Full-Step-DPO: Self-Supervised Preference Optimization with Step-wise Rewards for Mathematical Reasoning

Anonymous ACL submission

Abstract

Direct Preference Optimization (DPO) often struggles with long-chain mathematical reasoning. Existing approaches, such as Step-DPO, typically improve this by focusing on the first erroneous step in the reasoning chain. However, they overlook all other steps and rely heavily on humans or GPT-4 to identify erroneous steps. To address these issues, we propose Full-Step-DPO, a novel DPO framework tailored for mathematical reasoning. Instead of optimizing only the first erroneous step, it leverages step-wise rewards from the entire reasoning chain. This is achieved by training a self-supervised process reward model, which automatically scores each step, providing rewards while avoiding reliance on external signals. Furthermore, we introduce a novel step-wise DPO loss, which dynamically updates gradients based on these step-wise rewards. This endows stronger reasoning capabilities to language models. Extensive evaluations on both in-domain and out-of-domain mathematical reasoning benchmarks across various base language models, demonstrate that Full-Step-DPO achieves superior performance compared to state-of-the-art baselines [1].

1 Introduction

Large Language Models (LLMs) have attracted massive interest due to their remarkable capabilities across various tasks (Kaddour et al., 2023; Song et al., 2023; Wang et al., 2023a; Zheng et al., 2024; Wang et al., 2023b). However, they commonly encounter difficulties when tackling complex and symbolic multi-step reasoning, particularly in mathematical problem reasoning (Lightman et al., 2023; Huang et al., 2023). To improve the mathematical reasoning ability, some studies use Direct Preference Optimization (DPO) (Rafailov et al., 2024) with solution-wise preference data but find its benefit limited (Pal et al., 2024; Xu et al., 2024; Jiao et al., 2024). Recent works attribute this limitation to DPO's inability to perform process supervision and instead builds preference data based on reasoning steps rather than entire solutions (Lai et al., 2024; Chen et al., 2024; Xie et al., 2024a; Lu et al., 2024). For example, Step-DPO (Lai et al., 2024) focuses on optimizing only the first erroneous step in the reasoning chain, demonstrating notable improvements.

However, despite their improvements, these existing methods face the following limitations: (1) Some focus solely on the first erroneous step and ignore all other useful steps in the reasoning chain (Lai et al., 2024), as shown in Figure 1.

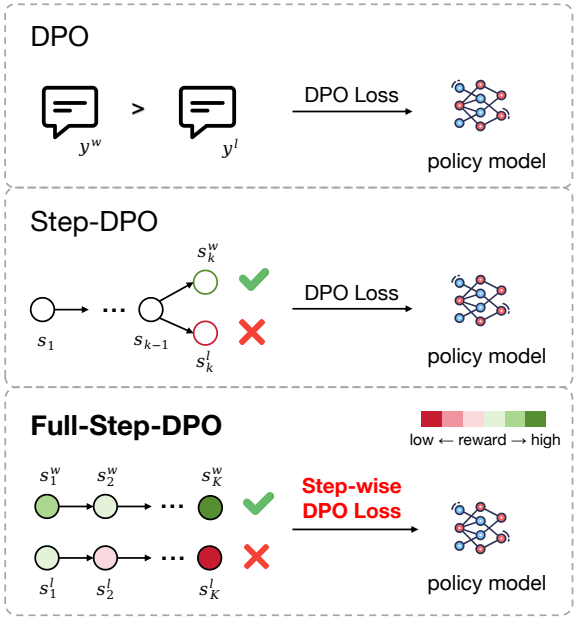

Figure 1: Comparison between DPO, Step-DPO, and our **Full-Step-DPO**. DPO operates on solution-wise preference data. Step-DPO advances to step-wise data but optimizes only a single step. Full-Step-DPO optimizes all steps with a novel step-wise DPO loss, effectively enhancing the model's reasoning capability.

[1] Our code, data, and models are available at https://github.com/anonymous.

As a result, they fail to fully optimize the reasoning chains, leading to suboptimal performance. (2) Their loss function still follows the early vanilla DPO in a solution-wise approach (Lu et al., 2024; Xie et al., 2024a). Consequently, it cannot directly leverage rewards in a step-wise fashion for learning. (3) They heavily rely on costly and resource-intensive annotations from GPT-4 or humans to detect erroneous steps (Lai et al., 2024; Lightman et al., 2023), significantly limiting their practicality.

To address the above limitations, we propose **Full-Step-DPO**, a novel DPO framework for mathematical reasoning. As illustrated in Figure 1, unlike vanilla DPO, which operates solution-wise, or Step-DPO, which focuses solely on the first erroneous step, Full-Step-DPO utilizes each step in the entire reasoning chain and optimizes them using step-wise rewards. We first train a Process Reward Model (PRM) (Lightman et al., 2023; Wang et al., 2023c) in a self-supervised way, utilizing data generated by the model itself. This approach enables the PRM to automatically score each step in the reasoning chain, eliminating the reliance on external annotations such as GPT-4 or humans. Then, we propose a novel **Step-wise DPO Loss**, which employs dynamic gradient updates to optimize each step based on its corresponding reward. This approach shifts the optimization focus from solution-wise to step-wise, enabling the policy model to achieve superior reasoning capabilities.

We conduct experiments on both in-domain and out-of-domain mathematical reasoning datasets with four widely used backbone LLMs. Experimental results demonstrate that our Full-Step DPO consistently outperforms the DPO and Step-DPO baselines, validating its effectiveness in enhancing reasoning performance. Our contributions can be summarized as follows:

- We propose the Full-Step-DPO framework with a novel step-wise DPO loss that dynamically adjusts each step's gradient based on its reward, enabling step-wise optimization rather than solution-wise and enhancing reasoning ability.
- We train a self-supervised PRM to provide step-wise rewards for preference learning and explore a more efficient approach for automatically constructing PRM training data.
- Extensive experiments on widely used mathematical benchmarks and base language models showcase the remarkable effectiveness of our method.

2 Related Work

Mathematical Reasoning Mathematical reasoning task is one of the most challenging tasks for LLMs. Various approaches have been explored to improve or elicit the mathematical reasoning ability of LLMs. A number of approaches have either continually pre-trained the base model on a vast of mathematical datasets (Azerbayev et al., 2023; Shao et al., 2024) or used supervised fine-tuning with substantial synthetic datasets distilled from cutting-edge models (Luo et al., 2023; Yu et al., 2023b; Mitra et al., 2024; Xu et al., 2024). Another line of work focuses on enhancing test-time computation by generating multiple solutions, developing separate reward models at either the outcome or process level to rerank these solutions (Cobbe et al., 2021a; Lightman et al., 2023), or employing decoding strategies guided by the reward model (Yu et al., 2023a; Xie et al., 2024b; Wang et al., 2023c; Wu et al., 2024). In addition, Reinforcement Learning's potential in general domains, demonstrated by Achiam et al. (2023) and Touvron et al. (2023), some studies have explored its use in mathematical reasoning (Wang et al., 2023c; Mitra et al., 2024; Pal et al., 2024).

Preference Learning Recently, preference learning (Ethayarajh et al., 2024) has attracted significant attention due to its ability to align with human preferences and distinguish between positive and negative examples. While these methods, like DPO (Rafailov et al., 2024), have proven effective in general domains, it offers only marginal benefits for mathematical reasoning (Pal et al., 2024). Some works (Chen et al., 2024; Lai et al., 2024) suggest that DPO's focus on coarse solution-wise preferences makes it less effective at correcting errors in multi-step reasoning, hindering reasoning improvement. Therefore, Step-DPO (Lai et al., 2024) was proposed, which first identifies the first erroneous step, and then optimizes only this erroneous step along with the corresponding correct one. Although this approach enhances mathematical reasoning capabilities, it totally overlooks the other steps in long-chain reasoning, which also provide valuable information and should not be completely disregarded. Building on this consideration, we propose Full-Step-DPO, which fully accounts for each step by dynamically optimizing all steps in the reasoning process.

Step-wise Supervision Recent findings by Lightman et al. (2023) suggest that step-wise supervision outperforms outcome-wise, due to the provision of more detailed feedback. However, training a PRM requires either costly manual annotation (Lightman et al., 2023) or significant computational resources (Khalifa et al., 2023; Wang et al., 2023c), which hinders the advancement and practical application of PRM. Therefore, in this paper, we aim to build a PRM for mathematical reasoning without relying on human annotation and with reduced computational resources. Additionally, we explore the effectiveness of the PRM in decoding and preference learning scenarios.

3 Full-Step DPO

In this section, we elaborate the proposed Full-Step DPO framework. We begin by reviewing the background of previous DPO and Step-DPO. Then we introduce the novel Step-wise DPO Loss which optimizes with step-wise rewards, and the Process Reward Model which automatically generate these step-wise rewards. Finally we outline the complete training pipeline of our Full-Step-DPO.

3.1 Preliminary

DPO. Direct Preference Optimization (DPO) (Rafailov et al., 2024) is one of the most popular preference optimization methods. Instead of learning an explicit reward model, DPO directly uses pair-wise preference data to optimize the policy model with an equivalent optimization objective. Specifically, given an input prompt x, and a preference data pair (y^w, y^l), DPO aims to maximize the probability of the entire preferred solution y^w and minimize that of the dispreferred solution y^l. The optimization objective of DPO is:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}_{(x,y^w,y^l)\sim\mathcal{D}}\Big[\log\sigma\Big(\beta\log\frac{\pi_\theta(y^w \mid x)}{\pi_{\text{ref}}(y^w \mid x)} - \beta\log\frac{\pi_\theta(y^l \mid x)}{\pi_{\text{ref}}(y^l \mid x)}\Big)\Big]$$

where $\pi_\theta(\cdot|x)$ is the policy model to be optimized, $\pi_{\text{ref}}(\cdot|x)$ is the reference model, (x, y^w, y^l) are preference pairs, σ is the sigmoid function, β is a parameter controlling the deviation from the reference model.

Step-DPO. Although DPO performs well on chat benchmarks, it is less effective for long-chain reasoning tasks like mathematical problems. Step-DPO (Lai et al., 2024) attributes this to DPO's inability to consider the sequential nature of mathematical reasoning, as rejecting an entire dispreferred solution may inadvertently penalize correct preceding steps, introducing significant noise. To address this, Step-DPO optimizes only the first incorrect step. As shown in Figure 1, given a math problem and a series of initial correct reasoning steps $\{s_1, ..., s_{k-1}\}$, Step-DPO aims to maximize the probability of the correct next step s_k^w and minimize the probability of the incorrect one s_k^l. Note that s_k^w and s_k^l refer to single steps, not all subsequent steps. The loss function used is still the vanilla DPO loss.

3.2 Step-wise DPO Loss

We now introduce the novel Step-wise DPO loss, which performs step-wise optimization using step-wise rewards. Although the motivation behind Step-DPO is reasonable, focusing solely on optimizing the first erroneous step and neglecting the valuable information provided by other steps may not be optimal. Additionally, we contend that it is not truly a step-wise DPO, as it still relies on the standard solution-wise DPO loss and resembles more of a data construction method.

To address this, we modify the vanilla DPO loss to the step-wise DPO loss, dynamically weighting the gradients of each step based on its reward, thereby enabling true step-wise optimization. Let's start with the gradient of the loss function $\mathcal{L}_{\text{DPO}}$. The gradient with respect to the parameters θ can be written as:

$$\nabla_\theta\mathcal{L} = -\beta\mathbb{E}_{(x,y^w,y^l)\sim\mathcal{D}}[\sigma\left(\hat{r}_\theta(x,y^l) - \hat{r}_\theta(x,y^w)\right) [\nabla_\theta\log\pi_\theta(y^w \mid x) - \nabla_\theta\log\pi_\theta(y^l \mid x)]]$$

where $\hat{r}_\theta(x,y) = \beta\log\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$. Intuitively, the gradient indiscriminately increases the likelihood of whole y^w and decreases the likelihood of whole y^l. To achieve dynamically weighting, we break $\nabla_\theta\log\pi_\theta(y \mid x)$ into a step-wise form and weight the gradient as follows:

$$\nabla_\theta\mathcal{L} = -\beta\mathbb{E}_{(x,y^w,y^l)\sim\mathcal{D}}\Bigg[\sigma\left(\hat{r}_\theta(x,y^l) - \hat{r}_\theta(x,y^w)\right) \Bigg[\sum_{i=1}^{K^w}\alpha_i^w\nabla_\theta\log\pi_\theta(s_i^w \mid x, s_{:<i}^w) - \sum_{i=1}^{K^l}\alpha_i^l\nabla_\theta\log\pi_\theta(s_i^l \mid x, s_{:<i}^l)\Bigg]\Bigg]$$

where s_i represents the i-th reasoning step of the solution y, $s_{:<i}$ denotes all reasoning steps preceding s_i , K is the total number of steps, and α_i is

the weight coefficient of s_i, calculated based on the reward of s_i as shown below:

$$\alpha_i = \begin{cases} \frac{e^{\gamma r_{s_i}}}{\sum_j e^{\gamma r_{s_j}}}, & s_i \in y^w \\ \frac{e^{-\gamma r_{s_i}}}{\sum_j e^{-\gamma r_{s_j}}}, & s_i \in y^l \end{cases}$$

where r_{s_i} is the reward of the step s_i, which will be introduced in the next subsection, and γ is the temperature of the Softmax operation. It is important to note that the calculation of α_i differs between the preferred solution y^w and the dispreferred solution y^l. For preferred solutions, a higher reward indicates a greater likelihood of correct reasoning in that step, so the model should perform gradient ascent with greater intensity. Conversely, for dispreferred solutions, a lower reward suggests a higher chance of incorrect reasoning, and the model should apply gradient descent with greater intensity accordingly. This approach allows us to leverage all steps and adaptively adjust the weight of each step based on its probability of correctness, achieving true step-wise optimization.

Compared to Step-DPO methods that focus solely on a single step, our method optimizes all steps simultaneously, enabling better global optimization. Noted that as $\gamma \to 0$, all steps will have equal weights, making Full-Step-DPO equivalent to vanilla DPO.

3.3 Process Reward Models

To obtain step-wise rewards, we train a Process Reward Model (PRM). The biggest challenge in training a PRM is constructing a process supervision dataset. Previous studies (Uesato et al., 2022; Lightman et al., 2023) utilize human annotators to obtain step-wise labels, which requires advanced annotator skills and is quite costly. Later, MathShepherd (Wang et al., 2023c) proposes using Monte Carlo estimation (Coulom, 2006) to automatically gather step-wise supervision, but it remained computationally expensive. In this section, we first examine the principles of Monte Carlo estimation, then present our simplified solution that significantly improves the efficiency of data construction.

Monte Carlo estimation. This approach assumes that the gold label y_{s_i} of a step s_i can be defined as the probability to deduce the correct answer a^*, and it includes both sampling and simulation phase. Specifically, given a math problem, it first randomly samples M solutions, with each solution consisting of K reasoning steps $\{s_1, s_2, \ldots, s_K\}$, and a represents the decoded answer from the last step s_K. Then, to estimate the quality of reasoning step s_i in a given solution, it simulates N subsequent reasoning processes from this step: $\{(s_{i+1,j}, \ldots, s_{K,j})\}_{j=1}^{N}$. The golden label for s_i is calculated as follows:

$$y_{s_i} = \frac{\sum_{j=1}^{N} \mathbb{I}(a_j = a^*)}{N}$$

where a_j is the decoded answer for the j-th simulated solution, and $\mathbb{I}$ is the indicator function that returns 1 if $a_j = a^*$ and 0 otherwise. This two-stage approach is highly time-consuming, as it requires N simulations for each of K step across all M solutions, resulting in a time complexity of $O(MNK)$.

Our efficient approach. It is important to note that there is a trade-off between the sampling number M and the simulation number N when computational resources are limited. A larger M can provide more data for training the PRM, while a larger N can result in higher accuracy of the labels y_i. In this paper, we found that the trained PRM performs reasonably well even with $N = 1$ when M is large, such as 32. This is likely because a larger M introduces more diversity into the training data, making the PRM more tolerant to slight reductions in data precision caused by the limited simulation number. This setting simplifies the PRM data construction by requiring only the sampling of M solutions without the need for simulation, significantly reducing computational resources and lowering the time complexity to $O(M)$. As a result, the gold label for step s_i can be simplified as follows:

$$y_{s_i} = \begin{cases} 1 & \text{if } a = a^* \\ 0 & \text{otherwise} \end{cases}$$

then the PRM could be trained as shown below:

$$\mathcal{L}_{\text{PRM}} = -\sum_{i=1}^{K} [y_{s_i} \log r_{s_i} + (1 - y_{s_i}) \log(1 - r_{s_i})]$$

where y_{s_i} is the golden label for s_i, r_{s_i} is the sigmoid score assigned by the PRM. With the above PRM, we can automatically score each step in the reasoning chain, providing reward signals for the step-wise DPO loss and enabling step-wise optimization.

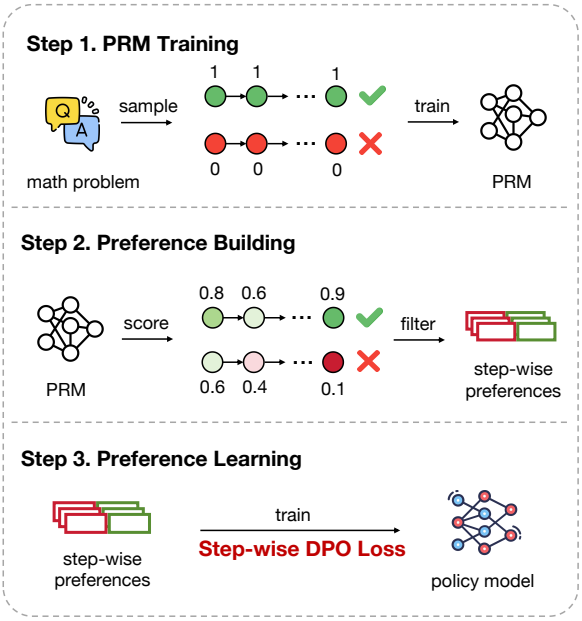

Figure 2: The overall framework of Full-Step-DPO consists of three steps: (1) Training the PRM using the model itself and generated solutions. (2) Using the PRM to score and filter solutions to form preference data with step-wise rewards. (3) Training the policy model with the proposed step-wise DPO loss.

3.4 Training Pipeline

Following previous methods (Wang et al., 2023c; Shao et al., 2024), we adopt a standard training pipeline illustrated in Figure 2: (1) We begin by training a PRM with self-generated data, where higher reward values indicate a stronger likelihood of correct reasoning, while lower values suggest potential errors. (2) The trained PRM is then used to construct preference pairs with step-wise rewards. Specifically, we generate M solutions for each math problem, score each step of these solutions with the PRM to produce a reward sequence, and calculate the average reward across all steps as the overall reward for each solution. We select the top T correct solutions with the highest rewards and the bottom T incorrect solutions with the lowest rewards to form T^2 step-wise preference pairs. (3) Finally, we update the policy model using the proposed step-wise DPO loss and the step-wise preference pairs, as described in Section 3.2.

During the inference, a well-trained PRM can guide the decoding process and enhance the model's performance. Therefore, in addition to the standard greedy decoding, we explore three alternative decoding methods: (1) Self-Consistency (SC) (Wang et al., 2022): given a problem in the test set, we sample K candidate solutions from the policy model. Instead of relying on the first decoded solution, we select the final answer based on majority voting over the answers provided by all sampled solutions. SC is a simple yet highly effective verification strategy. (2) Best-of-N (BoN): we similarly sample K candidate solutions, score them using the reward model, and select the highest-scoring solution as the final answer. Following previous work (Lightman et al., 2023; Wang et al., 2023c), we use the minimum score across all steps as the final score assigned to a solution by the PRM. (3) Step-wise Beam Search (SBS) (Yu et al., 2023a): the PRM provides feedback at each step, offering more fine-grained guidance. Specifically, for each step, we first sample b_1 candidate subsequent steps, then score them using the PRM. The top b_2 steps are retained, and decoding continues until b_2 final solutions are reached. The detailed algorithm is provided in Appendix A.

4 Experiments

4.1 Experimental Setup

Backbones. To comprehensively validate the effectiveness of our proposed method, we adopt four popular open-source LLMs as the backbone models: MetaMath-Mistral-7B (Yu et al., 2023b), Llama-3-8B (Touvron et al., 2023), DeepSeekMath-Base-7B (Shao et al., 2024) and Qwen2-7B (Bai et al., 2023). To improve these backbones' reasoning ability, Step-DPO (Lai et al., 2024) finetunes DeepSeekMath-Base-7B and Qwen2-7B on two open-source synthetic math datasets, MetaMath (Yu et al., 2023b) and MMIQC (Liu and Yao, 2024), resulting in DeepSeekMath-Base-SFT [2] and Qwen2-7B-SFT [3], which greatly outperform their previous versions. Following Step-DPO, we further finetune Llama3-8B to produce Llama3-8B-SFT. MetaMath-Mistral-7B has already been finetuned on MetaMath, so no additional finetuning was performed.

Baselines. For closed-source baselines, we compare our approach with OpenAI's GPT-3.5 and GPT-4 (Achiam et al., 2023). We also benchmarked our method against recent high-performing mathematical LLMs, including WizardMath (Luo et al., 2023), MetaMath (Yu et al., 2023b), InternLM-Math-7B (Ying et al., 2024), Qwen2-7B-

[2] https://huggingface.co/xinlai/DeepSeekMath-Base-SFT

[3] https://huggingface.co/xinlai/Qwen2-7B-SFT

Instruct(Bai et al., 2023), DeepSeekMath-Instruct (Shao et al., 2024), InternLM-Math-20B (Ying et al., 2024), and Llama-3-70B-Instruct (Touvron et al., 2023).

Additionally, we compare it against DPO (Rafailov et al., 2024) and Step-DPO (Lai et al., 2024). Among these, Lai et al. (2024) publicly release DeepSeekMath-Base-SFT-Step-DPO [4] and Qwen2-7B-SFT-Step-DPO [5], which we directly used for evaluation. Additionally, we trained MetaMath-Mistral-7B-Step-DPO and Llama-3-8B-SFT-Step-DPO using their publicly available code and dataset.

Datasets. To ensure a fair comparison, we use the same training dataset [6] provided by Step-DPO (Lai et al., 2024), which is synthesized from the training set of GSM8K (Cobbe et al., 2021b) and MATH (Hendrycks et al., 2021). Noted that we only use the problem prompts in this dataset and do not use the step labels marked by GPT-4.

For in-domain evaluation, we conduct experiments on GSM8K and MATH, which contain 1,319 and 5,000 test problems, respectively. We also evaluate on two more challenging out-of-domain (OOD) test sets OCWCourses (OCW) (Lewkowycz et al., 2022) and GaoKao2023 (GK2023) (Liao et al., 2024). OCW contains of 272 undergraduate-level STEM problems requiring multi-step reasoning for most questions, while GK2023 includes 385 mathematics problems from the 2023 Chinese higher education entrance exam, translated into English. Accuracy serves as the evaluation metric.

Implementation Details. During PRM training, we first randomly sample $M = 32$ solutions for each math problem using Qwen2-7B-SFT and then label them as described in Section 3.3, resulting in the PRM training set. Then, we add a classification-head to Qwen2-7B-SFT and train it on the PRM training set for one epoch. The batch size is 256, and the learning rate is 5e-7.

To build preference learning datasets, we first sample $M = 32$ solutions for each math problem. The trained PRM then scores each solution, and we select $T = 4$ solutions with the highest average rewards and $T = 4$ with the lowest average rewards to randomly form 16 preference pairs.

During preference learning, the batch size is 64, the learning rate is 5e-7, β is 0.05, and the reward temperature γ is 0.5. We use the AdamW (Loshchilov and Hutter, 2017) optimizer with a linear decay learning rate scheduler and only train one epoch. The warm-up ratio is 0.05.

During the decoding phase, we conduct experiments with two settings for SC and BoN, using $K = 5$ and $K = 15$. For Step-wise Beam Search, to ensure fair comparison, we test two configurations: $b_1 = 5, b_2 = 1$ (corresponding to $K = 5$) and $b_1 = 5, b_2 = 3$ (corresponding to $K = 15$). The sampling temperature is set to 0.8.

All the experiments are conducted on a server equipped with 8 NVIDIA A100-80GB GPUs and 512GB of system RAM. The implementation frameworks are PyTorch (Paszke et al., 2017), DeepSpeed (Rasley et al., 2020), and Huggingface (Wolf et al., 2019).

4.2 Main Results

Table 1 provides a comprehensive comparison of various models on both MATH and GSM8K, including open-source and closed-source LLMs. We find that: (1) Consistent with previous studies (Pal et al., 2024), DPO exhibits notable instability. Its performance shows slight degradation on MetaMath-Mistral-7B and MetaMath-Mistral-7B-SFT backbones, while the accuracy drops sharply to around 20% on Qwen2-7B-SFT. It achieves a slight performance improvement only when applied to the DeepSeekMath-Base-SFT. (2) Step-DPO achieves only minimal improvements across all backbones, with gains generally around 1% and, in some settings, even slight performance drops. We evaluate the publicly released Step-DPO model using its official script, and the results may differ slightly from those reported in the Step-DPO paper. Similar issues have also been observed by other researchers [7]. (3) Our Full-Step-DPO consistently outperforms Step-DPO across all backbones. Specifically, when applied to MetaMath-Mistral-7B and Llama-3-8B-SFT, our model achieves improvements of approximately 2.3% to 3.7%, while applied to the stronger backbones, DeepSeekMath-Base-SFT and Qwen2-7B-SFT, our method still delivers gains exceeding 1%. These results clearly

[4] https://huggingface.co/xinlai/DeepSeekMath-Base-SFT-Step-DPO

[5] https://huggingface.co/xinlai/Qwen2-7B-SFT-Step-DPO

[6] https://huggingface.co/datasets/xinlai/Math-Step-DPO-10K

[7] https://github.com/dvlab-research/Step-DPO/issues/2

Model	MATH (%)	GSM8K (%)
GPT-3.5	34.1	80.8
GPT-4	53.6	93.6
WizardMath	10.7	54.9
MetaMath	19.8	66.5
InternLM-Math-7B	34.6	78.1
Qwen2-7B-Instruct	49.6	82.3
DeepSeekMath-Instruct	46.8	82.9
InternLM-Math-20B	37.7	82.6
Llama-3-70B-Instruct	50.4	93.0
MetaMath-Mistral-7B	28.2	77.7
+ DPO	24.8 -3.4	70.7 -7.0
+ Step-DPO	28.9 +0.7	79.6 +1.9
+ Full-Step-DPO	**30.5** +2.3	**81.4** +3.7
Llama-3-8B-SFT	32.6	78.5
+ DPO	23.4 -9.2	62.3 -16.2
+ Step-DPO	31.8 -0.8	80.1 +1.6
+ Full-Step-DPO	**35.0** +2.4	**82.0** +3.5
DeepSeekMath-Base-SFT	51.7	86.4
+ DPO	51.7 -0	87.3 +0.9
+ Step-DPO	52.9 +1.2	86.6 +0.2
+ Full-Step-DPO	**53.2** +1.5	**87.9** +1.5
Qwen2-7B-SFT	53.9	88.3
+ DPO	20.0 -23.9	27.3 -61.0
+ Step-DPO	54.9 +1.0	88.4 +0.1
+ Full-Step-DPO	**55.4** +1.5	**89.3** +1.0

Table 1: Performance comparison of various models on MATH and GSM8K with greedy decoding.

demonstrate the effectiveness of our proposed approach, which considers all steps in the reasoning process rather than focusing on solution-wise preferences or only a single step. A case study can be found in Appendix B.3.

4.3 Results on OOD Datasets

To further demonstrate the superiority of Full-Step-DPO, we evaluate the models on OOD datasets GK2023 and OCW, as shown in Table 2. On these competition-level math problems, DPO and Step-DPO often exhibit performance degradation under various settings, while our Full-Step-DPO consistently achieves performance improvements. The only exception occurs on the OCW dataset with MetaMath-Mistral-7B, where Full-Step-DPO shows a slight 0.8% drop in accuracy. However, this drop is notably smaller than 3.0% with DPO and the 3.7% with Step-DPO. These results demonstrate the superior stability and resilience of Full-Step-DPO, particularly in handling challenging mathematical reasoning tasks. More experimental results on additional datasets can be found in Appendix B.1.

Model	GK2023 (%)	OCW (%)
MetaMath-Mistral-7B	15.8	**10.7**
+ DPO	15.8 -0	7.7 -3.0
+ Step-DPO	15.1 -0.7	7.0 -3.7
+ Full-Step-DPO	**20.5** +4.7	9.9 -0.8
Llama-3-8B-SFT	20.5	12.5
+ DPO	11.7 -8.8	9.9 -2.6
+ Step-DPO	19.7 -0.8	13.6 +1.1
+ Full-Step-DPO	**22.1** +1.6	**15.1** +2.6
DeepSeekMath-Base-SFT	30.4	19.1
+ DPO	31.2 +0.8	18.4 -0.7
+ Step-DPO	31.2 +0.8	18.0 -1.1
+ Full-Step-DPO	**31.7** +1.3	**20.2** +1.1
Qwen2-7B-SFT	33.0	15.8
+ DPO	8.8 -24.2	8.1 -7.7
+ Step-DPO	32.5 -0.5	15.8 -0
+ Full-Step-DPO	**33.5** +0.5	**18.4** +2.6

Table 2: Performance comparison on out-of-domain math problems.

4.4 Results on Various Verification Strategies

Figure 3 presents the performance of different verification strategies on GSM8K under two settings: $K = 5$ and $K = 15$. We find that: (1) SC serves as a simple yet powerful validation method that significantly improves performance across all models. Even for the high-performing Qwen2-7B-SFT, which achieves an accuracy of 89.3% with greedy decoding, SC further improves the accuracy to 93% when the sampling size $K = 15$. This result is already comparable to GPT-4's accuracy of 93.6%. (2) Compared to SC, BoN often achieves further improvements on MetaMath-Mistral-7B and Llama-3-8B-SFT. However, on the highly capable DeepSeekMath-Base-SFT and Qwen2-7B-SFT, BoN underperforms SC, indicating that the benefits of the reward model diminish for very strong baseline models. (3) SBS performs worse than both SC and BoN across most settings, yet consistently surpasses Greedy decoding, aligning with findings from previous studies (Yu et al., 2023a; Khalifa et al., 2023). This may be because, during the early stages of inference, the reward model struggles to effectively distinguish the correctness of steps.

4.5 Analysis of PRMs

As discussed in Section 3.3, the quality of the PRM may be influenced by the sampling number M and simulation number N. To assess this, we conducted a controlled experiment with fixed $M = 32$ and varying N. The trained PRM is then used to guide the decoding of MetaMath-Mistral-7B-Full-Step-

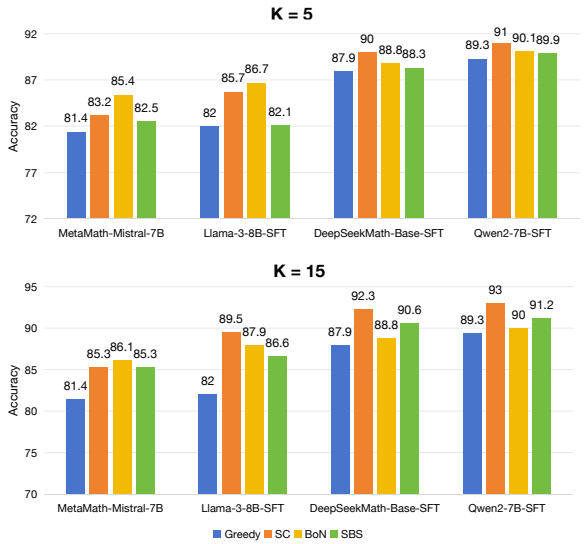

Figure 3: Performance comparison of various verifications on GSM8K, with all models trained using our Full-Step-DPO.

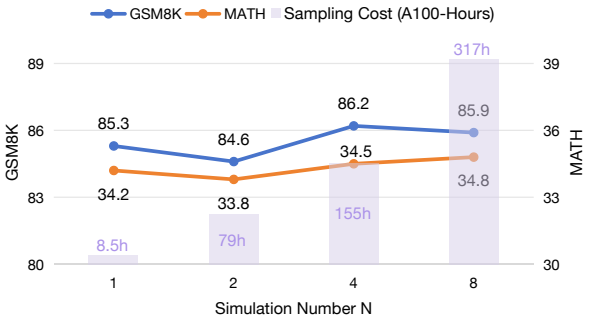

Figure 4: Accuracy of MetaMath-Mistral-7B-Full-Step-DPO using BoN decoding with $K = 15$. The PRM uses a fixed sampling number $M = 32$, while the simulation number N varies. Purple bars indicate the A100-Hours cost for constructing PRM training data.

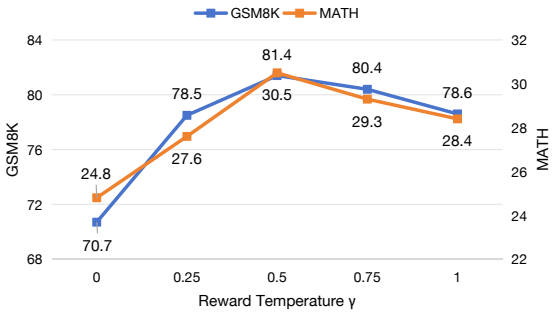

Figure 5: Accuracy of MetaMath-Mistral-7B-Full-Step-DPO with different reward temperature γ.

DPO using BoN strategy with $K = 15$.

As shown in Figure 4, the purple bars indicate the A100-Hours cost for constructing PRM training data, while blue and orange lines show GSM8K and MATH accuracy. When $N = 1$, which represents our proposed method, the model achieves competitive performance with 85.3% accuracy on GSM8K and 34.2% on MATH, while only requiring 8.5h with a time complexity of $O(M)$. As N increases, we observe that the performance fluctuation remains relatively minimal (within approximately 1% range), while the sampling cost grows substantially with a complexity of $O(MNK)$. This empirical evidence suggests that our approach with $N = 1$ achieves a good balance between sampling efficiency and model performance. Additionally, we provide a comprehensive comparison between our PRM and other publicly available PRMs in Appendix B.2.

4.6 Sensitivity of Hyperparameters

In step-wise DPO loss, the reward temperature γ reflects the level of trust in the PRM. As γ increases, the PRM model has a greater impact on the gradients. When $\gamma \to 0$, it indicates complete distrust in the PRM model, assigning equal weight coefficient to all steps, degrading in vanilla DPO. Conversely, when $\gamma \to \infty$, the loss function optimizes only the single step with the maximum or minimum reward in the solution, similar to Step-DPO. Figure 5 presents the accuracy of MetaMath-Mistral-7B-Full-Step-DPO with different γ values. The experimental results indicate that introducing a PRM to weight the gradients indeed effectively enhances optimization efficiency and improves performance. Additionally, this experiment demonstrates that there is a sweet spot for the reward temperature γ; excessively high or low γ will reduce accuracy.

5 Conclusion

In this work, we propose Full-Step-DPO, a novel framework for mathematical reasoning that optimizes each step in the entire reasoning chain using step-wise rewards. To achieve this, we train a self-supervised Process Reward Model to automatically score reasoning steps, eliminating reliance on external annotations. We also propose a novel Step-Wise DPO Loss that dynamically updates gradients based on the rewards for individual steps, enabling step-wise optimization and enhancing the reasoning ability of policy models. Experimental results on various benchmarks validate the effectiveness of Full-Step-DPO, paving the way for its application to other reasoning-intensive tasks.

Limitations

While we have conducted comprehensive experiments to demonstrate the effectiveness of Full-Step-DPO, several limitations remain. First, recent advancements suggest that generative reward models outperform the discriminative reward model used in this work. Exploring how generative reward models can further enhance mathematical reasoning capabilities would be a valuable direction for future research. Second, during preference data construction, the current strategy of selecting samples based on average reward is relatively simple. Investigating more advanced sample selection strategies may lead to further improvements. Finally, the step-wise DPO loss proposed in this paper is highly adaptable to other reasoning tasks, such as code generation. Conducting experiments on a broader range of tasks would provide additional evidence of the advantages of our method.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024. Step-level value preference optimization for mathematical reasoning. *arXiv preprint arXiv:2406.10858*.

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

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

Rémi Coulom. 2006. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

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

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*.

Fangkai Jiao, Chengwei Qin, Zhengyuan Liu, Nancy F Chen, and Shafiq Joty. 2024. Learning planning-based reasoning by trajectories collection and process reward synthesizing. *arXiv preprint arXiv:2402.00658*.

Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*.

Muhammad Khalifa, Lajanugen Logeswaran, Moontae Lee, Honglak Lee, and Lu Wang. 2023. Grace: Discriminator-guided chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15299–15328.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. 2024. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *arXiv preprint arXiv:2406.18629*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Minpeng Liao, Wei Luo, Chengxi Li, Jing Wu, and Kai Fan. 2024. Mario: Math reasoning with code interpreter output–a reproducible pipeline. *arXiv preprint arXiv:2401.08190*.

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*.

Haoxiong Liu and Andrew Chi-Chih Yao. 2024. Augmenting math word problems via iterative question composing. *arXiv preprint arXiv:2401.09003*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Zimu Lu, Aojun Zhou, Ke Wang, Houxing Ren, Weikang Shi, Junting Pan, Mingjie Zhan, and Hongsheng Li. 2024. Step-controlled dpo: Leveraging stepwise error for enhanced mathematical reasoning. *arXiv preprint arXiv:2407.00782*.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2021. A diverse corpus for evaluating and developing english math word problem solvers. *arXiv preprint arXiv:2106.15772*.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. 2024. Orca-math: Unlocking the potential of slms in grade school math. *arXiv preprint arXiv:2402.14830*.

Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddartha Naidu, and Colin White. 2024. Smaug: Fixing failure modes of preference optimisation with dpo-positive. *arXiv preprint arXiv:2402.13228*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. *NIPS 2017 Workshop Autodiff Submission*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Y Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Yifan Song, Weimin Xiong, Dawei Zhu, Cheng Li, Ke Wang, Ye Tian, and Sujian Li. 2023. Restgpt: Connecting large language models with real-world applications via restful apis. *arXiv preprint arXiv:2306.06624*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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. *arXiv preprint arXiv:2211.14275*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023b. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*.

Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Y Wu, and Zhifang Sui. 2023c. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *CoRR, abs/2312.08935*.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Zhenyu Wu, Qingkai Zeng, Zhihan Zhang, Zhaoxuan Tan, Chao Shen, and Meng Jiang. 2024. Enhancing mathematical reasoning in llms by stepwise correction. *arXiv preprint arXiv:2410.12934*.

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. 2024a. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. 2024b. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36.

Yifan Xu, Xiao Liu, Xinghan Liu, Zhenyu Hou, Yueyan Li, Xiaohan Zhang, Zihan Wang, Aohan Zeng, Zhengxiao Du, Wenyi Zhao, et al. 2024. Chatglm-math: Improving math problem-solving in large language models with a self-critique pipeline. *arXiv preprint arXiv:2404.02893*.

Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma, Jiawei Hong, Kuikun Liu, Ziyi Wang, et al. 2024. Internlm-math: Open math large language models toward verifiable reasoning. *arXiv preprint arXiv:2402.06332*.

Fei Yu, Anningzhe Gao, and Benyou Wang. 2023a. Outcome-supervised verifiers for planning in mathematical reasoning. *arXiv preprint arXiv:2311.09724*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023b. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

A Step-wise Beam Search

Unlike conventional beam search, which relies on token-level probabilities, our method integrates the reward model with an associated reranking criterion. This enables for step-wise beam search (SBS) (Yu et al., 2023a; Chen et al., 2024), effectively selecting the preferred solution path in mathematical reasoning, while incurring a lower computational cost compared to Monte Carlo Tree Search. Specifically, for each step t, suppose the sampling size is b_1, the policy model π_θ produces a set of candidate steps $\mathbb{S}^{(1:t+1)} = \{\mathcal{S}_i^{(1:t+1)}\}_{i=1}^{b_1}$, where $\mathcal{S}_i^{(1:t+1)} = \left[s_i^1, ..., s_i^{t+1}\right]$ is the i-th partial solution up to step $t+1$. Given the PRM π_r that can score each step, we select the top-scoring steps with beam size b_2. The algorithm is detailed in Algorithm 1. By focusing on the quality of each reasoning step rather than just the final solution, our method enhances the overall reasoning capabilities of the model.

Algorithm 1 Step-wise Beam Search

Input: Math problem q, Sampling size b_1, Beam size b_2, Maximum step C
Output: Best solution for q
Models: Policy model π_θ and PRM π_r
function StepLevelBeamSearch(q, b_1, b_2, C)
    Initialize step sequences $\mathbb{S} \leftarrow \{\}$
    Use π_θ to sample initial steps $\{s_1^1, \ldots, s_{b_1}^1\}$
    Use π_r to score all initial steps $\{r_1^1, \ldots, r_{b_1}^1\}$
    Select top-b_1 steps and add to $\mathbb{S}$
    Set current step counter $t \leftarrow 1$
    while $t < C$ **do**
        if All sequences in $\mathbb{S}$ are complete **then**
            Break
        end if
        $\mathbb{S}_{\text{new}} \leftarrow \{\}$
        $\mathbb{R} \leftarrow \{\}$;
        for each solution $\mathcal{S}^{(1:t)}$ in $\mathbb{S}$ **do**
            for $i = 1$ to b_1 **do**
                $\mathcal{S}_i^{(1:t+1)} = \pi_\theta(\mathcal{S}^{(1:t)}; q)$
                $r_i^{(1:t+1)} = \pi_r(\mathcal{S}_i^{(1:t+1)}; q)$
                $\mathbb{S}_{\text{new}} \leftarrow \mathbb{S}_{\text{new}} + \{\mathcal{S}_i^{(1:t+1)}\}$
                $\mathbb{R} \leftarrow \mathbb{R} + \{r_i^{(1:t+1)}\}$
            end for
        end for
        $\mathbb{S}_{\text{new}} \leftarrow$ top-b_2 rewarded solutions in $(\mathbb{S}_{\text{new}}, \mathbb{R})$
        $\mathbb{S} \leftarrow \mathbb{S}_{\text{new}}$
        $t \leftarrow t + 1$;
    end while
    return solution with highest final reward in $\mathbb{S}$
end function

B Additional Experiments

B.1 More OOD Datasets

We evaluate our method on five additional OOD mathematical reasoning datasets as our testbed. As shown in Table 3, our Full-Step-DPO consistently improves performance across all datasets, demonstrating the effectiveness and generalization ability of our approach on OOD mathematical reasoning tasks.

- SVAMP (Patel et al., 2021) includes 1000 math questions of up to fourth grade difficulty. These questions can be solved by expressions requiring no more than two operators.

- AddSub (Hosseini et al., 2014) contains 395 math questions that involve addition and subtraction operations.

- ASDiv (Miao et al., 2021) contains 2215 English math questions of different problem types. Each question provides the corresponding equation and answer.

- GSM-IC2 and GSM-ICM (Shi et al., 2023) are mathematical reasoning datasets containing irrelevant conditions within the problem descriptions each consisting of 1000 problems. Problems in GSM-IC2 require two steps to solve, while problems in GSM-ICM require more than two steps to solve.

Model	SVAMP (%)	AddSub (%)	ASDiv (%)	GSM-IC2 (%)	GSM-ICM (%)	Average (%)
MetaMath-Mistral-7B	79.1	86.6	81.2	**77.9**	76.5	80.3
+ DPO	72.7	53.4	73.8	60.5	65.2	65.1
+ Step-DPO	80.3	86.9	82.7	76.4	76.3	80.5
+ Full-Step-DPO	**81.7**	**88.6**	**83.9**	75.9	**76.6**	**81.3**
Llama-3-8B-SFT	82.8	88.4	85.0	80.1	79.5	83.2
+ DPO	72.2	60.0	74.2	64.1	66.3	67.4
+ Step-DPO	81.7	88.5	85.3	80.7	81.2	83.5
+ Full-Step-DPO	**82.9**	**88.8**	**86.4**	**82.1**	**81.6**	**84.4**
DeepSeekMath-Base-SFT	84.2	**87.6**	91.0	85.4	85.2	86.7
+ DPO	85.1	86.6	90.6	85.2	85.0	86.5
+ Step-DPO	85.3	85.3	90.7	85.9	**86.2**	86.7
+ Full-Step-DPO	**85.9**	86.6	**91.2**	**87.8**	85.3	**87.4**
Qwen2-7B-SFT	88.7	**92.7**	91.6	93.7	91.6	91.7
+ DPO	23.5	25.6	27.0	30.1	29.5	27.1
+ Step-DPO	88.1	92.2	91.8	**93.8**	91.9	91.6
+ Full-Step-DPO	**89.5**	**93.1**	**92.4**	93.6	**92.7**	**92.3**

Table 3: Performance comparison on five additional OOD math problems. The best and second-best results are highlighted in bold and underlined, respectively.

B.2 Performance Comparison of Different PRMs

We compare our PRM with Math-Shepherd-PRM-7B using MetaMath-Mistral-7B as the base model under the Best-of-8 decoding strategy. During decoding, we sampled eight responses with temperature = 0.7. For both PRMs, each response was scored by the minimum reward among its reasoning steps, and the response with the highest score was selected as the final output.

As shown in Table 4, our PRM achieves 84.3% accuracy on GSM8K and 33.2% on MATH, outperforming Math-Shepherd-PRM-7B (82.4% on GSM8K, 32.6% on MATH). While these results suggest the effectiveness of our approach, we acknowledge that the comparison may not be entirely comprehensive due to differences in backbone architecture and training data between the PRMs. Nevertheless, these findings, combined with our analysis in Section 4.5, demonstrate that our PRM can achieve competitive performance while significantly reducing the computational cost of PRM construction.

Model	MATH	GSM8K
MetaMath-Mistral-7B	28.2	77.7
+ Self-Consistency	32.4	81.3
+ Math-Shepherd-PRM-7B	32.6	82.4
+ Our-PRM-7B	**33.2**	**84.3**

Table 4: Performance comparison of different scoring strategies using Best-of-8 decoding on MATH and GSM8K benchmarks.

B.3 Case Study

We conducted a detailed case study using a sample problem from the AddSub dataset, as shown in Table 5. We compared outputs from DeepSeekMath-Base-SFT model and its Step-DPO and Full-Step-DPO variants. While both the base SFT model and Step-DPO model made incorrect reasoning by including the cost of unbought CDs, our Full-Step-DPO model generated correct reasoning steps. Our PRM scores accurately reflect the correctness of each reasoning step, with higher scores for correct steps and lower scores for incorrect ones.

Question	For his car, Mike spent $118.54 on speakers and $106.33 on new tires. Mike wanted 3 CD's for $4.58 but decided not to. In total, how much did Mike spend on car parts?
Gold Answer	224.87
SFT Response	**Step 1**: Mike spent $118.54 on speakers and 106.33 on new tires. So the total cost of these two items is $118.54 + $106.33 = $224.87. **Score: 0.9348**
	Step 2: Mike wanted to buy 3 CD's for $4.58 each, but he decided not to. So the total cost of the CD's he didn't buy is 3 * $4.58 = $13.74. **Score: 0.3646**
	Step 3: To find out how much Mike spent in total on car parts, we need to add the total cost of the speakers and tires to the total cost of the CD's he didn't buy. So the total cost is $224.87 + $13.74 = $238.61. The answer is: 238.61 **Score: 0.0427**
Step-DPO Response	**Step 1**: First, Mike spent $118.54 on speakers and $106.33 on new tires. So the total cost of these two items is $118.54 + $106.33 = $224.87. **Score: 0.9406**
	Step 2: Mike also wanted 3 CD's, but he decided not to buy them. Each CD costs $4.58, so the total cost of the CD's he didn't buy is 3 * $4.58 = $13.74. **Score: 0.3036**
	Step 3: To find out how much Mike spent in total on car parts, we need to add the total cost of the speakers and tires to the total cost of the CD's he didn't buy. So, the total cost is $224.87 + $13.74 = $238.61. The answer is: 238.61 **Score: 0.0445**
Full-Step-DPO Response	**Step 1**: Mike spent $118.54 on speakers and $106.33 on new tires. So the total amount he spent on these two items is $118.54 + $106.33 = $224.87. **Score: 0.9313**
	Step 2: Mike wanted to buy 3 CD's for $4.58 each, but he decided not to. So the total amount he would have spent on the CD's is 3 * $4.58 = $13.74. **Score: 0.7408**
	Step 3: To find out how much Mike spent in total on car parts, we need to add the total amount he spent on speakers and tires to the amount he would have spent on CD's. So the total amount is $224.87 + $0 = $224.87. The answer is: 224.87 **Score: 0.7363**

Table 5: Comparison of reasoning steps and their corresponding PRM scores across different models for AddSub dataset. The base model is DeepSeekMath-Base-SFT.