6.4 with detailed analyses in Appendix C.

### 6.1 Effect of Contrastive Targets

Prior work on RLVR frequently leverages model-internal likelihood as a confidence signal (Hatamizadeh et al., 2025; Liu et al., 2025; Zhao et al., 2025). However, Table 2 reveals that relying solely on the gold answer (i.e., the first term in Eq. 3) introduces a reward-hacking issue.

| Reward Type | ROC-AUC (↑) | Wasserstein (↑) |
|---|---|---|
| MC | 0.759 | 0.092 |
| PAV | 0.757 | 0.105 |
| CPMI | 0.765 | **0.135** |
| CPMI_Merge | **0.766** | 0.133 |
| Rand_Merge | 0.379 | 0.007 |

Table 3: Distribution-level analysis of PRM scores on the ProcessBench-Math split. Wasserstein reports the 1-Wasserstein distance between the two score distributions in probability space. Bold values indicate the best performance for each metric.

The model may spuriously increase the probability of the correct final answer without learning to reject logically incorrect or misleading steps, which yields poor discrimination on process-level metrics such as ROC-AUC, PB, and PRM. Conversely, using only the negative-target term suppresses the likelihood of wrong answers but still lacks explicit guidance toward the correct reasoning trajectory, resulting in limited performance on downstream tasks such as Math and MMLU.

In contrast, our CPMI objective combines both positive and contrastive negative signals, significantly improving step-level discrimination and leading to better downstream reasoning performance. These results suggest that explicitly penalizing plausible yet incorrect alternatives is essential for mitigating reward hacking and inducing faithful reasoning behavior.

### 6.2 Efficiency

Table 1 summarizes the trade-off between model quality and computational cost in constructing process-level datasets. The CPMI method achieves a drastic reduction in construction costs, requiring only $\mathbf{16}\%$ of the total runtime and $\mathbf{2}\%$ of the generated tokens compared to the MC baseline. The efficiency gain is even larger relative to the PAV, as it requires additional rollouts of the prover model. The CPMI_Merge variant also offers a balanced compromise between performance and budget, maintaining comparable or superior quality to other baselines while reducing both runtime and token usage by more than $\mathbf{70}\%$.

As shown in the last three columns of relative efficiency, both CPMI and CPMI_Merge consistently achieve over $\mathbf{6}\times$ and $\mathbf{3}\times$ improvements, respectively. These results demonstrate that CPMI-based reward estimation provides a stable and scalable trade-off between cost and performance without substantial quality degradation.

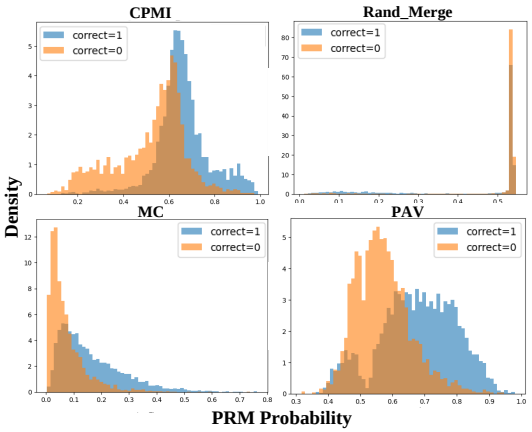


Figure 2: PRM probability distributions on ProcessBench Math split. The x-axis indicates the PRM probabilites. The y-axis shows the normalized density of step occurrences. Blue and orange histograms correspond to correct and incorrect reasoning steps, respectively.

### 6.3 Process-Level Evaluation

**ProcessBench.** We evaluate the verifiability of the trained PRM on the ProcessBench. To assign labels, we follow the ProcessBench gold first-error index, treating all steps before the first detected error as correct and all subsequent steps as incorrect. Since this assumption may not perfectly reflect real-world reasoning dynamics, it inevitably produces partially overlapping distributions, as visualized in Figure 2. Despite this overlap, the distributional metrics in Table 3 show that CPMI-based rewards achieve higher ROC-AUC compared to MC and PAV, indicating a stronger ability to rank correct steps above incorrect ones. The Wasserstein distance in probability space highlights a clearer difference at the global distribution level. MC exhibits almost no separation between correct and incorrect scores, while CPMI and CPMI_Merge substantially enlarge this distance by about $\mathbf{80}\%$ and exceed PAV. These results indicate that CPMI-based rewards not only preserve step-level discriminative power but also push correct and incorrect steps into more distinct regions of the score space, aligning with our goal of emphasizing steps that move the model toward the gold answer space.

Results in Table 4 show that the CPMI-based method consistently outperforms the MC reward across both in-domain and out-of-domain datasets, demonstrating strong generalization in step-level error detection. Moreover, CPMI_Merge surpasses the PAV baseline while maintaining lower computational cost, confirming the effectiveness of integrating local CPMI signals with global Monte Carlo

| Reward Type | GSM8K | Math | Olympiad | Omni | Average |
|---|---|---|---|---|---|
| MC | 35.2 | 38.8 | 17.9 | 18.9 | 27.7 |
| PAV | 51.8 | **43.8** | 27.6 | 23.1 | 36.6 |
| CPMI | 49.4 | 39.6 | 24.5 | 24.8 | 34.6 |
| CPMI_Merge | **52.0** | 40.8 | **28.7** | **25.6** | **36.8** |
| Rand_Merge | 21.7 | 24.9 | 12.2 | 15.9 | 18.7 |
| MS* | 47.9 | 33.9 | 24.8 | 19.8 | 31.5 |
| RLHFlow* | 50.4 | 33.4 | 13.8 | 15.8 | 28.4 |

Table 4: Performance by reward type on ProcessBench (F1 only). The **Average** column indicates the mean F1 across the four datasets. Rows marked with * correspond to reported performance in Wang et al. (2024). Bold values indicate the best F1 in each column.

supervision. Additionally, the large performance gap between CPMI_Merge and RAND_MERGE highlights that the observed improvements are not merely due to the MC component, but arise from the meaningful CPMI-based supervision. These findings collectively demonstrate that CPMI provides a more stable and informative reward signal for process-level evaluation.

**PRMBench.** We further evaluate step quality across the nine dimensions in PRMBench, grouped into three categories (Song et al., 2025), as detailed in Section 5.4. Table 5 shows that CPMI_Merge achieves the strongest performance across all category-level averages as well as the overall score. CPMI_Merge improves the overall PRMBench score by +11.03% over PAV and +21.87% over MC, with the largest gains on Soundness, indicating more logically consistent and less redundant step-level judgments.

### 6.4 BoN Inference

Across different values of $N$ and various mathematical reasoning benchmarks, the CPMI-based reward methods, shown in green and red in Figure 3, consistently outperform both MC and PAV. As $N$ increases, CPMI and CPMI_Merge exhibit stable and consistent scaling behavior, suggesting that their PRMs provide a reliable ranking signal for selecting high-quality trajectories under larger sampling budgets. In contrast, MC shows more non-monotonic trends on some datasets (e.g., GSM8K), indicating that noisier step-level supervision can limit the gains from increasing $N$. RAND_MERGE further highlights this effect, as its performance degrades with larger $N$, suggesting that uninformative or misaligned reward signals can be amplified rather than corrected by increased sampling. Overall, these results indicate that BoN inference is highly sensitive to the quality of the underlying reward model.

Another notable finding from out-of-domain evaluations is that CPMI achieves 76.31% on MMLU-Stem and 15.18% accuracy on Omni, compared with 73.45% and 13.57% obtained by MC, demonstrating stronger generalization under BoN inference.

| Reward Type | Simplicity | Soundness | Sensitivity | Total |
|---|---|---|---|---|
| MC | 40.88 | 36.92 | 35.06 | 38.80 |
| PAV | 47.16 | 49.56 | 47.20 | 49.64 |
| CPMI | 52.56 | 60.21 | 55.59 | 58.75 |
| CPMI_Merge | **54.65** | **62.30** | **56.50** | **60.67** |
| Rand_Merge | 27.65 | 21.64 | 18.65 | 23.20 |

Table 5: PRMBench results. Average scores over nine categories into three step-level dimensions (Simplicity, Soundness, Sensitivity) and the overall average for each reward method.

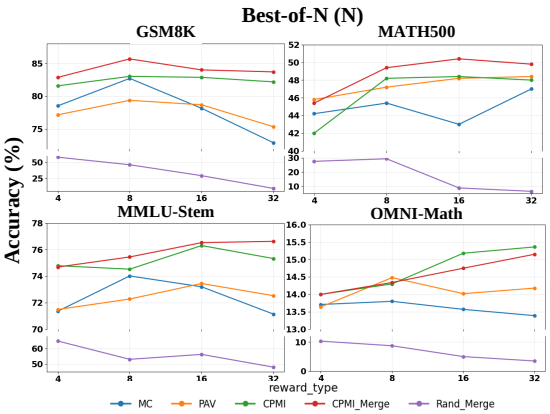


Figure 3: Best-of-$N$ accuracy on math benchmarks. The x-axis denotes the number of samples $N$ used for Best-of-$N$ selection, and the y-axis reports accuracy.

## 7 Conclusion

We introduced CPMI, a novel method for automatically generating process-level supervision without relying on human annotation or incurring the substantial computational cost required by traditional MC methods. By providing dense, localized contrastive reward signals, CPMI enables the construction of effective PRMs while reducing generation time by more than 80% and token usage by 90%. The resulting PRMs exhibit strong step-quality assessment capabilities and yield consistent improvements in BoN inference across various mathematical benchmarks. These results demonstrate that CPMI provides a practical and resource-efficient alternative to MC-based supervision, particularly in settings with limited computational budgets.

## Limitations

We validate our methods only during the inference stage. While reward or verifier models are typically employed within online reinforcement learning frameworks, our CPMI-based approach could naturally be extended to such training settings. In this work, we focus on its efficiency and discriminative capacity under offline evaluation. Another limitation lies in the scale of our trajectory data. We collect around 80K process-level samples, which is relatively small compared with recent large-scale PRM datasets. Future work can explore larger and more diverse trajectories, as well as varying PRM model sizes beyond the current Qwen3-4B configuration, to better understand scalability and robustness across different model capacities.

## Acknowledgments

This work is in part supported by the National Research Foundation of Korea (NRF, RS-2024-00451435(20%), RS-2024-00413957(20%)), Institute of Information & communications Technology Planning & Evaluation (IITP, RS-2025-02305453(15%), RS-2025-02273157(15%), RS-2025-25442149(15%) RS-2021-II211343(15%)) grant funded by the Ministry of Science and ICT (MSIT), Institute of New Media and Communications(INMAC), and the BK21 FOUR program of the Education, Artificial Intelligence Graduate School Program (Seoul National University), and Research Program for Future ICT Pioneers, Seoul National University in 2026.

## References

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024. Alphamath almost zero: Process supervision without process. *arXiv preprint arXiv:2405.03553*.

Wenhu Chen and 1 others. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

Jie Cheng, Ruixi Qiao, Lijun Li, Chao Guo, Junle Wang, Gang Xiong, Yisheng Lv, and Fei-Yue Wang. 2025. Stop summation: Min-form credit assignment is all process reward model needs for reasoning. *arXiv preprint arXiv:2504.15275*. PURE: PRM-supervised reinforcement fine-tuning for reasoning.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Łukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, and .... 2024. Omni-math: A universal olympiad level mathematic benchmark for large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. *Preprint*, arXiv:2211.10435.

Ali Hatamizadeh, Syeda Nahida Akter, Shrimai Prabhumoye, Jan Kautz, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, and Yejin Choi. 2025. Rlp: Reinforcement as a pretraining objective. arXiv preprint arXiv:2510.01265.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

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

Geunyeong Jeong, Juoh Sun, and Harksoo Kim. 2025. Watch your step: A fine-grained evaluation framework for multi-hop knowledge editing in large language models. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, CIKM '25, page 4842–4846, New York, NY, USA. Association for Computing Machinery.

Xiongnan Jin, Zhilin Wang, Jinpeng Chen, Liu Yang, Byungkook Oh, Seung-won Hwang, and Jianqiang Li. 2025. Hlmea: Unsupervised entity alignment based

on hybrid language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(11):11888–11896.

Wei Liu, Siya Qi, Xinyu Wang, Chen Qian, Yali Du, and Yulan He. 2025. Nover: Incentive training for language models via verifier-free reinforcement learning.

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

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. 2025. PRMBench: A fine-grained and challenging benchmark for process-level reward models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vienna, Austria. Association for Computational Linguistics.

Weiqi Sun and 1 others. 2025. An efficient and precise training data construction framework for process-supervised reward model in mathematical reasoning. *arXiv preprint arXiv:2503.02382*.

Richard S. Sutton. 1988. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44.

Qwen Team. 2025. Qwen3 technical report. Technical Report arXiv:2505.09388, Qwen Model Team, Qwen Language Model. Open weight large language model family (0.6B–235B parameters) with thinking/non-thinking modes, multilingual capabilities and public release under Apache 2.0.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *CoRR*, abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS) 35*, pages 24824–24837.

Wei Xiong, Wenting Zhao, Weizhe Yuan, Olga Golovneva, Tong Zhang, Jason Weston, and Sainbayar Sukhbaatar. 2025. Stepwiser: Stepwise generative judges for wiser reasoning. *arXiv preprint arXiv:2508.19229*.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. *CoRR*, abs/2501.07301. ArXiv preprint arXiv:2501.07301v2.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. 2025. Learning to reason without external rewards.

Chen Zheng and 1 others. 2025. Processbench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations (ICLR)*.

## A Practical Implementations

In practice, LLMs are highly sensitive to prompt phrasing, which can bias the estimated step contributions and increase variance. To obtain stable and reliable reward estimates, we incorporate two key design choices: (*i*) Prompt diversification and the construction of hard negative answers, (*ii*) length normalization when computing the CPMI.

**Prompt diversification.** To mitigate the sensitivity of prompt wording, we employ $K$ distinct prompt templates. Each template presents the problem statement $q$ with minor structural or formatting variations. Averaging across these templates reduces prompt-specific variance and yields more stable step-level rewards. We use the four prompts shown in Figure 4, which convey the same instruction but are phrased in slightly different ways.

We use the prompt templates in Figure 5, varying the number of samples $M$.

**Prompt #1**

```
You are a careful math solver. Follow the
steps methodically. Keep each step concise.\n
At the end, output exactly one line in the
format:\n The answer is: <final answer>\n\n
Problem:\n{q}\n Solution: Let's think step by
step.\n
```

**Prompt #2**

```
Solve the problem with numbered steps. Be
precise. Finally print exactly:\n The answer is:
<final answer>\n If numeric, use plain digits
only (no punctuation).\n\n Problem:\n{q}\n
Solution (step-by-step):\n
```

**Prompt #3**

```
Work through the solution briefly, then verify
and conclude. Conclude with exactly:\n The
answer is: <final answer>\n\n Problem:\n{q}\n
Solution: Let's proceed carefully.\n
```

**Prompt #4**

```
You are solving a math problem. Compute step by
step. End with exactly:\n The answer is: <final
answer>\n\n Problem:\n{q}\n Solution: Let's
think step by step.\n
```

Figure 4: The four prompts used for diversification.

**Hard-Negative Generation Prompt Template**

```
<Base Prompt containing the Problem Question>
(Gold answer: <Gold Truth Value>)
Generate several plausible but incorrect
answers to the given question. Identify any
wrong step and take that step's wrong computed
result or the mistaken number used in it as
a hard-negative example. Try to include all
earlier steps' wrong intermediate outputs if
possible. If all steps seem correct, return a
plausible wrong number (±1, ±2, or ±10%).
Output exactly one short line after 'The answer
is:' no units or words. \n
The answer is:
```

Figure 5: Prompt for generating hard-negative answers.

**Length Normalization.** When computing the pointwise mutual information in Eq. 3, we compute the log-probability as $\log p_\theta(a \mid X) = \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(a_l \mid a_{<l}, X)$, applying length normalization to mitigate the influence of varying sequence lengths.

## B Statistics of CPMI80k

| Statistic | Ratio (%) / Count |
|---|---|
| Total samples | 80,000 |
| Task Source (GSM8K : MATH) | 1 : 1 |
| Composition (mixed : pure) | 7 : 3 |

Table 6: Statistics of the CPMI-80K dataset. All values are reported as counts or ratios.

When constructing the 80k-sample from the Math-Shepherd corpus, we sample an equal number of problems from GSM8K and MATH to ensure coverage of both easier and more challenging reasoning tasks. We exclude extremely difficult problems, as PRMs fail to extract meaningful learning signals when all step-wise rewards collapse to zero. Additionally, we intentionally include a higher proportion of *mixed* trajectories, containing both positive (+) and negative (−) step labels, than *pure* trajectories with a single label type, as mixed trajectories provide richer supervisory signals for training step-level reward models.

## C Fine-Grained Process-Level Evaluation

We provide extended results on ProcessBench and PRMBench in Table 7 and Table 8, respectively. Table 7 reports three metrics for each dataset—error accuracy (Err), correct accuracy (Corr), and F1—along with their mean F1 in the **Average** column. We observe that the performance

| | GSM8K | | | Math | | | Olympiad | | | Omni | | | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Reward Type** | **Err** | **Corr** | **F1** | **Err** | **Corr** | **F1** | **Err** | **Corr** | **F1** | **Err** | **Corr** | **F1** | **F1** |
| MC | 22.2 | 85.0 | 35.2 | 25.9 | 76.8 | 38.8 | 10.7 | 54.3 | 17.9 | 11.5 | 53.9 | 18.9 | 27.7 |
| PAV | 35.7 | 93.8 | 51.8 | 32.2 | 68.5 | **43.8** | 17.5 | 64.3 | 27.6 | 14.5 | 57.3 | 23.1 | 36.6 |
| CPMI | 35.7 | 79.8 | 49.4 | 29.0 | 62.8 | 39.6 | 15.9 | 53.7 | 24.5 | 17.5 | 42.3 | 24.8 | 34.6 |
| CPMI_Merge | 37.2 | 86.5 | **52.0** | 31.0 | 59.6 | 40.8 | 20.1 | 49.9 | **28.7** | 18.4 | 41.9 | **25.6** | **36.8** |
| Rand_Merge | 13.5 | 55.4 | 21.7 | 16.5 | 50.5 | 24.9 | 7.1 | 43.4 | 12.2 | 9.6 | 46.1 | 15.9 | 18.7 |
| MS* | 32.4 | 91.7 | 47.9 | 18.2 | 69.2 | 33.9 | 15.1 | 77.1 | 24.8 | 12.4 | 72.3 | 19.8 | 31.5 |
| RLHFlow* | 33.8 | 99.0 | 50.4 | 21.4 | 72.2 | 33.4 | 8.2 | 43.1 | 13.8 | 9.6 | 45.2 | 15.8 | 28.4 |

Table 7: Detailed Performance by reward type on ProcessBench. The last two rows marked with * correspond to the reported performance in Wang et al. (2024). Bold values indicate the best F1 score in each column.

| | Simplicity | | Soundness | | | | Sensitivity | | | Overall |
|---|---|---|---|---|---|---|---|---|---|---|
| **Reward Type** | NR | NCL | ES | SC | DC | CI | PS | DR | MS | |
| MC | 37.56 | 44.20 | 43.18 | 26.84 | 36.72 | 40.94 | 42.28 | 42.30 | 20.61 | 38.8 |
| PAV | 45.69 | 48.62 | 55.86 | 39.09 | 46.96 | 56.32 | 50.17 | 53.18 | 38.25 | 49.64 |
| CPMI | 50.22 | 54.89 | 63.76 | 56.99 | 56.58 | 63.52 | 57.70 | 62.02 | **47.06** | 58.75 |
| CPMI_Merge | **51.27** | **58.02** | **65.99** | **58.96** | **58.70** | **65.53** | **59.28** | **63.34** | 46.89 | **60.67** |
| Rand_Merge | 20.14 | 35.15 | 26.40 | 19.26 | 21.11 | 19.79 | 25.71 | 23.24 | 7.00 | 23.20 |

Table 8: PRMBench results across reward designs. Bold values indicate the best PRMScore.

gaps between MC, PAV, and our CPMI-based methods are more pronounced in challenging out-of-domain evaluations such as Olympiad and Omni, highlighting CPMI's improved generalization to complex reasoning tasks.

Table 8 presents fine-grained performance according to the nine evaluation dimensions introduced in Song et al. (2025): Non-Redundancy (NR), Non-Circular Logic (NCL), Empirical Soundness (ES), Step Consistency (SC), Domain Consistency (DC), Confidence Invariance (CI), Prerequisite Sensitivity (PS), Deception Resistance (DR), and Multi-Solution Consistency (MS). These results demonstrate that CPMI enhances the model's capability to distinguish diverse failure modes in step-level reasoning, offering more reliable reward signals across a wide range of logical and mathematical phenomena.

# D Robustness of CPMI

## D.1 Robustness Across Labeling Model Capacities

We further evaluate CPMI on a different base model family, focusing on whether the proposed reward construction remains effective even when the labeling model is much smaller.

Table 9 shows that CPMI remains consistently effective even with compact labeling models from two different families, `Qwen2.5-1.5B` and `Llama3.2-1B`. In both cases, replacing MC labeling with CPMI-based labeling improves the step-level discrimination metric (ROC-AUC) and yields large gains on process-level benchmarks, especially on ProcessBench and PRMBench. Importantly, these quality improvements are achieved while substantially reducing reward construction cost, requiring 65–73% less labeling time and 82% fewer generated tokens.

These results suggest that although stronger pre-trained models can provide cleaner intrinsic signals, CPMI does not depend on a single high-capacity model regime. Instead, it offers a robust and cost-effective reward proxy even with lightweight labeling backbones, making the overall pipeline more practical for resource-constrained settings.

## D.2 Generalization to Logical Reasoning

To further validate the robustness of our reward signal beyond mathematical reasoning, we additionally evaluate PRMs trained with different reward constructions on out-of-distribution logical reasoning benchmarks. Following the main evaluation setup, we use Qwen3-4B-Base under zero-shot Best-of-8 inference.

As shown in Table 10, both CPMI and

| Labeler | Reward | ROC-AUC ↑ | PB ↑ | PRMB ↑ | MATH500 ↑ | ΔTime | ΔToken |
|---|---|---|---|---|---|---|---|
| Qwen2.5-1.5B | MC | 0.72 | 28.4 | 16.2 | 41.6 | 0% | 0% |
| | CPMI-based | **0.75** | **36.1** | **51.8** | **45.8** | **-65%** | **-82%** |
| Llama3.2-1B | MC | 0.73 | 29.5 | 18.6 | **48.0** | 0% | 0% |
| | CPMI-based | **0.76** | **37.0** | **59.1** | 47.6 | **-73%** | **-82%** |

Table 9: Robustness of CPMI across smaller labeling models from different families.

| Reward Type | FOLIO ↑ | LogiQA 2.0 ↑ | LogicNLI ↑ |
|---|---|---|---|
| MC | 54.19 | 33.39 | 25.4 |
| CPMI | **59.61** | **37.71** | **32.1** |
| CPMI_Merge | 58.62 | 35.66 | 30.8 |
| Rand_Merge | 42.86 | 24.54 | 24.5 |

Table 10: Generalization to out-of-distribution logical reasoning benchmarks under zero-shot BoN@8.

| Loss | Reward | ProcessBench | PRMBench | MATH500 |
|---|---|---|---|---|
| BCE | MC | 27.7 | 38.8 | 45.4 |
| | CPMI | **34.6** | **58.8** | **48.2** |
| PQM | MC | **37.9** | 45.4 | 44.0 |
| | CPMI | 35.6 | **56.7** | **46.2** |
| MSE | MC | 31.0 | 43.1 | 45.4 |
| | CPMI | **35.5** | **59.9** | **47.2** |

Table 11: Robustness of CPMI across different PRM training objectives.

CPMI_Merge consistently outperform MC across all three benchmarks, including FOLIO, LogiQA 2.0, and LogicNLI. These results indicate that the benefits of CPMI are not limited to in-domain math reasoning, but also transfer to logical reasoning tasks that require process-level verification.

### D.3 Robustness Across Training Objectives

Our core contribution lies in the labeling signal and dataset construction, which are largely agnostic to the downstream training objective. We use BCE in the main experiments to match the Math-Shepherd training setup as closely as possible, ensuring a fair comparison with MC-based baselines. Since CPMI provides continuous step-wise supervision after normalization, it can naturally be paired with objectives beyond BCE.

To verify this, we additionally train PRMs with two alternative objectives: mean squared error (MSE), which treats reward prediction as a regression problem, and pairwise ranking loss (PQM), which emphasizes relative ordering between steps. As shown in Table 11, CPMI remains consistently competitive across all training objectives and continues to outperform MC in PRMBench and MATH500 under both BCE and MSE.

Interestingly, PQM+MC achieves the highest ProcessBench score, suggesting that pairwise ranking can act as a highly localized error detector. However, this gain does not transfer to broader process-level quality measures, where CPMI-based supervision remains substantially stronger, especially on PRMBench. Overall, these results support our claim that the advantage of CPMI primarily comes from the reward signal itself rather than from a specific loss design.

## E Ablation Experiments

We conduct a series of ablation studies to evaluate the robustness and generalizability of our CPMI reward design. First, we vary the number of hard-negative targets to analyze how contrastive signals influence performance (Section E.1). We then examine the efficiency and performance–cost trade-offs of different step at which CPMI replaces MC used in the CPMI_Merge variants (Section E.2). Finally, we demonstrate the flexibility of CPMI by merging it with PAV-based baselines (Section E.3).

### E.1 Ablations on the Hard-Negative Term

Figure 6 varies the number of hard-negative answers $M$ used in the CPMI objective. Transitioning from $M = 0$ (Gold_only) or using only the negative term (Neg_only) to any setting with at least one negative sample ($M \geq 1$) yields substantial improvements across all evaluation metrics, as mentioned in Section 6.1. Interestingly, contrary to the expectation that increasing $M$ would yield more stable performance by reducing variance, $M = 1$ achieves the strongest Math accuracy and competitive performance on the other benchmarks. We attribute this behavior to a trade-off between signal strength and averaging: when $M = 1$, the CPMI reward focuses on the single hardest negative, providing a strong contrastive signal, whereas larger $M$ values average over easier or noisier negatives,

diluting the gradient despite reducing variance.

At the same time, the differences among $M \in \{1, 2, 4\}$ are relatively small (within a few points across benchmarks), suggesting that CPMI is not overly sensitive to the precise choice of $M$ once at least one hard negative is included. In our main experiments, we therefore fix $M = 4$ as a default, which offers a stable estimate of the negative term and reduces sensitivity to the particular sampled negatives, while maintaining performance close to the best ablation setting. Overall, these results indicate that the key factor is the presence of explicit contrastive negatives, and that moderate values of $M$ (e.g., $M = 1$–$4$) strike a reasonable balance between robustness and computational cost.

### E.2 Ablations on Different Merge Points

Table 12 reports the resulting performance across four metrics: step-level discriminative ability measured by ROC-AUC (**AUC**), average F1 on ProcessBench (**PB**), the overall PRMScore from PRM-Bench (**PRM**), and MATH500 BoN@8 accuracy using PRM-weighted trajectory selection (**MATH**). We use MC reward only at the first step as an initializer, we did ablation experiments of diverging the merge index to figure out the sweet spot for performance cost efficiency trade-off. We additionally report the computational cost of annotating step-level datasets. The relative efficiency score (**RelEff**) are computed using Eq. (9). As shown in Table 12, merging at index 1 yields the strongest overall performance across benchmarks while preserving a high level of efficiency, achieving relative efficiency gains in the range of 3.4× to 5.2×. Although later merge indices maintain competitive performance, their efficiency declines substantially due to the increased computational cost associated with relying on MC rewards for more steps. These results indicate that applying MC only at the earliest step and transitioning to CPMI immediately after provides the most favorable balance between quality and cost.

### E.3 Initialized with PAV

Since this initialization strategy is compatible with other labeling methods as well, we additionally evaluate a variant that uses the PAV reward for the first step and applies CPMI to all subsequent steps. We assess performance using four metrics, the same metrics used in Section. E.2. As shown in Table 13, initializing CPMI with PAV yields stable behavior and achieves a favorable performance–efficiency trade-off, mirroring the trends observed when MC is used as the initializer.

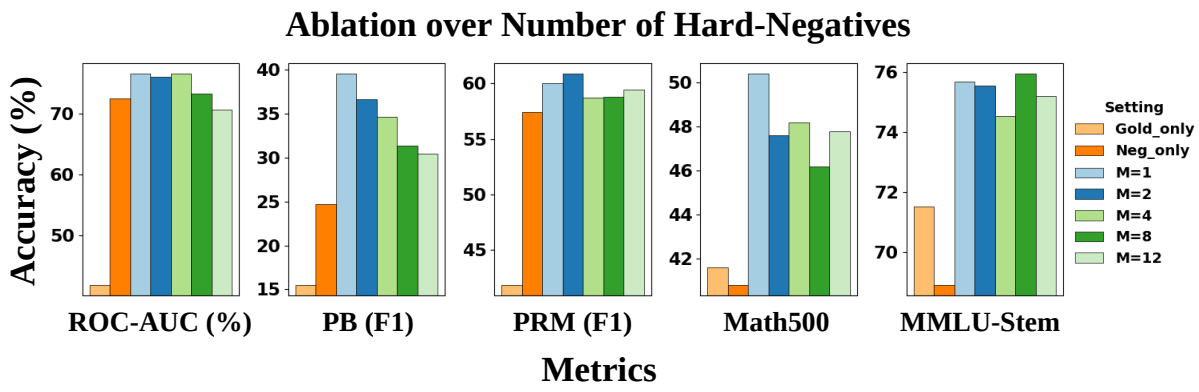


Figure 6: Ablation results varying the number of hard-negative targets $M$ in the CPMI objective. We report both process-level metrics (ROC-AUC, PB, PRM) and downstream performance (Math, MMLU).

| | | | Model Quality | | | | Computation Cost (Ratio) | | | RelEff (×) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Merge Index** | **MC(%)** | **CPMI(%)** | ROC-AUC | PB | PRMB | MATH | Time | ratio | gen_tokens | AUC | PB | PRMB | MATH |
| CPMI | 0 | 100 | 0.765 | 34.6 | 58.8 | 48.2 | 38,603 | 0.16 | 1,579,148 | **6.34** | **7.85** | **9.53** | **6.68** |
| CPMI_Merge1 | 16.6 | 83.4 | 0.766 | **36.8** | **60.7** | 49.4 | 72,553 | 0.30 | 16,420,803 | 3.38 | 4.45 | 5.24 | 3.64 |
| CPMI_Merge2 | 33.2 | 66.8 | 0.784 | 35.8 | 58.9 | 47.0 | 106,452 | 0.44 | 31,239,824 | 2.36 | 2.95 | 3.47 | 2.36 |
| CPMI_Merge3 | 48.0 | 52.0 | **0.804** | 36.6 | 56.7 | **56.6** | 136,579 | 0.56 | 44,410,422 | 1.88 | 2.35 | 2.60 | 2.22 |
| MC | 100 | 0 | 0.759 | 27.7 | 38.8 | 45.4 | 242,930 | 1.00 | 90,902,162 | 1.00 | 1.00 | 1.00 | 1.00 |

Table 12: Efficiency analysis of CPMI merging variants across quality, cost, and cost-effectiveness metrics.

| | Model Quality | | | | Computation Cost (Ratio) | | RelEff (×) | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Reward Type** | **AUC** | **PB** | **PRMB** | **MATH** | **Time** | **Token** | **AUC** | **PB** | **PRMB** | **MATH** |
| PAV | 0.757 | 36.6 | 49.6 | 47.2 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| CPMI | 0.765 | 34.6 | 58.8 | 48.2 | 0.14 (↓86%) | 0.01 (↓99%) | **7.46** | **6.99** | **8.75** | **7.55** |
| CPMI_Merge1 | 0.766 | 36.3 | **60.2** | **49.2** | 0.28 (↓72%) | 0.17 (↓83%) | 3.63 | 3.56 | 4.35 | 3.74 |
| CPMI_Merge2 | 0.750 | 37.5 | 56.8 | 49.6 | 0.42 (↓58%) | 0.34 (↓66%) | 2.34 | 2.43 | 2.71 | 2.49 |
| CPMI_Merge3 | **0.797** | **39.9** | 56.1 | 48.1 | 0.55 (↓45%) | 0.48 (↓52%) | 1.91 | 1.98 | 2.06 | 1.86 |

Table 13: Efficiency comparison of CPMI and PAV. Computation costs (Time, Generated Tokens) are accumulated over subset of 10K samples.