Efficient Process Reward Model Training via Active Learning

Keyu Duan[1,2*] **Zichen Liu**[1,2] **Xin Mao**[1] **Tianyu Pang**[1] **Changyu Chen**[1,3]

Qiguang Chen **Michael Qizhe Shieh**[2†] **Longxu Dou**[1†]

[1]Sea AI Lab [2]National University of Singapore [3]Singapore Management University

Abstract

Process Reward Models (PRMs) provide step-level supervision to large language models (LLMs), but scaling up training data annotation remains challenging for both humans and LLMs. To address this limitation, we propose an active learning approach, ActPRM, which proactively selects the most uncertain samples for training, substantially reducing labeling costs. During training, we use the PRM to estimate uncertainty after the forward pass, retaining only highly uncertain data. A capable yet costly reasoning model then labels this data. Then we compute the loss w.r.t. the labels and update the PRM's weights. We compare ActPRM vs. vanilla fine-tuning, on a pool-based active learning setting, demonstrating that ActPRM reduce 50% annotation, but achieving the comparable or even better performance. Beyond annotation efficiency, we further advance the actively trained PRM by filtering over 1M+ math reasoning trajectories with ActPRM, retaining 60% of the data. A subsequent training on this selected dataset yields a new state-of-the-art (SOTA) PRM on ProcessBench (75.0%) and PRMBench (65.5%) compared with same sized models [1].

1 Introduction

Recently, Large Language Models (LLMs) (DeepSeek-AI et al., 2025; Yang et al., 2024; OpenAI et al., 2024b) have shown remarkable advances in mathematical reasoning, yet they can make mistakes during chain-of-thought (CoT) reasoning despite correct final answers(Zheng et al., 2024). To address this challenge, process reward models (Lightman et al., 2023; Wang et al., 2024; Zhang et al., 2025) were proposed, aiming to identify process errors and provide finer-grained supervision of the training process.

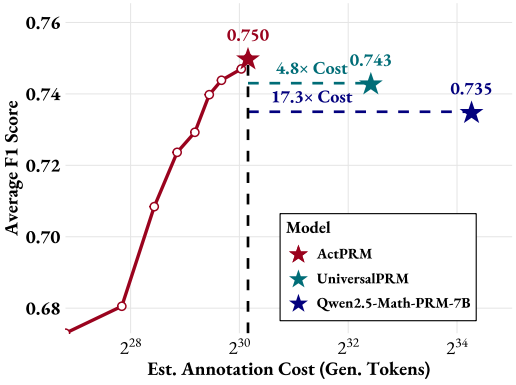

Figure 1: Average F1 score on ProcessBench (Zheng et al., 2024) versus estimated annotation cost. ActPRM outperforms prior SOTA models while requiring merely 20% of the annotation costs.

The main challenge in training Process Reward Models (PRMs) lies in obtaining fine-grained step-level annotations, which remain prohibitively expensive. Lightman et al. (2023) pioneered PRM training by employing human experts to label 75K questions at the step level. While their approach achieved high-quality results (reaching 57.5% on ProcessBench (Zheng et al., 2024)), it does not scale automatically due to the heavy reliance on manual annotation. To reduce human

*Work done during the internship in Sea AI Lab. Email: k.duan@u.nus.edu

†Corresponding authors.

[1]The code is available at https://github.com/sail-sg/ActivePRM

efforts, Monte Carlo (M.C.) Estimation Methods (Wang et al., 2024; Wei et al., 2024; Luo et al., 2024) were proposed. However, these approaches come with high computational costs (massive rollouts are required for accurate estimation) and struggle to accurately identify the first error step (Zheng et al., 2024). To address this challenge, Qwen2.5-Math-PRM (Zhang et al., 2025) proposed using LLM-as-Judge — leveraging LLMs to detect the first error step — and filter out unreliable M.C. labels. It significantly boosts the performance of PRM on both ProcessBench (Zheng et al., 2024) and PRMBench (Song et al., 2025). More recently, UniversalPRM (Tan et al., 2025) relies solely on LLM-as-Judge with ensemble prompting (via majority voting), achieving new SOTA performance on ProcessBench within the same model size. However, the annotation costs are still considerable. We estimate the labeling costs of Qwen2.5-Math-PRM (Zhang et al., 2025) and UniversalPRM (Tan et al., 2025) and illustrate them in Figure 1. It shows that Qwen-Math-PRM-7B and UniversalPRM consume over 2^{34} and 2^{32} generated tokens, respectively. Refer to Appendix C for estimation strategy.

To reduce annotation costs, we propose ActPRM, which uses a trained ensemble PRM to identify and select uncertain data for annotation by a high-capability reasoning model. Our approach trains a PRM with ensemble heads for uncertainty estimation. For each reasoning step, we compute the mean μ and standard deviation σ of ensemble predictions, identifying uncertain steps when prediction confidence is outside threshold $1 - \delta_{pred} < \mu < \delta_{pred}$ or variation exceeds δ_{std}. We consider a CoT trajectory uncertain **if any step up to and including the first predicted error meets these criteria**. By annotating only uncertain data and training exclusively on this subset, we significantly reduce labeling costs while maintaining PRM performance.

To validate the effectiveness and efficiency of ActPRM, we conducted comprehensive experiments in multiple settings:

- **Pool-based Evaluation (Section 5.1):** Using 100K labeled samples, ActPRM achieved performance comparable to full-data tuning while reducing annotation costs by 50%. It consistently outperformed random selection under identical budget constraints.
- **One-shot Active Learning (Section 5.2):** Starting with our pool-based model, we applied ActPRM to select uncertain samples from 1M+ unlabeled CoT trajectories from NuminaMath (Li et al., 2024). After annotation and fine-tuning, we achieved new SOTA performance of 75.0% on ProcessBench. As shown in Figure 1, ActPRM surpasses prior SOTA models with significantly lower costs—outperforming UniversalPRM (Tan et al., 2025) by 0.7% using only 20% of its annotation budget and exceeding Qwen2.5-Math-PRM-7B by 1.5% with just 6% of its annotation budget.

Our contributions are summarized as follows: ❶ We propose an uncertainty-aware active learning approach ActPRM for PRM training that selectively annotates informative reasoning steps using ensemble-based uncertainty estimation, significantly reducing labeling costs while maintaining performance. ❷ ActPRM achieves state-of-the-art results (75.0% on ProcessBench, 65.5% on PRMBench) while requiring only 20% of the annotation budget compared to prior SOTA method UniversalPRM. ❸ We release all trained models, datasets, and code to ensure reproducibility and facilitate community adoption.

2 Preliminaries

2.1 Process Reward Models

Problem Formulation. Given a math problem q and a corresponding solution trajectory $s = [s_1, s_2, \ldots, s_n]$, where s_i denotes i-th step., we require a PRM to identify the correctness of each step until a wrong step is identified. We only label the steps up to and including the first error step following prior works (Lightman et al., 2023; Zheng et al., 2024), since the afterward steps are genuinely difficult to define their correctness. As a result, in practice the labels for a solution trajectory are either $[1, 1, \ldots, 1]$ or $[1, 1, \ldots, 0]$. Then a PRM could be trained using the typical BCE loss:

$$\mathcal{L}_{BCE}(s, y|\theta) = -\frac{1}{|s|}\sum_{1}^{|s|} y_i \log(P_\theta(s_i|s_{[:i]}, q)) + (1 - y_i)\log(1 - P_\theta(s_i|s_{[:i]}, q)), \quad (1)$$

where P_θ is the PRM parameterized by θ and $s_{[:i]}$ denotes the steps before s_i. When using PRM for inference, we set a threshold δ (usually 0.5) to identify the first step that has a correctness probability $P_\theta(s_i|s_{[:i]}, q)$ less than δ.

PRM Implementation Details. A typical PRM is built upon a pretrained generative LLM by replacing the causal language model head with a binary classification head that outputs the probability of correctness at corresponding token position. In practice, we solely need the prediction at the end of each reasoning step and thus a prediction mask is used to mask out the prediction at the other positions.

2.2 Uncertainty Estimation for Classification

Aleatoric Uncertainty. As aforementioned, a typical PRM P_θ is trained as a binary classification task. The simplest way to measure the uncertainty for it is to use aleatoric uncertainty (Valdenegro-Toro & Saromo, 2022):[2].

$$\text{Aleatoric Uncertainty} \propto P_\theta(s_i) \log P_\theta(s_i). \quad (2)$$

Epistemic Uncertainty. In addition, ensemble of models is also a common way to estimate epistemic uncertainty (Valdenegro-Toro & Saromo, 2022) by quantifying the disagreement among ensemble models. For example, Liang et al. (2022); Gleave & Irving (2022) use an ensemble of reward models to estimate uncertainty in preference learning. for process reward modeling, we could leverage the standard deviation of the ensemble predictions as the uncertainty estimation:

$$\text{Epistemic Uncertainty} \propto \text{Var}(\{P_\theta(s_i)\}), \quad (3)$$

where $\{P_\theta\}$ is a set of models. It is worth noting that employing an ensemble of heads built upon a shared backbone is a common strategy to mitigate computational costs. We empirically study the combination of aleatoric and epistemic uncertainty and find that they are complementary to each other. Experimental results are shown in Section 5.1.

3 Related Work

Active Learning and Uncertainty Estimation. Active learning has been widely explored in the alignment of LLMs. Several studies adopt an online bandit formulation, leveraging uncertainty-aware reward models (RMs) for active exploration in response selection (Mehta et al., 2023; Dwaracherla et al., 2024; Liu et al., 2024; Melo et al., 2024; Gleave & Irving, 2022). For instance, Mehta et al. (2023) and Dwaracherla et al. (2024) use ensemble-based LLM heads to estimate epistemic uncertainty, prioritizing data most informative for preference learning. Similarly, Melo et al. (2024) propose an acquisition function combining both entropy (aleatoric uncertainty) and epistemic uncertainty. Our work builds on these approaches, empirically evaluating the role of both uncertainty types in the context of process reward modeling. Beyond active learning, ensemble methods—such as those in Coste et al. (2024)—have also proven effective in mitigating reward hacking (Amodei et al., 2016).

Process Reward Models. Different from outcome rewards (e.g., verifiable rewards (DeepSeek-AI et al., 2025) for mathematical reasoning problems) that assign rewards based on the final outcome, process rewards are assigned based on the intermediate steps of the problem-solving process. For a question and a corresponding solution with several steps, a PRM provides finer-grained rewards for each step. Til current stage, process reward modeling contains two categories: (i) *Process Reward as Q-values* and (ii) *Process Reward as Judger*. The former one (Wang et al., 2024; Luo et al., 2024; Wei et al., 2024; Li & Li, 2024) regards the process reward as the Q-values of the steps that estimate the probability of the policy model

[2]For simplicity, we use $P_\theta(s_i)$ as the aleatoric probability for the i-th step in the solution trajectory, where the full representation is $p_\theta(s_i|s_{[:i]}, q)$ as in Equation 1

Algorithm 1 PRM Active Learning with Cold Start.

1: // The difference with full data tuning is colored.
Input: Ensemble PRM P_θ, dataset $\mathcal{D} = \{(q, s)\}$, uncertainty thresholds δ_{pred} and δ_{std}, generative LLM M, batch size B, learning rate η
2: **for** $\mathcal{B} \subset \mathcal{D}$ **do**
3: $P_\theta(\mathcal{B}) \leftarrow \text{Forward}(\mathcal{B})$
4: $\widetilde{\mathcal{B}} = \{\}$
5: **for** $(q, s) \in \mathcal{B}$ **do**
6: **if** $\mathcal{U}_\theta^{\text{alea}}(s) \vee \mathcal{U}_\theta^{\text{epis}}(s)$ **then** ▷ Equation 5
7: $\widetilde{\mathcal{B}} \leftarrow \widetilde{\mathcal{B}} \cup \{(q, s)\}$
8: **end if**
9: **end for**
10: $Y_{\widetilde{\mathcal{B}}} \leftarrow \text{labeling}(\widetilde{\mathcal{B}})$ ▷ Labeling using generative LLM
11: $\mathcal{L} \leftarrow \frac{1}{|\widetilde{\mathcal{B}}|} \sum_{(s,y) \in (\widetilde{\mathcal{B}}, Y_{\widetilde{\mathcal{B}}})} \mathcal{L}(s, y)$ ▷ Equation 4
12: $\nabla_\theta \mathcal{L} \leftarrow \text{Backward}(\mathcal{L})$
13: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
14: **end for**
Output: P_θ

to reach the final correct answer. Specifically, they leverage the policy model that generates the solution steps to do Monte Carlo Estimation for each step. The estimated Q-values are used as the process rewards. However, recent works (Zhang et al., 2025; Zheng et al., 2024) show that this kind of process reward modeling suffers from identifying the process errors because it highly depends on the policy model and has large bias with the ground truth distribution. In contrast, the latter one (Lightman et al., 2023; Zhang et al., 2025) regards the process reward model as a proxy for identifying the intermediate process errors and the corresponding trained model achieves better performance on several benchmarks (Zheng et al., 2024; Song et al., 2025). In this work, we follow the latter one and regard the process reward as a judge that tries to identify the first error steps in the solutions if any. In addition, there are other works related to PRM. For example, Yuan et al. (2024) tries to train a PRM with a fashion of outcome reward modeling (ORM). Cheng et al. (2025); Cui et al. (2025) proposed RL training frameworks that integrate PRM as finer-grained supervison.

4 Efficient Process Reward Labeling via Active Learning

Labeling the process rewards for a large-scale dataset is very expensive as it either requires human experts to annotate the correctness of each step for each solution as in the previous work (Lightman et al., 2023) or leverages highly capable generative models to imitate human experts (Zhang et al., 2025). Even though the latter one is automated, it is still resource-consuming since the test time scales up with the difficulty of math problems.

To mitigate this issue, we propose to leverage active learning to make the PRM proactively select the data that is most informative to train on. Specifically, we train a PRM with ensemble heads to enable uncertainty estimation following Liang et al. (2022); Gleave & Irving (2022). As shown in Algorithm 1, We forward the data candidates to the ensemble PRM (line 3) and estimate the prediction uncertainty for each data point (line 5-6). Then we omit the data that the ensemble PRM is confident about (line 7) and only label the other retained data with a generative reasoning LLM (line 10). Then, we only backpropagate from the loss of labeled data (line 11). By doing so, we could considerably reduce the labeling cost while maintaining the PRM performance. Now we introduce our two key differences with the original finetuning: *ensemble PRM training* and *uncertain data selection*.

Ensemble PRM Training. In this work, we use ensemble of PRMs to estimate the epistemic uncertainty following Gleave & Irving (2022); Liang et al. (2022). Specifically, we use a shared LLM backbone and build multiple binary classification heads on top of it. In our training, the diversity of ensemble models is ensured by two ways: (i) the random initialization of the head layers and (ii) a diversity regularization term (Dwaracherla et al., 2024): $\mathcal{L}_{\text{div}} = \lambda \cdot \frac{1}{n} \sum_{i=1}^{n} ||\phi^i - \phi^i_{\text{init}}||_2$, where $\{\phi^i\}$ are the parameters of the ensemble heads and n is the number of ensemble heads. It is a $L2$ term penalizing the distance between

Listing 1: Pseudo code of uncertainty estimation.

```
# Compute ensemble predictions (num_ensemble, num_step)
score = llm(input_ids)
means, stds = score.mean(0), score.std(0)  # Per-step statistics
# Equ. 5 left
epistemic_uncertainty = stds >= std_threshold
# Equ. 5 right
alearotic_uncertainty = (means <= pred_threshold) & (means >= 1 -
    pred_threshold)
# Alg. 1 line 6
uncertainty = any(epistemic_uncertainty | alearotic_uncertainty)
```

the ensemble head parameters and their initial parameters. Our training objective for the ensemble PRM is therefore formulated as follows

$$\mathcal{L}(y,s) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathcal{L}_{BCE}(y,s|\theta,\phi^i) + \lambda||\phi^i - \phi^i_{\text{init}}||_2\right), \tag{4}$$

where θ denotes the backbone parameters and $\mathcal{L}_BCE$ is from Equation 1, that computes the loss for a certain head.

Uncertain Data Selection. Considering a batch of data candidates $\mathcal{D} = \{(q,s)\}$, we first forward the data to the ensemble PRM P_θ to get the ensemble predictions $P_\theta(\mathcal{D}) \in \mathbb{R}^{n\times|\mathcal{D}|\times|s|}$. for each data $(q,s) \in \mathcal{D}$, we could determine the hard-value aleatoric and epistemic uncertainty with pre-set thresholds. Briefly, the aleatoric (or epistemic) uncertainty is defined as 1 if uncertainty occurs at any step prior to the first predicted error step; otherwise, it is 0. A formal definition is as follows:

$$\mathcal{U}^{\text{alea}}_\theta(s) = \bigvee_{i=0}^{\mathcal{E}(s)}\left(\max\left(\mu(P_\theta(s_i)), 1-\mu(P_\theta(s_i))\right) < \delta_{pred}\right) \;;\; \mathcal{U}^{\text{epis}}_\theta(s) = \bigvee_{i=0}^{\mathcal{E}(s)}\left(\sigma(P_\theta(s_i)) > \delta_{std}\right), \tag{5}$$

where $\mu(\cdot)$ and $\sigma(\cdot)$ are the mean and standard deviation of the ensemble predictions among ensemble heads and $\vee$ denotes the logical 'OR' operation. Moreover, the $\mathcal{E}(s)$ denotes the first error step in the solution trajectory s, defined as $\mathcal{E}(s) = \min\{j \mid \mu(s_j) < \delta\}$, where δ is the threshold for the correctness, typically set to 0.5. This is because we only care about the correctness of the steps before the first error step since it is genuinely difficult to define the correctness of the steps afterwards. For further illustration, we also provide the pseudo code of the uncertainty estimation as in Listing 1. By following the uncertainty estimation strategy, we retain the data in $\mathcal{D}$ that satisfies either $\mathcal{U}^{\text{alea}}_\theta$ or $\mathcal{U}^{\text{epis}}_\theta$ as $\widetilde{\mathcal{D}}$. Then we could leverage expensive generative LLMs as judger (Zheng et al., 2024) to label the retained data in $\widetilde{\mathcal{D}}$.

5 Experiments

In Section 5.1, we first validate ActPRM in a pool-based active learning setting using 100K labeled samples, including ablation studies on our uncertainty estimation strategy. Based on the optimal configuration, we then scale up to 1M unlabeled samples in Section 5.2, further proving our pipeline's efficiency and effectiveness.

5.1 Pool-Based Active Learning

5.1.1 Experimental Settings

To evaluate our active learning strategy's effectiveness, we first conduct experiments in an offline setting where ActPRM iteratively selects the most informative examples from a large unlabeled pool as detailed in Algorithm 1. We establish a strong baseline by comparing

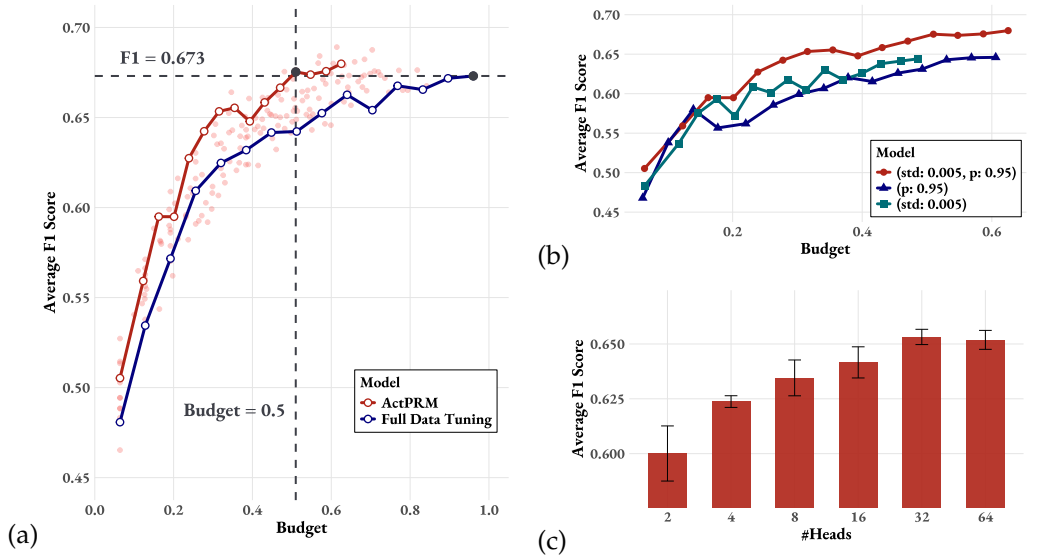

Figure 2: (a) Comparison of the average F1 score on ProcessBench between ACTPRM and random selection, plotted against the normalized budget positively correlated the number of labeled data instances consumed. The *semi-transparent* points represent all results in grid searching w.r.t. δ_{pred} and δ_{std}. For the highlighted ACTPRM curve in the figure, $\delta_{pred} = 0.95$ and $\delta_{std} = 0.005$. (b) Ablation: uncertainty estimation strategies. (c) Ablation: number of ensemble PRM heads.

against full data tuning, where a model is trained on the complete dataset labeled by a single annotator. It is worth noting that as our data is randomly shuffled, the performance of full data tuning at intermediate training steps is essentially equivalent to the performance of random selection with the corresponding budget.

Evaluation Benchmark. We utilize ProcessBench (Zheng et al., 2024) to evaluate the effectiveness of PRMs. The test data in ProcessBench contains intermediate step errors and requires the PRM to identify the first error step. ProcessBench contains four subsets, and we report the average F1 score following the original work.

Models. We train ACTPRM based on Qwen2.5-Math-7B-Instruct.

Training Dataset. For dataset construction, we randomly select 100K data from Numinamath (Li et al., 2024) dataset after decontamination against the ProcessBench (Zheng et al., 2024) and PRMBench (Song et al., 2025). We leverage Qwen-2.5-Math-7B-Instruct to generate CoT reasoning trajectories for the selected data and further use QwQ-32B as a judge to annotate the process correctness for all trajectories following Zhang et al. (2025). For completeness, we provide the prompt template in Appendix A.

5.1.2 *Experimental Results*

ACTPRM achieves comparable performance while reducing annotation costs by 50%. We compare ACTPRM with full data tuning across a normalized budget, as illustrated in Figure 2 (a). The results demonstrate that ACTPRM achieves an average F1 score of 0.673 on ProcessBench, matching baseline performance while using only half the annotation budget. Furthermore, ACTPRM consistently outperforms random selection under the same budget constraints. Notably, at 50% budget, ACTPRM surpasses random selection by a significant margin of 3.3%. at the end of pool-based active training, ACTPRM achieves a better performance of 0.680 on ProcessBench while consuming solely 62.5% budget.

ACTPRM Consistently Outperforms Random Selection Under Diverse δ_{pred} and δ_{std}. As shown in Figure 2 (a), the semi-transparent blue points represent all results of a grid searching over $\delta_{pred} \in \{0.9, 0.95, 0.97\}$ and $\delta_{std} \in \{0.01, 0.005, 0.002, 0.001\}$. One can see that most blue points are above the baseline (gray line) with the same budget, further demonstrating the effectiveness and robustness of ACTPRM.

Ablation Study on Uncertainty Estimation Strategies. We conduct an ablation study on uncertainty estimation strategies, i.e. using epistemic and aleatoric uncertainty. We selected the best setting ($\delta_{std} = 0.005, \delta_{pred} = 0.95$) searched by a grid search as in Figure 2 and ablates epistemic and aleatoric uncertainty by setting $\delta_{std} = \text{inf}$ and $\delta_{pred} = 0.5$, respectively. As shown illustrated in Figure 2 (b), solely use either epistemic or aleatoric uncertainty underperforms using both, indicating that epistemic and aleatoric uncertainty are complementary to each other.

Ablation Study on Number of Heads for Ensemble PRM. The number of heads for ensemble PRM controls how accurate our estimated epistemic uncertainty is. To find the trade-off between good estimation and computational overhead, we conduct an ablation study regarding it and show the results in Figure 2 (c), where we only consider epistemic uncertainty by setting $\delta_{std} = 0.005, \delta_{pred} = 0.5$ and report the averaged results with 3 runs. We empirically find that the performance continually grows with the number of heads and converges at about 32.

5.2 Achieving New SOTA Performance on ProcessBench (75.0%) with Solely 6% Annotation Cost.

Obtaining high-quality process supervision labels is costly. To demonstrate the efficiency of ACTPRM, we evaluate it in a one-shot active learning setting. Starting with the model trained in Section 5.1, we select the most uncertain samples from over 1M+ unlabeled examples and annotate them using a powerful reasoning model.

Figure 3 compares our estimated labeling costs with those of other real-world datasets for training PRMs, including MathShepherd (Wang et al., 2024), Consensus Filtering (Zhang et al., 2025), and Ensemble Prompting (Tan et al., 2025). Since the training data for Consensus Filtering is not publicly available, we estimate costs based on our data statistics. We introduce our estimation strategy in Appendix C.

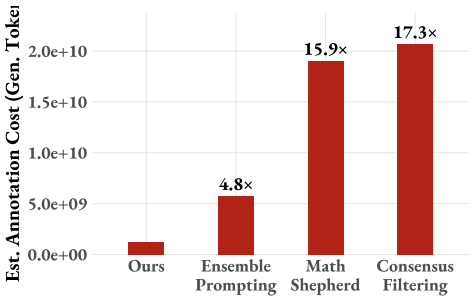

Figure 3: Estimated annotation costs (generated tokens) comparison between ACTPRM and popular methods, including Ensemble Prompting (Tan et al., 2025), MathShepherd (Wang et al., 2024) and Consensus Filtering (Zhang et al., 2025).

Training a Qwen2.5-Math-7B-Instruct on our data, Ensemble Prompting data, MathShepherd data, and Consensus Filtering data yields ACTPRM, UniversalPRM, Qwen2.5-Math-7B-Math-Shepherd, and Qwen2.5-Math-PRM-7B in Table 1. We evaluated the performance of models trained on this labeled data on both ProcessBench and PRMBench benchmarks.

5.2.1 *Experimental Settings*

Data Filtering with ACTPRM. We used Qwen2.5-Math-7B-Instruct and Qwen2.5-Math-72B-Instruct to collect over 1 million (1,061,763) Chain-of-Thought (COT) trajectories from the Numinamath problem set (Li et al., 2024), after decontamination against the test benchmarks. ACTPRM was then applied to filter out high-confidence ($\delta_{pred} > 0.95$ and $\delta_{std} < 0.005$ following Section 5.1) data instances that were unnecessary for training, retaining the remaining data for labeling and training. This process resulted in a final dataset of 563,030 PRM data points labeled by QwQ-32B, reducing annotation costs by 47.0%.

Models. Obtaining the dataset, we continually train our ACTPRM in Section 5.1 on our filtered dataset. In addition, we empirically find that our retrained data is generally useful to other PRMs. Specifically, we also continually train Qwen2.5-Math-PRM-7B (the previous SOTA model on ProcessBench) on our constructed data. The resultant model is named ACTPRM-X[3].

[3]X stands for extended version.

Models	GSM8K	MATH	Olympiad Bench	OmniMath	Average F1
LLM-as-judge					
o1-Mini$^\diamond$	0.932	0.889	0.872	0.824	0.879
Deepseek-R1-Distill-32B	0.817	0.739	0.659	0.585	0.700
QwQ-32B	0.871	0.834	0.787	0.771	0.816
Process Reward Models (72B)					
Qwen2.5-Math-PRM-72B$^\diamond$	0.873	0.806	0.743	0.711	0.783
Process Reward Models (7B+)					
Math-Shepherd-PRM-7B$^\diamond$	0.479	0.295	0.248	0.238	0.315
Qwen2.5-Math-7B-Math-Shepherd$^\diamond$	0.625	0.316	0.137	0.077	0.289
EurusPRM-Stage2$^\diamond$	0.473	0.357	0.212	0.209	0.313
Qwen2.5-Math-7B-PRM800K$^\diamond$	0.683	0.626	0.507	0.443	0.565
Ensemble-PRM-PRM800K (ours)	0.705	0.630	0.472	0.433	0.560
PURE-PRM-7B	0.690	0.665	0.484	0.459	0.575
Qwen2.5-Math-PRM-7B$^\diamond$	0.824	0.776	0.675	0.663	0.735
Universal-PRM	**0.858**	0.777	0.676	0.664	0.743
ACTPRM (ours)	0.816	0.798	0.714	0.670	0.750
ACTPRM-X (ours)	0.827	**0.820**	**0.720**	**0.673**	**0.760**

Table 1: Performance comparison on ProcessBench. We report the results in the same calculation method with ProcessBench. $^\diamond$ denotes the results are from Qwen PRM's report (Zhang et al., 2025).

Benchmarks. We use ProcessBench (Zheng et al., 2024) and PRMBench (Song et al., 2025) to evaluate the effectiveness of our trained model. Different from ProcessBench that collects intermediate errors from real-world generative models, PRMBench heuristically builds intermediate errors by manipulating correct steps.

Baselines. We compare with the following PRMs: ❶ *Qwen2.5-Math-PRM-7B* (Zhang et al., 2025): This model uses consensus filtering for labeling. It labels 860K data twice using two methods (LLM-as-judge [Zheng et al., 2024] and Mathshepherd [Wang et al., 2024]) and filters out 40% of the data where the labels disagree. ❷ *Pure-PRM-7B* (Cheng et al., 2025): A Qwen2.5-Math-based PRM trained on PRM800K using a two-stage strategy: warming up the PRM head and then fine-tuning the entire model. ❸: *EurusPRM-Stage2* (Cui et al., 2025): A PRM resulting from the Implicit PRM approach (Yuan et al., 2024), which derives process rewards from an ORM. ❹ *Universal-PRM* (Tan et al., 2025): A Qwen2.5-Math-based model trained with data augmentation techniques like ensemble prompting and reverse verification. ❺ *Math-Shepherd-PRM-7B* (Wang et al., 2024): a PRM trained on process labels that estimates hard Q-values for the policy model. ❻ *Qwen2.5-Math-7B-Math-Shepherd* (Zhang et al., 2025): a PRM trained on 860K data labeled using MathShepherd. ❼ *Ensemble-PRM-PRM800K (ours)*: a model with ensemble heads trained by ourselves on PRM800K without active learning.

5.2.2 *Experimental Results*

ACTPRM and ACTPRM-X achieve new SOTA performance on ProcessBench compared with same size models. The evaluation results on ProcessBench are shown in Table 1. ACTPRM achieves an average F1 score of 0.750, outperforming Qwen2.5-Math-PRM-7B by a margin of 1.5%. Furthermore, ACTPRM-X training based on Qwen2.5-Math-PRM-7B achieves a new SOTA performance on ProcessBench with an average F1 of 0.760, outperforming the second-place model (Universal-PRM) with a margin of 1.7% and improve the performance of Qwen2.5-Math-PRM-7B by a significant margin of 2.5%.

QwQ-32B (our PRM label annotator) outperforms all PRMs on ProcessBench. As shown in Table 1, QwQ-32B outperforms all PRMs including 72B models. It indicates the reliability of utilizing QwQ-32B as a PRM label annotator as it provides a high empirical upperbound for the training PRMs.

ACTPRM-X achieves new SOTA performance on PRMBench, on-par with GPT-4o. We further test our models on PRMBench and show the results in Table 2. As on the leaderboard, ACTPRM achieves the best performance within 7B PRMs and ACTPRM-X achieves new

#	Models	Simlicity	Soundness	Sensitivity	Average
	LLM-as-judge				
1	Gemini-2.0-thinking-exp-1219	0.662	0.718	0.753	0.688
1	o1-mini	0.646	0.721	0.755	0.688
4	GPT-4o	0.597	0.709	0.758	0.668
6	Gemini-2.0-flash-exp	0.627	0.673	0.754	0.660
	Process Reward Models (72B)				
3	Qwen-2.5-Math-PRM-72B	0.546	0.739	0.770	0.682
	Process Reward Models (7B+)				
7	Qwen2.5-Math-PRM-7B	0.521	0.710	0.755	0.655
9	Pure-PRM-7B	0.522	0.702	**0.758**	0.653
7	ACTPRM (ours)	0.536	0.713	0.752	0.655
5	ACTPRM-X (ours)	**0.545**	**0.727**	0.756	**0.667**

Table 2: Performance comparison on PRMBench. All results of the other models are from the o

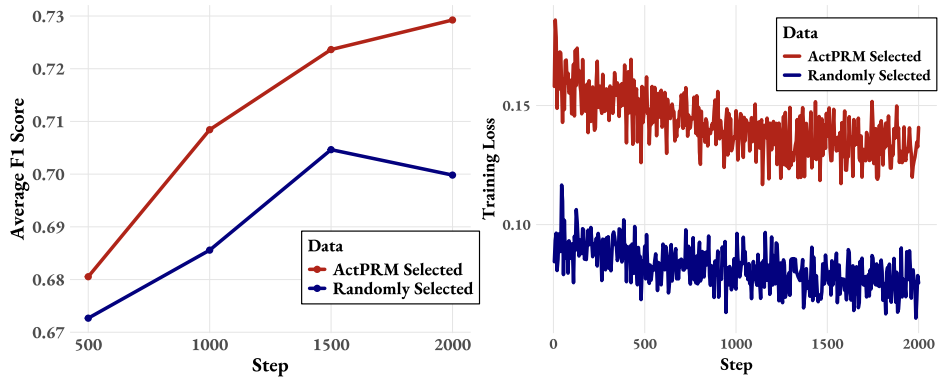

Figure 4: ProcessBench performance (*left*) and training loss (*right*): ActPRM v.s. random data selection on 1M NuminaMath Rollouts.

SOTA performance (0.667), outperforming the other models by a large margin of at least 1.2% and on-par with GPT-4o (OpenAI et al., 2024a).

5.2.3 *Comparative Experiment with Random Selection*

A potential concern is that while ACTPRM achieves state-of-the-art (SOTA) performance on several benchmarks, this success might be attributed solely to the high quality of our collected data pool, rather than the method itself. To address this, we conducted a comparative study with random selection. Specifically, we randomly selected 256K data points from our retained dataset as the experimental group. For the control group, we randomly selected the same number of data points from the entire data pool (over 1M) and used the same annotator to label any unlabeled data (i.e., data not in the retained set). We then continually train ACTPRM checkpoint, as in Sec 5.1, on both datasets. The results, including performance on ProcessBench and training loss, are shown in Figure 4.

ACTPRM outperforms random selection of the same amount of data. As illustrated in Figure 4 (left), the model trained on data selected by ACTPRM consistently achieves significantly better results than the model trained on randomly selected data. To further validate this, we compare their training losses in Figure 4 (right). The model trained on ACTPRM-selected data exhibits a consistently higher training loss, with a margin of 0.05, suggesting that the data selected by ACTPRM is more challenging and informative, thereby enhancing the learning process.

6 Conclusion and Future Work

In this work, we address the high annotation costs associated with training Process Reward Models (PRMs) by proposing ACTPRM, an uncertainty-aware active learning framework

that selectively annotates the most informative reasoning steps. By leveraging an ensemble PRM to estimate uncertainty and strategically labeling only uncertain data, ACTPRM significantly reduces annotation costs while maintaining competitive performance. Extensive experiments demonstrate that ACTPRM achieves a new state-of-the-art (75.0% on ProcessBench) with merely at most 20% of the labeling budget required by prior methods. Our results highlight the potential of efficient data selection for scalable PRM training, and we commit to releasing all models, datasets, and code to foster further research in this direction.

To further enhance PRM's performance, several promising directions can be explored. First, leveraging larger base models and more advanced LLM judges (e.g., O1-mini) could yield significant improvements. Second, implementing the framework in an online setting would ultimately enable PRM to iteratively refine its performance through active learning. Additionally, integrating online PRM training with reinforcement learning frameworks—such as actor-critic methods—presents an exciting avenue for research.

Acknowledgement

This project was partly supported by the MOE AcRF Tier 1 grant with grant number 251RES2514.

References

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety, 2016. URL https://arxiv.org/abs/1606.06565.

Jie Cheng, Lijun Li, Gang Xiong, Jing Shao, and Yisheng Lv. Stop gamma decay: Min-form credit assignment is all process reward model needs for reasoning, 2025. Notion Blog.

Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward Model Ensembles Help Mitigate Overoptimization, 2024.

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

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo,

Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Vikranth Dwaracherla, Seyed Mohammad Asghari, Botao Hao, and Benjamin Van Roy. Efficient exploration for llms. In *International Conference on Machine Learning*, 2024.

Adam Gleave and Geoffrey Irving. Uncertainty Estimation for Language Reward Models, 2022.

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, 2021. URL https://arxiv.org/abs/2103.03874.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath. [https://huggingface.co/AI-MO/NuminaMath-1.5](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024.

Wendi Li and Yixuan Li. Process reward model with q-value rankings. *arXiv preprint arXiv:2410.11287*, 2024.

Xinran Liang, Katherine Shu, Kimin Lee, and Pieter Abbeel. Reward Uncertainty for Exploration in Preference-based Reinforcement Learning, 2022.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's Verify Step by Step, 2023.

Zichen Liu, Changyu Chen, Chao Du, Wee Sun Lee, and Min Lin. Sample-efficient alignment for llms, 2024. URL https://arxiv.org/abs/2411.01493.

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. Improve Mathematical Reasoning in Language Models by Automated Process Supervision, 2024.

Viraj Mehta, Vikramjeet Das, Ojash Neopane, Yijia Dai, Ilija Bogunovic, Jeff Schneider, and Willie Neiswanger. Sample efficient reinforcement learning from human feedback via active exploration. *arXiv preprint arXiv:2312.00267*, 2023.

Luckeciano C. Melo, Panagiotis Tigas, Alessandro Abate, and Yarin Gal. Deep bayesian active learning for preference modeling in large language models, 2024. URL https://arxiv.org/abs/2406.10023.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Madry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun

Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. Gpt-4o system card, 2024a. URL https://arxiv.org/abs/2410.21276.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. Openai o1 system card, 2024b. URL https://arxiv.org/abs/2412.16720.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. PRMBench: A Fine-grained and Challenging Benchmark for Process-Level Reward Models, 2025.

Xiaoyu Tan, Tianchu Yao, Chao Qu, Bin Li, Minghao Yang, Dakuan Lu, Haozhe Wang, Xihe Qiu, Wei Chu, Yinghui Xu, and Yuan Qi. Aurora:automated training framework of universal process reward models via ensemble prompting and reverse verification, 2025. URL https://arxiv.org/abs/2502.11520.

Matias Valdenegro-Toro and Daniel Saromo. A deeper look into aleatoric and epistemic uncertainty disentanglement, 2022. URL https://arxiv.org/abs/2204.09308.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations, 2024.

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

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

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.

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, 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, 2024.

A LLM-as-Judger Prompt Template

For LLM-as-Judger, we follow the prompt in Zhang et al. (2025).

```
I will provide a math problem along with a solution. They will be formatted as follows:
[Math Problem]
<math_problem>
...(math problem)...
</math_problem>
[Solution]
<paragraph_1>
...(paragraph 1 of solution)...
</paragraph_1>
...
<paragraph_n>
...(paragraph n of solution)...
</paragraph_n>

Your task is to review each paragraph of the solution in sequence, analyzing,
verifying, and critiquing the reasoning in detail. You need to provide the
analyses and the conclusion in the following format:
<analysis_1>
...(analysis of paragraph 1)...
</analysis_1>
...
<analysis_n>
...(analysis of paragraph n)...
</analysis_n>
<conclusion>
Correct/Incorrect
</conclusion>

* When you analyze each paragraph, you should use proper verification, recalculation, or
    ↪ reflection to indicate whether it is logically and mathematically valid. Please
    ↪ elaborate on the analysis process carefully.
* If an error is detected in any paragraph, you should describe the nature and cause of the
    ↪ error in detail, and suggest how to correct the error or the correct approach. Once a
    ↪ paragraph is found to contain any error, stop further analysis of subsequent
    ↪ paragraphs (as they may depend on the identified error) and directly provide the
    ↪ conclusion of "Incorrect." For instance, given a solution of five paragraphs, if an
    ↪ error is found in the third paragraph, you should reply in the following format:
<analysis_1>
...(analysis of paragraph 1)...
</analysis_1>
<analysis_2>
...(analysis of paragraph 2)...
</analysis_3>
<analysis_3>
...(analysis of paragraph 3; since an error is found here, also provide detailed critique and
    ↪ correction guideline)...
</analysis_3>
<conclusion>
Incorrect
</conclusion>
Note that the analyses of paragraphs 4 and 5 should be skipped as the paragraph 3 has been
    ↪ found to contain an error.
* Respond with your analyses and conclusion directly.
--------------------------------------------------
The following is the math problem and the solution for your task:
[Math Problem]
{tagged_problem}
[Solution]
{tagged_response}
```

B More Experiment Results

B.1 Problem diversity is important for Training PRMs

PRM800K (Lightman et al., 2023) is a widely used and human-annotated dataset for PRM training, which contains 800K step-level labels across 75K tree-of-thoughts solutions to 12K MATH (Hendrycks et al., 2021). Our empirical results show that models trained on our dataset (100K samples from 100K diverse questions) consistently and significantly outper-

	# Problem set	# CoT Trajectories	ProcessBench F1 score
PRM800K	7,500	460,000	0.575
NuminaMath (Random Selected)	100,000	100,000	0.673

Table 3: Comparison between PRM800K and 100K data collected from NuminaMath labeled by Qwen-QwQ.

form those trained on PRM800K[4] (369K samples from only 12K questions) on ProcessBench. These findings suggest that problem diversity plays a more crucial role in PRM training than the number of step-level annotations.

C Annotation Cost Estimation

We estimate the labeling cost based on the statistics of our 1M data collected from NuminaMath Li et al. (2024) using Qwen2.5-Math-7B-Instruct and Qwen2.5-Math-72B-Instruct. We introduce the statistics in Table 4.

	Value	Source
# Reasoning Steps (S)	8.845	Qwen Models' rollouts
# Tokens per Rollout (R)	625.098	Qwen Models' rollouts
# Tokens per Critic Response from Judge (C)	1,919.860	Qwen-QwQ's responses as LLM-as-Judge

Table 4: Statistics of 1M NuminaMath CoT Trajectories collected by Qwen2.5-Math Models.

In addition to the statistics, we also use N to denote the data number of the dataset and show this statistic of each model's training dataset in Table 5

Dataset	# Labeled Data
ACTPRM	624,000 (labeled in two stages)
Qwen2.5-Math-PRM-Math-shepherd	860,000
Qwen2.5-Math-PRM	860,000
UniversalPRM	690,000

Table 5: Data number of datasets.

Using the statistics, we compute the estimated labeling cost for ACTPRM, Qwen2.5-Math-PRM-Math-shepherd (Zhang et al., 2025), Qwen2.5-Math-PRM(Zhang et al., 2025), UniversalPRM Tan et al. (2025) as follows:

- Qwen2.5-Math-PRM-Math-shepherd: $N \times S \times 8 \times R/2$, where 8 is the number of rollouts per step set in Zhang et al. (2025). We divided by two since the number of tokens for rollouts varies based on the position of reasoning step. For latter reasoning step, it requires less reasoning tokens. As a result, the expectation of tokens per rollout should be half of the number of tokens of the complete rollout.
- Qwen2.5-Math-PRM: $N \times S \times 8 \times R/2 + N * C$. It used consensus filtering for each data, the cost is both from MathShepherd ($S \times 8 \times R/2$) and LLM-as-Judge (C).
- UniversalPRM: $N \times C \times 4 + N \times S$, where 4 is the number of ensemble prompts from the original paper (Tan et al., 2025) and another $N \times S$ is for its semantic-based step seperation.
- ACTPRM: $N \times C$. We solely use Qwen-QwQ as Judge and do not include any other operations.

[4] https://huggingface.co/datasets/HuggingFaceH4/prm800k-trl-dedup