Error Typing for Smarter Rewards: Improving Process Reward Models with Error-Aware Hierarchical Supervision

Tej Deep Pala[1], **Panshul Sharma**[1]
Amir Zadeh[2], **Chuan Li**[2], **Soujanya Poria**[1]

[1]Nanyang Technological University
[2]Lambda Labs

Abstract

Large Language Models (LLMs) are prone to hallucination, especially during multi-hop and reasoning-intensive tasks such as mathematical problem solving. While Outcome Reward Models verify only final answers, Process Reward Models (PRMs) score each intermediate step to steer generation toward coherent solutions. We introduce `PathFinder-PRM`, a novel hierarchical, error-aware discriminative PRM that first classifies math and consistency errors at each step, then combines these fine-grained signals to estimate step correctness. To train `PathFinder-PRM`, we construct a 400K-sample dataset by enriching the human-annotated PRM800K corpus and RLHFlow Mistral traces with three-dimensional step-level labels. On PRMBench, `PathFinder-PRM` achieves a new state-of-the-art PRMScore of 67.7, outperforming the prior best (65.5) while using 3× less data. When applied to reward guided greedy search, our model yields prm@8 48.3, a +1.5 point gain over the strongest baseline. These results demonstrate that decoupled error detection and reward estimation not only boost fine-grained error detection but also substantially improve end-to-end, reward-guided mathematical reasoning with greater data efficiency. [1].

1 Introduction

Large language models (LLMs) have achieved remarkable success on many natural language tasks, including open-ended generation and complex reasoning (Brown et al., 2020; Wei et al., 2022). However, they remain prone to *hallucinations* and subtle logical errors when generating multi-step solutions, particularly in domains such as mathematical problem solving (Wang et al., 2023; Zheng et al., 2024). Traditional outcome-only verifiers (Outcome Reward Models) can check a final answer but fail to catch intermediate missteps that lead reasoning astray (Wang et al., 2024b).

To address this gap, *Process Reward Models* (PRMs) have been proposed, which assign individual rewards to each reasoning step. (Uesato et al., 2022; Liu et al., 2025). As such, PRMs can filter out erroneous chains of thought and guide generation toward more reliable reasoning trajectories (Lightman et al., 2023; Zhang et al., 2025).

Recent interest in explicitly reasoning-centric LLMs such as DeepSeek-R1 and OpenAI's GPT-o series models underscores the field's growing emphasis on human-like thinking and the ability to flexibly scale test-time compute (Guo et al., 2025; OpenAI, 2025). These models demonstrate extended deliberation and use structured reasoning traces to solve complex problems. In such settings, effective *process supervision* is crucial: rather than merely verifying a final answer, it must guide and correct the reasoning process at every step, ensuring logical coherence and factual accuracy throughout. PRMs are thus an essential component in aligning reasoning LLMs with reliable multi-step reasoning.

Despite recent advances, current PRMs still struggle with fine-grained error types. For example, the PRMBench benchmark reveals that many state-of-the-art PRMs fall short of detecting subtler faults such as non-redundancy violations, domain inconsistencies, or deceptive logical steps (Song et al., 2025). Moreover, existing methods typically combine *error detection* (is this step wrong?) with *path optimality* (how helpful is this step in reaching the solution?) in a single prediction, leaving each signal underutilized (Zhang et al., 2025; Xia et al., 2025).

In this work, we argue that error detection and value estimation are complementary but distinct objectives. By decoupling them into two sequential subtasks, first explicitly identifying specific error categories and using those error signals to

[1]Our code can be found at `https://github.com/declare-lab/PathFinder-PRM`

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 14940–14954
November 4-9, 2025

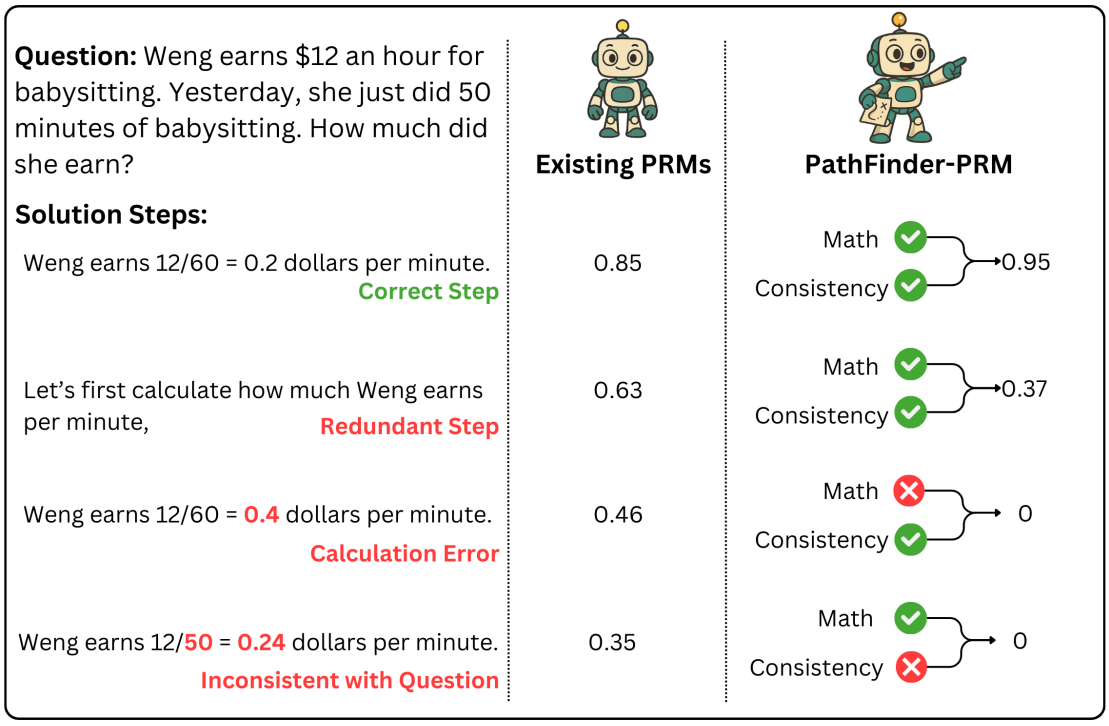

Figure 1: Comparing the Methodology of Existing PRMs against `PathFinder-PRM`.

compute a step-level reward, we can obtain richer supervision and stronger guidance for downstream generation (depicted in Figure 1). To this end, we introduce `PathFinder-PRM`, an error-aware hierarchical PRM that (1) classifies each step to detect the presence of math or consistency errors, and (2) combines these fine-grained error labels to produce a final reward score.

We construct a new training corpus by augmenting the human-annotated *PRM800K* (Lightman et al., 2023) and automated RLHFlow Mistral data (Xiong et al., 2024) with our three-dimensional labels, yielding around 400K richly annotated reasoning trajectories.

Our experiments on ProcessBench (Zheng et al., 2024), PRMBench (Song et al., 2025), and a suite of end-to-end math benchmarks show that `PathFinder-PRM` not only establishes the new state of the art among PRMs trained on PRM800K-only data but also continues to scale gracefully when incorporating additional automated annotations. We summarize our main contributions:

- We propose `PathFinder-PRM`, the first hierarchical PRM that explicitly types errors into *math* and *consistency* categories before estimating step optimality.
- We curate a multi-source dataset of 400K mathematical reasoning trajectories with three-dimensional step-level labels, combining PRM800K and RLHFlow data under our unified schema.
- We demonstrate that error-aware hierarchical supervision yields substantial gains on ProcessBench and PRMBench, and leads to more accurate and robust end-to-end math problem solving under reward-guided search.

2 Related Work

Process Reward Models: Process reward models (PRMs) evaluate the quality of intermediate steps in reasoning processes. PRMs are a core component of test-time scaling, enabling small policy models to outperform much larger models on reasoning tasks by reinforcing productive reasoning pathways (Liu et al., 2025). Most recent work on PRMs has varied along two primary axes: (1) the choice of the base model architecture and (2) the methods used to synthesize step-level supervision labels (Wang et al., 2024b; Zhang et al., 2025; Khalifa et al., 2025).

Discriminative PRMs are designed to classify the correctness of individual process steps within reasoning trajectories (Uesato et al., 2022). Cor-

responding training data synthesis approaches include gold human step-level annotations (Lightman et al., 2023), Monte Carlo estimation (Wang et al., 2024b; Luo et al., 2024), and consensus filtering methods (Zhang et al., 2025). FG-PRM (Li et al., 2024) introduced a fine-grained system of six separate discriminative PRMs, each specialized to detect a different type of hallucination or reasoning error.

Generative PRMs scale verifier compute, utilizing the language modeling head of the PRMs to generate a chain of thought (CoT) before producing step-level correctness classifications. Associated training data is typically obtained by filtering LLM-as-a-judge reasoning traces against human step-level labels (Khalifa et al., 2025; She et al., 2025) or relative progress estimation (Zhao et al., 2025).

Limitations of Current PRMs: Current PRMs face significant challenges in detecting nuanced error types. Although frontier models excel at identifying obvious mistakes (Zheng et al., 2024), their performance deteriorates markedly when confronted with more subtle error types. The recently introduced PRMBench benchmark demonstrates this limitation, revealing substantial performance drops across fine-grained error categories such as redundancy, circular logic, step inconsistency and domain inconsistency (Song et al., 2025).

3 Methodology

3.1 Process Reward Modeling

The primary task of a Process Reward Model in mathematical problem solving is to evaluate the correctness of each intermediate reasoning step generated by an LLM. Unlike ORMs, which assess only the final answer, PRMs evaluate at the step level, enabling fine-grained supervision and improved interpretability.

Formally, given a multi-step solution $\mathcal{S} = \{s_1, s_2, \ldots, s_T\}$ produced by a language model for a math problem Q, the PRM assigns a scalar reward $r_t \in \mathbb{R}$ to each step s_t, reflecting its mathematical and logical correctness. These rewards serve two main purposes:

1. **Error Detection:** By evaluating each step individually, PRMs can identify inaccurate or hallucinated reasoning in the solution process, even when the final answer may coincidentally be correct.
2. **Guidance Toward Correct Solutions:** The step-level feedback provided by PRMs can be used to steer generation policies, in reinforcement learning or reward-guided generation, towards valid and logically coherent trajectories, thereby improving the overall quality of generated solutions.

Process Reward Models enhance the robustness of mathematical reasoning systems by aligning training and inference with process-based correctness, rather than relying only on outcome validation.

3.2 PathFinder-PRM

Existing Process Reward Models tackle both error detection and optimal path guidance jointly. Given a mathematical problem Q and a sequence of solution steps generated by a policy model, $\mathcal{S} = \{s_1, s_2, \ldots, s_T\}$, these models assign each step a reward score conditioned on the question and all previous steps:

$$R_t = PRM\big(Q, \mathcal{S}_{<t}, s_t\big).$$

where $\mathcal{S}_{<t} = \{s_1, s_2, \ldots, s_{t-1}\}$ The resulting score implicitly reflects both the presence of errors in s_t and its contribution towards a correct solution.

In contrast, our approach takes a hierarchical perspective by decomposing the reward assignment into two sequential subtasks: (a) detecting errors in each reasoning step, and (b) using the error information to inform step optimality. Specifically, we first categorize errors into two types:

1. **Math Errors:** mistakes in arithmetic or algebraic manipulation, incorrect formula application, or invalid implications.
2. **Consistency Errors:** logical inconsistencies with the question, prior steps, or established constraints.

As shown in Figure 1, PathFinder-PRM performs two forward passes per step. In the first pass, it predicts the probabilities of the Math error, M_t, and Consistency error, C_t, using masked token prediction. In the second pass, these predictions are reinserted into the input, and the model estimates the step reward R_t, explicitly conditioned on the detected error labels:

$$M_t, C_t = \texttt{PathFinder-PRM}\big(Q, \mathcal{S}_{<t}, s_t\big),$$
$$R_t = \texttt{PathFinder-PRM}\big(Q, \mathcal{S}_{<t}, s_t, M_t, C_t\big).$$

By structuring reward modeling hierarchically, PathFinder-PRM leverages fine-grained error signals to improve reward estimation, while maintaining a clear separation between error detection and correct path guidance.

To investigate this, we propose the following hypothesis:

Hypothesis: A hierarchical supervision strategy—first detecting error types, then using them to compute rewards—is more effective than existing methods that compute rewards directly without identifying the presence of errors explicitly.

3.3 Inference Design

To realize this hierarchical design, we adopt a two-forward-pass approach rather than autoregressive decoding. This prevents cascading errors between dependent predictions: Math and Consistency labels are predicted independently in the first pass, avoiding the risk of one label influencing the other. In the second pass, these predicted labels condition the Correctness reward, yielding modular and interpretable supervision. Detailed inference steps and algorithmic description are provided in Appendix C.

3.4 Creating the PathFinder-PRM Dataset

The PathFinder-PRM dataset includes step-level fine-grained labels across three categories: (1) *mathematical reasoning accuracy*, (2) *consistency* with prior steps and mathematical domain, and (3) *step correctness*. Here, the third category *step correctness* identifies whether a step is both error-free and optimally contributes to solving the problem. For each process step, we assign a three-dimensional categorical score vector $\mathbf{c}_t = (c_t^{\text{math}}, c_t^{\text{consistency}}, c_t^{\text{correctness}})$, where each component $c_t^{(i)} \in \{0, 1\}$ represents a binary label for the respective category. We construct this dataset by leveraging two existing datasets: the PRM800k (Lightman et al., 2023) and Mistral-PRM-Data by RLHFlow (Xiong et al., 2024). These datasets contain step-level correctness annotations generated through human evaluation and Monte Carlo estimation, respectively.

PRM800K Integration: The original PRM800k dataset contains over 800,000 gold step-level correctness labels $l_t \in \{-1, 0, 1\}$ for each reasoning step s_t, where -1 corresponds to an incorrect step, 0 corresponds to a correct but suboptimal step, and 1 corresponds to a correct and optimal step. We transform each correctness label into our categorical score vector as follows:

- Steps with $l_t = 1$ (correct and optimal) are mapped to $\mathbf{c}_t = (1, 1, 1)$.
- Steps with $l_t = 0$ (correct but suboptimal) are mapped to $\mathbf{c}_t = (1, 1, 0)$.

This mapping reflects our interpretation that human labels $l_t \in \{0, 1\}$ indicate error-free reasoning, with $l_t = 0$ specifically denoting non-optimal process steps. For erroneous steps ($l_t = -1$), the original correctness labels provide insufficient information to determine scores across our three evaluation categories. Therefore, we employ DeepSeek-R1-Distill-Qwen-32 B to generate binary labels for each category for these steps (DeepSeek-AI, 2025). To maintain dataset quality, we subsequently filter out samples with categorical score vectors that are inconsistent with -1 human annotated labels (i.e., $\mathbf{c}_t = (1, 1, 1)$).

Mistral-PRM-Data Integration: Since this dataset lacks gold standard step-level correctness labels, we utilize DeepSeek-R1-Distill-Qwen-32B to assign binary categorical labels to a small, randomly selected subset of process steps. To ensure data quality, we implement a consistency filtering mechanism. This removes score assignments that are logically incompatible with the existing Monte Carlo (MC) estimation labels. Specifically:

- For steps with MC estimation '+' labels (indicating positive assessment), we retain only samples with assignments of $\mathbf{c}_t = (1, 1, 1)$.
- For steps with MC estimation '-' labels (indicating negative assessment), we retain samples with categorizations where at least one component equals 0, i.e., $\mathbf{c}_t \neq (1, 1, 1)$.

In totality, the PathFinder-PRM dataset contains about 400K reasoning trajectory samples with step-level categorical score vectors $\mathbf{c}_t$. Of these 400K trajectories, approximately 345K are sourced from PRM800k and the other 55K reasoning paths are sourced from Mistral-PRM-Data. We train two variants of the model, PathFinder-PRM-7B and PathFinder-PRM-7B-PRM800k trained on the full dataset and just the PRM800K subset respectively.

3.5 Training Recipe for PathFinder-PRM

Previous studies demonstrate that a model's mathematical reasoning ability correlates with its performance as a process reward model (Xia et al., 2025).

Consequently, we initialize PathFinder-PRM from Qwen2.5-Math-7B-Instruct, which achieves state-of-the-art results on multiple math benchmarks (Yang et al., 2024). Unlike recent PRMs that swap the language modeling head for a scalar value head (Zhang et al., 2025; Xia et al., 2025; Tan et al., 2025), we preserve the original LM architecture and extend the tokenizer with two special tokens, <+> and <->, to represent positive and negative step labels.

Training Objective Each training example is structured in two parts, mirroring the inference passes:

1. *Error Detection Target:*

   Prompt + Math: <+>/<->, Consistency: <+>/<->

2. *Reward Estimation Target:* Append the predicted error labels and the token

   Prompt + Math: [Math label], Consistency: [Consistency Label] + Correctness: <+>/<->

For each sample, we compute the cross-entropy loss only on these label tokens.

4 Experimental Setup

4.1 Evaluation Benchmarks

For math steps error detection, we use ProcessBench and PRMBench. **ProcessBench** is a benchmark designed to evaluate language models' ability to identify errors in mathematical reasoning processes. It comprises 3,400 test cases, primarily sourced from different math reasoning benchmarks. Each case includes a step-by-step solution annotated by human experts to indicate the earliest step containing an error or to confirm the correctness of all steps. Models are tasked with pinpointing the first erroneous step in a solution or affirming the solution's correctness. **PRMBench** is a fine-grained benchmark aimed at evaluating Process-Level Reward Models (PRMs) on their capability to detect nuanced errors in reasoning steps. It consists of 6,216 problems with a total of 83,456 step-level labels, assessing models across multiple dimensions: Simplicity (non-redundancy, non-circular logic), Soundness (empirical soundness, step consistency, domain consistency, confidence invariance), and Sensitivity (prerequisite sensitivity, deception resistance, multi-solution consistency). The benchmark uses both synthetic and human-verified data, with rigorous quality control measures, including manual verification of a subset of data. A composite metric, PRMScore, is introduced, combining positive and negative F1 scores for a balanced evaluation.

To evaluate the effectiveness of PathFinder-PRM in guiding step-by-step mathematical problem-solving, we employ it to assign scores to individual reasoning steps generated by large language models (LLMs), selecting only those with the highest overall rewards to build upon. This evaluation is conducted across several widely recognized math reasoning benchmarks, including AIME24, AMC23, MATH, Olympiad Bench, College MATH, and Minerva MATH[2].

4.2 Baselines

Our evaluation utilizes a diverse set of discriminative process reward models from recent literature as baselines: *Math-Shepherd* (Wang et al., 2024b), *Math-PSA* (Wang et al., 2024a), *RLHFlow-Mistral and RLHFlow-DeepSeek* (Xiong et al., 2024), *Skywork-PRM-7b* (o1 Team, 2024), *ReasonEval-7B* (Xia et al., 2025), *Llemma-PRM800k-7B* (Sun et al., 2024), *Qwen2.5-Math-PRM-7B* and *Qwen2.5-Math-7B-PRM800K* (Zhang et al., 2025). We selected these baselines to cover a diverse range of training regimes, including models trained on human annotations, automated annotations, and hybrid approaches, as well as varying scales of training data.

5 Results

5.1 Main Results

PathFinder-PRM-7B is the SOTA for PRMBench: Table 1 shows the PRMBench results of the selected baselines, PathFinder-PRM-variants as well as LLM-as-judge performance of strong open-source and proprietary LLMs. In the discriminative PRM category, PathFinder-PRM-7B achieves the highest overall PRM score (67.7), outperforming Qwen2.5-Math-PRM-7B (65.5) and ReasonEval-7B (60.0). The variant PathFinder-PRM-7B-PRM800K, trained on a fraction of our dataset, achieves a competitive score of 65.0. Notably, PathFinder-PRM-7B outperforms nearly all LLM-as-Judge models,

[2]Following (She et al., 2025), we use their subset of 200 test samples for Olympiad Bench and College MATH.

Model	Simplicity			Soundness					Sensitivity				Overall
	NR.	NCL.	Avg.	ES.	SC.	DC.	CI.	Avg.	PS.	DR.	MS.	Avg.	
LLM-as-judge, Open-source Language Models													
Qwen-2.5-Math-72B*	55.3	54.9	55.1	55.5	71.6	58.1	59.1	61.1	47.4	53.8	**100.0**	67.1	57.4
QwQ-Preview-32B*	**57.2**	**55.6**	**56.4**	**67.4**	**72.3**	**66.2**	**66.9**	**68.2**	**57.8**	**62.7**	**100.0**	**73.5**	**63.6**
LLM-as-judge, Proprietary Language Models													
GPT-4o*	57.0	62.4	59.7	72.0	69.7	70.7	71.1	70.9	**62.5**	**65.7**	99.2	**75.8**	66.8
Gemini-2.0-flash-exp*	67.2	58.1	62.7	70.4	65.7	66.0	67.3	67.3	61.8	66.2	98.2	75.4	66.0
Gemini-2.0-thinking-exp-1219*	**68.5**	**63.8**	**66.2**	**72.9**	**71.3**	**71.0**	**71.8**	**71.8**	60.3	**65.7**	**99.8**	75.3	**68.8**
Discriminative Process Reward Models													
Math-Shepherd-7B*	44.0	50.3	47.1	49.4	44.5	41.3	47.7	45.7	47.2	48.6	86.1	60.7	47.0
Math-PSA-7B†	47.6	55.1	51.3	56.5	49.4	47.1	54.2	51.8	51.7	54.1	88.9	64.9	52.3
RLHFlow-Mistral-8B*	46.1	47.3	46.7	56.6	55.1	54.4	63.8	57.5	51.5	56.2	97.9	68.5	54.4
RLHFlow-DeepSeek-8B*	46.4	48.9	47.6	55.7	55.0	53.2	66.2	57.5	49.0	55.4	**99.8**	68.1	54.2
Lemma-PRM800k-7B*	49.3	53.4	51.4	56.4	47.1	46.7	53.3	50.9	51.0	53.5	93.6	66.0	52.0
Skywork-PRM-7B*	35.7	41.2	38.4	36.7	29.1	30.6	34.4	32.7	36.8	37.4	88.8	54.3	36.2
ReasonEval-7B*	**61.0**	50.1	55.5	62.1	65.9	61.5	66.0	63.9	55.6	58.0	99.5	71.0	60.0
Qwen2.5-Math-7B-PRM800K†	48.6	47.8	48.2	62.1	59.4	58.7	68.5	62.2	52.9	64.0	**99.8**	72.2	58.3
Qwen2.5-Math-PRM-7B*	49.0	55.1	52.1	71.8	67.3	66.3	**78.5**	71.0	57.6	69.1	99.7	75.5	65.5
⊛ PathFinder-PRM-7B-PRM800K	51.5	61.3	56.4	69.7	67.6	65.9	71.9	68.8	58.7	66.6	99.4	74.9	65.0
w/o Separate Subtask Prediction	58.9	**66.7**	**62.8**	68.6	62.4	62.4	66.7	65.0	60.2	64.9	97.8	74.3	64.4
⊛ PathFinder-PRM-7B	52.1	65.8	58.9	73.1	68.7	66.3	75.0	70.8	**61.7**	**69.8**	99.2	**76.9**	**67.7**
w/o Separate Error Categories	51.7	62.0	56.9	**73.2**	**70.0**	**66.9**	75.8	**71.5**	60.3	69.2	99.6	76.4	67.3
w/o Separate Subtask Prediction	57.9	66.4	62.1	69.1	62.6	62.2	68.7	65.7	61.0	65.4	98.2	74.9	64.9

Table 1: Performance on **PRMBench**. Results marked with * and †come from Song et al. and She et al. respectively. Bold text denotes the best results within each category. ⊛ represents the models we trained.

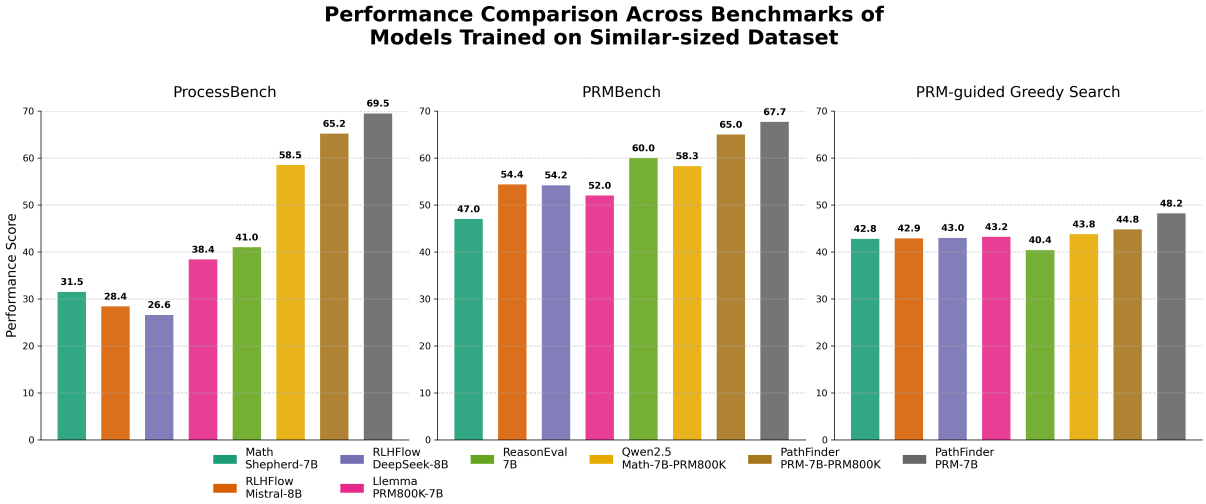

Figure 2: Performance comparison of language models across three benchmarks. The PathFinder-PRM-7B model (gray) shows the highest performance across all benchmarks.

including GPT-4o, QwQ-Preview-32B and Gemini-2.0-flash-exp. PRMBench is a benchmark designed to test a model's ability to detect subtle and complex errors. Our results affirm that our hierarchical PRM approach enables the model to detect these nuanced errors, leading to stronger process-level understanding and supervision.

PathFinder-PRM Excels on ProcessBench: Table 2 presents F1 results on ProcessBench. When trained exclusively on PRM800K, PathFinder-PRM-7B-PRM800K attains an average F1 of 65.2, beating the previous best (Qwen2.5-Math-7B-PRM800K, 58.5) by 6.7 points and outperforming all other PRM800K-only baselines across every category: GSM8K (+5.9),

Model	# Samples	GSM8K	MATH	Olympiad Bench	OmniMath	Avg. F1
Trained on Automated Annotation Data						
Math-Shepherd-7B*	445K	47.9	29.5	24.8	23.8	31.5
RLHFlow-Mistral-8B*	273K	50.4	33.4	13.8	15.8	28.4
RLHFlow-DeepSeek-8B*	253K	38.8	33.8	16.9	16.9	26.6
Qwen2.5-Math-PRM-7B*	∼1.5M	**82.4**	**77.6**	**67.5**	**66.3**	**73.5**
Trained on Human Annotated Data (PRM800K)						
Llemma-PRM800K-7B	∼350K	48.4	43.1	28.5	33.4	38.4
ReasonEval-7B†	∼350K	41.0	48.9	36.7	37.4	41.0
Qwen2.5-Math-7B-PRM800K*	264K	68.2	62.6	50.7	44.3	58.5
PathFinder-PRM-7B-PRM800K	∼350K	<u>74.1</u>	<u>71.3</u>	58.7	56.6	<u>65.2</u>
w/o Separate Subtask Prediction		71.4	71.1	<u>59.2</u>	<u>58.0</u>	64.9
Trained on a Mix of Human and Automated Annotation Data						
Math-PSA-7B†	∼860K	62.4	41.9	31.5	25.2	40.3
Skywork-PRM-7B*	unk	70.8	53.6	22.9	21.0	42.1
⊛ PathFinder-PRM-7B	∼400K	<u>77.9</u>	<u>75.3</u>	<u>65.0</u>	<u>59.7</u>	<u>69.5</u>
w/o Separate Error Categories		76.1	73.8	61.4	56.6	67.0
w/o Separate Subtask Prediction		73.9	72.6	63.9	59.9	67.6

Table 2: Performance (F1) on ProcessBench. Results marked with * come from Zhang et al..The best performance across all categories is in **bold** and the best performance within a category is <u>underlined</u>.# Samples denotes the number of training samples used by each model.

Setting	AIME24	AMC23	MATH	Olympiad Bench	College MATH	Minerva MATH	Avg.
pass@1*	11.2	47.8	73.0	38.0	38.6	37.2	41.0
major@8*	20.0	57.5	79.6	47.0	41.5	42.7	48.0
pass@8*	33.3	82.5	88.8	58.5	47.5	57.7	61.4
Reward Guided Search (prm@8)							
Math-Shepherd-7B*	13.3	52.5	74.6	38.5	36.5	41.2	42.8
Math-PSA-7B*	6.7	57.5	79.8	42.5	41.0	39.3	44.5
RLHFlow-PRM-Mistral-8B*	10.0	57.5	73.4	37.5	38.0	41.2	42.9
RLHFlow-PRM-DeepSeek-8B*	13.3	52.5	74.8	39.5	37.0	40.8	43.0
Lemma-PRM800k-7B*	13.3	57.5	73.8	40.0	36.5	38.2	43.2
Skywork-PRM-7B*	10.0	57.5	77.8	41.5	39.0	43.4	44.9
ReasonEval-7B*	3.3	55.0	73.0	37.5	35.5	37.9	40.4
Qwen2.5-Math-7B-PRM800K*	**23.3**	45.0	78.2	42.0	35.5	38.6	43.8
Qwen2.5-Math-PRM-7B*	16.7	60.0	**81.0**	43.5	39.0	40.4	46.8
⊛ PathFinder-PRM-7B-PRM800K	20.0	55.0	79.0	36.0	55.0	36.4	46.9
w/o Separate Subtask Prediction	6.6	55.0	82.2	36.0	53.5	36.0	45.0
⊛ PathFinder-PRM-7B	20	62.5	78.8	36.5	55.0	36.7	**48.3**
w/o Separate Error Categories	13.3	52.5	80.4	35.5	53.5	37.5	45.4
w/o Separate Subtask Prediction	10.0	55.0	81.6	37.0	53.5	36.0	45.5

Table 3: The performance of PRM guided greedy search with Qwen2.5-7B-Instruct as the policy model. Results marked with * come from She et al.

MATH (+8.7), Olympiad Bench (+8.0) and OmniMath (+12.3).

Leveraging a larger, mixed human + auto-annotated dataset further boosts performance. PathFinder-PRM-7B achieves an average F1 of 69.5, setting new state–of–the–art among mixed-data models and closing the gap to the top automated-annotation model (Qwen2.5-Math-PRM-7B*, 73.5) to just 4 points. Notably, PathFinder-PRM-7B also leads in every individual benchmark—GSM8K (77.9), MATH (75.3), Olympiad Bench (65.0), and OmniMath (59.7), demonstrating the scalability and robustness of our hierarchical reward modeling approach.

Improved Reward-Guided Search with Better PRMs: Finally, we assess the utility of our PRM in guiding solution search. Using Qwen2.5-Instruct-7B as a generator and ranking sampled steps in completions using our PRM, Table 3 shows that PathFinder-PRM-7B yields the highest average prm@8 score (48.3), outperforming Qwen2.5-Math-PRM-7B (46.8). The advantage holds across tasks, including challenging subsets

such as AIME24 and College MATH, indicating better inductive bias and alignment with ground-truth solution quality.

`PathFinder-PRM` is Competitive to Qwen2.5-Math-PRM-7B Despite Using ∼3× Less Data: Although Qwen2.5-Math-PRM-7B was trained on roughly 1.5M automated annotations, our `PathFinder-PRM-7B`, trained on only ∼400K samples, matches or exceeds its performance in key benchmarks and reward-guided search. On ProcessBench, `PathFinder-PRM-7B` performs competitively to Qwen2.5-Math-PRM-7B in average F1 69.5 vs 73.5. More importantly, `PathFinder-PRM-7B` surpasses Qwen2.5-Math-PRM-7B on PRMBench overall (67.7 vs 65.5), and drives higher pass@8 in reward-guided greedy search (48.3 vs. 46.8). This demonstrates that our hierarchical, error-aware training yields more data-efficient and robust PRMs, achieving superior process supervision with far fewer samples.

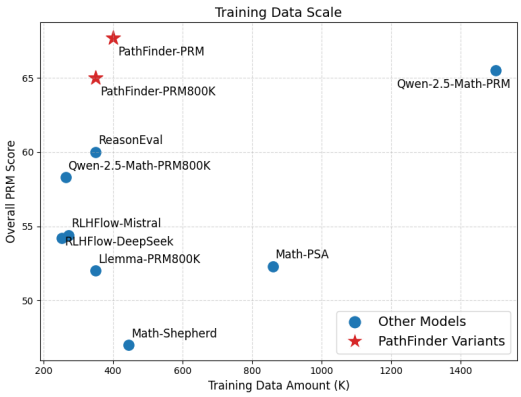

Figure 3: PRMBench Overall PRMscore against the data scales of different baselines and `PathFinder-PRM` variants.

As shown in Figure 2, when comparing against PRMs trained on similarly sized datasets, `PathFinder-PRM` consistently achieves superior performance across all benchmarks. Figure 3 presents the performance of various PRMs on PRMBench. The results demonstrate that `PathFinder-PRM` not only surpasses other PRMs trained on comparably sized datasets but also outperforms Math-PSA and Qwen-2.5-Math-PRM, despite those models being trained on 2–3 times more data.

`PathFinder-PRM` Scales Consistently Across Policy Sizes: As shown in Figure 4, When paired with a larger 14B policy model (Qwen2.5-14B-Instruct) to perform reward guided greedy search, `PathFinder-PRM-7B` provides a +1.4% improvement in average accuracy over Qwen2.5-Math-PRM-7B—closely matching the +1.5% gain observed with a 7B policy model. This consistent margin across scales indicates that the benefits of our hierarchical reward modeling generalizes to larger policy models.

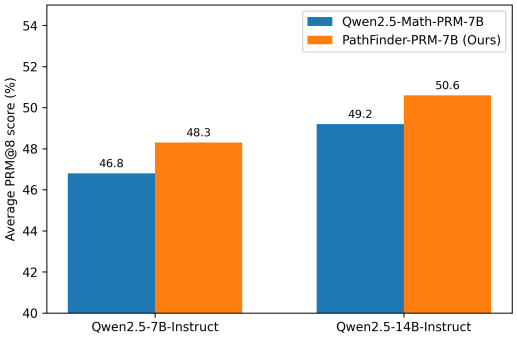

Figure 4: Average accuracy with 7B and 14B policy models. `PathFinder-PRM-7B` yields consistent gains (+1.4–1.5%) over Qwen2.5-Math-PRM-7B, demonstrating scalability.

5.2 Ablations

In our approach, we made two main claims: (1) decoupling the subtasks of Error Detection and Correct Path Guidance, and (2) categorizing errors into two prominent error categories in math will boost PRM performance. To verify these claims, we performed ablation experiments by modifying parts of our method:

1. `PathFinder-PRM` **w/o Separate Subtask Prediction**: Following existing PRM approaches, we trained the model to jointly learn to tackle both error detection and correct path guidance using only the step correctness labels.

2. `PathFinder-PRM` **w/o Separate Error Categories**: In this approach, we still do a hierarchical prediction but we modify step 1. Instead of detecting the presence of 2 error categories, we combined the categories and predicted the presence of an error in the step.

`PathFinder-PRM` Benefits from Separating Error Categories: On ProcessBench, explicitly distinguishing math and consistency errors yields a clear overall boost: `PathFinder-PRM-7B` scores 69.5 Avg. F1, versus 67.0 for the

PathFinder-PRM-7B w/o Separate Error Categories. We also observe a similar drop in performance on PRMBench, the PathFinder-PRM w/o Separate Error Categories shows a small drop in performance (0.4 points) compared to PathFinder-PRM-7B

Crucially, reward-guided search highlights the practical impact of error typing: when ranking eight candidate solutions, PathFinder-PRM-7B achieves 48.3 prm@8, compared to just 45.4 for PathFinder-PRM w/o Separate Error Categories (+2.9 points). This jump in real-world problem-solving performance highlights that fine-grained error signals not only improve diagnostic metrics but can also translate directly into selecting higher-quality solution paths.

PathFinder-PRM Benefits from Error-Aware Hierarchical Supervision: Across ProcessBench, PRMBench, and reward-guided search, PathFinder-PRM consistently outperforms the PathFinder-PRM w/o separate subtask prediction, demonstrating the importance of hierarchical modeling of the subtasks. On ProcessBench, PathFinder-PRM-7B improves from 67.6 to 69.5 F1 (+1.9), and on PRMBench, from 64.9 to 67.7 (+2.8). In reward-guided search, the improvement is similarly clear: 48.3 prm@8 versus 45.5. These results highlight the value of decoupling feedback prediction into discrete reasoning components.

Scalable Performance with Additional Training Data: Training on an additional 50K samples from a broader, automatically annotated dataset greatly boosted the performance of PathFinder-PRM and helped it reach state-of-the-art performance. PathFinder-PRM-7B outperforms PathFinder-PRM-7B-PRM800K across ProcessBench, PRMBench, and reward-guided greedy search, demonstrating the benefits of scaling beyond PRM800K dataset. On ProcessBench, PathFinder-PRM-7B achieves 69.5 Avg.F1 versus 65.2, while on PRMBench it improves from 65.0 to 67.7 F1. In reward-guided search, PathFinder-PRM-7B raises pass@8 from 46.9 to 48.3. These gains highlight that error-aware hierarchical modeling scales well with increased data diversity and quantity, enabling stronger generalization and robustness.

A key distinction is that the 50K additional data was collected using Mistral-7B as a generator, while PRM800K was produced using a fine-tuned GPT-4 model. Exposure to reasoning traces from a smaller and weaker model may have helped PathFinder-PRM better learn to recognize and correct common failure patterns, contributing to improved robustness and generalization across benchmarks. We leave a deeper investigation of this hypothesis to future work.

6 Conclusion

In this work, we introduced PathFinder-PRM, a hierarchical discriminative process reward model that decouples error detection from step optimality guidance, classifying math and consistency errors before computing step rewards. Our evaluation and ablation trials demonstrate that error-aware hierarchical supervision yields notable improvements on PRM benchmarks, with PathFinder-PRM-7B achieving state-of-the-art performance among discriminative PRMs on PRMBench and strong results on ProcessBench despite using less than three times the data used to train the current best performing model. A similar performance boost is observed in reward-guided search evaluation, and we further show that these gains scale consistently when paired with larger policy models, affirming our hypothesis about the efficacy of hierarchical error-aware reward generation. Our approach is, therefore, a promising direction for more robust and interpretable process reward models, with potential for further gains when scaled to larger architectures.

7 Limitations

Due to computational constraints, our experiments were limited to 7B models. While this scale provides a strong foundation for evaluating our proposed methodology, we hypothesise that larger models could further enhance modeling accuracy and better leverage process supervision signals due to their improved mathematical reasoning capabilities.

Acknowledgments

This research/project is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG3-GV-2023-010). This work is also supported by the National Research Foundation, Singapore under its National Large Language Models Funding Initiative (AISG Award No: AISG-NMLP-2024-005), and the NTU SUG project #025628-00001:Post-training to Improve Embodied AI Agents.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. 2025. Process reward models that think. *arXiv preprint arXiv:2504.16828*.

Ruosen Li, Ziming Luo, and Xinya Du. 2024. Fg-prm: Fine-grained hallucination detection and mitigation in language model mathematical reasoning. *Preprint*, arXiv:2410.06304.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. 2025. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling. *arXiv preprint arXiv:2502.06703*.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024. Improve mathematical reasoning in language models by automated process supervision. *Preprint*, arXiv:2406.06592.

Skywork o1 Team. 2024. Skywork-o1 open series. `https://huggingface.co/Skywork`. Accessed: 2025-05-18.

OpenAI. 2025. Introducing openai o3 and o4-mini. Accessed: 2025-05-19.

Shuaijie She, Junxiao Liu, Yifeng Liu, Jiajun Chen, Xin Huang, and Shujian Huang. 2025. R-prm: Reasoning-driven process reward modeling. *arXiv preprint arXiv:2503.21295*.

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

Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. 2024. Easy-to-hard generalization: Scalable alignment beyond human supervision. *arXiv preprint arXiv:2403.09472*.

Xiaoyu Tan, Tianchu Yao, Chao Qu, Bin Li, Minghao Yang, Dakuan Lu, Haozhe Wang, Xihe Qiu, Wei Chu, Yinghui Xu, and 1 others. 2025. Aurora: Automated training framework of universal process reward models via ensemble prompting and reverse verification. *arXiv preprint arXiv:2502.11520*.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback. *Preprint*, arXiv:2211.14275.

Jun Wang, Meng Fang, Ziyu Wan, Muning Wen, Jiachen Zhu, Anjie Liu, Ziqin Gong, Yan Song, Lei Chen, Lionel M Ni, and 1 others. 2024a. Openr: An open source framework for advanced reasoning with large language models. *arXiv preprint arXiv:2410.09671*.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed 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*, volume 35, pages 24824–24837. Curran Associates, Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. 2025. Evaluating mathematical reasoning beyond accuracy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 27723–27730.

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

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

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. *arXiv preprint arXiv:2501.07301*.

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and Bowen Zhou. 2025. Genprm: Scaling test-time compute of process reward models via generative reasoning. *arXiv preprint arXiv:2504.00891*.

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

A Extended Results

A.1 ProcessBench

Table 4 contains the extended evaluation on ProcessBench with LLM-as-Judge Baselines using both Proprietary and Open-Source Language Models.

Model	# Samples	GSM8K	MATH	Olympiad Bench	OmniMath	Avg. F1
LLM-as-judge, Proprietary language models						
GPT-4o*	unk	79.2	63.6	51.4	53.5	61.9
o1-mini*	unk	**93.2**	88.9	87.2	82.4	87.9
LLM-as-judge, Open-source language models						
Llama-3.3-70B-Instruct*	unk	82.9	59.4	46.7	43.0	58.0
Qwen2.5-Math-72B-Instruct*	unk	65.8	52.1	32.5	31.7	45.5
Qwen2.5-72B-Instruct*	unk	76.2	61.8	54.6	52.2	61.2
Discriminative Process Reward Models						
Trained on Automated Annotation Data						
Math-Shepherd-7B*	445K	47.9	29.5	24.8	23.8	31.5
RLHFlow-Mistral-8B*	273K	50.4	33.4	13.8	15.8	28.4
RLHFlow-DeepSeek-8B*	253K	38.8	33.8	16.9	16.9	26.6
Qwen2.5-Math-PRM-7B*	∼1.5M	**82.4**	**77.6**	**67.5**	**66.3**	**73.5**
Trained on Human Annotated Data (PRM800K)						
Llemma-PRM800K-7B	∼350K	48.4	43.1	28.5	33.4	38.4
ReasonEval-7B†	∼350K	41.0	48.9	36.7	37.4	41.0
Qwen2.5-Math-7B-PRM800K*	264K	68.2	62.6	50.7	44.3	58.5
`PathFinder-PRM-7B-PRM800K`	∼350K	<u>74.1</u>	<u>71.3</u>	58.7	56.6	<u>65.2</u>
w/o Separate Subtask Prediction		71.4	71.1	<u>59.2</u>	<u>58.0</u>	64.9
Trained on a Mix of Human and Automated Annotation Data						
Math-PSA-7B†	∼860K	62.4	41.9	31.5	25.2	40.3
Skywork-PRM-7B*	unk	70.8	53.6	22.9	21.0	42.1
⊛ `PathFinder-PRM-7B`	∼400K	<u>77.9</u>	<u>75.3</u>	<u>65.0</u>	<u>59.7</u>	<u>69.5</u>
w/o Separate Error Categories		76.1	73.8	61.4	56.6	67.0
w/o Separate Subtask Prediction		73.9	72.6	63.9	59.9	67.6

Table 4: Performance (F1) on ProcessBench. Results marked with * come from Zhang et al.. The best performance across all categories is in **bold** and the best performance within a category is <u>underlined</u>.

B Scaling Effects of Additional RLHFlow Mistral Data

To further probe the effects of training data scaling, we increased the amount of RLHFlow Mistral data from 50K to 200K samples and retrained `PathFinder-PRM`. As shown in Table 5, this increase from 400K to 550K total training samples did not result in improved performance. In fact, we observe a slight drop across all benchmarks: ProcessBench performance decreased from 69.5 to 68.75 Avg.F1, PRMBench from 67.7 to 67.4, and reward-guided search accuracy dropped from 48.3 to 47.5.

These findings suggest that the benefits of augmenting training with weaker model-generated traces (e.g., from Mistral-7B) may saturate quickly. Simply increasing the volume of such data does not necessarily lead to improved generalization, and may even slightly degrade performance. This underscores the importance of data quality and the nuanced role of diversity over quantity in training PRMs. We leave further study into effective feedback curation and dataset composition to future work.

Model	# Total samples	# Mistral samples	ProcessBench	PRMBench	Reward Guided Search (PRM@8)
`PathFinder-PRM-7B-PRM800K`	350K	0	65.2	65.0	46.9
`PathFinder-PRM-7B`	400K	50K	69.5	67.7	48.3
`PathFinder-PRM-7B`	550K	200K	68.75	67.4	47.5

Table 5: Effect of scaling RLHFlow Mistral data on `PathFinder-PRM` performance.

C Design Choice: Two Forward Passes over Autoregressive Multi-Token Prediction

In PathFinder-PRM, we adopt a two-forward-pass approach to predict intermediate error labels—Math Error and Consistency Error—followed by a final reward score for correctness. This contrasts with autoregressive decoding, which would generate all three labels sequentially. The motivation behind this design is to minimize error cascading between dependent predictions. Specifically, we aim to predict Math and Consistency labels independently in the first forward pass to prevent the Consistency prediction from being influenced by the previously generated Math label. This is a potential issue in the autoregressive setup since each token depends on the ones before it.

In our setup, a single language modeling head is used without any task-specific heads. During the first forward pass, we construct the input as: `[PRM Input], Math: <mask>, Consistency: <mask>`, where the two special `<mask>` tokens represent the target positions for Math and Consistency labels. The model is trained to produce the correct label tokens at these masked positions, allowing us to decode both predictions simultaneously without one influencing the other.

In the second forward pass, we supply the previously predicted Math and Consistency tokens in place, and append a third `<mask>` token to infer the final correctness label: `[PRM Input], Math: predicted_label, Consistency: predicted_label, Correctness: <mask>`. The probability corresponding to the positive reward token at the final mask position is then used as the model's output reward score.

This inference method avoids sequential decoding and uses only masked forward passes, enabling clearer modular supervision and avoiding implicit dependency leakage between intermediate labels. During training, a similar masked format is used. This clean separation helps PathFinder-PRM to better capture independent error signals and contributes to its robustness.

Algorithm 1 PathFinder-PRM Inference via Two Forward Passes

Require: Prompt P, PathFinder-PRM

// First Forward Pass: Predict Math and Consistency Labels
$Input_1 \leftarrow concatenate(P, \text{"Math: <mask>"}, \text{"Consistency: <mask>"})$
$Logits_1 \leftarrow \texttt{PathFinder-PRM}.forward(Input_1)$
$pred_{math} \leftarrow argmax(Logits_1$ at Math mask position$)$
$pred_{consistency}$ ¸$argmax(Logits_1$ at Consistency mask position$)$

// Second Forward Pass: Predict Correctness/Reward Label
$Input_2 \leftarrow concatenate(P, \text{"Math: "}, pred_{math},$
"Consistency: ", $pred_{consistency}$,
"Correctness: <mask>")

$Logits_2 \leftarrow \texttt{PathFinder-PRM}.forward(Input_2)$
$reward_{prob} \leftarrow softmax(Logits_2$ at Correctness mask position$)$
$pred_{reward} \leftarrow reward_{prob}$ of <+> token
return $pred_{math}, pred_{consistency}, pred_{reward}$

D Data Annotation Prompt

The prompt below was utilized with DeepSeek-R1-Distill-Qwen-32B to synthesize the 3-dimensional categorical score vectors assigned to each sample in the dataset used to train `PathFinder-PRM`

Prompt for dataset labelling

You are an analytical math instructor grading a student's work. Think step-by-step through your analysis. Below is the math question, the previous steps by the student, and the current step to evaluate.

{context}

Your task is to rigorously examine the current step and determine if it contains ANY mathematical errors. Assign binary scores (0 = wrong, 1 = correct) based on three criteria:

A) Mathematical logic — Is the current step, **on its own**, mathematically valid? Check for: • Calculation errors • Incorrect formula application • Invalid operations or simplifications • Algebra mistakes or sign errors • Incorrect assertions

B) Consistency — Is the current step logically consistent with: • Established ground truth • Previous steps • Any constraints or conditions established earlier • The mathematical domain applicable to this problem

C) Simplicity and optimality — is this step an efficient next step toward the solution? Check for: • Redundant statements: factually correct statements that do not help progress toward the solution. • Circular logic: does this step come to a conclusion already previously established? • Non-clarity: Are the assertions made in this step ambiguous in a way that obsfucates their purpose? • Optimality: is the **idea** of this step the near optimal approach one would take to solve the problem?

Double check all listed criterion here explicitly in your reasoning. In your analysis, be sensitive to subtle issues like missing pre-requisites/assumptions, correct-looking statements with slight errors and high confidence statements containing errors.

IMPORTANT POINTS:
- If you find ANY error, even a minor one, you MUST assign a score of 0 to the appropriate criteria. Be skeptical and verify all claims thoroughly.
- For incorrect steps, wherever possible, attempt to categorize the issue as violating **one of the three criterion** (i.e., assign score 0 to **only one category**). Assign multiple 0 scores only for serious errors.

You must format your answer as below:

Reasoning:
{{Provide detailed analysis, showing all verification steps and explicitly identifying any errors found}}

Final answers:
Score A:
{{0 or 1 only}}
Score B:
{{0 or 1 only}}
Score C:
{{0 or 1 only}}

E Training HyperParameters

Table 6 contains the hyperparameters used to train `PathFinder-PRM`. We used the Transformers library Trainer implementation to train our model in a seq-to-seq manner (Wolf et al., 2020).

Parameter	Value
Model	QwenQwen2.5-Math-7B-Instruct
Torch Data type	bfloat16
Attention Implementation	flash attention 2
Per-device Train Batch Size	2
Gradient Accumulation Steps	32
Learning Rate	1.0e-05
Number of Training Epochs	4
LR Scheduler Type	cosine
Max Gradient Norm	1.0
Warmup Ratio	0.1
Seed	42
BF16	true
Optimizer	adam
Gradient Checkpointing	True

Table 6: Training Configuration for Qwen2.5-Math-7B-Instruct.

F Scalability to Larger Models (Detailed Results)

In addition to the bar chart presented in the main text (Figure 4), we provide detailed benchmark-level results for Reward Guided Greedy Search experiments with Qwen2.5-14B-Instruct as the policy model. Table 7 reports performance across six math benchmarks. `PathFinder-PRM-7B` achieves the highest average accuracy (50.6%) and shows a +1.4% improvement over Qwen2.5-Math-PRM-7B with the 14B policy model. This matches the +1.5% relative gain observed with the 7B policy model, confirming that the effectiveness of our reward model generalizes across policy scales.

Process Reward Models	AIME24	AMC23	MATH	Olympiad Bench	College MATH	Minerva MATH	Avg.
Qwen2.5-Math-PRM-7B	85.0	65.0	10.0	54.5	39.0	41.5	49.2
`PathFinder-PRM-7B-PRM800K` (Ours)	83.4	57.5	23.3	56.0	37.0	41.9	49.9
`PathFinder-PRM-7B` (Ours)	82.2	62.5	23.3	53.5	40.5	41.5	50.6

Table 7: Scalability results with Qwen2.5-14B-Instruct across six math benchmarks.