# Step-level Value Preference Optimization for Mathematical Reasoning

**Guoxin Chen, Minpeng Liao, Chengxi Li, Kai Fan***
Tongyi Lab
chenguoxin22@mails.ucas.ac.cn
{minpeng.lmp,xiji.lcx,k.fan}@alibaba-inc.com

## Abstract

Direct Preference Optimization (DPO) using an implicit reward model has proven to be an effective alternative to reinforcement learning from human feedback (RLHF) for fine-tuning preference aligned large language models (LLMs). However, the overall preference annotations of responses do not fully capture the fine-grained quality of model outputs in complex multi-step reasoning tasks, such as mathematical reasoning. To address this limitation, we introduce a novel algorithm called *Step-level Value Preference Optimization* (SVPO). Our approach employs Monte Carlo Tree Search (MCTS) to automatically annotate step-level preferences for multi-step reasoning. Furthermore, from the perspective of learning-to-rank, we train an explicit value model to replicate the behavior of the implicit reward model, complementing standard preference optimization. This value model enables the LLM to generate higher reward responses with minimal cost during inference. Experimental results demonstrate that our method achieves state-of-the-art performance on both in-domain and out-of-domain mathematical reasoning benchmarks. Our code is available at https://github.com/MARIO-Math-Reasoning/Super_MARIO.

## 1 Introduction

Recently, large language models (LLMs) have demonstrated remarkable capability across a wide range of natural language processing (NLP) tasks (OpenAI, 2023; Du et al., 2022; Team et al., 2023; Chen et al., 2023; Anil et al., 2023; Bai et al., 2023; AI@Meta, 2024). However, they continue to encounter significant challenges when engaging in complex and symbolic multi-step reasoning, particularly in mathematical reasoning (Chen et al., 2022; Azerbayev et al., 2023; Yu et al., 2023b; Shao et al., 2024; Chen et al., 2024b; Kang et al., 2024; Chen et al., 2024a).

*Corresponding Author.

| Training Paradigm | Training Data | | Annotation from GPT-4 | Preference |
|---|---|---|---|---|
| | Pos. | Neg. | | |
| SFT | ✓ | ✗ | ✓ | ✗ |
| DPO | ✓ | ✓ | ✓ | Solution-level |
| SVPO (Ours) | ✓ | ✓ | ✗ | Step-level |

Table 1: Comparison of different training paradigm.

Most existing studies (Wang et al., 2023; Yue et al., 2023; Gou et al., 2023; Lu et al., 2024; Liao et al., 2024) have significantly improved the mathematical reasoning capabilities through fine-tuning on high-quality positive supervision data (*i.e*, correct solutions) annotated by GPT-4. In this process, a large number of negative examples generated by GPT-4 are wasted, and the model blindly imitates successful cases without understanding what the wrong solutions are. Therefore, preference learning, such as Direct Preference Optimization (DPO) (Rafailov et al., 2023), has been proposed to align with human preferences and enable the model to distinguish between positive and negative examples. However, most current efforts (Yuan et al., 2024; Chen et al., 2024d; Pang et al., 2024) focus on solution-level preferences, relying on humans or GPT-4 to generate and score complete solutions for training. This approach is expensive and often provides only a coarse preference relationship, which does not reflect the natural process by which humans learn to solve mathematical problems. This discrepancy arises because solution-level preferences pursue a solution to its final answer, without informing which step in the negative solution (*i.e.,* $\mathbf{y}^l$) led to the mistake. Unlike these approaches, humans tend to identify and analyze their mistakes step by step when learning to solve mathematical problems, thereby preventing repeated errors. In this manner, humans progressively learn to make informed decisions in similar states.

Furthermore, while DPO reparameterizes the reward function in reinforcement learning from hu-



*Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7889–7903
November 12-16, 2024 

man feedback (RLHF) (Ouyang et al., 2022) to improve simplicity and training stability, it also discards the state-value function $V(\mathbf{s})$, which is used to evaluate the expected return from the current state. Recent work (Liu et al., 2023; Liao et al., 2024) has demonstrated the effectiveness of the value model in improving the reasoning capabilities of policy models, but it is limited by the need for additional annotated data or the complexity of the reinforcement learning process.

To address the above issues, we propose *Step-level Value Preference Optimization* (SVPO), a novel preference learning framework that focuses on more fine-grained step-level preferences via Monte Carlo Tree Search (MCTS) to significantly enhance mathematical reasoning capabilities. Specifically, as illustrated in Figure 1, step-level preferences are autonomously generated through the MCTS framework (Silver et al., 2016, 2017). This approach not only avoids labor-intensive annotation but also provides detailed insights into which steps may lead to mistakes in $\mathbf{y}^l$, as indicated by the $Q$-value at each node. Compared to the forced knowledge infusion through GPT-4 annotated data, the preferences obtained through self-exploration are better aligned with the capabilities of the current LLM, highlighting the reasoning errors that the model is more prone to making. Furthermore, we integrate an explicit value model with DPO, where the value model is designed not only to assist the policy model (*i.e.*, LLM) in navigating more effective reasoning paths but also to steer preference learning. In our work, the value model is trained based on both $Q$-values and step-level preference relationships derived from MCTS, thereby bypassing the need for additional annotations and simplifying the training process.

We conduct extensive experiments on both in-domain and out-of-domain mathematical reasoning datasets. Our SVPO significantly outperforms state-of-the-art methods, achieving comparable or even superior results to GPT-4 on 7B LLMs. The experiments demonstrate three key points: **(1)**, the self-exploration process via MCTS naturally provides step-level preference relationships and highlights potential reasoning errors by $Q$-values; **(2)**, compared to solution-level preferences, step-level preferences can significantly enhance the reasoning capabilities of the policy model; **(3)**, the value model effectively guides the policy model's preference learning and reasoning.

## 2 Background

In standard RLHF framework, it first learns a reward model $r(\mathbf{x}, \mathbf{y})$ with Bradley-Terry (Bradley and Terry, 1952) preference optimization.

$$\mathcal{L}(r) = -\mathbb{E}_{\mathbf{x},\mathbf{y}^w,\mathbf{y}^l}\left[\log\sigma\left(r(\mathbf{x},\mathbf{y}^w) - r(\mathbf{x},\mathbf{y}^l)\right)\right] \quad (1)$$

where the expectation is taken over a preference dataset that includes tuples of prompts and preference responses $(\mathbf{x}, \mathbf{y}^w \succ \mathbf{y}^l)$. Following this, the policy model $\pi$ is optimized using the learned reward model $r$ and the proximal policy optimization (PPO) algorithm (Schulman et al., 2017). Typically, PPO requires maintaining 4 models in the training pipeline: a reward model $r$, a policy model $\pi$, a reference policy model $\pi'$, and a value model $V$, making it a complex procedure.

Instead of learning an explicit reward model, DPO only maintains 2 policy models and minimize the following objective.

$$\mathcal{L}_{\text{DPO}}(\pi) = -\mathbb{E}_{\mathbf{x},\mathbf{y}^w,\mathbf{y}^l}\left[\log\sigma\left(\beta\log\frac{\pi(\mathbf{y}^w|\mathbf{x})\pi'(\mathbf{y}^l|\mathbf{x})}{\pi'(\mathbf{y}^w|\mathbf{x})\pi(\mathbf{y}^l|\mathbf{x})}\right)\right] \quad (2)$$

where reference policy $\pi'$ is typically a supervised fine-tuning (SFT) model. The implicit reward model is characterized by the log-likelihood ratio between two policy models. Although DPO simplifies the training process, it discards the value model, which has been proven effective in improving the reasoning capabilities of the policy model. Additionally, the coarse preferences derived from existing annotation methods also limit its performance in multi-step reasoning tasks.

## 3 Method

In this section, we present our SVPO in detail to further explore the potential of preference learning in multi-step reasoning tasks, particularly in mathematical reasoning.

### 3.1 Step-level Preference Annotation

Unlike the traditional annotations that only provide solution-level preferences, we employ the MCTS framework to encourage LLMs to autonomously explore step-level generations as well as infer step-level preferences, as shown in Figure 1. In this manner of self-exploration, we obtain more fine-grained preferences, while the $Q$-values at each step (tree node) indicate potential reasoning errors that traditional annotations cannot achieve.

The policy model for running MCTS is a SFT model of multi-step reasoning, denoted as

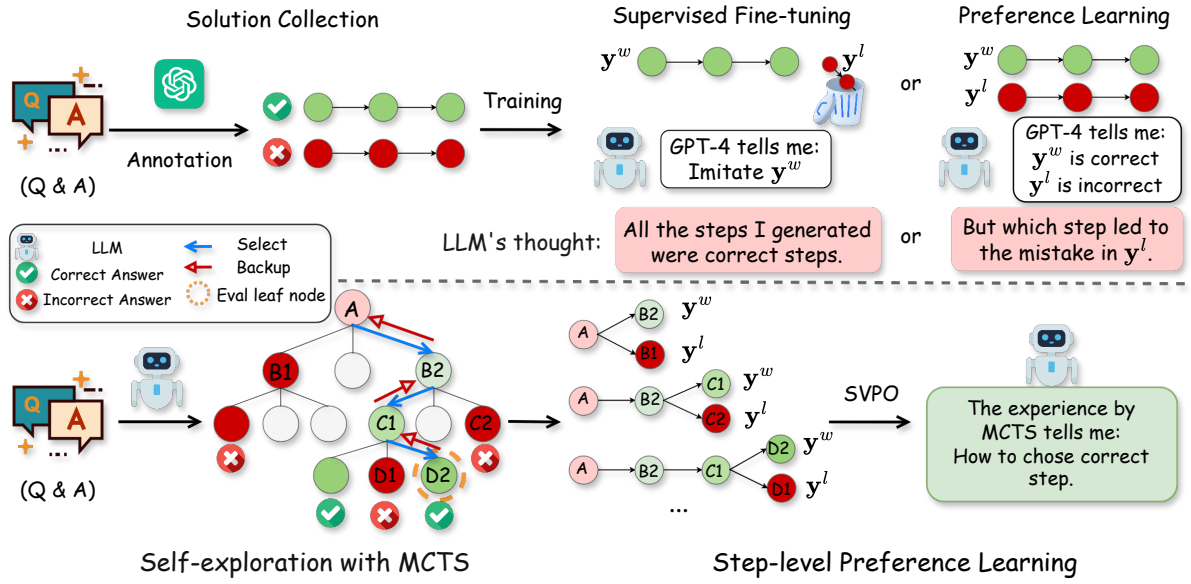



Figure 1: Comparison of different frameworks: SFT, DPO, and SVPO. The top panel shows the typical pipeline of SFT and DPO, where GPT-4 does not indicate which step in $\mathbf{y}^l$ led to the mistake. The bottom panel illustrates the pipeline of SVPO. Step-level preferences are autonomously generated via MCTS, where $Q$-values (represented by node colors) indicate potential reasoning errors.

$\pi(\mathbf{y}_{1:T}|\mathbf{x})$, where $\mathbf{x}$ is the prompted question and $\mathbf{y}_t$ represents the $t$-th step. In the parlance of reinforcement learning, the state and action are defined as $\mathbf{s}_t = \mathbf{y}_{<t}$ and $\mathbf{a}_t = \mathbf{y}_t$, respectively. In addition, the state transition function is deterministic as $\mathbf{s}_{t+1} = \text{ConCat}[\mathbf{s}_t, \mathbf{a}_t]$.

Our primary objective in annotating preferences is to compare the quality of two potential step-level generations. Concretely, this can be transformed into comparing the $Q$-values of two possible actions for the same previous state:

$$Q(\mathbf{s}_t, \mathbf{a}_t^{(1)}) \quad v.s. \quad Q(\mathbf{s}_t, \mathbf{a}_t^{(2)}) \tag{3}$$

Next, we will introduce the detailed MCTS process to automatically derive the $Q$-values. Specifically, we will iterate through the following four operations until convergence.

**Selection** Given the current tree $\mathcal{T}$, MCTS first needs to select a leaf node as a candidate for further exploration. By initializing the state $\mathbf{s}$ as the root, we use the PUCT criterion (Rosin, 2011) until a leaf node is encountered.

$$\arg\max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a}) + c_{\text{puct}} \frac{\sum_j \pi(a_j|\cdot)}{|\mathbf{a}|} \frac{\sqrt{N_{\text{parent}}(\mathbf{a})}}{1 + N(\mathbf{s}_t, \mathbf{a})} \tag{4}$$

where $N(\cdot)$ represents the visit count, and $a_j$ is the $j$-th token in the step.

**Expansion and Evaluation** Given the state represented by the selected leaf node, we sample multiple possible candidate actions for the next step. To encourage diversity, a higher temperature, typically ranging from 0.6 to 1, is used.

For efficient evaluation, we reuse the expanded nodes and simply apply a one-step rollout. If the rollout action is not terminal, we directly set the value of the current leaf node to 0. Otherwise, the final answer in the terminal action is evaluated for equivalence to the ground truth. If the final answer is correct, the reward $R$ will be 1; otherwise, it will be -1. Therefore, the value can be written as follows.

$$V(\mathbf{s}) = \mathbb{I}_{\text{terminal}}(\mathbf{a}) \cdot R(\mathbf{s}, \mathbf{a}) \tag{5}$$

where $\mathbb{I}(\cdot)$ is the indicator function.

**Backup** For the terminal nodes reached during the rollout and the current leaf node, MCTS performs a backward update of the visit count and $Q$-value for every $(\mathbf{s}', \mathbf{a}')$ along the path from current node to the root.

$$\begin{aligned} N(\mathbf{s}', \mathbf{a}') &\leftarrow N(\mathbf{s}', \mathbf{a}') + 1 \\ Q(\mathbf{s}', \mathbf{a}') &\leftarrow Q(\mathbf{s}', \mathbf{a}') + \frac{1}{N(\mathbf{s}', \mathbf{a}')}(V(\mathbf{s}) - Q(\mathbf{s}', \mathbf{a}')) \end{aligned} \tag{6}$$

As shown in Figure 1, we obtain a solution tree $\mathcal{T}$ with many branches after running the above MCTS process for several iterations. From this

tree, we can extract a partial solution and its two different next steps along with their corresponding $Q$-values. The step with the larger $Q$-value will be annotated as the preferred example.

### 3.2 Step-level Preference Learning

Given our autonomously generated step-level preference annotations, we propose an approach called step-level value preference optimization–SVPO. In contrast to DPO, we maintain 3 models with an additional value model $V_\phi$. Unlike in PPO, our value model is lightweight, achieved by adding an auxiliary value head directly over the policy model. This value head consists of a single linear layer with a `tanh` activation function, running parallel to the linear layer used for token prediction.

For notation simplification, we denote the annotated step-level preference instance $(\mathbf{s}_t, \mathbf{a}_t^w \succ \mathbf{a}_t^l)$ as $(\mathbf{s}_{t+1}^w \succ \mathbf{s}_{t+1}^l)$, where the two multi-step generations are only different at their last steps. According to our previous definition, the state $\mathbf{s}_{t+1}$ also represents the first $t$ steps $\mathbf{y}_{1:t}$ .

**Pre-Training** In DPO, the policy model is pre-trained with a standard SFT loss. In our approach, due to the weights sharing architecture between $\pi$ and $V_\phi$, our pre-training adopts the multi-task loss.

$$\begin{aligned} \hat{V}(\mathbf{s}_{t+1}) &= \begin{cases} R(\mathbf{s}_t, \mathbf{a}_t), & \mathbf{a}_t \text{ is terminal} \\ Q(\mathbf{s}_t, \mathbf{a}_t), & \text{otherwise} \end{cases} \\ \mathcal{L} &= \mathcal{L}_{\text{SFT}} + \mathbb{E}\left[(V_\phi(\mathbf{s}_{t+1}) - \hat{V}(\mathbf{s}_{t+1}))^2\right] \end{aligned} \tag{7}$$

The mean squared error (MSE) loss is employed to pre-train the value head, which is also the pointwise approach in ranking algorithm (Liu et al., 2009). The label for the value prediction is either the $Q$-value of the intermediate step or the final reward.

**SVPO** As indicated by DPO, the difference of implicit rewards for a pair of preference annotations can be re-parameterized as follows:

$$\Delta r_\pi(\mathbf{s}_{t+1}^w, \mathbf{s}_{t+1}^l) = \beta \log \frac{\pi(\mathbf{s}_{t+1}^w)\pi'(\mathbf{s}_{t+1}^l)}{\pi'(\mathbf{s}_{t+1}^w)\pi(\mathbf{s}_{t+1}^l)} \tag{8}$$

In our SVPO, we aim to optimize both policy and value models through preference learning. Accordingly, we define the explicit value difference.

$$\Delta r_\phi(\mathbf{s}_{t+1}^w, \mathbf{s}_{t+1}^l) = V_\phi(\mathbf{s}_{t+1}^w) - V_\phi(\mathbf{s}_{t+1}^l) \tag{9}$$

We then propose the following SVPO loss function, which includes three different objectives.

$$\begin{aligned} \mathcal{L}_{\text{SVPO}} &= -\log \sigma(\Delta r_\pi(\mathbf{s}_{t+1}^w, \mathbf{s}_{t+1}^l)) \\ &+ \max(0, \gamma - \Delta r_\phi(\mathbf{s}_{t+1}^w, \mathbf{s}_{t+1}^l)) \\ &+ \left(\Delta r_\pi(\mathbf{s}_{t+1}^w, \mathbf{s}_{t+1}^l) - \text{sg}\left[\Delta r_\phi(\mathbf{s}_{t+1}^w, \mathbf{s}_{t+1}^l)\right]\right)^2 \end{aligned} \tag{10}$$

where the margin $\gamma \geq 0$ is tunable hyper-parameter, and $\text{sg}[\cdot]$ denotes the stop gradient operator.

The first objective essentially replicates the original DPO loss $\mathcal{L}_{\text{DPO}}$ in (2), applied to the automatically annotated step-level preference data.

The second objective is a margin loss for value preference learning, inspired by the pairwise ranking algorithm (Liu et al., 2009). Given a non-negative margin $\gamma$, minimizing this loss encourages the value of the positive example to be larger than that of the negative one by at least $\gamma$. The detailed theoretical analysis can refer to (Chen et al., 2009).

The third objective is a regularization term adapted MSE loss, which aims to ensure a similar preference scale between the implicit reward model and our proposed explicit value model. In this loss, we use $\Delta r_\phi$ as the targeted label and detach its gradient to prevent model degeneration.

**Analysis of Regularization** A natural question regarding the regularization term is whether designing the value output via `tanh` can match the reward defined in the log-likelihood ratio. We can first derive the possible theoretical matching range.

$$\begin{aligned} &\Delta r_\phi \in [-2, 2] \\ \Rightarrow &\frac{\pi(\mathbf{y}^w)\pi'(\mathbf{y}^l)}{\pi'(\mathbf{y}^w)\pi(\mathbf{y}^l)} = e^{\Delta r_\pi/\beta} \in [e^{-2/\beta}, e^{2/\beta}] \end{aligned} \tag{11}$$

Therefore, the range is determined by $\beta$.

**(1)** $\lim_{\beta\to 0}[e^{-2/\beta}, e^{2/\beta}] = (0, +\infty)$: when $\beta$ is small in DPO or PPO, it can prevent the policy model from deviating too far. For example of the commonly used $\beta = 0.1$, the allowed matching range becomes $[e^{-20}, e^{20}]$, which is actually equivalent to $(0, +\infty)$ in the context of numerical precision. In other words, for smaller $\beta$, our regularization loss can easily match the scale between implicit and explicit preferences.

**(2)** $\lim_{\beta\to\infty}[e^{-2/\beta}, e^{2/\beta}] = \{1\}$: when $\beta$ becomes large, the allowed matching range will gradually center around 1, forcing the distance between $\pi$ and $\pi'$ to be closer. In other words, for larger $\beta$, our regularization loss can also play the role of preventing the policy model from deviating too far.

### 3.3 Step-level Inference

Even without the value model, one can still directly apply greedy decoding to the policy model. However, incorporating the value model and an associated reranking criterion allows step-level beam search (SBS) (Yu et al., 2023a; Chen et al., 2024a) to effectively select the preferred solution path in

mathematical reasoning, all while incurring a lower computational cost compared to MCTS. Since our value preference learning is optimized at the step level and utilizes ranking loss, our approach seamlessly integrates with the inference framework of step-level beam search.

## 4 Experiments

### 4.1 Experimental Setup

We validate the applicability of our framework across various base models, including math domain-specific pre-trained models such as DeepseekMath-Base-7B (Shao et al., 2024), as well as general pre-trained models such as Llama3 (AI@Meta, 2024). In this study, we mainly focus on how to improve mathematical reasoning skills through step-level preference learning. Therefore, we obtain the corresponding multi-step SFT models using 27k MARIO seed data (Liao et al., 2024) in XML format, as detailed in Appendix A.5.

**Step-level Preference Annotation via MCTS** Given a multi-step SFT model for mathematical reasoning, we only extract the 15k questions from the GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) datasets. Following the methodology described in Section 3.1, we employ the MCTS framework to automatically generate both multi-step solutions and step-level preferences. This process requires no supervision from either humans or GPT-4. Particularly, for each question, we continue constructing trees until we obtain four complete and correct multi-step solutions or until the number of trees reaches 10. We then extract step-level preferences from the trees in a top-down manner, maintaining an approximate ratio of 1:4 between positive and negative examples. Consequently, we acquire a total of 56k complete positive instances $\mathbf{y}^w$. Additional details are provided in Appendix A.2.

**Test sets** The in-domain test sets from GSM8K and MATH share the same distribution as our training data. Meanwhile, we evaluate our final checkpoint on the out-of-domain (OOD) datasets GaoKao2023 (Liao et al., 2024) and OCWCourses (Lewkowycz et al., 2022). These OOD test sets are even more challenging than the MATH dataset but inherently require multi-step reasoning.

**Baselines** For commercial and popular open-source models, we compared our approach with OpenAI's ChatGPT and GPT-4 (OpenAI, 2023), Llama2 (Touvron et al., 2023), and Llemma (Azerbayev et al., 2023) using the Chain of Thought (CoT) (Wei et al., 2022) and Program-Aided Language (PAL) (Gao et al., 2023). Additionally, we benchmarked our method against recent high-performing fine-tuned mathematical LLMs, including MAmmoTH (Yue et al., 2023), MathCoder (Wang et al., 2023), ToRA (Gou et al., 2023), MARIO (Zhang et al., 2024), MathGenie (Lu et al., 2024), DeepSeekMath-Instruct (Shao et al., 2024), and AlphaMath (Chen et al., 2024a). Similar to our approach, these models leverage a Python code interpreter for numerical calculations. Further implementation details are provided in Appendix A.

### 4.2 Main Results

For a fair comparison, in Table 2, we report the in-domain and out-of-domain results of our SVPO based on DeepSeekMath-Base-7B, which is consistent with the state-of-the-art methods, such as DeepSeekMath-Instruct (Shao et al., 2024) and AlphaMath (Chen et al., 2024a).

**Greedy Decoding** Without the assistance of a value model, we first evaluate the policy model using greedy decoding, which is comparable to most related works. The main conclusion is that for more difficult problems requiring more reasoning steps, our approach shows greater advantages. As the difficulty increases for GSM8K, MATH, GaoKao2023 (GK2023), and OCWCourses (OCW), our approach achieves improvements of -2.0% / +2.1% / +3.2% / +16.2% over the previous state-of-the-art, DeepSeekMath-Instruct.

We slightly lag behind in GSM8K, which could be attributed to two possible reasons. First, GSM8K usually requires single step solution and less logical reasoning. Second, the diversity of our training dataset is limited. While DeepSeekMath utilized 776k high-quality supervised data, we only autonomously generated 56k complete positive examples based on 15k questions.

**SBS** With the value model optimized by step-level value preference learning, we can utilize the computationally efficient step-level beam search (SBS) to investigate the role of the value model in facilitating mathematical reasoning. Compared to greedy decoding, the value model significantly assists the policy model in navigating more effective reasoning paths, rather than solely relying on prior probabilities. Compared with AlphaMath, our

| Model | Size | Tool | Zero Shot | In-Domain GSM8k | In-Domain MATH | OOD GK2023 | OOD OCW |
|---|---|---|---|---|---|---|---|
| Proprietary Models | | | | | | | |
| GPT-4 | - | ✗ | ✗ | 92.0 | 42.5 | - | - |
| GPT-4 (PAL) | - | ✓ | ✗ | 94.2 | 69.7 | 43.6 | 30.1 |
| ChatGPT | - | ✗ | ✗ | 80.8 | 35.5 | - | - |
| ChatGPT (PAL) | - | ✓ | ✗ | 78.6 | 38.7 | - | - |
| Open-Source Models | | | | | | | |
| Llama-2 | 7B | ✗ | ✗ | 13.3 | 4.1 | - | 3.7 |
| CodeLlama | 7B | ✗ | ✗ | 10.5 | 4.5 | - | 4.7 |
| CodeLlama(PAL) | 7B | ✓ | ✗ | 27.1 | 17.2 | - | - |
| Llemma | 7B | ✗ | ✗ | 36.4 | 18.0 | - | 7.7 |
| Llemma (PAL) | 7B | ✓ | ✗ | 40.1 | 21.5 | - | - |
| DeepSeekMath-Base(PAL) | 7B | ✓ | ✗ | 66.9 | 31.4 | - | - |
| Tuning Models | | | | | | | |
| MAmmoTH-Coder | 34B | ✓ | ✓ | 72.7 | 43.6 | 25.2 | 14.0 |
| MathCoder | 34B | ✓ | ✓ | 81.7 | 46.1 | - | - |
| ToRA-Code | 34B | ✓ | ✓ | 80.7 | 50.8 | 31.7 | 5.5 |
| MARIO | 34B | ✓ | ✓ | 78.2 | 53.5 | 42.6 | 30.2 |
| MathGenie | 34B | ✓ | ✓ | 84.1 | 55.1 | - | - |
| Llama-2 SFT | 7B | ✗ | ✓ | 41.3 | 7.2 | - | - |
| Llama-2 RFT | 7B | ✗ | ✓ | 51.2 | - | - | - |
| MAmmoTH-Coder | 7B | ✓ | ✓ | 59.4 | 33.4 | 15.3 | 11.0 |
| MathCoder | 7B | ✓ | ✓ | 67.8 | 30.7 | - | - |
| ToRA | 7B | ✓ | ✓ | 68.8 | 40.1 | 19.5 | 2.6 |
| ToRA-Code | 7B | ✓ | ✓ | 72.6 | 44.6 | 23.9 | 4.8 |
| MARIO | 7B | ✓ | ✓ | 74.5 | 48.3 | 34.5 | 21.7 |
| MathGenie | 7B | ✓ | ✓ | 76.0 | 48.3 | - | - |
| DeepSeekMath-Instruct | 7B | ✓ | ✓ | **83.7** | 57.4 | 43.9 | 18.0 |
| AlphaMath | 7B | ✓ | ✓ | 73.5 | 53.6 | 40.5 | 26.1 |
| + SBS ($B_1 = 1$) | | ✓ | ✓ | 81.1 | 62.8 | 46.2 | 30.5 |
| + SBS ($B_1 = 3$) | | ✓ | ✓ | 84.1 | 66.3 | 51.4 | 33.1 |
| SVPO (Ours) | 7B | ✓ | ✓ | 81.7 | **59.5** | **47.1** | **34.2** |
| + SBS ($B_1 = 1$) | | ✓ | ✓ | 85.9 | 64.4 | 54.6 | 36.8 |
| + SBS ($B_1 = 3$) | | ✓ | ✓ | **86.5** | **67.2** | **55.3** | **40.8** |

Table 2: Main results. The best results for greedy decoding and step-level beam search (SBS) are highlighted in bold and blue box, respectively. By default, we set the beam size $B_2 = 5$ in SBS.

SVPO achieves an average improvement of 5.3% / 3.7% on $B_1 = 1$ / $B_1 = 3$, respectively, which demonstrates the effectiveness of our approach. We will further analyze the value model in subsequent ablation studies. It is worth noting that with the help of the value model, our SVPO on 7B LLMs achieves comparable or even better results than GPT-4 in the challenging datasets.

### 4.3 Analysis 1: Policy Model

In this section, we will investigate the impact of step-level preferences on the policy model and explore the performance of different base models in our SVPO framework.

**Ablation Study of Training Paradigm** As shown in Table 3, we compare the performance of the policy model under different preference optimization. Our principal findings are as follows: **(1)** Compared to SFT, which blindly imitates positive examples $\mathbf{y}^w$, preference learning encourages the policy model to distinguish between $\mathbf{y}^w$ and $\mathbf{y}^l$, thereby enhancing its reasoning capability. However, solution-level DPO is limited by its coarse

| Training Paradigm | In-Domain GSM8k | In-Domain MATH | OOD GK2023 | OOD OCW |
|---|---|---|---|---|
| SFT | 77.7 | 56.9 | 43.1 | 27.5 |
| DPO† | 78.9 | 57.1 | 45.4 | 28.3 |
| SVPO (Ours) | 81.7 | 59.5 | 47.1 | 34.2 |
| - w/o regularization | 80.2 | 58.3 | 46.2 | 32.1 |

Table 3: Ablation study of training paradigm on policy model. †Solution-level DPO.

| Model | In-Domain GSM8K | In-Domain MATH | OOD GK2023 | OOD OCW |
|---|---|---|---|---|
| Llama3-8B + SFT | 75.9 | 46.5 | 33.2 | 10.3 |
| Llama3-8B-Instruct | 79.6 | 30.0 | - | - |
| Llama3-70B-Instruct | **93.0** | 50.4 | - | - |
| Llama3-8B + SVPO | 81.3 | 48.8 | 35.6 | 11.1 |
| + SBS ($B_1 = 1$) | 84.3 | 54.2 | 40.0 | 13.3 |
| + SBS ($B_1 = 3$) | 85.5 | **56.3** | **43.7** | **16.6** |

Table 4: Performance comparison of Llama3 series.

preference relationships, which do not indicate which specific step in the negative solutions $\mathbf{y}^l$ led to the mistakes. **(2)** Compared to solution-level DPO, our proposed step-level preferences can significantly enhance the reasoning performance of the policy model on both in-domain and out-of-domain datasets. This can be attributed to the more granular information of reasoning steps reflected by $Q$-value in the Monte Carlo tree. **(3)** The value model can further guide the optimization of the policy model, as evidenced by the performance decreases when the regularization term is removed.

**Discussion of Different Base Models** We further investigate the performance of the general pre-trained model, Llama3 (AI@Meta, 2024), within our framework. As shown in Table 4, we have the following main findings: **(1)** Compared to DeepSeekMath-Base-7B in Table 2, the overall performance of the general pre-trained model Llama3 is relatively insufficient. This is because DeepSeekMath-Base is pre-trained on a substantial math-related corpus and is believed to process more necessary mathematical knowledge, resulting in higher quality preference data. **(2)** Our SVPO outperforms the instruction-tuned Llama3 and approached the performance of the 70B model. Furthermore, compared to the SFT model, we achieved significant improvements, demonstrating the effectiveness and applicability of our approach.

| Method | SBS | In-Domain GSM8K | In-Domain MATH | OOD GK2023 | OOD OCW |
|---|---|---|---|---|---|
| SVPO (Ours) | $B_1 = 1$ | 85.9 | 64.4 | 54.6 | 36.8 |
| | $B_1 = 3$ | 86.5 | 67.2 | 55.3 | 40.8 |
| w/o Margin loss | $B_1 = 1$ | 85.4 | 62.5 | 49.6 | 34.9 |
| | $B_1 = 3$ | 85.2 | 63.7 | 52.2 | 37.5 |
| w/o MSE loss | $B_1 = 1$ | 83.8 | 60.5 | 52.2 | 30.8 |
| | $B_1 = 3$ | 82.1 | 56.7 | 45.7 | 28.6 |

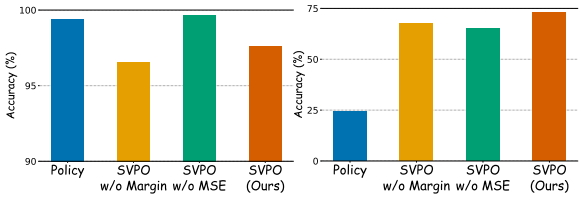
Table 5: Ablation study of value model.

### 4.4 Analysis 2: Value Model

In this section, we further investigate the impact of value loss (mainly including the MSE loss in Eq. (7) and Margin loss in Eq. (10) on performance and evaluate the accuracy of identifying preferences.

**Ablation Study of Value Loss** As shown in Table 5, we compare the performance of step-level beam search in different setups for value loss. Our principal findings are as follows: **(1)** When margin loss is omitted, which describes the local relationships in preference data, the performance will decrease. As explained in the method, this can be attributed to the ability of margin loss to further distinguish the value of candidate actions. **(2)** MSE loss is crucial for the value model, as it provides global information for each node in the Monte Carlo tree. Relying solely on preference relationships by margin loss may cause the value model to lose the ability to screen cousin nodes (*i.e.*, candidate actions at the same level but with different previous states). This explains why the performance of $B_1 = 3$ is significantly lower than that of $B_1 = 1$ when MSE loss is omitted. In summary, MSE loss and margin loss provide complementary information, and their combined effect leads to a better value model.

**Win Rate of Preference** We conduct a further investigation into the accuracy of the policy and value model by Eq. (8) and (9) in assessing preference relationships. We randomly select 200 questions

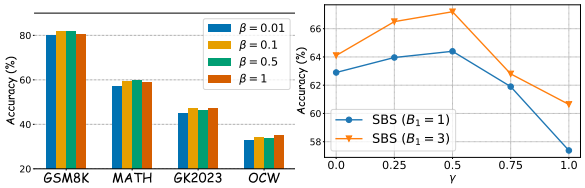



(a) Training Set (b) Test set

Figure 2: Win Rate of Preference.

from the test sets of GSM8K and MATH respectively, and utilize MCTS to build preferences as the test set in the win rate. As shown in Figure 2, we have the following main findings: **(1)** The preference relationships in training sets can be easily mastered, as evidenced in Figure 2a, where the accuracy of both "Policy" and "SVPO w/o MSE" significantly surpasses that of others. However, the poor performance of "Policy" on the test set indicates that the implicit reward model (*i.e.*, policy model) is highly susceptible to overfitting. **(2)** Compared to the implicit reward model, our proposed explicit value model is relatively stable even if it only learns preference relationships by margin loss. This further demonstrates the effectiveness of our value model.

### 4.5 Sensitivity of $\beta$ and $\gamma$




(a) Beta $\beta$ (b) Margin $\gamma$

Figure 3: Hyperparameter sensitivity analysis.

$\beta$ in Eq. (8) controls the implicit reward model, while the margin $\gamma$ in Eq. (10) controls the explicit value model. Thus, we investigate the impact of the two key hyper-parameters.

**Beta $\beta$** Following DPO (Rafailov et al., 2023), we investigate the impact of different $\beta$ on the policy model, as shown in Figure 3a. We observe that the optimization of the policy model remains relatively stable across different $\beta$. This can be attributed to the regularization term, as analyzed in Section 3.2. The explicit value model can prevent the policy model $\pi$ from deviating too far from the reference policy model $\pi'$ through the regularization term, thereby improving the stability.

**Margin $\gamma$** As shown in Figure 3b, we evaluate the results of SBS in MATH with varying $\gamma$ between $[0, 1]$. We observe that an excessively large $\gamma$ will cause the value model to degenerate. This can be attributed to the fact that a large margin $\gamma$ compresses the predicted values towards either -1 or 1, making it difficult for the value model to correctly differentiate between candidate actions at the same level in SBS. Moreover, setting $\gamma$ to 0 also leads to a performance degradation, indicating that it is not the optimal target margin. Although with $\gamma = 0$ the value model still maintains the value preference learning, an appropriate gamma is conducive to increase the confidence in scoring and enhances the model's generalization.

## 5 Related Work

**Mathematical Reasoning** Recent work (Gou et al., 2023; Liao et al., 2024; Lu et al., 2024; Shao et al., 2024) has achieved remarkable progress in mathematical reasoning. However, most efforts focus solely on supervised fine-tuning, which makes LLMs blindly imitate positive solutions without understanding what the wrong solutions are.

**Preference Learning** Recently, preference learning (Rafailov et al., 2023; Ethayarajh et al., 2024; Chen et al., 2024c) has attracted significant attention due to its ability to align with human preferences and distinguish between positive and negative examples. However, due to focusing solely on coarse solution-level preferences, most existing work is limited in performance on multi-step reasoning tasks, particularly in mathematical reasoning. Compared to previous work, our SVPO autonomously annotates step-level preferences through MCTS, and reflects potential reasoning errors through the $Q$-values at each step, thereby significantly improving the performance of preference learning on multi-step reasoning tasks.

**Value Model** The value model is derived from the state-value function in reinforcement learning (RL), which is used to evaluate the expected return of the current state. Recent work (Liu et al., 2023) has found that the value model can effectively enhance the reasoning capability of the policy model but limited by the complex training process of RL. In our study, we propose step-level value preference optimization, which achieves higher quality value models in a simpler training process.

## 6 Conclusion

In this study, we introduce *Step-level Value Preference Optimization* (SVPO) by extending Direct Preference Optimization (DPO) through the integration of a lightweight step-level value model. The training framework of SVPO is much more computationally efficient compared to Proximal Policy Optimization (PPO). Extensive experimental results demonstrate that for tasks involving

multi-step mathematical reasoning, our approach significantly enhances performance, particularly with the support of the proposed value model.

## Limitations

First, we consider our work, SVPO, as a trade-off approach between DPO and PPO, offering a relatively lower computational cost. Beyond being an Empirical Method in Natural Language Processing (EMNLP), the theoretical foundation of margin loss in the area of learning-to-rank has been extensively discussed in Chen et al. (2009), and we also theoretically analyze how the regularization loss in $\mathcal{L}_{\text{SVPO}}$ impacts the preference learning of the policy model. Nevertheless, in the future work, we need to establish a more solid theoretical foundation to connect the implicit reward model and the explicit value model.

Secondly, although our method has achieved excellent performance in multi-step reasoning, particularly in mathematical reasoning, there is still an issue that deserves further exploration in future work: whether our SVPO can enhance mathematical reasoning capabilities in the context of multimodal data. This may include mathematical reasoning from multiple combinations of modalities, such as language, images, tables, or audio, which is an increasingly prevalent and demanding type of reasoning in real-world scenarios. In future work, we plan to extend our SVPO to accommodate multimodal scenarios.

Additionally, although in this study we integrate the step-level value preference optimization into DPO as an example, our approach is broadly applicable to various types of preference learning algorithms (Ethayarajh et al., 2024; Chen et al., 2024c; Hong et al., 2024). In future work, we will explore incorporating our SVPO into these preference learning algorithms.

## Ethics Statement

This work primarily focuses on mathematical reasoning tasks, and our contributions are entirely methodological. Therefore, this work does not have direct negative social impacts. For the experiments, we have open-sourced the code and utilized openly available datasets commonly used in previous research, without any sensitive information to our knowledge. The authors of this work adhere to the ACL ethical guidelines, and the application of this work does not present any apparent issues that may lead to ethical risks.

## Acknowledgments

This work was supported by Alibaba Research Intern Program.

## References

AI@Meta. 2024. Introducing Meta Llama 3: The most capable openly available LLM to date.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

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*.

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024a. Alphamath almost zero: process supervision without process. *Preprint*, arXiv:2405.03553.

Guoxin Chen, Yiming Qian, Bowen Wang, and Liangzhi Li. 2023. MPrompt: Exploring multi-level prompt tuning for machine reading comprehension. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5163–5175, Singapore. Association for Computational Linguistics.

Guoxin Chen, Kexin Tang, Chao Yang, Fuying Ye, Yu Qiao, and Yiming Qian. 2024b. SEER: Facilitating structured reasoning and explanation via reinforcement learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5901–5921, Bangkok, Thailand. Association for Computational Linguistics.

Huayu Chen, Guande He, Hang Su, and Jun Zhu. 2024c. Noise contrastive alignment of language models with explicit rewards. *CoRR*, abs/2402.05369.

Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhi-Ming Ma, and Hang Li. 2009. Ranking measures and loss functions in learning to rank. *Advances in Neural Information Processing Systems*, 22.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *CoRR*, abs/2307.08691.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. GLM: General language model pretraining with autoregressive blank infilling. In *ACL*.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. KTO: model alignment as prospect theoretic optimization. *CoRR*, abs/2402.01306.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: program-aided language models. In *ICML*, Proceedings of Machine Learning Research.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, et al. 2023. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*.

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. In *NeurIPS*.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. ORPO: monolithic preference optimization without reference model. *CoRR*, abs/2403.07691.

Jikun Kang, Xin Zhe Li, Xi Chen, Amirreza Kazemi, and Boxing Chen. 2024. Mindstar: Enhancing math reasoning in pre-trained llms at inference time. *arXiv preprint arXiv:2405.16265*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *SOSP*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *NeurIPS*.

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*.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. 2023. Making PPO even better: Value-guided monte-carlo tree search decoding. *CoRR*, abs/2309.15028.

Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*. OpenReview.net.

Zimu Lu, Aojun Zhou, Houxing Ren, Ke Wang, Weikang Shi, Junting Pan, Mingjie Zhan, and Hongsheng Li. 2024. Mathgenie: Generating synthetic data with question back-translation for enhancing mathematical reasoning of llms. *arXiv preprint arXiv:2402.16352*.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

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

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. 2024. Iterative reasoning preference optimization. *arXiv preprint arXiv:2404.19733*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*.

Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. 2021. Zero-infinity: breaking the GPU memory wall for extreme scale deep learning. In *International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM.

Christopher D Rosin. 2011. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

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*.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the game of go without human knowledge. *Nature*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Ke Wang, Houxing Ren, Aojun Zhou, Zimu Lu, Sichun Luo, Weikang Shi, Renrui Zhang, Linqi Song, Mingjie Zhan, and Hongsheng Li. 2023. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning. *Preprint*, arXiv:2310.03731.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

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*.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

Boning Zhang, Chengxi Li, and Kai Fan. 2024. MARIO eval: Evaluate your math LLM with your math LLM-A mathematical dataset evaluation toolkit. *CoRR*, abs/2404.13925.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *CoRR*, abs/2403.13372.

# A Implementation Details

## A.1 Detailed Setup

**For step-level preference annotation via MCTS**, we set $c_{\text{puct}}$ to 1.25, set the temperature within the range of 0.6 to 1, limit the maximum tree depth to 8, set each node to expand 5 child nodes, and simulate at most 60 times. For each question in training set, we construct at most 10 trees. Following Chen et al. (2024a), we define two types of steps in MCTS, $\mathcal{C}$-step and $\mathcal{A}$-step. The $\mathcal{C}$-step is responsible for code execution and consists of text analysis, code snippets, and execution results. The $\mathcal{A}$-step is responsible for summarizing the answers, comprising text analysis and the final answer. We organize these two steps in the following XML format:

$\mathcal{C}$-step

```
<step>
<p>
{textual analysis}
</p>
<code>
{code snippets}
</code>
<p>
{code output}
</p>
</step>
```

$\mathcal{A}$-step

```
<step>
<p>
{textual analysis}
</p>
<p>
Final Answer: {predicted answer}
</p>
</step>
```

**For Pre-training via SFT**, we convert the pre-trained model into a corresponding multi-step SFT model through the pre-training loss in Eq. (7). We set the learning rate to 2e-5, the batch size to 512, fix the MSE weight to 0.01, and train for 10 epochs. We employ the AdamW optimizer (Loshchilov and Hutter, 2019) and a cosine learning rate scheduler, setting the warm-up rate to 0.03.

**For SVPO**, we set $\beta$ to 0.1, $\gamma$ to 0.5, learning rate to 5e-6, batch size to 512, and train for 1 epoch. Since preference learning may easily degenerate model, it is common practice to incorporate SFT loss in RLHF or DPO training (Ouyang et al., 2022; Pang et al., 2024) to mitigate this issue. Thus, we also use the pre-training loss including standard SFT loss and MSE value loss in preference optimization stage. Specifically, we fixed the weights for the margin loss and MSE loss at 0.25, the weight for the regularization term at 0.001, and the weight for the SFT loss at 5. In addition, we also employ the AdamW optimizer (Loshchilov and Hutter, 2019) and the cosine learning rate scheduler with a warmup rate of 0.03.

| Dataset | OOD? | # Training | # Test |
|---|---|---|---|
| GSM8K | In-Domain | 7473 | 1319 |
| MATH | In-Domain | 7500 | 5000 |
| GaoKao2023 | OOD | - | 385 |
| OCWCourses | OOD | - | 272 |

Table 6: Datasets Statistics

## A.2 Datasets Details

**Mathematical Reasoning Benchmarks** Table 6 provides a detailed overview of the mathematical reasoning benchmarks. The training and test sets are divided in accordance with previous studies (Cobbe et al., 2021; Hendrycks et al., 2021). GSM8K (Cobbe et al., 2021) is a dataset focused on multi-step mathematical reasoning, featuring high-quality, diverse grade school math word problems crafted by human authors. The MATH dataset (Hendrycks et al., 2021) contains complex competitive mathematics problems. GaoKao2023 (Liao et al., 2024) includes math problems from the 2023 Chinese National College Entrance Examination, the 2023 American Mathematics Competitions, and the 2023 American College Testing. OCWCourses (Lewkowycz et al., 2022) is a compilation of 272 STEM problems targeted at the undergraduate level, most of which require multi-step reasoning.

**Preference Test set in the Win Rate** In Section 4.4, we evaluate the accuracy of the policy model and the value model in assessing preferences. These models can accurately assess the preferences on the training set, as shown in Figure 2a. To accurately evaluate the generalization of the value model, we randomly sample 200 questions from the test sets of GSM8K and MATH respectively, and constructed total 10633 preference pairs using MCTS.

**Step-level Preference Pairs Construction in Monte Carlo tree** After step-level preferences annotation via MCTS, we need to filter the preference pairs from the Monte Carlo tree for training. Given step-level beam search, we need to consider the preference relationships among sibling nodes (at the same layer with same previous state), cousin nodes (at the same layer but with different previous states), and non-same-level terminal nodes (at different layer with terminal nodes).

Algorithm 1 outlines the process of our step-level preference pairs construction. First, we label each node along the path based on the correctness of the terminal node (Lines 2-4). Then, we iteratively construct step-level preference pairs in a top-down manner (Lines 5-23). In this process, we can specify the quantities of the three different types of preference relationships. In this study, we set the number of sibling nodes to 2, the number of cousin nodes and non-same-level terminal nodes to 1, respectively. This maintains an approximate ratio of 1:4 between positive and negative examples.

### A.3 Policy-value model Details

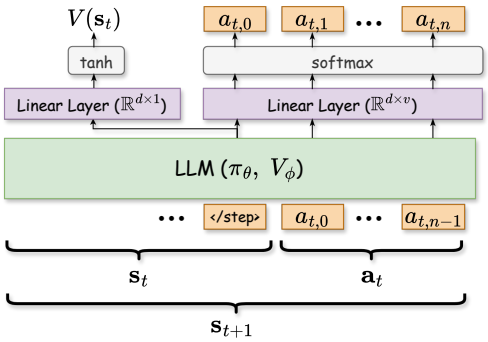



Figure 4: An overview of our policy-value model. $d$ represents the dimension of the hidden state in LLM, and $v$ represents the size of the vocabulary.

As shown in Figure 4, the value model $V_\phi$ and the LLM policy model $\pi_\theta$ are the same model but with different final layers. This design implies that these two models, $\pi_\theta$ and $V_\phi$, share the majority of their parameters. In practical implementation of the value loss, the value is only predicted on the last token of current reasoning step, representing the step-level preference.

### A.4 Experiment Environments

All experiments were conducted on Ubuntu 22.04 equipped with 8 * NVIDIA A100 GPUs. Our code mainly depends on Python 3.11 and PyTorch 2.2.1. We use our customized *Llama Factory* (Zheng et al., 2024) as the training framework and our customized *vLLM* (Kwon et al., 2023) as the inference framework[1]. We trained all models with *DeepSpeed ZeRO Stage2* (Rajbhandari et al., 2021) and *Flash-Attention 2* (Dao, 2023). The pre-trained LLMs are sourced from *HuggingFace*[2].

### A.5 Prompt Example of our XML format

To train the SFT model in executing mathematical reasoning, we utilize an XML format alongside zero-shot learning. This approach is adopted because the math-related pre-training corpora are predominantly harvested from the Internet, where HTML tags serve to distinguish various types of content, including text, equations, and code snippets. In this work, each solution consists of both text analysis and code snippet, as shown in Figure 5.

[1]We have released our customized framework in our Github Repository.

[2]https://huggingface.co

**Algorithm 1** Step-level Preference Pairs Construction

**Require:** Monte Carlo trees $\mathcal{T}$, prompted question $\mathbf{x}$.
**Ensure:** Step-level Preference Pairs $\mathcal{P}$.

$\mathcal{P} = [\,]$ ▷ *Initialization*
**for** terminal node $n$ in $\mathcal{T}$ **do**
  **if** $n$ has correct final answer **then**
    Backpropagation labels each node as *correct* along the path from root to $n$
$n \leftarrow$ root node in $\mathcal{T}$
**while** $n$ is non-terminal node **do** ▷ *In a top-down manner*
  $n_w \leftarrow \arg\max_{child \in n.children}\{Q(child)|child \text{ is correct node.}\}$ ▷ *Ensure $\mathbf{y}^w$ is a correct solution*
  $\mathbf{y}^w \leftarrow$ partial solution from root to $n_w$ in $\mathcal{T}$
  $n_s \leftarrow$ randomly select a non-correct node in $n.children$ ▷ *Pairs in sibling nodes*
  **if** $n_s \neq \emptyset$ **then**
    $\mathbf{y}^l \leftarrow$ partial solution from root to $n_s$ in $\mathcal{T}$
    Add $(\mathbf{x}, \mathbf{y}^w, \mathbf{y}^l)$ to $\mathcal{P}$
  $n_c \leftarrow$ randomly select a non-correct node at the same level of $n_w$ ▷ *Pairs in cousin nodes*
  **if** $n_c \neq \emptyset$ **then**
    $\mathbf{y}^l \leftarrow$ partial solution from root to $n_c$ in $\mathcal{T}$
    Add $(\mathbf{x}, \mathbf{y}^w, \mathbf{y}^l)$ to $\mathcal{P}$
  $n_t \leftarrow$ randomly select a non-correct terminal node at the different level of $n_w$ ▷ *Pairs in non-same-level terminal nodes*
  **if** $n_t \neq \emptyset$ **then**
    $\mathbf{y}^l \leftarrow$ partial solution of $n_t$ in $\mathcal{T}$
    Add $(\mathbf{x}, \mathbf{y}^w, \mathbf{y}^l)$ to $\mathcal{P}$
  **if** $n_s = n_c = n_t = \emptyset$ **then**
    (Optional) Find $\mathbf{y}^l$ in another tree $\mathcal{T}'$. ▷ *If no negative example found in all trees, all possible generated solutions are correct. No preference learning needed for this question.*
  $n \leftarrow n_w$ ▷ *Go to next layer*

An example of our SFT XML format:

```
<question>Haley grows at the rate of 3 inches every year. If she is currently 20 inches tall,
what will be her height after 10 years?</question>
<step>
<p>
To calculate Haley's height after 10 years, I need to add 10 times the growth rate of 3 inches to
her current height.
</p>
<code>
```python
current_height = 20
growth_rate = 3
years = 10
future_height = current_height + (growth_rate * years)
print(future_height)
```
</code>
<output>
50
</output>
</step>
<step>
<p>
I have calculated Haley's height after 10 years. Haley will be 50 inches tall
after 10 years.
</p>
<p>
Final Answer: $50$
</p>
</step>
```

Figure 5: An example of our SFT XML format. The text in black is prompt, and the text in red is model generation.