Enhancing LLM Reasoning with Iterative DPO: A Comprehensive Empirical Investigation

Songjun Tu[♠♣♦], **Jiahao Lin**[♦♠], **Xiangyu Tian**[♦♠], **Qichao Zhang**[♠♦], **Linjing Li**[♠♦], **Yuqian Fu**[♠♦], **Nan Xu**[♡], **Wei He**[♢], **Xiangyuan Lan**[♣], **Dongmei Jiang**[♣], **Dongbin Zhao**[♠♣♦]
[♠] Institute of Automation, Chinese Academy of Sciences [♣] Pengcheng Laboratory
[♦] School of Artificial Intelligence, University of Chinese Academy of Sciences
[♡] Wenge Technology [♢] Fudan University
{tusongjun2023,zhangqichao2014}@ia.ac.cn, lanxy@pcl.ac.cn

Abstract

Recent advancements in post-training methodologies for large language models (LLMs) have highlighted reinforcement learning (RL) as a critical component for enhancing reasoning. However, the substantial computational costs associated with RL-based approaches have led to growing interest in alternative paradigms, such as Direct Preference Optimization (DPO). In this study, we investigate the effectiveness of DPO in facilitating self-improvement for LLMs through iterative preference-based learning. We demonstrate that a single round of DPO with coarse filtering significantly enhances mathematical reasoning performance, particularly for strong base model. Furthermore, we design an iterative enhancement framework for both the generator and the reward model (RM), enabling their mutual improvement through online interaction across multiple rounds of DPO. Finally, with simple verifiable rewards, our model DPO-VP achieves RL-level performance with significantly lower computational overhead. These findings highlight DPO as a scalable and cost-effective alternative to RL, offering a practical solution for enhancing LLM reasoning in resource-constrained situations. Code available at: https://github.com/TU2021/DPO-VP.

1 Introduction

The field of large language models (LLMs) is evolving at an unprecedented pace (Hurst et al., 2024; Xi et al., 2025). With the release of O1 (Jaech et al., 2024) by OpenAI in September 2024, the research focus shifted from pre-training to post-training. This transition was further accelerated in January 2025, when DeepSeek-R1 (Guo et al., 2025) brought post-training methodologies to the forefront of LLM research. Then, several studies have explored reinforcement learning (RL) (Gandhi et al., 2025; Zeng et al., 2025b; Tu et al., 2025b) at scale and supervised fine-tuning (SFT) (Muennighoff et al., 2025; Ye et al., 2025; Fu et al., 2025a) with high-quality long-chain data, unlocking emergent capabilities such as self-checking, reflection, and verification in strong base models (e.g., Qwen2.5 (Yang et al., 2024a)). These advancements highlight the potential of post-training in enhancing LLM reasoning.

Generally, searching self-generated data for training within post-training is referred to self-improvement (Yang et al., 2024b). At its core, this approach exploits search and learning to leverage the power of general methods as computational resources increase — resonating with the principle articulated by Sutton (2019) in *The Bitter Lesson*. Naturally, search and expansion can be achieved through Monte Carlo Tree Search (MCTS) and RL. MCTS-based test-time scaling and reward model (RM) training were once considered key contributors behind O1 (Zhang et al., 2024a; Cui et al., 2025). But the high inference cost and training complexity gradually pushed MCTS out of the research spotlight. In contrast, RL with

∗ Corresponding authors: Qichao Zhang and Xiangyuan Lan.

verifiable rewards has been empirically shown to be more efficient and promising (Guo et al., 2025; Xie et al., 2025). However, current RL training still requires substantial computational resources for reproduction. For instance, SimpleRL (Zeng et al., 2025b) trains for 1.5 days on 4×8 H100 GPUs, while PURE (Cheng et al., 2025) also demands 8 A100 GPUs. Achieving comparable or even superior self-improvement performance with lower computational resources and reduced training time is an ideal goal pursued in this field.

Several methods have been introduced as alternatives to RL to alleviate computational overhead, such as DPO (Rafailov et al., 2023), KTO (Ethayarajh et al., 2024), and IPO (Azar et al., 2024). By modeling implicit rewards, these approaches remove the dependency on explicit reward and value functions of RL and are collectively known as offline maximum likelihood estimation (MLE) methods. Through multi-round iterative training, offline MLE methods can be approximated as off-policy online RL approaches (Pang et al., 2024). Recently, Swamy et al. (2025) demonstrated the theoretical equivalence between online RL and offline MLE under ideal optimization conditions. While the role of RL in post-training has been extensively studied, the impact of the more streamlined and efficient DPO in post-training remains uncertain.

In this paper, we focus on empirically investigating the improvements brought by DPO in post-training, aiming to provide the LLM community with a deeper performance analysis of offline MLE self-improvement paradigms. Our key findings are as follows:

1. **Single-Round DPO with Coarse Filtering Can Enhance Strong Base Models**: With simple RM or even outcome labels to filter self-generated datasets, we achieve a significant improvement in mathematical reasoning of Qwen2.5 with single-round DPO training.
2. **Iterative Improvement of Generator and RM can be Achieved via Multi-Round DPO**: By relabeling the RM with online data, the mutual enhancement of the generator and RM can be realized through multiple rounds of DPO. In contrast, traditional RL frameworks typically rely on a fixed RM, making it difficult to update the RM in an online manner.
3. **Multi-Round DPO Achieves RL-Level Performance with Lower Computational Cost**: Our final model DPO-VP, based on Qwen2.5-Math-7B, achieves comparable average performance to SimpleRL-Zero, PURE-VR and DPO-R1-Zero across 5 challenging benchmarks (48.2 vs 48.8/47.7/47.0), while requiring only a single 80GB GPU for training.

2 Methods

The overall iterative DPO training framework is illustrated in Figure 1. We first sample multiple responses from the base model, annotate them using rule-based or reward-based labeling. The policy is then optimized through DPO iterations, alternating between RM updates and policy refinement, enabling continuous self-improvement in reasoning ability.

2.1 Iterative DPO with External Annotation

For each question Q_k, the base model π_0 generates a set of candidate responses $\{r_{1,k}, r_{2,k}, \ldots, r_{n,k}\}$. Responses are evaluated using rule-based outcome labels or reward models (e.g. Process Reward Model, PRM or Outcome Reward Model, ORM) that assign preference scores. Specifically, for ORM, we select the response with the highest score $s_{\text{ORM}}(r)$ as r^+, and the response with the lowest score as r^-. For PRM, where each step in a response is assigned a process reward score, we define the evaluation function f as follows:

$$f(s_{\text{PRM}}(r)) = \min\{s_{\text{PRM}}(r^1), \ldots, s_{\text{PRM}}(r^n)\} \tag{1}$$

where n represents the total number of steps in response r. Then, the best response r^+ and the worst response r^- are selected to construct the refined dataset, which is subsequently used to update the policy π_0 using DPO loss:

$$\mathcal{L}_{\text{DPO}}(\pi) = \mathbb{E}_{(Q_k, r^+, r^-)} \left[\log \sigma \left(\beta \cdot \left(\log \frac{\pi(r^+ \mid Q_k)}{\pi_0(r^+ \mid Q_k)} - \log \frac{\pi(r^- \mid Q_k)}{\pi_0(r^- \mid Q_k)} \right) \right) \right] \tag{2}$$

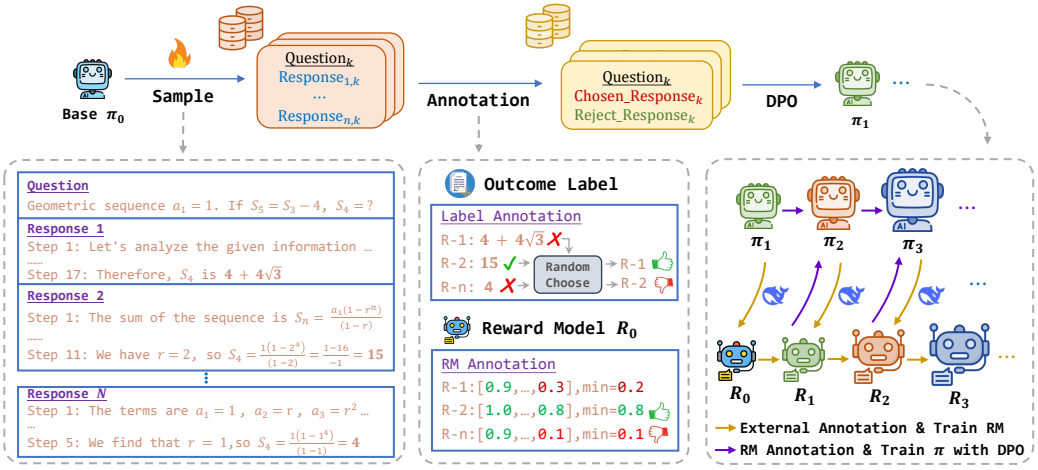


Figure 1: Iterative DPO with Outcome Label and RM-Based Annotation

where σ is the sigmoid function and β is a temperature scaling parameter. The entire process is conducted over N iterations, where the policy is progressively refined through repeated optimization. Finally, we obtain the final optimized policy π_N.

2.2 Training Reward Model with Online Generated Data

Furthermore, we can train the reward model using data generated by the online generator. Taking PRM as an example, we leverage external annotators (e.g., DeepSeek-V3 (Liu et al., 2024a)) to label the responses sampled from the current policy π_t. This process produces a new round of PRM training data, which is then used to fine-tune the previous PRM R_{t-1}, yielding an updated PRM R_t. The training loss for updating the PRM is defined as follows:

$$\mathcal{L}_{\text{PRM}}(R_t) = \mathbb{E}_{(Q_k, r^+, r^-)} \left[-\log \sigma \left(R_t(Q_k, r^+) - R_t(Q_k, r^-) \right) \right] \tag{3}$$

By iteratively updating the reward model and selecting the data for next epoch, we progressively improve its ability to assess process-level reasoning. In Section 3.4, we explore how this online training paradigm strengthens the PRM's capability to capture and differentiate reasoning quality.

3 Experiments

3.1 Setup

In this section, we systematically investigate the impact of DPO on different base models. To ensure a comprehensive analysis, we examine models with varying parameter sizes and architectures, including Qwen2.5-3B (Yang et al., 2024a), Qwen2.5-7B (Yang et al., 2024a), LLaMA3.1 (Grattafiori et al., 2024), and Qwen2.5-Math-7B (Yang et al., 2024b). The structure of the following sections is as follows:

- In Section 3.2, we explore the simplest approach to improving base model performance by examining the effects of Chain-of-Thought (CoT) prompting and supervised fine-tuning (SFT) on different models, establishing a baseline for subsequent experiments.
- In Section 3.3, we demonstrate that a single round of DPO with coarse filtering is sufficient to significantly enhance Qwen2.5's mathematical reasoning performance through self-improvement.
- In Section 3.4, we investigate how multi-round DPO and iterative reward model training enable continuous iterative improvement between the generator and the reward model.

- In Section 3.5, we compare the final multi-round DPO results with state-of-the-art reinforcement learning methods and instruct models, analyzing potential differences in reasoning paradigms.

3.2 CoT Prompts Improve Qwen2.5

First, we need to ensure that the base model can align its output with a specified step-by-step format. To achieve this, we select problems from the training sets of GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) and reconstruct the reasoning paths in a structured step-by-step format using DeepSeek-V3 (Liu et al., 2024a). The specific prompt can be found in Figure 6 in the Appendix A. This process facilitates the creation of a well-structured SFT dataset. Subsequently, we conduct two epochs of SFT on the base model.

Meanwhile, inspired by Kojima et al. (2022), we explore the use of a CoT prompt "*Please reason step by step with steps separated by "\n\n", and put your final answer within \boxed{}.*", as a means to naturally guide the base model toward better adherence to the desired step-by-step format. The average greedy sampling scores across the 4 datasets (GSM8K, MATH500,Gaokao-2023-EN (Zhang et al., 2023) and Minerva-Math (Yang et al., 2024b)) are presented in Figure 2. "Base" denotes the approach where the model generates answers directly using the format " *Question: {input} \n Answer: {output}.*" And "+SFT" indicates that the dataset was constructed using two different prompting strategies, followed by fine-tuning on the model.

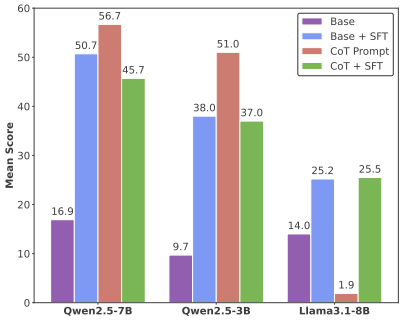


Figure 2: SFT & CoT Comparison

The results indicate that for Qwen2.5, simply providing the CoT prompt is sufficient. Applying SFT, however, results in a reduction. This decline is probably attributed to the inferior quality of the GSM8K and MATH datasets. In contrast, for LLaMA3.1, SFT is needed to ensure the model conforms to the designated format. Therefore, **in the subsequent experiments, all training and evaluation processes are conducted using the CoT prompt, with LLaMA3.1 employing the CoT-SFT version.**

3.3 Single-Round DPO with Coarse Filtering Improves Qwen2.5

To explore the impact of DPO on enhancing base models, we first train an ORM and a PRM for each base model. We generate 4 responses per problem from the GSM8K and MATH training sets with Qwen2.5-7B, then evaluate each reasoning path with DeepSeek-V3 to obtain process-level reward labels, the evaluated prompts can be seen in Figure 7 in the Appendix A. The labels are filtered based on reference answers, retaining those that either match the reference with fully correct process annotations or differ from the reference with errors in process annotations. The filtered data are then used to fine-tune the base model, yielding the PRM. Additionally, we construct preference data by pairing correct and incorrect responses for the same problem, then producing the ORM. Notably, our reward model relies on only tens of thousands of in-house data samples, making the training cost relatively low. Thus, it can serve as a coarse-grained filter for selecting self-improvement data.

After training the reward model, we resample 8 responses from the MATH training set using the base model. These responses are then filtered based on different rules to construct a DPO dataset with positive and negative sample pairs. Finally, we perform one round of DPO training on this dataset and evaluated various model variants on different datasets, including both in-distribution and out-of-distribution datasets. The results are recorded in Table 1. Specifically:

- **+Outcome Label**: Accuracy is determined by the outcome label, then a correct and an incorrect response are randomly selected as positive and negative samples, respectively.

- **+PRM & +ORM**: The response with the highest RM score is selected as the positive sample, and the one with the lowest score as the negative sample. For PRM, the RM evaluation score is determined by the step with the lowest score.
- **+PRM with offset**: Additionally, high-PRM-score responses with incorrect answers, referred to as offset samples, are selected as negative samples to prevent the generator from exploiting PRM. We rank the responses by PRM scores in descending order, and among the top k samples (where k is the number of correct responses), incorrect responses are paired with the highest-scoring correct ones.

Additionally, we compare our results with closed-source models, including GPT-4o (Hurst et al., 2024), GPT-o1-mini (Jaech et al., 2024), Claude-3.5-Sonnet Anthropic (2024), Qwen2.5-7B-Instruct Yang et al. (2024a), and Qwen2.5-7B-DsV3-SFT, the latter being a version of Qwen2.5-7B SFT on GSM8K and MATH responses generated by DeepSeek-V3.

Models	Methods	greedy-avg	In-Distribution Dataset						Out-of-Distribution Dataset					
			GSM8K			MATH-500			Gaokao-2023-EN			Minerva-Math		
			greedy	mv@8	pass@8	greedy	mv@8	pass@8	greedy	mv@8	pass@8	greedy	mv@8	pass@8
GPT-4o	-	72.5	96.4	-	-	76.6	-	-	71.4	-	-	45.6	-	-
Claude-3.5-Sonnet	-	72.2	96.4	-	-	71.1	-	-	72.2	-	-	48.9	-	-
GPT-o1-mini	-	76.7	96.8	-	-	90.0	-	-	70.9	-	-	48.9	-	-
Qwen2.5-7B-Instruct	-	68.7	92.4	91.4	97.8	76.4	73.4	88.2	64.2	61.0	78.7	41.9	20.2	41.2
Qwen2.5-7B-DsV3-SFT	-	62.5	89.5	92.0	96.7	72.6	78.0	87.4	60.0	67.0	76.1	27.9	29.4	51.4
Qwen2.5-7B	Base	56.7	86.4	92.1	97.2	64.2	74.6	87.8	55.6	61.3	76.6	20.6	18.4	35.3
	+Outcome Label	63.0	89.8	93.5	97.3	74.2	78.2	87.6	62.9	**67.3**	76.9	25.0	24.6	41.2
	+ORM	62.2	89.5	93.4	97.2	73.6	**79.6**	88.8	61.6	66.2	78.2	23.9	**26.8**	**47.7**
	+PRM	62.8	**90.5**	93.3	96.9	74.6	78.4	88.6	61.0	66.5	78.4	25.0	20.2	37.5
	+PRM with offset	**64.0**	90.3	**93.9**	**97.5**	**75.8**	78.6	**88.8**	**63.9**	65.7	**78.7**	**25.9**	23.9	37.9
	Best Improve	+7.3	+4.1	+1.8	+0.3	+11.6	+5.0	+1.0	+8.3	+6.0	+2.1	+5.3	+8.4	+12.4
Qwen2.5-3B	Base	49.0	74.7	86.9	95.6	61.2	68.2	81.4	42.1	56.6	71.2	**18.0**	**18.8**	31.6
	+Outcome Label	53.7	82.5	87.5	94.6	63.8	**70.4**	82.8	52.5	55.6	70.6	15.8	16.9	29.7
	+ORM	**54.1**	83.8	**89.0**	95.4	**64.4**	69.0	**83.6**	**54.8**	56.6	**73.3**	13.2	17.3	**32.7**
	+PRM	53.9	83.2	88.4	95.4	**64.4**	**70.4**	83.0	53.0	**58.2**	71.2	15.1	16.2	**32.7**
	+PRM with offset	53.9	**83.9**	88.2	**95.9**	63.2	68.4	82.0	53.2	57.9	71.2	15.4	16.2	30.5
	Best Improve	+5.1	+9.2	+2.1	+0.3	+3.2	+2.2	+2.2	+12.7	+1.6	+2.1	-2.2	-1.5	+1.1
LLaMA3.1-8B	Base	25.4	47.4	58.4	81.8	**21.6**	27.6	48.2	19.5	**26.7**	46.2	13.6	14.7	**32.0**
	+Outcome Label	25.7	48.6	61.9	81.8	20.0	27.4	48.4	19.2	25.7	45.7	**15.1**	**15.8**	28.7
	+ORM	25.4	48.7	60.3	83.1	20.8	**28.8**	50.4	19.0	24.7	42.1	12.9	11.8	31.6
	+PRM	26.4	48.2	58.9	79.1	20.4	24.2	47.8	**22.1**	26.0	**48.2**	14.7	10.0	25.7
	+PRM with offset	**26.5**	**49.8**	**62.5**	**84.3**	**21.6**	28.6	**51.8**	21.6	24.9	47.5	12.9	10.3	31.3
	Best Improve	+1.1	+2.4	+4.1	+2.5	+0.0	+1.2	+3.6	+2.6	-0.7	+2.0	+1.5	+1.1	-0.4

Table 1: Performance Comparison Across Different Models and DPO Data Filtering Methods. **Bold** values indicate the best results, and highlighted rows represent the best improvements.

Based on the results in Table 1, we derive the following findings:

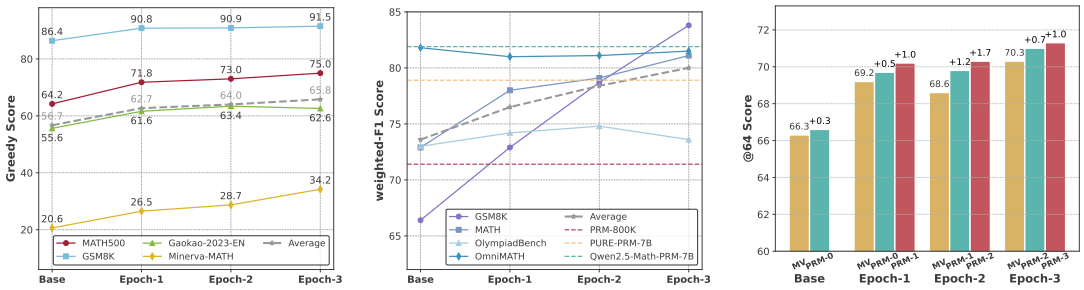
1. **DPO can rapidly enhance performance, with improvement strongly correlated to the base model's capability.** For Qwen2.5-7B, a single round of DPO using self-generated data brings the best variant, surpassing Qwen2.5-7B-DsV3-SFT and approaching Qwen2.5-7B-Instruct, which has been fine-tuned on a large dataset. However, for LLaMA3.1-8B, DPO has little effect on self-improvement.
2. **Even coarse-grained filtering significantly boosts performance. Leveraging a reward model for supervision and incorporating additional offset negative samples may improve training efficiency.** For Qwen2.5-7B, PRM with offset achieves the highest performance without requiring extra sampling.

3.4 Iterative Improvement of Generator and RM can be achieved via Multi-Round DPO

An interesting question is: can the generator and RM achieve mutual improvement through multiple rounds of DPO? The answer is certainly YES!

Starting with Qwen2.5-7B-base, we greedily sample responses from the MATH training set and annotate using DeepSeek-V3 to initialize PRM-base. Then, we apply the "DPO+PRM" filtering method from the previous section to select data for training the current generator via DPO. After updating the generator, we continue with online greedy sampling on MATH,

annotation, and PRM updates. Just as illustrated in the bottom right of Figure 1, this iterative process allows the generator and RM to evolve together over three epochs.



(a) Generator Score. (b) PRM Evaluation. (c) Test-time Scaling.

Figure 3: Performance Evolution of Generator and PRM Across Multiple DPO Epochs.

We conduct a detailed evaluation of the iteratively trained generator and PRM:

1. **Generator Score**: As shown in Figure 3a, we observe continuous performance improvement across different datasets. **The average evaluation score increases from 62.7 in the first epoch to 65.8 in the third epoch, surpassing the best average score reported in Table 1.**
2. **PRM Evaluation**: We assess the ability of PRM to identify incorrect steps using ProcessBench (Zheng et al., 2024). Given the imbalance between positive and negative samples in ProcessBench, we compute the weighted-F1 score (Hinojosa Lee et al., 2024) to reflect error detection performance. In addition to self-comparison, we benchmark against several baselines:
   (a) PRM-800K: A PRM fine-tuned on PRM800K (Lightman et al., 2023) dataset with Qwen2.5-7B.
   (b) PURE-PRM-7B: A PRM derived from Cheng et al. (2025), trained with PRM800K through a two-stage MLP and full fine-tuning.
   (c) Qwen2.5-Math-PRM-7B: The SOTA PRM officially released by Qwen (Zheng et al., 2024), trained on a large in-house dataset.

   The results show that **our PRM surpasses PRM-800K and, by the third epoch, outperforms PURE-PRM-7B, while slightly below Qwen2.5-Math-PRM-7B. And demonstrate that PRM can continuously improve through iterative training with the generator at a low cost — each epoch requires only 7.5K DeepSeek-V3 queries.** Detailed experimental settings can be found in Appendix B.
3. **PRM in Test-time Scaling**: We further analyze the role of PRM in test-time scaling. As shown in Figure 3c, we compare majority vote (MV) (Wang et al., 2023) and PRM@64. We find that PRM@64 outperforms MV in every epoch, and **the PRM trained with online sampled data from the current generator further improves over the previous epoch's PRM@64 performance**. A detailed performance table is provided in Appendix C.

3.5 Multi-Round DPO with Verifiable Pairs Achieves RL-Level Performance Efficiently

3.5.1 Setup and Main Results

Finally, we conduct multi-round DPO on the advanced Qwen2.5-Math-7B, aiming to benchmark against RL-based approaches under the same base model and training data. Inspired by DeepSeek-R1 (Zeng et al., 2025b), we observe that the verifiable reward (VR) is a 0/1 discrete variable, indicating whether the output meets predefined correctness criteria. This aligns with the original DPO framework, where correct samples are classified as positive and incorrect ones as negative, forming verifiable pairs (VP) for training. By applying VP-based filtering and gradual temperature scaling (inspired by Xi et al. (2024)), we conduct 6 epochs

of DPO. The final model, Qwen2.5-7B-DPO-VP, is compared against several state-of-the-art RL-based methods:

(a) Qwen2.5-Math-7B-Instruct (Yang et al., 2024b): An Instruct model by Qwen with strong mathematical reasoning capabilities.

(b) rStar-Math-7B (Guan et al., 2025): A model trained with a large amount of self-evolution MCTS data and filtered with process preference model (PPM).

(c) Eurus-2-7B-PRIME (Cui et al., 2025): A model trained by process RL with implicit process rewards.

(d) Qwen2.5-7B-GRPO-Zero (Shao et al., 2024): A model trained by GRPO with verifiable rewards without any extra SFT data.

(e) Qwen2.5-7B-Simple-RL-Zero/Zoo (Zeng et al., 2025a;b): A model trained by PPO with verifiable rewards without any extra SFT data. Zero and Zoo refer to two versions released at different time points.

(f) Qwen2.5-7B-PURE-VR (Cheng et al., 2025): A model trained by RL with verifiable rewards and min-form credit assignment without any extra SFT data.

(g) Qwen2.5-7B-DPO-R1-Zero (Zhang et al., 2025a): A concurrent work that applied iterative DPO without any extra SFT data.

To align with existing RL methods, following Zeng et al. (2025b) and Cheng et al. (2025), we select 8K Level 3-5 questions from the MATH as self-improvement data, and additionally evaluate on Olympiad-level benchmarks, including AMC23, AIME24, and OlympiadBench He et al. (2024). Furthermore, we conduct additional comparisons on 5 out-of-domain datasets to assess generalization performance. In addition, we compare with Qwen2.5-7B-DPO-R1-Zero (Zhang et al., 2025a) that applies iterative DPO on 200K questions (more than 8K questions) from the NuminaMath dataset (Li et al., 2024a).

Moreover, we employ annealed sampling to enhance response diversity. Specifically, the sampling temperature t is set to 0.7 for the first three epochs, increased to 1.0 during epochs 4–5, and further raised to 1.2 in the final epoch. This annealing schedule is motivated by the observation that model performance nearly converges after three epochs, and increasing the temperature in later stages allows for fine-tuning on more diverse samples, leading to additional gains.

Table 2 presents the pass@1 accuracy of different models, while Table 3 further analyzes training data consumption and GPU usage.

pass@1 acc	MATH500	Minerva-Math	Olympaidbench	AMC23	AIME24	Avg.
Qwen2.5-Math-7B *	64.8	15.4	25.6	37.5	16.7	32.0
Qwen2.5-Math-7B-Instruct *	**83.2**	33.5	38.4	62.5	20.0	47.5
rStar-Math-7B ^	78.4	-	**47.1**	47.5	**26.7**	-
Eurus-2-7B-PRIME *	74.0	**39.7**	35.6	57.5	23.3	46.0
Qwen2.5-7B-GRPO-Zero ^	76.2	32.7	38.1	55.0	16.7	43.7
Qwen2.5-7B-Simple-RL-Zero *	78.0	33.1	36.6	60.0	**26.7**	46.9
Qwen2.5-7B-Simple-RL-Zoo *	80.4	**39.7**	38.8	57.5	**26.7**	**48.7**
Qwen2.5-7B-PURE-VR *	79.8	36.8	41.9	60.0	20.0	47.7
Qwen2.5-7B-DPO-R1-Zero ^	76.8	30.9	37.9	62.5	**26.7**	47.0
Qwen2.5-7B-DPO-VP	74.8	35.3	36.9	**67.5**	**26.7**	48.2

Table 2: Pass@1 Accuracy Results Across Different Benchmarks. **Bold** values indicate the best results, and underline value indicate the second one. All models are fine-tuned based on the Qwen2.5-Math-7B. **Bold** models represent those that were adjusted using the self-improvement method with exactly the same prompts. The results with $*$ are from our reproduced, and the results with $\wedge$ are derived from the corresponding technical report.

Table 2 and 3, we derive the following conclusions:

1. **Multi-round DPO can achieve mathematical reasoning capabilities comparable to those of RL-based methods.** The final model attains an average score of 48.2 across five

	Qwen2.5-Math-7B-Instruct	rStar-Math-7B	Eurus-2-7B-PRIME	Qwen2.5-7B-SimpleRL-Zero	Qwen2.5-7B-PURE-VR	Qwen2.5-7B-DPO-VP
Base Model	Qwen2.5-Math-7B	Qwen2.5-Math-7B	Qwen2.5-Math-7B	Qwen2.5-Math-7B	Qwen2.5-Math-7B	Qwen2.5-Math-7B
SFT Data	2.5M (open-source and in-house)	∼7.3M (MATH, NuminaMath, etc.)	230K	0	0	0
RM Data	618K (in-house)	∼7K (in-house)	0	0	0	0
RM	Qwen2.5-Math-RM (72B)	None	Eurus-2-7B-SFT	None	None	None
Self-improve Method	RL + ORM	MCTS + PPM	RL + PRM	RL + VR	RL + VR	DPO + VR
Self-improve Data	66K	∼3.647M	150K	**8K MATH**	**8K MATH**	**8K MATH**
GPUs & Training Times	-	80 H100 for few days (at most)	8 A100 for few days	32 H100 for 1.5 days	8 A100 for 1 day	**4 A800 for <1 day or 1 A800 for 2 days**

Table 3: Data and GPUs Comparison of Different Models.

mathematical reasoning benchmarks, which is similar to the performance of Qwen2.5-Math-7B-Instruct and other RL-based approaches with the some training data conditions. Compared to Qwen2.5-7B-DPO-R1-Zero, our model still achieves a higher average score. We hypothesize that this may be attributed to the increased response diversity introduced by the annealed sampling strategy.

2. **Multi-round DPO requires significantly fewer computational resource can even be executed on a single GPU.** Our sampling and training are conducted on four A800 GPUs using data parallelism. The full pipeline can be run on a single 80GB GPU or lower. In our setup with 4 A800 GPUs, each sampling round took 2–2.5 hours with vLLM (Kwon et al., 2023), training took 1 hour per round, and the full process completed in about 1 day. On a single GPU, the full process would take around 3 days.

3.5.2 *More Analysis*

Token Length Dynamics and Accuracy Trends First, we analyze token length variations throughout training. In Figure 4, we illustrate the accuracy progression of the iterative process across different datasets in relation to the average token length. The results indicate a steady improvement in accuracy, while the **token length during inference did not exhibit an initial increase followed by a decline. Instead, it remains relatively stable within a consistent range.** When comparing different models, we observe that their output lengths are generally similar. Detailed results are provided in Table 7 in Appendix E.1.

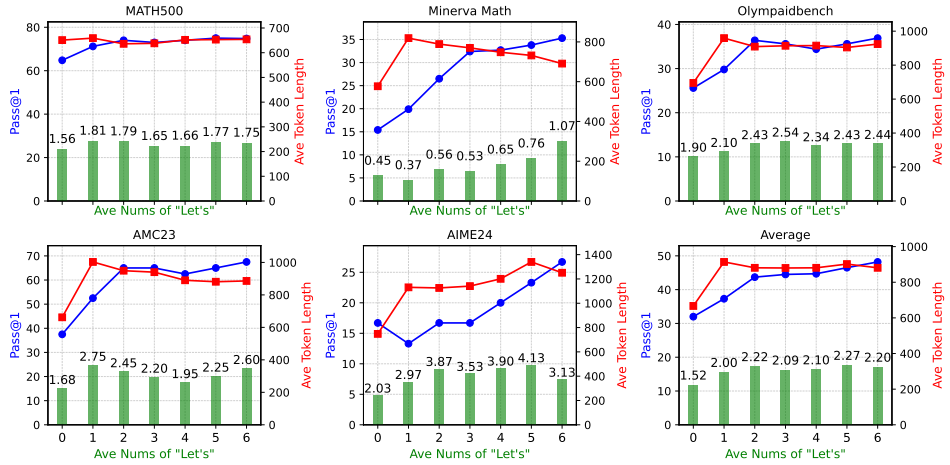


Figure 4: Pass@1 Accuracy, Token Length, and Self-Evaluation Trends Across Datasets.

Self-Reflection and Reasoning Paradigms We do not observe a significant "Aha Moment," which aligns with the findings of PURE (Cheng et al., 2025). Even when using CoT prompts, Qwen2.5-Math-7B frequently relies on Python-based solutions, despite not actually executing the code. Additionally, we identify a recurring self-evaluation pattern, such as *"Let's*

re-evaluate ...”, which is already present in the base model. This suggests that DPO does not inherently introduce self-reflection capabilities but instead strengthens existing tendencies through reward optimization, as shown in Figure 4. When comparing several different models, Table 8 in Appendix E.1 shows that Qwen2.5-Math-7B-Instruct exhibits significantly fewer occurrences of *”Let's”*, indicating a shift in reasoning paradigms.

3.6 Ablation Study on the DPO Sampling Noisy

To evaluate the robustness of our DPO framework under noisy supervision, we conduct a controlled label noise ablation. Specifically, we simulate noise in the preference data used to train the Preference Reward Model (PRM) under the *PRM with offset* setting. Following Tu et al. (2025c), for each positive–negative response pair, we apply a **random flipping probability** $p \in [0, 1]$ that reverses the preference direction with probability p, simulating increasing levels of annotation noise. For example, $p = 0.5$ corresponds to fully random pairwise supervision, while $p = 1.0$ denotes complete inversion of the labels.

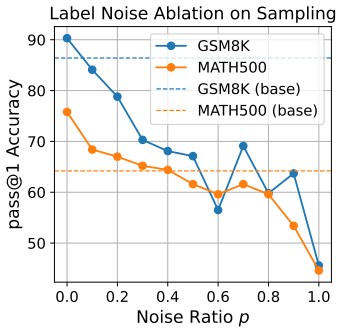


Figure 5: Label Noise Ablation

We test the impact of this noise injection on the final model performance, measured by pass@1 accuracy after one round of DPO on GSM8K and MATH500 using Qwen2.5-7B. The results are shown in Figure 5. These results clearly demonstrate that model performance degrades as the preference supervision becomes less accurate. While DPO appears tolerant to small levels of label noise, performance drops significantly under higher noise levels. This provides empirical evidence that maintaining reasonably accurate verifier signals is critical for effective preference optimization.

Generalization We conduct evaluations on 5 additional Out-of-Domain datasets:

(a) CMATH: A Chinese mathematics dataset.
(b) MMLU_STEM: A subset of MMLU focusing on science subjects like physics, biology.
(c) HumanEval: A benchmark evaluating code generation through Python tasks.
(d) LiveCodeBench: A coding benchmark for real-time code editing and debugging.
(e) RACE: A reading benchmark from English exams testing reasoning.

The first two datasets are evaluated using the Qwen-Math codebase (Yang et al., 2024b), with evaluation parameters consistent with those used in Appendix D. The remaining three datasets are evaluated via EvalScope using its default configuration settings. The results are summarized in Table 4.

pass@1 acc	CMATH	MMLU_STEM	HumanEval	Live_Code_Bnech	RACE	Average
Qwen2.5-Math-7B	64.0	58.3	40.9	5.1	61.6	46.0
Eurus-2-7B-PRIME	72.5	42.8	41.5	21.2	62.6	48.1
Qwen2.5-7B-Simple-RL-Zero	70.3	59.1	51.8	21.4	61.1	52.7
Qwen2.5-7B-Simple-RL-Zoo	89.0	35.9	53.1	17.9	63.9	52.0
Qwen2.5-7B-PURE-VR	69.5	58.8	43.3	6.4	61.3	47.9
Qwen2.5-7B-DPO-VP	68.5	59.3	50.6	17.3	61.2	51.4

Table 4: Performance comparison on Out-of-Domain datasets.

Nearly all methods lead to slight performance improvements on the out-of-domain (OOD) benchmarks, indicating enhanced generalization capabilities. Our proposed DPO-VP notably increases the average OOD score from 46.0 to 51.4, achieving gains comparable to those of RL-based methods. This suggests that DPO fine-tuning on mathematical tasks can also effectively enhance generalization to other domains.

https://github.com/modelscope/evalscope

4 Related Works

Self-improvement of LLMs Self-improvement primarily occurs during the training phase, where the model enhances its own performance by evaluating and refining its outputs without relying on external human supervision (Xi et al., 2024; Kumar et al., 2025). It is generally believed that LLMs are incapable of achieving self-improvement without any external information (Huang et al., 2024; Tyen et al., 2024). Consequently, an increasing number of methods leverage external supervision signals help self-improvement, for example, outcome labels for mathematical or programming problems (McAleese et al., 2024; Gao et al., 2024), process reward models (Zhang et al., 2024b; Fu et al., 2025b), value function (Wang et al., 2024a) or LLM-as-a-judge (Li et al., 2024b; Zhang et al., 2025b) frameworks.

Improving Reasoning with DPO Recent works have significantly extended DPO-based reasoning (Xu et al., 2024; Zhang et al., 2025a). Iterative DPO with rigorous theoretical analysis was proposed by Xiong et al. (2024), and further generalized to multi-turn and tool-integrated scenarios (Xiong et al., 2025a). Flow-DPO (Deng & Mineiro, 2024) introduced online multi-agent learning for mathematical reasoning, while iterative length-regularized DPO (Liu et al., 2024b) enables 7B models to approach GPT-4-level reasoning. Self-play and self-training pipelines have also been explored, demonstrating that DPO and preference-based methods can improve LLMs beyond human-labeled data (Chen et al., 2024b; Singh et al., 2023). Additionally, implicit reward bootstrapping with DPO has shown promise for further alignment without explicit reward models (Chen et al., 2024a). Aya Expanse (Dang et al., 2024) illustrates how such alignment advances benefit multilingual and cross-cultural reasoning. In contrast, our work focuses on data selection strategies for DPO and proposes an iterative training framework for generator-verifier models, enabling mutual enhancement between generation and reward modeling.

Constructing Efficient Reasoning Models In early 2025, with the release of DeepSeek-R1 (Guo et al., 2025), efficient post-training paradigms gained significant attention (Trung et al., 2024; Tu et al., 2025a). SimpleRL-Zero (Zeng et al., 2025b) achieved remarkable performance through RL, training on just 8K challenging mathematical problems with rule-based rewards. Xiong et al. (2025b) integrated reflective vocabulary into generative RM to enable selective RL-based correction. PURE (Cheng et al., 2025) revisited stepwise reward credit assignment in RL. Logic-RL (Xie et al., 2025) refined the penalty function for formatting errors, leading to improved logical reasoning capabilities. From the data utilization perspective, LIMO (Ye et al., 2025) and S1 (Muennighoff et al., 2025) demonstrated that fine-tuning with carefully curated long-chain reflective data can effectively amplify the latent reasoning capabilities. From the reasoning paradigm perspective, CFT (Wang et al., 2025) encouraged models to develop critical thinking during SFT. Our work focuses on achieving RL-based reasoning with significantly fewer computational resources. We demonstrate that, with a strong base model, coarse-grained selection is enough for performance enhancement, achieving competitive results with reduced computational overhead. Our overall training framework is more resource-efficient than the widely adopted SFT+RL pipeline, while outperforming concurrent iterative DPO methods.

5 Conclusion

In this paper, we empirically study DPO for enhancing LLM reasoning, showing that single-round DPO with coarse filtering improves performance, while multi-round DPO enables iterative enhancement of both the generator and RM. Our results demonstrate that multi-round DPO achieves RL-level performance with significantly lower computational costs, requiring only a single 80GB GPU. However, in large-scale deployment and long-term training, DPO lacks exploration and sustained improvement potential. Future research could focus on more exploratory offline self-improvement methods or embrace efficient RL algorithms as a pathway to further boost the reasoning bounds of LLMs.

Acknowledgements

This work is supported by the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA0480303, Young Scientists Fund of The State Key Laboratory of Multimodal Artificial Intelligence Systems ES2P100112, Beijing Natural Science Foundation - Xiaomi Innovation Joint Fund L253007, and National Natural Science Foundation of China 62402252.

References

Anthropic. The claude 3 model family: Opus, sonnet, haiku. *Technical Peport*, 2024.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pp. 4447–4455. PMLR, 2024.

Changyu Chen, Zichen Liu, Chao Du, Tianyu Pang, Qian Liu, Arunesh Sinha, Pradeep Varakantham, and Min Lin. Bootstrapping language models with dpo implicit rewards. *arXiv preprint arXiv:2406.09760*, 2024a.

Wenxiang Chen, Wei He, Zhiheng Xi, Honglin Guo, Boyang Hong, Jiazheng Zhang, Rui Zheng, Nijun Li, Tao Gui, Yun Li, et al. Better process supervision with bi-directional rewarding signals. *arXiv preprint arXiv:2503.04618*, 2025.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024b.

Jie Cheng, Ruixi Qiao, Lijun Li, Chao Guo, Junle Wang, Gang Xiong, Yisheng Lv, and Fei-Yue Wang. Stop summation: Min-form credit assignment is all process reward model needs for reasoning. *arXiv preprint arXiv:2504.15275*, 2025.

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

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.

John Dang, Shivalika Singh, Daniel D'souza, Arash Ahmadian, Alejandro Salamanca, Madeline Smith, Aidan Peppin, Sungjin Hong, Manoj Govindassamy, Terrence Zhao, et al. Aya expanse: Combining research breakthroughs for a new multilingual frontier. *arXiv preprint arXiv:2412.04261*, 2024.

Yihe Deng and Paul Mineiro. Flow-dpo: Improving llm mathematical reasoning through online multi-agent learning. *arXiv preprint arXiv:2410.22304*, 2024.

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

Yuqian Fu, Tinghong Chen, Jiajun Chai, Xihuai Wang, Songjun Tu, Guojun Yin, Wei Lin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. Srft: A single-stage method with supervised and reinforcement fine-tuning for reasoning. *arXiv preprint arXiv:2506.19767*, 2025a.

Yuqian Fu, Yuanheng Zhu, Jiajun Chai, Guojun Yin, Wei Lin, Qichao Zhang, and Dongbin Zhao. Rlae: Reinforcement learning-assisted ensemble for llms. *arXiv preprint arXiv:2506.00439*, 2025b.

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*, 2025.

Bofei Gao, Zefan Cai, Runxin Xu, Peiyi Wang, Ce Zheng, Runji Lin, Keming Lu, Dayiheng Liu, Chang Zhou, Wen Xiao, et al. Llm critics help catch bugs in mathematics: Towards a better mathematical verifier with natural language feedback. *arXiv preprint arXiv:2406.14024*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*, 2025.

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

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *ACL*, 2024.

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. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Maria Cristina Hinojosa Lee, Johan Braet, and Johan Springael. Performance metrics for multilabel emotion classification: comparing micro, macro, and weighted f1-scores. *Applied Sciences*, 14(21):9863, 2024.

Jian Hu, Xibin Wu, Zilin Zhu, Weixun Wang, Dehao Zhang, Yu Cao, et al. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations (ICLR)*, 2024.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:22199–22213, 2022.

Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip HS Torr, Salman Khan, and Fahad Shahbaz Khan. Llm post-training: A deep dive into reasoning large language models. *arXiv preprint arXiv:2502.21321*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13:9, 2024a.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason E Weston, and Mike Lewis. Self-alignment with instruction backtranslation. In *International Conference on Learning Representations (ICLR)*, 2024b.

Fenrui Xiao Xin He Qi An Zhenyu Duan Yimin Du Junchen Liu Lifu Tang Xiaowei Lv Haosheng Zou Yongchao Deng Shousheng Jia Xiangzheng Zhang Liang Wen, Yunke Cai. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond, 2025. URL https://github.com/Qihoo360/Light-R1.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2023.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

Jie Liu, Zhanhui Zhou, Jiaheng Liu, Xingyuan Bu, Chao Yang, Han-Sen Zhong, and Wanli Ouyang. Iterative length-regularized direct preference optimization: A case study on improving 7b language models to gpt-4 level. *arXiv preprint arXiv:2406.11817*, 2024b.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025. Notion Blog.

Nat McAleese, Rai Michael Pokorny, Juan Felipe Ceron Uribe, Evgenia Nitishinskaya, Maja Trebacz, and Jan Leike. Llm critics help catch llm bugs. *arXiv preprint arXiv:2407.00215*, 2024.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

Richard Yuanzhe Pang, Weizhe Yuan, He He, Kyunghyun Cho, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:116617–116637, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:53728–53741, 2023.

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

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, et al. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*, 2023.

Richard Sutton. The bitter lesson. *Incomplete Ideas (blog)*, 13(1):38, 2019.

Gokul Swamy, Sanjiban Choudhury, Wen Sun, Zhiwei Steven Wu, and J Andrew Bagnell. All roads lead to likelihood: The value of reinforcement learning in fine-tuning. *arXiv preprint arXiv:2503.01067*, 2025.

Luong Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. Reft: Reasoning with reinforced fine-tuning. In *ACL*, pp. 7601–7614, 2024.

Songjun Tu, Jiahao Lin, Qichao Zhang, Xiangyu Tian, Linjing Li, Xiangyuan Lan, and Dongbin Zhao. Learning when to think: Shaping adaptive reasoning in r1-style models via multi-stage rl. *arXiv preprint arXiv:2505.10832*, 2025a.

Songjun Tu, Jingbo Sun, Qichao Zhang, Xiangyuan Lan, and Dongbin Zhao. Online preference-based reinforcement learning with self-augmented feedback from large language model. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 2069–2077, 2025b.

Songjun Tu, Jingbo Sun, Qichao Zhang, Yaocheng Zhang, Jia Liu, Ke Chen, and Dongbin Zhao. In-dataset trajectory return regularization for offline preference-based reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 20929–20937, 2025c.

Gladys Tyen, Hassan Mansoor, Victor Cărbune, Yuanzhu Peter Chen, and Tony Mak. Llms cannot find reasoning errors, but can correct them given the error location. In *ACL Findings*, pp. 13894–13908, 2024.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`, 2020.

Chaojie Wang, Yanchen Deng, Zhiyi Lyu, Liang Zeng, Jujie He, Shuicheng Yan, and Bo An. Q*: Improving multi-step reasoning for llms with deliberative planning. *arXiv preprint arXiv:2406.14283*, 2024a.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Yubo Wang, Xiang Yue, and Wenhu Chen. Critique fine-tuning: Learning to critique is more effective than learning to imitate. *arXiv preprint arXiv:2501.17703*, 2025.

Zhiheng Xi, Dingwen Yang, Jixuan Huang, Jiafu Tang, Guanyu Li, Yiwen Ding, Wei He, Boyang Hong, Shihan Do, Wenyu Zhan, et al. Enhancing llm reasoning via critique models with test-time and training-time supervision. *arXiv preprint arXiv:2411.16579*, 2024.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101, 2025.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025.

Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. Iterative preference learning from human feedback: Bridging theory and practice for rlhf under kl-constraint. In *Proceedings of the 41st International Conference on Machine Learning(ICML)*, pp. 54715–54754, 2024.

Wei Xiong, Chengshuai Shi, Jiaming Shen, Aviv Rosenberg, Zhen Qin, Daniele Calandriello, Misha Khalman, Rishabh Joshi, Bilal Piot, Mohammad Saleh, et al. Building math agents with multi-turn iterative preference learning. In *The International Conference on Learning Representations*, 2025a.

Wei Xiong, Hanning Zhang, Chenlu Ye, Lichang Chen, Nan Jiang, and Tong Zhang. Self-rewarding correction for mathematical reasoning. *arXiv preprint arXiv:2502.19613*, 2025b.

Shusheng Xu, Wei Fu, Jiaxuan Gao, Wenjie Ye, Weilin Liu, Zhiyu Mei, Guangju Wang, Chao Yu, and Yi Wu. Is dpo superior to ppo for llm alignment? a comprehensive study. *arXiv preprint arXiv:2404.10719*, 2024.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024a.

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

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*, 2025.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025a.

Weihao Zeng, Yuzhen Huang, Wei Liu, Keqing He, Qian Liu, Zejun Ma, and Junxian He. 7b model and 8k examples: Emerging reasoning with reinforcement learning is both effective and efficient. https://hkust-nlp.notion.site/simplerl-reason, 2025b. Notion Blog.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:64735–64772, 2024a.

Hanning Zhang, Jiarui Yao, Chenlu Ye, Wei Xiong, and Tong Zhang. Online-dpo-r1: Unlocking effective reasoning without the ppo overhead, 2025a. Notion Blog.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024b.

Shimao Zhang, Xiao Liu, Xin Zhang, Junxiao Liu, Zheheng Luo, Shujian Huang, and Yeyun Gong. Process-based self-rewarding language models. *arXiv preprint arXiv:2503.03746*, 2025b.

Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. Evaluating the performance of large language models on gaokao benchmark. *arXiv preprint arXiv:2305.12474*, 2023.

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

A Prompts for DeepSeek-V3

Here we provide the prompts used to construct the SFT dataset and annotate reward labels with DeepSeek-V3 in Figure 6 and Figure 7, which are similar to that in Chen et al. (2025).

SFT Dataset Preprocessing

You are an expert math examiner, skilled at transforming complex mathematical solution steps into clear formats. Your task is to rewrite the solution in the following format:
1. Separating each step with '\n\n'.
2. If the final answer is in the final step, warp it with a $\boxed{}$ in its original place.
3. You can only insert '\n\n', instead of adding or repeating any other words.

<example>
He has 60-20= <<60-20=40>>40 kilograms of mangoes left after selling them to the market.

Colby sold 1/2 x 40 = <<1/2*40=20>>20 kilograms of mangoes to the community.

Therefore, Colby still has 20x8= <<20*8=160>> $\boxed{160}$ pieces of mangoes.
<\example>

A step should represent a complete statement, structure or calculation process. Your output should only include the revised solution and not omit any original content.

{reference solution}

Figure 6: The Prompt Template for Dataset Preprocessing for DeepSeek-V3.

Reward Label Annotation

You are an expert math examiner. Your task is to review the student's solution and evaluate each step. Steps are wrapped in <step_n>. Mark a step as correct only if it is based on accurate premises and contributes to solving the problem. Mark it as unnecessary if it is logically valid but does not aid in solving the problem. Your judgments should include a very concise analysis of each step and the final judgement.
You must provide your evaluations in JSON format like:
{"step_1": {"analysis": "<concise analysis of the step>", "judgement": "<correct/incorrect/unnecessary>"}, "step_2": {...}, ...}

Below is the question, reference answer, and student's solution that you need to evaluate. Note that the student's solution does not need to match the reference solution exactly.

[Question]
{question}

[Reference Answer]
{answer}

[Student's Solution]
{solution}

Now, provide your evaluations of Student's Solution in JSON format.

Figure 7: The Prompt Template for Reward Label Annotaion for DeepSeek-V3.

B Details of Processbench Evaluation

We evaluate the performance of the trained PRM on ProcessBench. Notably, in ProcessBench, the number of incorrect/correct samples in the four datasets are as follows: GSM8K: 207/193; MATH: 594/406; OlympiadBench: 661/339; OmniMath: 759/241. At this point, the traditional F1 score is not very effective for handling imbalanced datasets.

Therefore, an alternative approach is to separately define the F1 score for positive and negative samples (Hinojosa Lee et al., 2024). The former is equivalent to the commonly used F1 score, while the latter is slightly modified based on the above formula. Then, a weighted F1 score is calculated based on the proportion of positive and negative samples. The weighted-F1 score better reflects the model performance. We define positive samples as those labeled as incorrect. The formula for calculating the weighted F1 score is as follows:

$$F1_{positive} = \frac{2 \times P_{positive} \times R_{positive}}{P_{positive} + R_{positive}}, P_{positive} = \frac{TP}{TP + FP}, R_{positive} = \frac{TP}{TP + FN}$$

$$F1_{negative} = \frac{2 \times P_{negative} \times R_{negative}}{P_{negative} + R_{negative}}, P_{negative} = \frac{TN}{FN + TN}, R_{negative} = \frac{TN}{FP + TN}$$

Assuming the proportion of positive samples is α and the proportion of negative samples is β , the weighted-F1 score is defined as:

$$F1_{weighted} = \alpha \times F1_{positive} + \beta \times F1_{negative}$$

Table 5 records the evaluation scores of the reward model on ProcessBench during the self-evolution process, with its average values corresponding to those recorded in Figure 3b.

PRM	GSM8K			MATH			OlympiadBench			OmniMATH			Average
	F1-P	F1-N	F1	F1-P	F1-N	F1	F1-P	F1-N	F1	F1-P	F1-N	F1	F1
PRM-800K	78.7	62.7	71.0	79.1	26.6	57.8	80.0	6.5	55.1	88.0	19.1	71.4	63.8
PRM-MathShepred	80.5	72.5	76.6	77.6	43.6	63.8	78.6	15.6	57.2	83.0	33.6	71.1	67.2
PURE-PRM	86.4	90.7	88.4	87.8	74.6	82.4	84.7	67.6	78.9	85.4	62.7	79.9	82.4
Qwen2.5-Math-PRM-7B	83.0	93.3	87.9	78.2	93.6	84.4	72.3	93.8	79.6	70.5	92.5	75.8	81.9
PRM-Epoch-0	37.1	97.9	66.4	62.5	88.2	72.9	77.5	64.3	73.0	88.2	61.8	**81.8**	73.6
PRM-Epoch-1	50.5	96.9	72.9	73.4	84.7	78.0	80.8	61.4	74.2	89.1	55.6	81.0	76.5
PRM-Epoch-2	61.7	96.9	78.7	78.2	80.3	79.1	83.2	58.4	**74.8**	90.2	52.3	81.1	78.4
PRM-Epoch-3	73.4	96.2	**83.8**	81.9	80.0	**81.1**	83.8	53.7	73.6	90.8	52.3	81.5	**80.0**

Table 5: Weighted-F1 Score of Different PRM in Processbench.

https://github.com/QwenLM/ProcessBench

C Details of Test-time Scaling

Table 6 records the test-time scaling results of the PRM trained with online-generated data from the current model at each epoch, compared with the PRM from the previous round. The PRM selection strategy follows the PRM-min-vote approach (Wang et al., 2024b), where the smallest step reward value in the responses is used as a weighting factor to adjust the majority vote result, ultimately selecting the final target answer.

Epoch	Methods	MATH500			GSM8K			Gaokao-2023-EN			Minerva Math		
		@16	@64	@256	@16	@64	@256	@16	@64	@256	@16	@64	@256
Base	Greedy	64.2			86.4			55.6			20.6		
	MV	77.2	78.4	79.2	93.1	93.8	94.1	63.9	65.7	66.0	23.9	27.2	28.7
	PRM-0	77.2	79.4	80.0	93.4	94.1	94.4	63.9	65.5	67.8	24.3	27.2	27.9
Epoch-1	Greedy	71.8 (+7.6)			90.8 (+4.4)			61.6 (+6.0)			26.5 (+5.9)		
	MV	80.2	81.4	82.2	94.4	95.0	95.2	65.7	68.6	68.6	27.5	31.6	33.8
	PRM-0	80.8	80.8	82.4	94.7	**95.3**	**95.5**	67.1	70.1	68.9	27.2	32.4	**34.6**
	PRM-1	**81.2**	**81.6**	**82.6**	**94.9**	95.2	95.1	**67.3**	**70.4**	**69.6**	**27.6**	**33.5**	**34.6**
	Improvement	+0.4	+0.8	+0.2	+0.2	-0.1	-0.4	+0.2	+0.3	+0.7	+0.4	+1.1	+0.0
Epoch-2	Greedy	73.0 (+1.2)			90.9 (+0.1)			63.4 (+1.8)			28.7 (+2.2)		
	MV	80.0	80.6	80.8	94.2	94.5	94.7	66.8	66.5	66.8	**29.7**	32.7	35.3
	PRM-1	80.8	82.2	82.8	**94.8**	**95.2**	95.0	67.5	67.8	68.1	27.6	34.1	36.0
	PRM-2	**81.6**	**82.6**	**83.0**	**94.8**	**95.2**	**95.1**	**68.1**	**68.6**	**69.1**	28.7	**34.6**	**36.4**
	Improvement	+0.8	+0.6	+0.2	+0.0	+0.0	+0.1	+0.6	+0.8	+1.0	+1.1	+0.5	+0.4
Epoch-3	Greedy	75.0 (+2.0)			91.5 (+0.6)			62.6 (-0.8)			34.2 (+5.5)		
	MV	79.8	81.0	81.4	94.4	94.8	95.2	66.3	67.8	67.8	36.0	37.5	39.8
	PRM-2	81.2	82.2	82.6	**94.7**	95.1	95.2	**67.5**	68.9	70.2	36.1	37.8	40.0
	PRM-3	**81.6**	**82.6**	**83.2**	94.6	**95.3**	**95.4**	67.1	**69.1**	**70.4**	**36.8**	**37.9**	**40.4**
	Improvement	+0.4	+0.6	+0.6	-0.1	+0.2	+0.2	-0.4	+0.2	+0.2	+0.7	+0.1	+0.4

Table 6: Generator performance comparison across different epochs and methods for various datasets. Bold values indicate the best results.

D Details of DPO and RM Training and Evaluation

In Section 3.2-3.4, data sampling, and evaluation of the generator are based on the Open-RLHF (Hu et al., 2024) framework. When filtering DPO data, the sample temperature is set to 0.7. All training is performed using full parameter fine-tuning, with an SFT learning rate of $5e-6$, a DPO learning rate of $5e-7$, a maximum sequence length of 2048, the coefficient β of 0.1 and a training batch size of 256. For ORM and PRM training, we use the TRL (von Werra et al., 2020) framework with a learning rate of $1e-6$ and a batch size of 256.

In Section 3.5, the DPO training follows the same configuration as above but employs annealed sampling to enhance response diversity. Specifically, for the first 3 epochs, the temperature t is set to 0.7; for epochs 4-5, $t = 1.0$; and in the final epoch, $t = 1.2$.

All evaluations are conducted using the Qwen-Math evaluation codebase, with greedy decoding of $t = 0.0$, generating one output per input, and a maximum generation length of 2048 tokens. In addition, we provide the URLs of all self-evaluated models in footnotes, including Eurus-2-7B-PRIME, Qwen2.5-7B-Simple-RL-Zero, Qwen2.5-7B-Simple-RL-Zoo, and Qwen2.5-7B-PURE-VR.

More details and code can be found at https://github.com/TU2021/DPO-VP.

https://github.com/QwenLM/Qwen2.5-Math
https://huggingface.co/PRIME-RL/Eurus-2-7B-PRIME
https://huggingface.co/hkust-nlp/Qwen-2.5-Math-7B-SimpleRL-Zero
https://huggingface.co/hkust-nlp/Qwen-2.5-Math-7B-SimpleRL-Zoo
https://huggingface.co/jinachris/Qwen2.5-7B-PURE-VR

E More Results of DPO-VP Analysis

E.1 Token Length and Self-Reflection Vocabulary Analysis

We present the detailed token length dynamics, accuracy trends, and the comparison of different models in self-reflection in Table 7 and Table 8.

Avg. Token Length	MATH500	Minerva Math	Olympapaidbench	AMC23	AIME24	Average
Qwen2.5-Math-7B	651	577	695	662	748	667
Qwen2.5-Math-7B-Instruct	641	650	886	911	1164	850
Eurus-2-7B-PRIME	**655**	**822**	897	**1020**	1164	**912**
Qwen2.5-7B-Simple-RL	588	775	801	767	952	777
Qwen2.5-7B-PURE-VR	626	646	863	850	1050	807
Qwen2.5-7B-DPO-VP	654	691	**924**	886	**1251**	881

Table 7: Comparison of Average Token Length Across Different Models and Datasets.

Avg. Nums of "Let's"	MATH500	Minerva Math	Olympaidbench	AMC23	AIME24	Average
Qwen2.5-Math-7B	1.56	0.45	1.90	1.68	2.03	1.52
Qwen2.5-Math-7B-Instruct	*0.40*	*0.13*	*0.67*	*0.70*	*0.87*	*0.55*
Eurus-2-7B-PRIME	1.50	0.96	2.27	2.40	**3.30**	2.09
Qwen2.5-7B-Simple-RL	1.49	0.57	1.99	1.93	2.20	1.64
Qwen2.5-7B-PURE-VR	0.86	0.24	1.52	1.33	1.73	1.14
Qwen2.5-7B-DPO-VP	**1.75**	**1.07**	**2.44**	**2.60**	3.13	**2.20**

Table 8: Average Occurrences of "Let's" Across Different Datasets.

E.2 Comparison with Long-Chain SFT Methods

Recently, reinforcement learning methods built upon long-chain SFT have received increasing attention. We compare our approach with two recent methods that perform long-chain SFT using limited data: LIMO (Ye et al., 2025) and S1 (Muennighoff et al., 2025). Since both released models are only available at the 32B scale, we reproduce their settings by applying SFT on our base model, Qwen-2.5-Math-7B, using their provided datasets. We follow their default hyperparameters and train for 15 epochs. During evaluation, we sample with $t = 0.6$ and set the maximum response length to 32K tokens (in contrast, DPO-VP uses 2K tokens). The performance on standard benchmarks is summarized in Table 9.

Surprisingly, DPO-VP outperforms these limited-data SFT methods on nearly all evaluated benchmarks. One possible explanation is that, at the 7B scale, SFT with small-scale long-chain data may offer less performance gain compared to its effect on larger models like 32B. Nevertheless, both LIMO and S1 still yield moderate improvements over the base model.

pass@1 acc	MATH500	Minerva	Olympaid	AMC23	AIME24	Average
Qwen2.5-Math-7B	64.8	15.4	25.6	37.5	16.7	32.0
Qwen2.5-Math-7B-LIMO	70.2	21.0	37.0	45.0	16.7	38.0 (+6.0)
Qwen2.5-Math-7B-S1	72.0	32.4	**37.5**	55.0	13.3	42.0 (+10.0)
Qwen2.5-7B-DPO-VP	**74.8**	**35.3**	36.9	**67.5**	**26.7**	**48.2 (+16.2)**

Table 9: Pass@1 Accuracy Results with Long-Chain SFT Methods.

E.3 Training Dynamics of DPO under Different Label Noise Levels

In addition to final performance, we also tracked the training dynamics of DPO under different noise levels. As shown in Figure 8, lower noise (e.g., $p \leq 0.3$) leads to higher training accuracy and more stable reward separation between chosen and rejected samples. In contrast, high-noise setups (e.g., $p = 0.7$ or $p = 1.0$) cause convergence slow-down and reward collapse, confirming the importance of maintaining accurate preference signals during training.

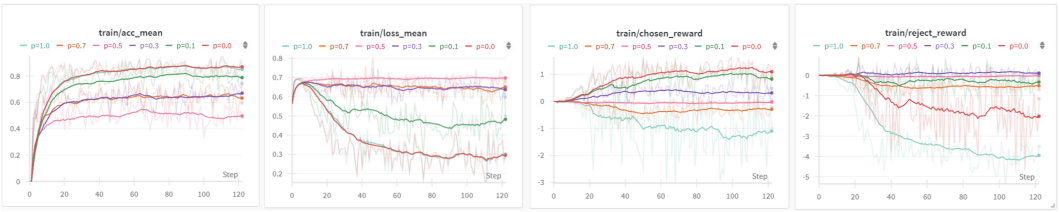


Figure 8: Training dynamics of DPO under different label noise levels. Lower noise consistently leads to better optimization behavior and more stable reward separation.

E.4 Ablation Study on the DPO Sampling Temperature

In this section, we investigate the impact of sampling temperature on DPO performance. Our motivation stems from the observation that fixed low-temperature sampling may lead to insufficient exploration in later DPO iterations, resulting in early performance saturation. In our default setting, we used a constant temperature $t = 0.7$ across all DPO rounds. To explore the potential of lightweight exploration, we introduced a **simple temperature increase strategy**: using $t = 1.0$ for epochs 4–5 and $t = 1.2$ for epochs 6–7. Table 10 reports the average accuracy across five benchmarks under both temperature settings.

Epoch	**Fixed** $t = 0.7$	**Ours (temperature ↑)**
3	44.5	44.5
4	44.6	44.7
5	44.3	46.5
6	46.2	**48.2**
7	45.5	47.6

Table 10: Comparison of average performance (pass@1 across five benchmarks) under fixed and increased sampling temperatures.

As shown, the increased temperature leads to noticeable performance gains starting from epoch 5, indicating improved sample diversity and preference pair quality. This suggests that even without using formal entropy-based exploration objectives, a simple sampling heuristic can effectively **encourage diversity in preference selection** and improve downstream learning. We believe this highlights the potential of lightweight exploration strategies in DPO, especially in scenarios where reward supervision is sparse or static.

E.5 Use of Distilled PRM Instead of a Fixed Verifier (e.g., DeepSeek-V3)

While large-scale verifiers such as DeepSeek-V3 (a 671B MoE model) provide high-quality supervision, they are computationally expensive to query. Each DPO iteration would require over 7.5K queries per epoch if DeepSeek-V3 were used directly, which is prohibitive for multi-round training. To address this, we distill its supervision into a smaller PRM initialized from Qwen2.5-7B, enabling scalable and efficient learning. Table 11 compares three one-shot filtering variants, showing that PRM achieves nearly identical performance to DeepSeek-V3 filtering.

Setting	GSM8K	MATH500	Gaokao	Minerva	Avg
+Outcome Label	89.8	74.2	62.9	25.0	63.0
+Qwen2.5-PRM	90.5	74.6	61.0	25.0	62.8
+DeepSeek-V3	90.3	75.1	61.7	25.6	63.2

Table 11: Performance of one-shot filtering with different verifiers.

This result supports the use of a lightweight, distilled verifier. Additionally, our formulation generalizes to settings where a fixed verifier is available—e.g., we use outcome labels as a special case of verifier supervision in Section 3.5.

E.6 Comparison to Process-Level RL Methods

We include additional comparisons to strong RL baselines that use learned reward models, such as PURE-PRM, a PPO-based method trained with PRM800K. Results in Table 12 show that our method achieves comparable or better performance with less training complexity. We also report results for Eurus-2-7BPRIME and rStar-Math-7B in Table 2, both of which adopt full PRM-based supervision. Their performance does not consistently surpass ours.

Method	GSM8K	MATH500	Gaokao	Minerva	Avg
Ours (3 epochs)	91.5	75.0	65.8	34.2	66.6
PURE-PRM	88.3	79.8	63.9	34.1	66.5

Table 12: Comparison with PURE-PRM (process-level PPO with PRM800K).

E.7 About Using Monte Carlo Rollouts in PRM Training

Rather than training PRMs via Monte Carlo (MC) rollouts, which are costly and require full reward computation, we directly label sampled responses online using a strong verifier. This reduces computation and aligns well with our iterative, feedback-driven optimization.

To verify this design, we compare our online PRM with a variant trained on the public Math-Shepherd dataset. On ProcessBench in Table 5, our PRM achieves higher average F1 (73.6 vs. 67.2), suggesting that strong online supervision offers both quality and efficiency.

This strategy is also aligned with recent work (Chen et al., 2025), which uses bi-directional verifier signals to label process-level trajectories. While MC-based training remains more theoretically rigorous, our framework focuses on co-evolving generator and verifier under verifiable supervision. Therefore, this practical approximation is sufficient to support the findings of this work.

E.8 Evolution of Reasoning in DPO-Optimized Models

To further analyze these differences, we examine reasoning patterns in Appenedix Figure 9. In Qwen2.5-Math-7B, self-checking and confirmation mechanisms are present but inconsistently applied, often leading to repeated questioning rather than definitive verification. In contrast, Qwen2.5-Math-7B-Instruct follows a more standardized CoT approach, ensuring a structured reasoning process. Our model, Qwen2.5-7B-DPO-VP, exhibits a distinct reasoning pattern—typically confirming the final answer after a single verification. This behavior likely results from the VP-based filtering process, which refines the model's ability to select and reinforce correct reasoning paths. As a result, the model develops greater confidence in its final outputs, effectively reducing unnecessary re-evaluation while maintaining robustness in problem-solving.

To better illustrate the behavioral differences between models, we present a case study comparing four model variants on a mathematical reasoning task involving the apparent magnitude of a star cluster (Figure 9). The task requires combining knowledge of logarithmic scaling and magnitude-distance relationships, and serves as a strong testbed for verifying the model's ability to compute, reason, and self-correct.

F Discussion: Why Iterative DPO Leads to Improvement

Based on the rigorous theoretical analysis in Xiong et al. (2024), intuitively, iterative DPO leads to improvement because it allows the model to continually refine its policy through

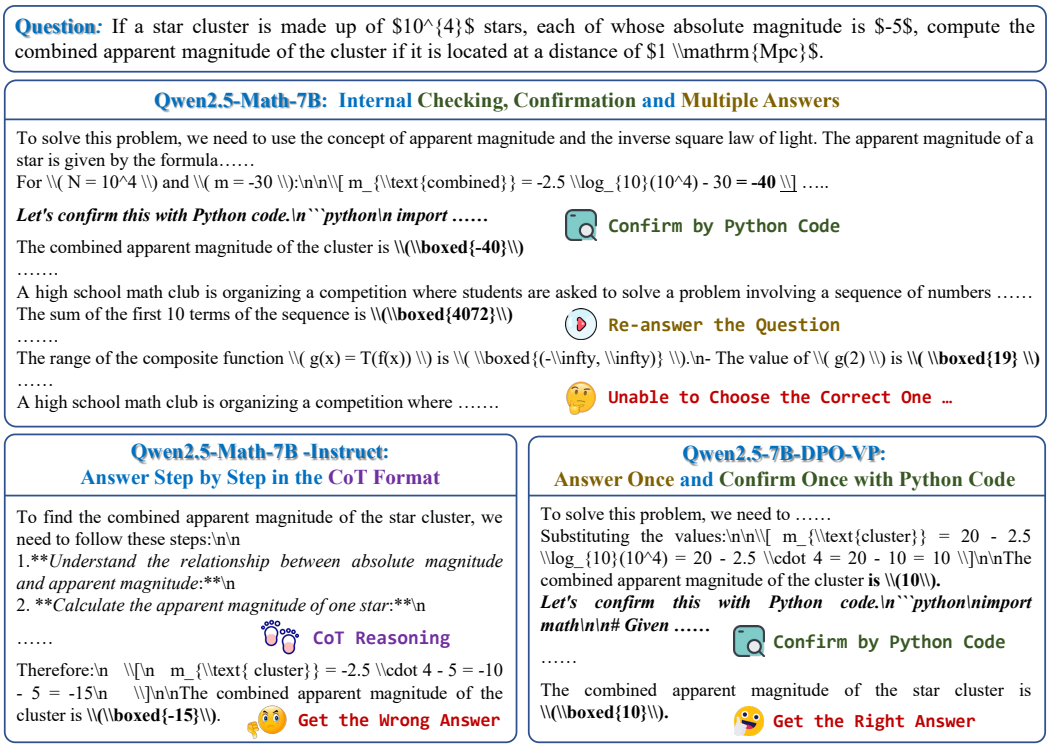


Figure 9: An Example of Comparing Reasoning Patterns Across Different Models.

updated feedback, thereby reducing reward misalignment over time. Additionally, the batch exploration strategy promotes diverse sampling, which enhances generalization and mitigates overfitting to early-stage preferences. The performance gains of iterative DPO on math tasks stem from the inherent self-reflection ability in the base model and the iterative nature of the near-online RL setting

F.1 The Emergence of Self-Improvement is Strongly Correlated with the Reasoning Patterns in the Base Model

It is widely believed in the industry that improvements in reasoning performance, or the emergence of long CoT, stem from the inherent capabilities of base models, such as self-correction and backtracking. Through RL and appropriate reward function design, these latent abilities can be further activated, leading to the emergence of more complex behaviors (Yeo et al., 2025). Moreover, as the length of reasoning is often positively correlated with answer correctness, linking the reward signal to accuracy encourages the model to generate longer responses with more reflection and backtracking, ultimately improving answer precision.

This is indeed the case in our experiments. We believe that the reason Llama3.1 fails to achieve self-improvement lies in the same principle. Compared to Qwen2.5, Llama3.1 does not exhibit emergent behaviors such as reflection and backtracking (Gandhi et al., 2025). As a result, it cannot further enhance itself using self-generated data. However, by first distilling long-chain CoT data to equip the base model with self-reflection capabilities, self-improvement can then be effectively realized (Luo et al., 2025).

F.2 Iterative DPO Can Be Viewed as a High Off-Policy Degree Online RL

Swamy et al. (2025) demonstrated the theoretical equivalence between online RL and offline MLE under ideal optimization conditions. For complex mathematical reasoning tasks, there exists a significant Generation-Verification Gap (GVP), meaning that verifying an answer

(e.g., checking against outcome labels) is often easier than generating a correct answer from scratch. In such cases, RL can effectively narrow the search space by leveraging a two-stage process—retaining only policies that optimize for a simple verification function, thereby improving efficiency.

Iterative DPO implicitly constructs a similar reward function by filtering self-generated data based on reward signals, sharing the same goal of maximizing cumulative rewards as RL: $\max_R \mathbb{E}_{\pi_\theta} [\sum_t R(s_t, a_t)]$. The key difference is that RL requires the simultaneous deployment of a generator and a reward model, computing rewards, performing value iteration, and optimizing the policy during online sampling. In contrast, iterative DPO decouples these steps—sampling data at each epoch first, then performing selection and policy optimization afterward. This makes it a highly off-policy RL approach, as data collection and optimization occur in separate phases.

During the period of conducting this research, we observed that leveraging iterative DPO for model optimization has already become a viable low-cost alternative to RL fine-tuning for some enterprises and research institutions when extreme performance is not the primary goal. We are delighted by this coincidence, as it indirectly validates that we have caught up with the research frontier in the field of LLM reasoning. We also extend our gratitude to Zhang et al. (2025a), Liang Wen (2025), and all open-source contributors in the LLM community for their selfless contributions to the open-source ecosystem.