Extended Abstract

Motivation Reinforcement learning from human feedback (RLHF) applies a single reward value only at solution termination, creating a sparse signal that is incapable of distinguishing careful reasoning from lucky correct answers. In multi-step mathematical reasoning tasks especially, early errors propagate through the solution chain unchecked and the reward gradient is delayed exhibiting high variance during training. This project addresses the problem by replacing sparse outcome rewards with dense step-level rewards from a **Process Reward Model (PRM)**, by assigning a correctness probability to each individual reasoning step.

Method The project is implemented using a 3-stage pipeline constituting a **(1) Supervised Fine-Tuning (SFT)** of Qwen2.5-7B-Instruct with LoRA to establish a reasoning baseline, **(2) PRM Training** of a scalar reward head on Math-Shepherd automated step annotations using binary cross-entropy (BCE) loss and **(3) Best-of-N re-ranking (during inference)** done for selecting the highest-PRM-scoring candidate from N sampled solutions. Finally a DPO strategy was attempted to generate pairs of problems to further train the PRM.

Implementation The SFT fine-tunes Qwen2.5-7B-Instruct with LoRA (rank r=16, α=32) on 7,473 GSM8K training examples for 3 epochs with a peak learning rate of 2×10^{-4} and cosine decay, yielding $\approx$40M trainable parameters. The PRM then adds a projection head $\mathbf{W} \in \mathbb{R}^{3584\times1}$ at each step-separator token positions, trained on 400,189 step pairs from Math-Shepherd via BCE loss.

Results The SFT baseline achieved **73.54%** on the 1,319-problem GSM8K test set. DPO fine-tuning with PRM-guided preference pairs resulted in marginal improvement (+0.15 pp to 73.69%), this is because it was limited by the scarcity of meaningful contrasts. Only 64 preference pairs emerged from 500 training problems because the SFT model already achieves $\approx$91% accuracy on those problems. Best-of-N (N=8) re-ranking assisted by the PRM achieved **86.58%**, a **+13.04 pp** improvement without any additional training. The PRM was able to achieve an AUC-ROC $= 0.9287$ and step-level accuracy $= 0.8463$ on a held-out Math-Shepherd annotated test-set.

Discussion The experiments conducted do show that dense process rewards are most effective during inference time especially when the base model is capable enough to generate diverse enough candidate solutions. The +13.04 pp gain using the Best-of-N strategy with zero additional training shows the PRM's ability to rank solution quality. The DPO's near-zero improvement shows that preference-based training requires problems where the SFT model can produce both correct and incorrect solutions with different reasoning quality. As this condition was not satisfied by GSM8K for a 91%-accurate model future work will apply DPO on harder benchmarks (MATH, AIME) where solution diversity is greater for more complex problems.

Conclusion The dense step-level rewards from a well-trained PRM can substantially improve long multi-step mathematical reasoning task accuracy at inference time. DPO for training fine-tuning requires a problem distribution that can truly challenge the base model. The PRM framework as a verifier is most accurate for models operating near the frontier of their individual capabilities.

Dense Step-Level Rewards via Process Reward Models for Mathematical Reasoning

Dhruv Arcot
Stanford University
dhruva98@stanford.edu

Abstract

Reinforcement learning from human feedback (RLHF) assigns a single outcome reward at the end of a solution, providing a sparse signal that is incapable of pinpointing where exactly the multi-step reasoning fails. This project proposes a Process Reward Model (PRM) on the automated Math-Shepherd step annotations in order to assign dense step-level rewards. The rewards are then applied in two ways to a Qwen2.5-7B-Instruct model fine-tuned with LoRA on GSM8K: (1) **Best-of-N re-ranking during inference time**, selecting the highest scoring among N sampled solutions and (2) **a DPO training time preference pair generation**, using the PRM scores to identify chosen and rejected solutions. Best-of-8 re-ranking achieves an accuracy of **86.58%** on the GSM8K test set, a **+13.04 pp** increase over the 73.54% SFT baseline. Due to a lack of useful preference contrasts the DPO gains only +0.15 pp when the base model (SFT) is already strong. The PRM achieves AUC-ROC $= 0.9287$ on held-out step annotations also confirming its stability and reliability as a step-level verifier.

1 Introduction

Large language models have shown remarkable capability on mathematical reasoning benchmarks, yet training them robustly remains challenging. The preferred paradigm, Reinforcement Learning from Human Feedback (RLHF) Ouyang et al. (2022); Christiano et al. (2017), applies a single scalar reward at the end of a solution. This outcome-level reward is poorly suited to multi-step reasoning as we have no context of exactly where the reasoning step broke down. This can result in a model reaching the correct final answer through incorrect reasoning steps, early errors propagate undetected through the chain and only at the final step will we identify them, and the delayed reward provides high variance in the gradient signal. The goal in simple terms is to ensure the evaluator (verifier) also assesses the intermediate steps given by a student (LLM) in order to reach the final conclusion.

The goal now becomes to reward each individual reasoning step as against the paradigm of only evaluating the final outcome signal. A **Process Reward Model (PRM)** is utilized for this where it learns to predict whether each step is on a correct reasoning path. This turns a single delayed reward into a dense sequence of intermediate signals for further refinement:

$$r_t = \mathrm{PRM}(x,\, s_1, \ldots, s_t) \in [0, 1]$$

The step-level rewards from the PRM are used during inference time as a verifier system among a pool of candidate solutions generated. An additional step is to generate pairs of responses (correct and wrong ones) in order to perform DPO.

In this work, a complete PRM-based pipeline on GSM8K Cobbe et al. (2021) is implemented and evaluated:

1. **SFT baseline**: Qwen2.5-7B-Instruct Team (2025) fine-tuned with LoRA on GSM8K training data was used to establish a strong baseline.

2. **PRM training**: A scalar reward head trained on Math-Shepherd Wang et al. (2024) via binary cross-entropy using the step level annotations provided by the dataset, yielded a reliable step-level verifier (AUC-ROC $= 0.9287$).
3. **Downstream applications**: (a) Best-of-N inference-time re-ranking using the PRM verifier scores; (b) DPO Rafailov et al. (2023) fine-tuning with PRM-guided preference pairs generating both correct and incorrect responses.

The main finding is that the Best-of-8 re-ranking implementation improves the GSM8K accuracy from 73.54% to **86.58%** (+13.04 pp) without any additional model training. The DPO provides only marginal improvement (+0.15 pp). This is primarily due to the SFT model's high accuracy on the training distribution leaving very little room for useful preference contrasts.

2 Related Work

Outcome-supervised verifiers: Cobbe et al. (2021) had trained solution verifiers on GSM8K using only the final signal of outcome correctness. While this was effective on the training data the solution was not able to diagnose intermediate reasoning errors and it also struggled to generalize over problems outside of its training distribution.

Process reward models: Lightman et al. (2023) showed that human-annotated step-level labels can significantly outperform the existing outcome-supervised verifiers on the MATH benchmark. This validated the process-reward approach. A drawback of this approach was in the collection of human annotated samples which is both expensive and does not scale. Wang et al. (2024) addressed the scalability problem of Lightman with the Math-Shepherd. Based on whether the trajectory was going to reach a correct answer, it labels each step yielding 400K+ annotated step pairs and at no annotation cost. The Math-Shepherd labels were used for the PRM's training signal.

Process verifiers for inference-time compute: Setlur et al. (2024) demonstrated that process verifiers can substantially improve best-of-N accuracy at inference time. This approach was also investigated by using DPO against the PRM scores generated.

Direct Preference Optimization: Rafailov et al. (2023) introduced DPO as an alternative to PPO-based RLHF. An approach to directly optimize from (chosen, rejected) pairs without any explicit reward model at training time can be used to further fine-tune the model. The intent was to use pair generation from the PRM in order to supply the DPO with informed preferences in order to perform training time fine-tuning.

Base model: Qwen2.5-7B-Instruct Team (2025) was used as the base model. It achieves $\approx 85\%$ zero-shot accuracy on the GSM8K dataset, giving us a strong foundation for evaluating whether PRM-based rewards can add measurable value beyond what the model already does.

3 Method

The pipeline includes three distinct stages: (1) SFT baseline training, (2) PRM training, and (3) downstream applications of the PRM. The SFT model generates N candidate solutions and each of them is scored by the PRM using the mean step-level reward. The highest-scoring candidate is returned as the final answer.

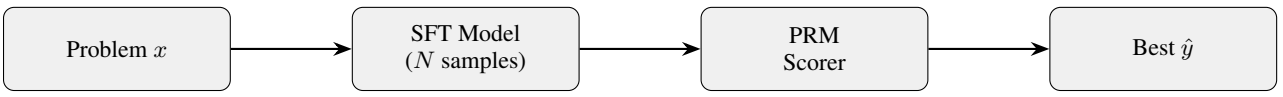

Figure 1: Best-of-N inference pipeline.

3.1 Stage 1: Supervised Fine-Tuning

The GSM8K dataset is split into a training set consisting of 7,473 problems. This data is then used to fine-tune a Qwen2.5-7B-Instruct Team (2025) using Low-Rank Adaptation (LoRA). LoRA adapters

with rank r=16 and scaling α=32 are applied to all the attention projection layers (query, key, value, output) and the MLP layers. The model resulted in approximately 40M trainable parameters. Training runs for 3 epochs hitting a peak learning rate of 2×10^{-4} and cosine decay, the training loss drops from 1.43 to $\approx$0.13 with token accuracy stabilizing at $\approx$96%.

The SFT model now serves two roles in subsequent stages. It acts as the *generator* for the Best-of-N candidate sampling and DPO pair collection. Its frozen weights form the *backbone* of the PRM verifier.

3.2 Stage 2: Process Reward Model

Architecture: The PRM is built on top of the frozen SFT backbone. The solutions are decomposed into reasoning steps which are separated by the `<step>` step-separator token (used in Math-Shepherd). At each separator position t, the backbone's hidden state $h_t \in \mathbb{R}^{3584}$ is extracted and then passed through a learned linear projection head:

$$\hat{r}_t = \sigma\big(\mathbf{W}^\top h_t\big), \qquad \mathbf{W} \in \mathbb{R}^{3584\times1} \tag{1}$$

Here $\sigma(\cdot)$ is the sigmoid activation function, giving us a reward $\hat{r}_t \in [0,1]$ as the step-level correctness probability.

Training supervision: Following Math-Shepherd's annotation convention Wang et al. (2024), step labels are defined as:

$$y_t = \begin{cases} 1 & \text{if step } s_t \text{ belongs to a correct answers trajectory} \\ 0 & \text{otherwise} \end{cases}$$

The projection head is then trained via binary cross-entropy over all the annotated step positions:

$$\mathcal{L}_{\text{PRM}} = -\sum_t \Big[y_t \log \hat{r}_t + (1-y_t)\log(1-\hat{r}_t) \Big] \tag{2}$$

The training uses 400,189 step pairs and 44,465 validation pairs from the Math-Shepherd dataset. Only $\mathbf{W}$ is updated while keeping the backbone frozen in order to preserve the SFT model's reasoning ability.

3.3 Stage 3a: Best-of-N Inference-Time Re-ranking

Given a test problem x, the SFT model now samples N independent solution candidates $\{y^{(1)},\dots,y^{(N)}\}$ using temperature based sampling. Each candidate $y^{(i)}$ consists of multiple T_i reasoning steps. The PRM then aggregates step-level scores by averaging across steps:

$$\text{score}(y^{(i)}) = \frac{1}{T_i}\sum_{t=1}^{T_i} \hat{r}_t^{(i)} \tag{3}$$

The final solution is then selected as the candidate with the highest score:

$$\hat{y} = \arg\max_{i\in\{1,\dots,N\}} \text{score}(y^{(i)}) \tag{4}$$

This procedure does not require any additional training after Stage 2. The PRM, once trained, now functions as a verifier for any generator model.

3.4 Stage 3b: DPO with PRM-Guided Pair Generation

For 500 problems from the GSM8K training set, N=4 solutions were sampled per problem from the SFT model and each of them were scored using the PRM. Preference pairs (chosen, rejected) are constructed using three strategies mentioned below:

1. **Mixed outcomes**: Among the N samples generated as the solution there exists at least one correct and one incorrect solution among them. The correct one is the solution which has the highest PRM score and the rejected one is the solution with the lowest scoring incorrect solution. This preference signal is the strongest one provided to the model.

2. **All correct**: All N samples from the SFT are correct. The solution with the highest PRM score, the best reasoned solution, is chosen and the lowest scored sample is labeled as the rejected one. This signal helps reinforce quality reasoning signals rather than correctness.
3. **All incorrect**: All N samples from the SFT are incorrect. The formed pairs are ranked by their PRM scores. A pair with a score gap <0.05 is not considered as the PRM verifier cannot distinguish them with confidence.

DPO Rafailov et al. (2023) is then applied to the SFT model using all the collected pairs:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{(x,\, y_w,\, y_l)} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)} \right) \right] \tag{5}$$

where

- y_w: chosen solution.
- y_l: rejected solution.
- π_θ: policy being trained.
- π_{ref}: frozen SFT reference policy.
- β: controls the strength of the KL-divergence penalty.

4 Experimental Setup

Datasets:

- **GSM8K** Cobbe et al. (2021): 8,500+ grade-school math word problems were divided into an 85-15 train-test split ratio. This then consisted of 7,473/1,319 training and testing samples respectively. Accuracy is measured by an exact numerical match on the full test set (1,319 problems).
- **Math-Shepherd** Wang et al. (2024): 444,654 step-annotated reasoning pairs were divided into a 90-10 train-validation split ratio consisting of 400,189 and 44,465 training and validation samples respectively. This data was used exclusively for PRM training and evaluation.

Base model: For its strong zero-shot GSM8K accuracy, $\approx 85\%$, the Qwen2.5-7B-Instruct is selected over alternatives (e.g., Llama-3-8B) as the baseline model. This ensures that improvements from the PRM are entirely attributable to the reward framework rather than the base policy which can be weaker/stronger.

Compute: All the experiments were run on Modal cloud infrastructure with NVIDIA A100-40GB GPUs. SFT training took approximately 3 hours while the PRM training to the evaluated checkpoint (approx step 12,200) took approximately 6 hours.

Evaluation: Primary metric: Accuracy, exact match accuracy, on the full 1,319-problem GSM8K test set. Secondary metric: AUC-ROC and step-level accuracy at threshold τ=0.5 on the held-out Math-Shepherd validation set. Along with the accuracy the standard error on the test-set is also reported. The standard error is defined as:

$$\text{SE} = \sqrt{\hat{p}(1-\hat{p})/n} \quad \text{where } \hat{p} \text{ is the observed accuracy,}$$

The standard error can help in determining if the difference between the methods is a genuine effect or simply the sampling noise from the specific test-set.

5 Results

5.1 Quantitative Evaluation:

The test accuracy across all evaluated methods is reported in table 1. The SFT baseline achieves 73.54% with greedy decoding. This shows a regression from the zero-shot baseline (discussed in

Section 6) which reports an accuracy of $\approx 85\%$. The Best-of-8 re-ranking with the trained PRM achieved an accuracy of **86.58%**, a **+13.04 pp** improvement over the SFT baseline. The accuracy achieved is also slightly above the zero-shot baseline. This was achieved without any additional model training after the PRM head. The DPO fine-tuning was only able to yield marginal improvement. It resulted in a +0.15 pp improvement to 73.69% from the SFT baseline.

The Best-of-8 gain of +13.04 pp is roughly $8.5\times$ the combined standard error of the SFT and Best-of-8 estimates (≈ 1.53 pp, computed as $\sqrt{\mathrm{SE}_{\mathrm{SFT}}^2 + \mathrm{SE}_{\mathrm{BoN}}^2}$). Using the standard error measurement, the improvement is statistically significant, $z \approx 8.5$, $p \ll 0.001$. In contrast, the DPO gain of +0.15 pp is significantly smaller than its combined standard error as compared to the SFT baseline (≈ 1.71 pp, $z \approx 0.09$), meaning it is statistically insignificant further validating the negligent effects of it.

Table 1: GSM8K test set accuracy (1,319 examples)

Method	**Accuracy ($\pm$ SE)**	**Correct / 1,319**
Qwen2.5-7B-Instruct (zero-shot)	$85.00\% \pm 0.98$ pp	1,121
SFT Greedy (baseline)	$73.54\% \pm 1.21$ pp	970
DPO Fine-tuned	$73.69\% \pm 1.21$ pp	972
Best-of-8 + PRM	$\mathbf{86.58\% \pm 0.94}$ pp	**1,142**

Table 2 reports the secondary metric, AUC-ROC, in order to measure the reliability and the intrinsic quality of the PRM approach. This was measured on the held-out Math-Shepherd step annotated test-set. An AUC-ROC of 0.9287 confirms that the scalar head is able to reliably distinguish correct from incorrect reasoning steps, providing the discriminative power of the verifier that enables the Best-of-8 result.

Table 2: PRM evaluation on held-out Math-Shepherd steps (checkpoint 12,200).

PRM Metric	**Value**
AUC-ROC	0.9287
Step Accuracy (τ=0.5)	0.8463
Steps Evaluated	10,150

5.2 Qualitative Analysis: DPO Pair Scarcity

During the implementation of the DPO pair generation process a fundamental limitation was exposed. Only **64 preference pairs** were generated when sampling 4 solutions per problem on 500 samples of the GSM8K training data. Since the SFT model achieved a 91% accuracy, in most cases all 4 samples per problem generated proved to be all correct, leaving no opportunity to form the mixed-outcome pairs to generate a quality high signal for DPO training. The all-correct pairs (strategy 2) carry a weaker signal, where the PRM score differences between two correct solutions discern reasoning style rather than correctness, providing a weaker gradient for DPO. Overfitting was seen almost immediately when training on 64 pairs, explaining the negligible +0.15 pp gain.

6 Discussion

Inference time dense rewards are highly effective: The Best-of-8 re-ranking strategy proved to be the central result of the paper as seen in the +13.04 pp improvement in accuracy. It clearly demonstrates that a PRM trained on automated step annotations (with no human labeling) can serve as a reliable inference time verifier. The higher AUC-ROC (0.9287) is the enabling factor, also showing that the PRM consistently scores correct reasoning solutions above incorrect ones, across diverse problem types. Critically, this gain was attained without any additional training to the SFT generator; the PRM is a drop-in downstream verifier applied only during inference.

DPO requires harder problems: The negligible improvement from the DPO (+0.15 pp, which is statistically indistinguishable at $z \approx 0.09$) is due to the lack of data collection where enough mixed outcome pairs were not generated from the SFT training. A 91%-accurate model on its own

training distribution produces almost no problems where both good and bad reasoning are observed simultaneously. To generate the diverse, high-contrast preference pairs that DPO requires, future work should source problems from held-out harder benchmarks (MATH, AIME) where the base model genuinely struggles, ensuring that sampled solutions span the full quality spectrum.

SFT regression vs. zero-shot: The SFT model underperforms, accuracy of 73.54%, as compared to the zero-shot baseline, accuracy of $\approx 85\%$. Two plausible explanations could be: the heuristic to extract the answers (`#### <int>`) could have missed the correct responses as their output format might have shifted after fine-tuning, or the LoRA training may have overfit the GSM8K solution style at the cost of generalization on harder test problems. With the Best-of-8 re-ranking approach, the implemented pipeline does surpass even the zero-shot baseline, accuracy of 86.58%. The PRM compensates for the SFT regression as compared to the zero-shot approach.

Reward hacking risk: By placing the step separator tokens at positions that can inflate step scores without improving reasoning, the PRM training runs the risk of reward hacking. In the DPO setup this risk is limited as the separators are inserted only during tokenization rather than being generated by the policy. When implementing the PPO-based online training, where the policy itself would generate the individual step boundaries dynamically, it is imperative to monitor the step-level error rates independently of final-answer accuracy to ensure no reward hacking is being performed by the policy.

7 Conclusion

This project clearly demonstrates that dense step-level rewards from a Process Reward Model do considerably improve mathematical reasoning at inference time. Training a scalar projection head on automated step annotations yields a reliable verifier (AUC-ROC $= 0.9287$), enabling the Best-of-8 re-ranking that can improve the GSM8K accuracy from 73.54% to 86.58% (+13.04 pp) without any additional model training. DPO with PRM-guided preference pairs was only able to improve the accuracy by +0.15 pp. This limitation is due to the near absence of meaningful preference contrasts (correct and incorrect pairs from SFT generation) when the base model is already saturated on the training distribution.

These results establish two key lessons: (1) PRM rewards are most effective during inference time, functioning as a plug-in downstream verifier that requires only an adept generator; and (2) training time preference learning via DPO requires a more challenging dataset entailing more complex reasoning-related problems that genuinely challenge the base model. The MATH or AIME could prove to be more appropriate target datasets for implementing the DPO. Future work will explore PPO with the PRM as an online dense reward function and extend the DPO pipeline to out-of-distribution harder problems where preference contrasts are richer and more informative. The intent is to also repeat each training stage, SFT< PRM and DPO, across various random seeds. This will help estimate run-to-run variance complementing the test set sampling uncertainty, standard error, as reported in the results.

8 Team Contributions

- **Dhruv Arcot (solo project)**

Changes from Proposal The core hypothesis and pipeline structure from the proposal remain unchanged. A notable update from the initial proposal was the choice of Qwen2.5-7B-Instruct over Llama-3-8B. This was motivated due to the former's strong zero-shot accuracy of 85% showing that it is a capable enough base model. This would also further ensure that any improvements on this number would be directly attributable only to the PRM's reward framework rather than a weaker base model. Due to the lack of mixed outcome based solutions from SFT, the potential benefits of DPO were not fully realized. PPO based online RL was also a methodology initially proposed but this was later replaced because of the results from the Best-of-N based inference re-ranking strategy yielding very good results.

References

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep Reinforcement Learning from Human Preferences. arXiv:1706.03741 [cs.LG]

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. arXiv:2110.14168 [cs.LG]

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:2305.20050 [cs.LG]

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. arXiv:2203.02155 [cs.CL]

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. arXiv:2305.18290 [cs.LG]

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2024. Rewarding Progress: Scaling Automated Process Verifiers for LLM Reasoning. arXiv:2410.08146 [cs.LG]

Qwen Team. 2025. Qwen2.5 Technical Report. arXiv:2412.15115 [cs.CL]

Peiyi Wang, Lei Li, Zhihong Sheng, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. arXiv:2312.08935 [cs.AI]

A Implementation Details

SFT hyperparameters: LoRA rank r=16, α=32; target modules: `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`. Peak learning rate 2×10^{-4} with cosine schedule and 100 warmup steps. Per-device batch size 4, gradient accumulation steps 4 (effective batch 16). Training epochs: 3.

PRM hyperparameters: Learning rate 1×10^{-4}; batch size 8. Architecture: frozen SFT backbone + `nn.Linear`$(3584, 1)$ head trained with BCE and sigmoid. Evaluated at checkpoint step 12,200.

Best-of-N sampling: Temperature $= 0.7$, top-p=0.95, N=8 solutions per problem. Solution score computed as mean of per-step $\hat{r}_t$ values (Eq. 3).

DPO hyperparameters: β=0.1; learning rate 5×10^{-5}; 3 epochs; per-device batch size 2 with gradient accumulation 8.

Compute: All stages run on Modal cloud (NVIDIA A100-40GB). SFT training ≈3 hours; PRM training to checkpoint 12,200 ≈6 hours; Best-of-8 inference on 1,319 test problems ≈2 hours.