LEARNING DISCRIMINATIVE PROCESS REWARD MODELS WITHOUT STEP LABELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Process reward models (PRMs) can improve LLM reasoning by providing step-level feedback, but training them typically depends on costly step annotations, which limits scalability. Final outcome labels, in contrast, are inexpensive to obtain; however, existing outcome-supervised methods either learn outcome-only reward models that offer no stepwise guidance, or adopt implicit reward formulations that increase inference cost and are susceptible to reward hacking. To address these limitations, we propose a new framework for learning discriminative PRMs using only outcome labels. Our approach treats step quality as a latent variable and connects it to the outcome via an aggregation function that emphasizes low-scoring steps (e.g. geometric mean). This enables end-to-end training by backpropagating the outcome loss through the aggregator to step-level scores. The resulting classifier-style PRMs are efficient at inference and achieve competitive performance on challenging reasoning tasks, both for test-time search and on PRM benchmarks.

1 INTRODUCTION

Modern large language models (LLMs) have demonstrated remarkable automated reasoning capabilities with chain-of-thought prompting (Wei et al., 2022). However, they still remain vulnerable to logical hallucinations and error propagation, where a single incorrect step can invalidate an entire reasoning trajectory. This fragility has necessitated the development of reward models, serving as specialized verifiers to score generated outputs and enhance the reliability of the reasoning process.

The efficacy of a reward model is largely determined by the granularity of its supervision. Outcome Reward Models (ORMs) evaluate the validity of a solution based solely on the final answer (Cobbe et al., 2021). While ORMs benefit from the abundance of easily obtainable outcome labels, their feedback is inherently sparse, failing to identify the precise location of reasoning errors. In contrast, process reward models (PRMs) assign scores to individual reasoning steps (Lightman et al., 2023), and such fine-grained feedback has proven highly effective for complex reasoning, substantially improving both test-time scaling (Lightman et al., 2023; Wang et al., 2024; Luo et al., 2024; Yuan et al., 2024; Li & Li, 2024; Yang et al., 2025; Xiong et al., 2025) and reinforcement fine-tuning (Setlur et al., 2024; Zou et al., 2025; Cui et al., 2025; Zha et al., 2025; Liu et al., 2026). However, training PRMs typically require step-level labels from expert annotation or extensive rollouts from intermediate steps, limiting their scalability. This dilemma has motivated weakly supervised approaches that attempt to learn process-level signals using only outcome-level labels. The prevailing methods in this category often rely on implicit reward model formulations (Yuan et al., 2024; Cui et al., 2025), where step scores are derived from the log-likelihood ratios of generative policies. While these methods successfully circumvent the need for step-level annotations, they introduce additional computational overhead. Furthermore, implicit reward models have been shown to be susceptible to reward hacking, potentially prioritizing superficial token patterns over semantic information (Razin et al., 2025).

In this work, we propose a novel framework for learning *discriminative* process reward models using only final outcome labels. Our approach retains the inference speed and architectural simplicity of standard discriminative classifiers while achieving the granular step-level supervison. The core intuition behind our method is the "weakest link" principle of logical reasoning: the quality of a multi-step solution is governed by the most problematic part of the reasoning trajectory (Pollock, 2001; Jacovi et al., 2024). We capture this intuition by treating step qualities as latent variables and linking them to the observed outcome through a reward aggregation objective that is sensitive to low-

scoring steps (e.g. minimum or geometric mean). Training then backpropagates the final-outcome loss through the aggregator, encouraging the model to place low scores on steps that most explain failures, thereby learning to localize errors without any explicit step-level supervision. Extensive experiments on complex reasoning tasks demonstrate that PRMs trained via our framework achieve performance competitive with strong baselines, both in test-time search effectiveness and on standard PRM benchmarks (Zheng et al., 2024; Song et al., 2025).

2 METHODS

Given a question q and a response $o = (o_1, \ldots, o_{|o|})$ with $|o|$ tokens, we segment o into M_o contiguous reasoning steps where step m is $s_m = (o_{t_m}, \ldots, o_{t_{m+1}-1})$ with $1 = t_1 < \cdots < t_{M_o+1} = |o| + 1$. Let $s_{<m} = (s_1, \ldots, s_{m-1})$ and define the content till step $m-1$ as $x_m = (q, s_{<m})$. Given a unlabeled dataset $\mathcal{D}$ that contains question-response pair (q, o), we consider two different kinds of labels for $\mathcal{D}$. The first one is *step-level* dataset $\mathcal{D}_{\text{step}}$ that contains the quality label for every step in $\mathcal{D}$. Concretely, given $(q, o) \in \mathcal{D}$, $\mathcal{D}_{\text{step}}$ contains tuples (x_m, s_m, c_m) for each step $m \in [M_o]$ where $c_m \in \{-1, 0, 1\}$ indicates whether step s_m is bad, okay or good given the prefix x_m. This three label classification approach follows Lightman et al. (2023). The second kind of dataset $\mathcal{D}_{\text{final}}$ only contains the *final correctness label* of a reponse. Specifically, for every $(q, o) \in \mathcal{D}$, $\mathcal{D}_{\text{final}}$ contains a tuple (q, o, c) where $c \in \{0, 1\}$ denotes whether the whole response o is correct given q.

2.1 EXISTING METHODS

Training PRM with Step Labels (Lightman et al., 2023). With step-level supervision, we treat PRM training as a three-way classification problem with labels $c_m \in \{-1, 0, 1\}$ corresponding to bad, okay and good for the step quality. Given a step context $x_m = (q, s_{<m})$ and step s_m, the PRM outputs three logits $\mathbf{z}_\theta(x_m, s_m) \in \mathbb{R}^3$ and the class probabilities are $p_\theta(y \mid x_m, s_m) = \text{softmax}(\mathbf{z}_\theta(x_m, s_m))_y, y \in \{-1, 0, 1\}$. We then train the PRM multiclass cross-entropy Eq. (1):

$$\min_\theta \sum_{(x_m, s_m, c_m) \in \mathcal{D}_{\text{step}}} -\log p_\theta(c_m \mid x_m, s_m). \tag{1}$$

Outcome Reward Model (Cobbe et al., 2021). With final correctness labels, a common baseline is an outcome reward model (ORM) r_θ that scores an entire response and is trained via Eq. (2).

$$\min_\theta \sum_{(q,o,c) \in \mathcal{D}_{\text{final}}} c \log\left(\sigma\left(r_\theta(q, o)\right)\right) + (1 - c) \log\left(1 - \sigma\left(r_\theta(q, o)\right)\right). \tag{2}$$

where σ denotes the sigmoid function.

Implict PRM (Yuan et al., 2024). With final correctness label only, Yuan et al. (2024) shows that it is possible to train a process reward model through implict reward model formulation. Specifically, given a question q, response o and reference policy π_{ref}, Yuan et al. (2024) defines $r_\theta(q, o) = \beta \log\left(\frac{\pi_\theta(o|q)}{\pi_{\text{ref}}(o|q)}\right)$ with $\beta = 0.05$ as log likelihood ratio of a learned policy and a reference policy, then optimizes it using the same outcome-level objective in Eq. (2). After training, Yuan et al. (2024) utilizes token level decomposition to evaluate the score of each step. For question q, response $o = (s_1, s_2, \ldots, s_{M_o})$ with $s_m = (o_{t_m}, o_{t_m+1}, \ldots, o_{t_{m+1}-1})$. The reward of step s_m given prefix x_m is defined as $r_\theta(x_m, s_m) = \sum_{i=t_m}^{t_{m+1}-1} \beta \log\left(\frac{\pi_\theta(o_i|q,o_{<i})}{\pi_{\text{ref}}(o_i|q,o_{<i})}\right)$. This yields a step-level signal. However, implicit reward models must evaluate the likelihood of both the learned and reference policies, making inference more expensive than discriminative models. Moreover, recent work suggests implicit reward models can be more easily hacked by superficial token-level changes (Razin et al., 2025).

2.2 OUR METHOD

To address the limitations of the approaches above, we propose a framework to train a discriminative PRM using only final correctness labels. This allows us to retain the low inference cost of discriminative classifiers while avoiding the annotation costs of explicit process supervision. Our method bridges the gap between the local nature of process rewards (step-level quality) and the global nature of the supervision (outcome correctness) via an aggregation function Agg. For every $(q, o, c) \in \mathcal{D}_{\text{final}}$ we define $A_\theta(q, o; c) = \text{Agg}\big(\{r_\theta(x_m, s_m)\}_{m=1}^{M_o}; c\big)$, where r_θ is a PRM parameterized by θ, and $\text{Agg}(\cdot; c)$ maps the set of step scores $\{r_\theta(x_m, o(s_m))\}_{m=1}^{M_o}$ to a scalar. Crucially, we allow the

aggregation function to depend on the outcome label c, enabling asymmetric aggregation: correct and incorrect samples can be summarized differently during training.

The aggregator Agg should summarize step scores into a response-level signal, assigning higher values to correct responses than to incorrect ones. Since an incorrect solution typically contains at least one flawed step, we focus on aggregators that emphasize low step scores, including

- $\text{Agg}_{\min}(\{r(x_m, s_m)\}_{m=1}^{M_o}) = \min_{m\in[M_o]} r(x_m, s_m)$ (minimum)
- $\text{Agg}_{\text{prod}}(\{r(x_m, s_m)\}_{m=1}^{M_o}) = \left(\prod_{m\in[M_o]} r(x_m, s_m)\right)^{\frac{1}{M_o}}$ (geometric mean)
- $\text{Agg}_{\text{soft}}(\{r(x_m, s_m)\}_{m=1}^{M_o}) = -\tau \log\left(\frac{1}{M_o}\sum_{m\in[M_o]} \exp(-\frac{1}{\tau} r(x_m, s_m))\right)$ (softmin)

All these aggregators are heavily influenced by the lowest step score, so a single poor step can sharply reduce the aggregated score. This aggregation perspective also matches common test-time usage of PRMs, where step-level scores are combined with emphasis on the weakest step to produce a response-level assessment (Lightman et al., 2023).

With these aggregators, an incorrect response should have lower aggregated score because at least one step is bad. To learn this, we utilize binary cross-entropy loss Eq. (3) that pushes the aggregated score up for correct responses and down for incorrect ones, which in turn drives the PRM to assign higher scores to steps in correct trajectories and to down-weight the erroneous steps in incorrect trajectories.

$$\min_\theta \sum_{(q,o,c)\in\mathcal{D}_{\text{final}}} \left[c \log \sigma(A_\theta(q, o, c)) + (1 - c) \log(1 - \sigma(A_\theta(q, o, c)))\right]. \tag{3}$$

In Section 3, we will compare the performance of PRM trained with different aggregators. We find that using *asymmetric* aggregation functions for correct and incorrect responses gives the most favorable results. In particular, $A_\theta(\cdot; 1) = \text{Agg}_{\min}(\cdot)$ and $A_\theta(\cdot; 0) = \text{Agg}_{\text{prod}}(\cdot)$ (i.e. use minimum aggregation for correct samples and geometric mean aggregation for incorrect samples) achieve the best performance. This aligns with the asymmetric nature of the problem: reasoning can fail due to a single erroneous step, but only succeed when every step is correct. We elaborate more in Appendix A.

3 Experiments

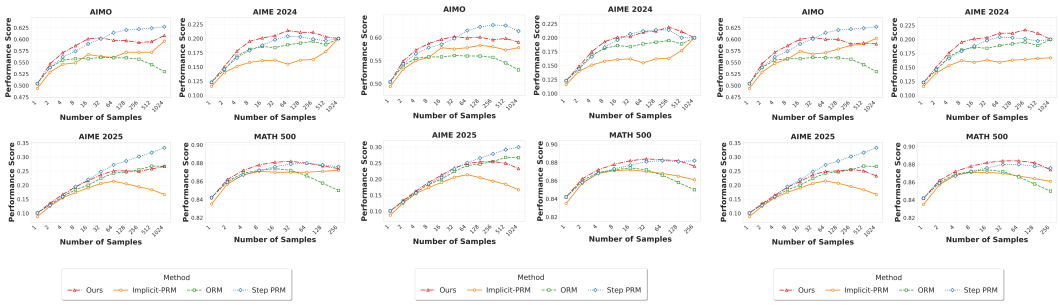
Training Dataset and Hyperparameters. Constructing the training dataset for PRMs required a two-stage process. First, we collected rollouts with Qwen-2.5-Math-7B-Instruct (Yang et al., 2024), with questions from the OpenThoughts Dataset (Guha et al., 2025). We then determine final answer correctness with the Math-Verify[1] library, and collect per-step labels with powerful QwQ-32B (Team, 2025). The step-level label is only used for the baseline method. Further details about dataset collection are provided in Appendix B.1. For our method, we choose $A_\theta(\cdot; 1) = \text{Agg}_{\min}(\cdot)$ and $A_\theta(\cdot; 0) = \text{Agg}_{\text{prod}}(\cdot)$ for the results in this section and more discussions on other aggregation methods are provided in Appendix C. We compare our approaches with PRM trained with step label (**Step PRM**), outcome reward model (**ORM**) and **Implicit PRM** with $\beta = 0.05$.

Models. We use Qwen-2.5-Math-1.5B-Instruct as the base model to train all PRMs. For implicit PRM (Yuan et al., 2024), since it is a generative reward model, we directly train on the base model. All other mehods are discriminative, so we replace the final softmax layer the base model with a two-layer MLP to serve as the classification head. The final layer has N outputs, where $N = 3$ for training PRM with step labels as shown in Section 2.1 and $N = 1$ for all other method.

Best-of-N Evaluation. We evaluate PRMs on the MATH-500 (Hendrycks et al., 2021), AIME2024, AIME 2025 and AIMO[2] datasets. For each question, we sample up to $N = 1024$ rollouts from Qwen-2.5-Math-7B-Instruct (Team, 2024). We segment each response into steps using the delimiter "\n\n", a standard heuristic (Zhang et al., 2025), and score each step with a PRM. For PRM trained with step labels, the step score is the predicted probability of the *good* class; for our method, the score of every step is the single scalar output of the reward model, and the score calculation for implicit PRM is discussed in Section 2.1. We then aggregate step scores using the functions in Section 2.2 to obtain a response-level score and select the top-scoring response among the N candidates, matching prior work (Lightman et al., 2023). For outcome reward models, we rank candidates directly by the outcome score. We report Best-of-N accuracy, i.e., the average correctness of the selected

[1] https://github.com/huggingface/Math-Verify

[2] https://huggingface.co/datasets/AI-MO/aimo-validation-aime



(a) Inference with $\text{Agg}_{\min}$ (b) Inference with Agg_{prod} (c) Inference with Agg_{soft}

Figure 1: **Best-of-N Performance Comparsion.** Our method is trained with $A_\theta(\cdot; 1) = \text{Agg}_{\min}(\cdot)$ and $A_\theta(\cdot; 0) = \text{Agg}_{\text{prod}}(\cdot)$, outperforming baselines trained with outcome labels during inference.

response over all questions, and show performance as a function of N in Figure 1 with different score aggregation method during inference. Across all datasets and inference aggregations, our method consistently outperforms both the ORM and the implicit PRM, achieving the best results among approaches trained solely from outcome labels. On certain dataset–aggregator pairs (e.g., AIME 2024 with $\text{Agg}_{\min}$), our method is even comparable to or better than the step-supervised PRM, despite not using dense per-step correctness labels. This highlights that inexpensive final-answer supervision can sometimes match the benefits of costly step-level annotation. Moreover, the implicit PRM method is 21% slower than our method at inference, indicating that our approach is both efficient and effective. Additional results are provided in Appendix C.

Benchmark Evaluation. We further evaluate PRMs trained with different methods on ProcessBench (Zheng et al., 2024) and PRMBENCH (Song et al., 2025). ProcessBench uses naturally occurring model solutions with possible intermediate errors and measures whether a PRM can localize the *first* incorrect step across four math subsets. In contrast, PRMBENCH is a controlled stress test that starts from correct multi-step solutions and injects targeted step-level errors, reporting performance by error type (e.g., *simplicity*, *soundness*, and *sensitivity*). As shown in Table 1 and Table 2, our method consistently outperforms Implicit-PRM on all ProcessBench subsets and across PRMBENCH evaluations. Additional benchmark results are provided in Appendix D.

	Method		
Dataset	Ours	Step PRM	Implicit-PRM
GSM8K	0.5575	0.5221	0.4485
MATH	0.5127	0.5521	0.4653
Olympiad Bench	0.3880	0.5247	0.3392
Omnimath	0.4216	0.5109	0.3744
Average	**0.4699**	**0.5275**	**0.4069**

Table 1: ProcessBench Scores

	Method		
	Ours	Step PRM	Implicit-PRM
Score	0.525	0.582	0.457

Table 2: PRMBench Scores

4 Conclusion

We proposed a framework for learning *discriminative* process reward models (PRMs) from *only* final outcome labels. By treating step quality as latent and training through a weakest-link aggregator that emphasizes low-scoring steps, our method backpropagates outcome supervision into step-level signals without step annotations. Experiments show consistent gains over outcome-supervised baselines and competitive performance with step-supervised PRMs in some settings. Future work includes testing our approach on larger models and integrate our PRM training to reinforcement finetuning.

REFERENCES

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

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

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*, 2025.

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

Alon Jacovi, Yonatan Bitton, Bernd Bohnet, Jonathan Herzig, Or Honovich, Michael Tseng, Michael Collins, Roee Aharoni, and Mor Geva. A chain-of-thought is as strong as its weakest link: A benchmark for verifiers of reasoning chains. *arXiv preprint arXiv:2402.00559*, 2024.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Wendi Li and Yixuan Li. Process reward model with q-value rankings. *arXiv preprint arXiv:2410.11287*, 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.

Haolin Liu, Dian Yu, Sidi Lu, Yujun Zhou, Rui Liu, Zhenwen Liang, Haitao Mi, Chen-Yu Wei, and Dong Yu. Save the good prefix: Precise error penalization via process-supervised rl to enhance llm reasoning. *arXiv preprint arXiv:2601.18984*, 2026.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.

John L Pollock. Defeasible reasoning with variable degrees of justification. *Artificial intelligence*, 133(1-2):233–282, 2001.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models, 2020. URL https://arxiv.org/abs/1910.02054.

Noam Razin, Yong Lin, Jiarui Yao, and Sanjeev Arora. Why is your language model a poor implicit reward model? *arXiv preprint arXiv:2507.07981*, 2025.

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.

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.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL https://qwenlm.github.io/blog/qwq-32b/.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

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

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

Zhaohui Yang, Chenghua He, Xiaowen Shi, Linjing Li, Qiyue Yin, Shihong Deng, and Daxin Jiang. Beyond the first error: Process reward models for reflective mathematical reasoning. *arXiv preprint arXiv:2505.14391*, 2025.

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.

Kaiwen Zha, Zhengqi Gao, Maohao Shen, Zhang-Wei Hong, Duane S Boning, and Dina Katabi. Rl tango: Reinforcing generator and verifier together for language reasoning. *arXiv preprint arXiv:2505.15034*, 2025.

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.

Jiaru Zou, Ling Yang, Jingwen Gu, Jiahao Qiu, Ke Shen, Jingrui He, and Mengdi Wang. Reasonflux-prm: Trajectory-aware prms for long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2506.18896*, 2025.

A Why Is Asymmetric Training Effective?

Assume there exists a ground-truth reward function $r^\star(x, s) \in \{0, 1\}$ that accurately indicates whether step s is correct given input x. We also assume that the final answer is correct if and only if every reasoning step is correct, which implies $c = \min_m r^\star(x_m, s_m)$. The learner's objective is to find r_θ that approximates $r^\star$.

However, the learning problem exhibits *asymmetric information*: when $c = 1$, the learner has *full information*—we know with certainty that $r^\star(x_m, s_m) = 1$ for all steps m. In contrast, when $c = 0$, the learner has only *partial information*—we know that at least one step is incorrect (i.e., $\exists m : r^\star(x_m, s_m) = 0$), but we cannot identify which step(s) caused the error.

Prediction errors arise in two distinct ways:

1. **False negative**: $c = 1$ but the model's aggregated score is low
2. **False positive**: $c = 0$ but the model's aggregated score is high

Given the asymmetry, these two error types suggest different correction strategies:

False negative: When all steps are correct but the aggregated score is low, the bottleneck must be the step with the lowest predicted reward. Increasing $\text{Agg}_{\min}(\{r_\theta(x_m, s_m)\}_{m=1}^{M_o})$ directly addresses this bottleneck and corrects the underestimation.

False positive: When at least one step is incorrect but the aggregated score is high, there is uncertainty about which step caused the error. Decreasing only $\text{Agg}_{\min}(\{r_\theta(x_m, s_m)\}_{m=1}^{M_o})$ may miss the actual erroneous step, particularly if r_θ assigns a high score to that step. Decreasing the mean score across all steps ensures the incorrect steps are penalized regardless of their position in the sequence.

Our asymmetric training paradigm with $A_\theta(\cdot; 1) = \text{Agg}_{\min}(\cdot)$ and $A_\theta(\cdot; 0) = \text{Agg}_{\text{prod}}(\cdot)$ exactly corresponds to the above strategy.

B Additional Training Details

B.1 Dataset Collection Details

For each question in the dataset, 32 possible rollouts are sampled with the following system prompt and a max length of 4096 tokens. Responses that don't have an answer identifiable by a parser are filtered out. We only keep question response pairs where the correct percentage of the rollout model's responses were between 20 and 80 percent, in order to ensure a reasonable difficulty level. To verify model response correctness with the ground truth, we use the Math-Verify[3] library.

System Prompt to Generate Rollouts

Please reason step by step and put your answer within \boxed{}.

To collect dense process rewards a model response is split into multiple reasoning steps, with the delimiter token '`\n\n`'. The steps are then merged from the beginning of the response, keeping every step length under a max length parameter. The '`<extra_0>`' token is added to indicate the end of a reasoning step, and where a process reward should be collected from the model. A final '`<extra_0>`' token is always appended to the end of the response.

We then leverage the QwQ-32B (Team, 2025) to judge each of the trajectories and assign a label per reasoning step, following the judge prompt in (Lightman et al. (2023)). There are three possible labels: {-1, 0, 1}, corresponding to the {Bad, Okay and Good} labels. The judge prompt is provided below. Each training sample in the dataset will have two types of labels, a per-reasoning step label and a final answer correctness label.

Prompt For Large Models to Get Step labels

Overview:
Your goal is to grade the model's steps to a solution of a math problem. The model will often say things that look okay at first, but will turn out to be wrong on closer inspection - stay vigilant!
Please mark each step with "Great", "Okay" or "Bad". Your task ends here - there is no need for you to solve the problem to the end.
Return a list of grades, one for each step in the solution, and choose from the words: "Great", "Okay" or "Bad". Enclose your list of grades within \boxed{}.
EXAMPLE ANSWER USING THIS FORMAT: \boxed {"Great", "Great", "Great", "Okay", "Okay", "Bad"} for a question with six steps.
Rating Solution Steps: You will rate each solution step. A Great option is anything a good student of math would try. Most of the time it's a clear cut step forward towards solving the problem. But it could also be a sub-optimal choice, as long as it looks like something a reasonably smart human might say while trying to solve the problem. An Okay option is anything that's reasonable for a person to say, but it's not offering any insight, doesn't further the solution by exploring an option, performing a calculation, or offering an idea for the next step. A Bad option is one that confidently says something incorrect, is off-topic/weird, leads

[3] https://github.com/huggingface/Math-Verify

the solution into a clear dead-end, or is not explained clearly enough for a human to follow along with (even if it is correct).

Great Step Critiera:
A Great option is all of the following:
- Correct
- Nothing it states is wrong
- Verifiable
- It should take you no more than about 30 seconds to verify that the statement is correct (more if you are rusty with the problem's general area, like trigonometry or number theory - use your judgment)
- You might need to use a calculator to check some simple calculations, but ifyou feel like you need to take out paper and pencil to check that the suggestion is right, mark it Bad because it wasn't explained well enough
- You might also need to look up a theorem (e.g. a trig identity) in order to verify correctness. If you can do this with a quick Google search you can mark it Great, if it's more obscure than that mark it Bad
- Appropriate
- It fits correctly into the context of the previous steps - If the previous steps contain a mistake that wasn't noticed before, it's ok for a Great option to point out the mistake
- Insightful
- They are reasonable things a smart human might try while solving the problem
- Even if it's ultimately the wrong direction (but not immediately obviously a wrong direction), it can still be insightful to try. We want to teach the AI to learn how to recover from trying something that doesn't work out right away!
- If the option is simply restating one of the previous steps, without adding any additional insight or setting direction for what to do next, mark it Okay
- If the option is a statement of encouragement ('Good job!') but doesn't offer any additional insight or setting direction for what to do next, mark it Okay
- Sometimes the suggestion might add the tiniest amount of further insight or guides the solution forward in a particular direction only slightly - it can be marked Great or Okay depending on context or even other suggestions that you see
- If the option has a final answer, it should not only be correct, but also clearly follow from the rest of the reasoning. Don't select options with incorrect final answers!

Great Step Example:
Problem: Four fair coins are to be flipped. What is the probability that all four will be heads or all four will be tails? Express your answer as a common fraction
Step: For the probability of getting all heads, the probability of the first coin being heads is 1/2.
Rating: Great

Okay Step Criteria:
Okay options sound like something a person would say, they just don't contribute anything of essence to the conversation.
They're reasonable, verifiably correct, and appropriate, but they're also redundant, stalling, or just don't add any value. For example they might just repeat a fact or the problem itself, provide some encouragement without furthering the conversation ("Great job!"), complain that the problem is hard or say that it's easy, etc.
Another kind of correct statement that should be marked Okay is one that makes progress along a direction, but it's stalling on making a more decisive amount of progress. For example if the problem is to find the last digit of 2^{10000}, the first couple of steps that look like
$2^1 = 2$, ends in 2
$2^2 = 4$, ends in 4
$2^3 = 8$, ends in 8,
$2^4 = 16$, ends in 6,
$2^5 = 32$, ends in 2,

$2^6 = 64$, ends in 4,
$2^7 = 128$, ends in 8,
$2^8 = 256$, ends in 6,
$2^7 = 512$, ends in 2, can all be marked Great because they are contributing to our understanding of the problem, but if this continues for too long, eventually it's just stalling on making the critical observation that there is a pattern in the last digits.
So at some point (which might reasonably be anywhere between 2^6 and 2^{10}) please stop marking the suggestions as Great and instead mark it as Okay.

Bad Step Criteria:
Any of these characteristics will make an option Bad:
- Hard to verify
- It's not explained well and you'd need to use paper and pencil to check that it's correct
- Wrong
- Even if most of the suggestion is correct, but it also states something that is wrong, mark it Bad
- Contains gibberish
- Contains off-topic text or non-sequiturs
- Suggests attempting something that is unreasonable for this problem
- Derails the conversation
- Leads the solution into an immediately obvious dead end or makes it go in circles
- Leads the solution into a repetitious pattern that should obviously be stopped
- Refers to an external link that it claims the solution relies on (e.g. a link to a graph or image)
- please don't click on computer-generated external links.
- Refers to a graph or picture that is not included, and the solution relies on it (i.e. it's hard to imagine what the graph or picture is supposed to be unambiguously)

Here is a sample question and examples of steps and their labels:
Question: If $f(x) = 3x + 1$, what is f(3)?
Bad Step: So the function f takes a number x and does three things to it: It first multiplies it by 3, then adds 1.
This step is bad because it is incorrect (it says f does 3 things to x but only lists 2).
Great Step: Let's plug 3 in for x in the function. So we get $f(3) = 3(3) + 1$. This step is Great because it moves the problem forward.

Okay Step: What's $f(3)$? This step is Okay because it is correct but does not move us forward.
Remember, return a list of grades, one for each step in the solution, and choose from the words: "Great", "Okay" or "Bad". Enclose your list of grades within \boxed{}.

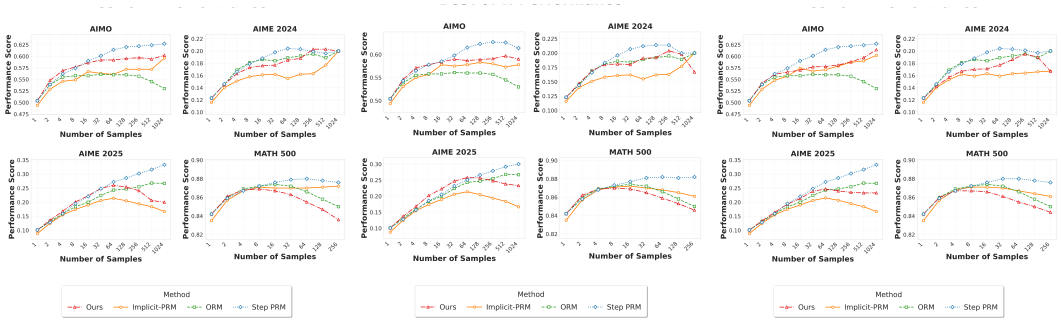
EXAMPLE ANSWER USING THIS FORMAT: \boxed {"Great", "Great", "Great", "Okay", "Okay", "Bad" }, for a question with six steps. Please label each of the following reasoning steps:
Question: question
Steps: steps
Grade:

B.2 Training Setup and Hyperparameters

All models are obtained by finetuning the Qwen2.5-Math-1.5B-Instruct model on our curated dataset. We use the DeepSpeed training framework Rajbhandari et al. (2020), and the Adam optimizer Kingma & Ba (2017) with a learning rate of 1×10^{-5}. The learning rate is warmed up over the first 1000 steps. We use ZeRo Stage 2 for all models, with a batch size of 32, across 4 H100 SXM5 96GB GPUs.

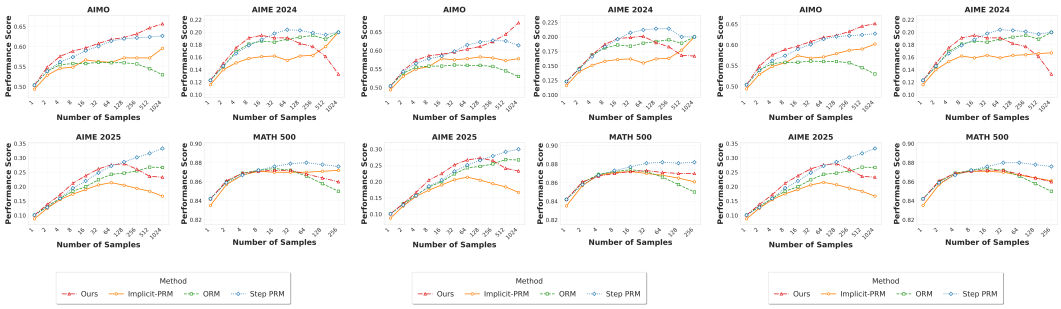
C Additional Best-of-N Plots and Analysis

We provide Best-of-N evaluation for our methods trained with different aggregators in this Figure 2, Figure 3 and Figure 4. Among these evaluations, implicit PRM does not perform well. A possible reason for Implicit-PRM's worse performance is that it uses the cumulative log probability ratio over the response to calculate reward: $\sum_i \frac{\pi_\theta(o_i|q,o_{<i})}{\pi_{\text{ref}}(o_i|q,o_{<i})}$. This quantity scales with sequence length, longer outputs can accumulate more error and exhibit greater variance, which may hurt performance. The authors mitigate this by multiplying the accumulative reward of a sequence by a constant coefficient of 0.001, but this is a heuristic and not a principled mitigation. As seen in Table 3, the AIME datasets have the longest response lengths. Consequently, we see the largest difference in performance for those datasets in the Best-of-N plots.



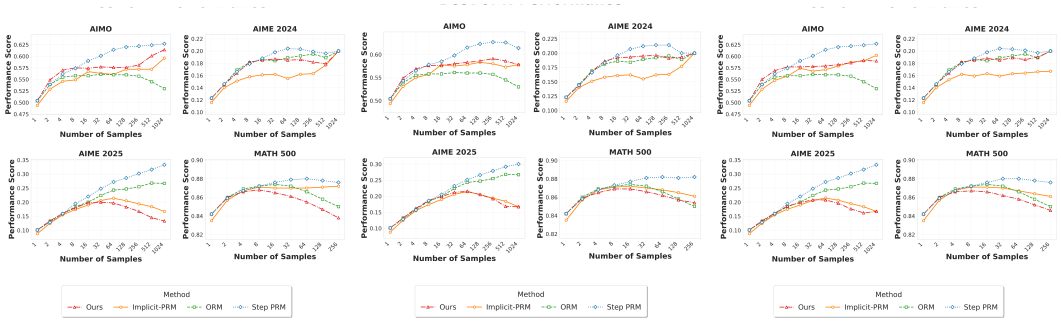
(a) Inference With $\text{Agg}_{\min}$ (b) Inference with Agg_{prod} (c) Inference with Agg_{soft}

Figure 2: **Best-of-N Performance Comparsion.** Our method is trained with $A_\theta(\cdot;1) = \text{Agg}_{\text{prod}}(\cdot)$ and $A_\theta(\cdot;0) = \text{Agg}_{\text{prod}}(\cdot)$.



(a) Inference With $\text{Agg}_{\min}$ (b) Inference with Agg_{prod} (c) Inference with Agg_{soft}

Figure 3: **Best-of-N Performance Comparsion.** Our method is trained with $A_\theta(\cdot;1) = \text{Agg}_{\min}(\cdot)$ and $A_\theta(\cdot;0) = \text{Agg}_{\min}(\cdot)$.



(a) Inference With $\text{Agg}_{\min}$ (b) Inference with Agg_{prod} (c) Inference with Agg_{soft}

Figure 4: **Best-of-N Performance Comparsion.** Our method is trained with $A_\theta(\cdot;1) = \text{Agg}_{\text{soft}}(\cdot)$ and $A_\theta(\cdot;0) = \text{Agg}_{\text{soft}}(\cdot)$.

Dataset	AIME 2024	AIME 2025	AIMO	MATH 500
Average length	3265	3520	2700	1900

Table 3: Average Input Length Over Questions

The inference time for Best-of-1024 of our method and implicit PRM is shown in Table 4, indicating that our method is both faster and more effective.

	Method	
Dataset	Ours	Implicit-PRM
AIME 2024	17.9	21.2
AIME 2025	16.9	20.1
AIMO	33.5	40.71
MATH	26.9	33.4

Table 4: Inference Times on Best-of-1024 (minutes)

D Additional Benchmark Scores

In this section, we provide benchmark scores for our method trained with different aggregators. We consider symmetric aggregator in this section such that $\mathrm{Agg}(\cdot; 0) = \mathrm{Agg}(\cdot; 1)$. Since the benchmarks aim to identify the first incorrect step while PRMs output scalar step scores, we must choose a threshold below which a step is classified as erroneous. We follow the method in the Implicit PRM codebase (Yuan et al., 2024), which searches over 10000 possible thresholds between 0 and 1, and then chooses the best score per subset.

	Training Aggregation		
Dataset	$\mathrm{Agg}_{\mathrm{min}}$	$\mathrm{Agg}_{\mathrm{soft}}$	$\mathrm{Agg}_{\mathrm{prod}}$
GSM8K	0.5774	0.5740	0.5691
MATH	0.5607	0.5459	0.5495
Olympiad Bench	0.4490	0.4534	0.4618
Omnimath	0.4478	0.4409	0.4445
Average	**0.5088**	**0.5036**	**0.5062**

Table 5: ProcessBench Results for Different Symmetric Aggregators

	Training Aggregation		
	$\mathrm{Agg}_{\mathrm{min}}$	$\mathrm{Agg}_{\mathrm{soft}}$	$\mathrm{Agg}_{\mathrm{prod}}$
Score	0.546	0.545	0.5305

Table 6: PRMBench Results for Different Symmetric Aggregators