HS-STAR: Hierarchical Sampling for Self-Taught Reasoners via Difficulty Estimation and Budget Reallocation

Feng Xiong[1]*, Hongling Xu*, Yifei Wang[1,2], Runxi Cheng[3], Yong Wang[1]†, Xiangxiang Chu[1]
[1]AMAP, Alibaba Group [2]University of Chinese Academy of Sciences [3]Tsinghua University
jingxun.xf@icloud.com, wangyong.lz@alibaba-inc.com

Abstract

Self-taught reasoners (STaRs) enhance the mathematical reasoning abilities of large language models (LLMs) by leveraging self-generated responses for self-training. Recent studies have incorporated reward models to guide response selection or decoding, aiming to obtain higher-quality data. However, they typically allocate a uniform sampling budget across all problems, overlooking the varying utility of problems at different difficulty levels. In this work, we conduct an empirical study and find that problems near the boundary of the LLM's reasoning capability offer significantly greater learning utility than both easy and overly difficult ones. To identify and exploit such problems, we propose HS-STAR, a **H**ierarchical **S**ampling framework for **S**elf-**Ta**ught **R**easoners. Given a fixed sampling budget, HS-STAR first performs lightweight pre-sampling with a reward-guided difficulty estimation strategy to efficiently identify boundary-level problems. Subsequently, it dynamically reallocates the remaining budget toward these high-utility problems during a re-sampling phase, maximizing the generation of valuable training data. Extensive experiments across multiple reasoning benchmarks and backbone LLMs demonstrate that HS-STAR significantly outperforms other baselines without requiring additional sampling budget.

1 Introduction

Large language models (LLMs) can improve their capabilities by training on self-generated data, characterizing them as self-taught reasoners (STaRs) (Zelikman et al., 2022; Yuan et al., 2023; Hosseini et al., 2024). This paradigm is also referred to as reinforced self-training (Gulcehre et al., 2023) (ReST) or self-improvement (Huang et al., 2023). For mathematical reasoning (Yang et al., 2025; Tian et al., 2025a; Wang et al., 2025a), pioneer STaRs generally follow an **iterative process**: (1) generating candidate responses for a given math problem via temperature sampling; (2) selecting responses based on answer correctness; and (3) updating the model using either SFT or DPO (Singh et al., 2024; Pang et al., 2024; Wu et al., 2025).

Building on previous efforts, recent work has focused on enhancing STaRs by leveraging additional reward models, which can be categorized into two main directions. One line of work, known as *reward-guided selection*, introduces an auxiliary reward model to re-rank or filter responses based on their estimated quality, encouraging the model to exploit higher-quality trajectories (Yang et al., 2024; Zeng et al., 2025b; Tu et al., 2025). Another line of work, *reward-guided decoding*, leverages Monte Carlo Tree Search (MCTS), in which a process reward model (PRM) (Wang et al., 2024a) is trained and used to guide the decoding process, aiming to improve both final answer accuracy and the quality of intermediate reasoning steps (Zhang et al., 2024; Chen et al., 2024a).

However, these studies primarily focus on response quality through such reward-guided approaches, neglecting the utility of the problems themselves. Specifically, a uniform allocation of sampling budget across all problems fails to account for the varying difficulty levels of individual problems and their differential impacts on the learning process (Zhu et al., 2024). Since autoregressive decoding is the principal bottleneck in STaRs, such an indiscriminate allocation strategy is highly inefficient. This issue raises two critical questions: *(i) Profiling: Which difficulty level of problems are most beneficial for self-taught reasoners?* Intuitively, problems that are too simple provide limited learning value, while those that are overly challenging may either waste sampling resources by requiring numerous attempts to generate correct responses or be beyond the model's capabil-

* Equal contribution.
† Corresponding author and project lead.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 5539–5555
November 4-9, 2025

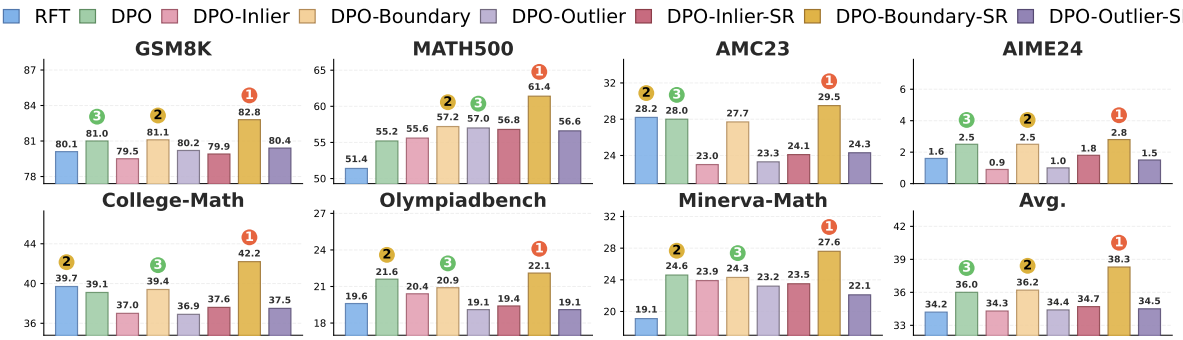


Figure 1: Pilot experiments on Qwen2.5-3B.

ities, hindering effective learning. *(ii) Allocation: How can sampling resources be allocated to maximize the utility of valuable problems?* Given the high expense of sampling, it is essential to identify and prioritize high-utility problems to optimize the trade-off between resource usage and performance improvement.

To address these questions, we first conduct a pilot study to analyze the utility of problems across varying difficulty levels (see Sec. 2 for details). We begin by defining model-specific problem difficulty based on the accuracy over multiple sampling attempts (Snell et al., 2025; Tong et al., 2024). As depicted in Fig. 1, we observe that training solely on either *Inlier* or *Outlier* problems leads to a significant decline in performance, whereas training exclusively on *Boundary* problems yields even better results than using the full set of problems. Furthermore, allocating additional sampling budget to these *Boundary* problems for self-training substantially improves model performance, which underscores their importance in guiding more effective learning in STaRs.

While the above findings highlight the high utility of *Boundary* problems, identifying them typically relies on statistical estimation with extensive sampling, limiting the practical applicability. To address this limitation and further tackle the second question, we propose the **H**ierarchical **S**ampling framework for **S**elf-**Ta**ught **R**easoners (HS-STAR). Given a fixed total sampling budget, HS-STAR begins with a *Difficulty Estimation* phase, where a small portion of the budget is used to estimate problem difficulty based on both answer accuracy and response quality, a process we refer to as reward-guided difficulty estimation. The remaining budget is then dynamically reallocated to problems estimated to be of high utility in a subsequent *Re-Sampling* phase, thereby maximizing the exploitation of valuable problems without incurring additional budget. Finally, the aggregated responses are used to construct a preference dataset for self-training in *Preference Optimization* phase, improving the overall effectiveness of STaRs.

Our contributions are summarized as follows:

- We conduct an empirical study that reveals the high utility of *Boundary* problems in self-taught reasoning. This motivates a problem-centric perspective for optimizing sampling resource allocation by identifying and prioritizing these problems.
- We propose HS-STAR, a hierarchical sampling framework that integrates reward-guided difficulty estimation to dynamically reallocate sampling budgets toward high-utility problems, significantly enhancing training effectiveness under a fixed sampling budget.
- Extensive experiments across seven reasoning benchmarks and various backbone LLMs demonstrate the superiority of our HS-STAR. Further analyses confirm the effectiveness of each component within the framework.

2 Pilot Experiments

To analyze the core challenges of *Profiling* and *Allocation*, we conduct a comprehensive empirical study on the utility of problems in STaRs.

2.1 Preliminary

We begin by formalizing the iterative self-training process of STaRs. At iteration t, we denote the policy model as $\mathcal{M}_t$ and the utilized dataset as $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i is a math problem and y_i is the corresponding answer. This process typically consists of three steps:

(1) Generation. For each problem $x \in \mathcal{D}_t$, the model $\mathcal{M}_t$ generates n responses by sampling,

forming the set $\mathcal{R}_{t,x} = \{r_j | r_j \sim \mathcal{M}_t(x)\}_{j=1}^{n}$.

(2) Selection. We apply a rule-based verifier $V(x, y, r) \in \{0, 1\}$ to assess response correctness, and omit the answer y from the notation hereafter for simplicity. For Rejection Sampling Fine-Tuning (RFT) (Yuan et al., 2023), we select only correct responses to form $\mathcal{D}_t^{\text{corr}}$. For DPO (Rafailov et al., 2023), we construct $\mathcal{D}_t^{\text{pairs}}$ by pairing correct and incorrect responses.

(3) Updating. For RFT, the model $\mathcal{M}$ is updated by minimizing the negative log-likelihood:

$$\mathcal{L}_{\text{RFT}} = -\log \mathcal{M}(r|x), \tag{1}$$

where $(x, r) \in \mathcal{D}_t^{\text{corr}}$. For DPO, the model $\mathcal{M}$ is updated by minimizing:

$$\mathcal{L}_{\text{DPO}} = -\log \sigma\left(\beta \left(\log \frac{\mathcal{M}(r_w|x)}{\mathcal{M}_{\text{t}}(r_w|x)} - \log \frac{\mathcal{M}(r_l|x)}{\mathcal{M}_{\text{t}}(r_l|x)}\right)\right), \tag{2}$$

where $(x, r_w, r_l) \in \mathcal{D}_t^{\text{pairs}}$, r_w is a correct response and r_l is an incorrect response for problem x.

Additionally, we introduce Statistical Difficulty Estimation (**SDE**) as an oracle for assessing problem difficulty. Following Snell et al. (2025), SDE computes accuracy using a substantial sampling budget (i.e., 100 samples per problem), providing a reliable proxy for the model-specific difficulty of each problem. Inspired by Chen et al. (2024b), we partition problem instances as *Inlier* (accuracy $>$ 87.5%), *Outlier* (accuracy $<$ 12.5%), or *Boundary* (otherwise), with their corresponding sets denoted as $\mathcal{D}_t^{\mathcal{I}}$, $\mathcal{D}_t^{\mathcal{O}}$, and $\mathcal{D}_t^{\mathcal{B}}$, respectively.

2.2 Analysis of Utility over Problem Difficulty

We conduct experiments of STaRs using our SDE-based partitioning on Qwen-2.5 3B (Qwen et al., 2025), as shown in Fig. 1. Training and dataset details are provided in Sec. 4.1.

Difficulty-Aware Training Analysis. Given the observed superiority of DPO over RFT, we adopt DPO as the default training objective throughout our study. We define three variants—`DPO-Inlier`, `DPO-Boundary`, and `DPO-Outlier`—each trained exclusively on one SDE-defined subset: $\mathcal{D}^{\mathcal{I}}$, $\mathcal{D}^{\mathcal{B}}$, and $\mathcal{D}^{\mathcal{O}}$, respectively, using a fixed sampling count of n as 8. As shown in Fig. 1, we find that both `DPO-Inlier` and `DPO-Outlier` yield significantly worse performance, while `DPO-Boundary` produces a slight improvement (+0.2%) than using all problems. These results highlight that boundary-level problems offer the highest utility for STaRs.

Sampling Budget Reallocation. We further examine whether allocating more sampling resources to different difficulty levels can enhance self-training effectiveness. We introduce three Sampling Reallocation (SR) variants: `DPO-Inlier-SR`, `DPO-Boundary-SR`, and `DPO-Outlier-SR`, where the total sampling budget ($8 \times |\mathcal{D}_t|$) is reallocated exclusively to one difficulty category. This concentrated sampling allows each selected problem to receive more candidate responses. Notably, `DPO-Boundary-SR` significantly achieves the best performance across all benchmarks, with an average score of 38.3%. These results reinforce that boundary-level problems are more sampling-efficient, indicating strategically prioritizing them is key to enhance self-training.

3 Methodology

In this section, we provide a detailed introduction to our HS-STAR, as shown in Fig. 2. Our approach is divided into three main phases: Difficulty Estimation, Re-Sampling, and Preference Optimization.

3.1 *Phase 1*: Difficulty Estimation

While SDE provides a reliable oracle for assessing problem difficulty, it requires extensive sampling and is computationally expensive. To enable practical difficulty estimation under limited resources, we propose a lightweight alternative, including pre-sampling and reward-guided estimation.

Pre-Sampling. We first perform a pre-sampling step, where a small portion of the sampling budget is used to generate responses for each question. Concretely, given a problem $x \in \mathcal{D}_t$, we derive n_p responses from the policy model $\mathcal{M}_t$, forming $\mathcal{R}_{t,x}^p = \{r_1, \ldots, r_{n_p} \mid r_i \sim \mathcal{M}_t(x)\}$. Here, n_p is set to a relatively small value, allowing more remaining budget to be reallocated toward high-utility problems in *Phase 2*.

Reward-Guided Estimation. Subsequently, we evaluate $\mathcal{R}_{t,x}^p$ using both the ground-truth answer and process reward model (PRM). Specifically, we propose a reward-guided difficulty estimation (RDE) strategy, which incorporates two complementary metrics: $\phi_{\text{a}}(\mathcal{R}_{t,x}^p)$ for assessing accuracy, and $\phi_{\text{r}}(\mathcal{R}_{t,x}^p)$ for evaluating the quality of the underlying reasoning process. $\phi_{\text{a}}(\mathcal{R}_{t,x}^p)$ is defined as the average accuracy over all generated responses: $\phi_{\text{a}}(\mathcal{R}_{t,x}^p) = \frac{1}{n_p} \sum_{i=1}^{n_p} V(x, r_i)$. The term $\phi_{\text{r}}(\mathcal{R}_{t,x}^p)$ assesses the quality of the reasoning process produced by the policy model, which

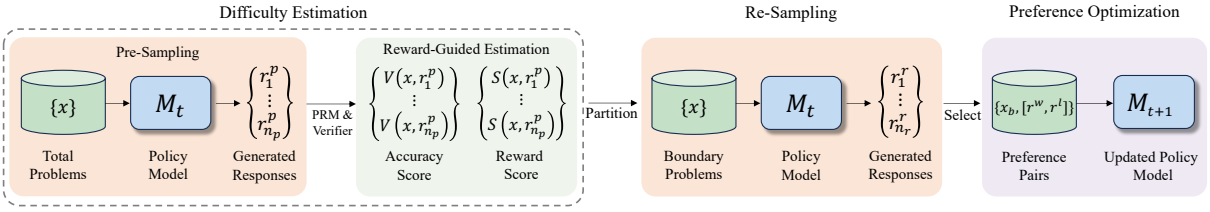


Figure 2: Illustration of the HS-STAR framework. Each iteration begins with a *Difficulty Estimation* phase, where a limited sampling budget is used to generate candidate responses for each query, referred to as pre-sampling. These responses are then evaluated using a reward-guided strategy to estimate problem difficulty. In the subsequent *Re-Sampling* phase, the remaining budget is allocated to high-utility boundary problems identified in the previous step. Finally, in the *Preference Optimization* phase, preference pairs are constructed from all collected responses and used to update the policy model.

is achieved by leveraging the scores provided by the PRM for each sampled response. We define an aggregate process quality score as the average of the reward values within all sampled responses: $\phi_{\mathrm{r}}(\mathcal{R}^p_{t,x}) = \frac{1}{n_p}\sum_{i=1}^{n_p} S(r_i)$, where $S(r_i)$ represents the reward score assigned to the i-th response r_i. Given that a response r_i consists of n_i reasoning steps, with step j assigned a reward score $s_{i,j}$, the overall process quality score for a complete response r_i is defined as the minimum reward score across all steps in the sequence (Tu et al., 2025): $S(r_i) = \min_{j\in\{1,2,\ldots,n_i\}}\{s_{i,j}\}$.

Based on these two critical dimensions of responses, we categorize the difficulty level for the given model $\mathcal{M}_t$ on a specific problem x into three distinct classes:

$$\Phi_{\mathcal{M}_t}(x)=\begin{cases} \textit{Inlier}, & \text{if } \phi_{\mathrm{a}}(\mathcal{R}^p_{t,x})=1 \wedge \phi_{\mathrm{r}}(\mathcal{R}^p_{t,x})>\tau_h \\ \textit{Outlier}, & \text{if } \phi_{\mathrm{a}}(\mathcal{R}^p_{t,x})=0 \wedge \phi_{\mathrm{r}}(\mathcal{R}^p_{t,x})<\tau_l \\ \textit{Boundary}, & \text{otherwise} \end{cases} \tag{3}$$

where τ_h and τ_l are predefined thresholds. This metric jointly captures the model's ability to solve a given problem by evaluating both the accuracy of the final answer and the soundness of the reasoning process, thereby providing an effective estimate of problem difficulty even with limited responses.

3.2 *Phase 2*: Re-Sampling

Building on the insights from Sec. 2, which highlight the critical role of boundary problems, we aim to maximize their exploitation through targeted resampling. Specifically, given an initial sampling budget of n_t per query, we subsequently assign an additional sampling count n_r to each boundary sample estimated by our RDE during the Re-Sampling phase, calculated as follows:

$$n_r = \left\lfloor \frac{(n_t - n_p)\times|\mathcal{D}_t|}{|\mathcal{D}_t^{\mathcal{B}}|} \right\rfloor, \tag{4}$$

where $\mathcal{D}_t^{\mathcal{B}}$ represents the subset of samples classified as *Boundary* in the Difficulty Estimation phase. This reallocation of the sampling budget enables us to focus computational resources on such instances, which offer greater potential for optimization. Subsequently, for each query $x_b \in \mathcal{D}_t^{\mathcal{B}}$, we utilize the policy model $\mathcal{M}_t$ to generate n_r candidate responses, forming $\mathcal{R}^r_{t,x_b} = \{r_1, r_2, \ldots, r_{n_r} \mid r_i \sim \mathcal{M}_t(x_b)\}$.

3.3 *Phase 3*: Preference Optimization

Based on the sampled responses from the aforementioned two phases, we construct a preference dataset to facilitate self-training via preference optimization. At iteration t, for each query x, the policy model $\mathcal{M}_t$ has generated a response set $\mathcal{R}_{t,x} = \mathcal{R}^p_{t,x} \cup \mathcal{R}^r_{t,x}$. To construct the preference dataset $\mathcal{D}_t^{\text{pairs}}$, these responses are systematically categorized based on their correctness. For each query x, the response set $\mathcal{R}_{t,x}$ is partitioned into two subsets: the set of correct responses $\mathcal{R}^{\text{corr}}_{t,x} = \{r \in \mathcal{R}_{t,x} \mid V(x,r) = 1\}$, and the set of incorrect responses $\mathcal{R}^{\text{incorr}}_{t,x} = \{r \in \mathcal{R}_{t,x} \mid V(x,r) = 0\}$.

Subsequently, the samples in sets $\mathcal{R}^{\text{corr}}_{t,x}$ and $\mathcal{R}^{\text{incorr}}_{t,x}$ are independently ranked in descending order according to their reward scores $S(r)$. This produces the ordered sequences $\widetilde{\mathcal{R}}^{\text{corr}}_t = (r^{\text{corr}}_{(1)}, r^{\text{corr}}_{(2)}, \ldots, r^{\text{corr}}_{(|\mathcal{R}^{\text{corr}}_t|)})$ and $\widetilde{\mathcal{R}}^{\text{incorr}}_t = (r^{\text{incorr}}_{(1)}, r^{\text{incorr}}_{(2)}, \ldots, r^{\text{incorr}}_{(|\mathcal{R}^{\text{incorr}}_t|)})$. The number of pairs k for $\mathcal{D}_t^{\text{pairs}}$ is defined as the minimum cardinality of these two sets: $k = \min(|\mathcal{R}^{\text{corr}}_t|, |\mathcal{R}^{\text{incorr}}_t|)$. Finally, the paired dataset is constructed as $\mathcal{D}_t^{\text{pairs}} = \{(s^{\text{corr}}_i, s^{\text{incorr}}_i) \mid i = 1, \ldots, k\}$, where each s^{corr}_i

and s_i^{incorr} is a unique sample from the top k elements of $\mathcal{R}_t^{\text{corr}}$ and $\mathcal{R}_t^{\text{incorr}}$, respectively.

By training on the given set of preference pairs, we derive the updated model $\mathcal{M}_{t+1}$, initialized from its predecessor $\mathcal{M}_t$. The optimization follows the DPO objective (Rafailov et al., 2023), as specified in Eq. 2.

4 Experiments

4.1 Setup

Dataset. Following Zhang et al. (2025a), we use NuminaMath-1.5 (Li et al., 2024a) for iterative self-taught reasoning. The original dataset contains approximately 900K math problems, and we apply a filtering pipeline to ensure the quality of questions and the verifiability of answers. In each iteration, we randomly sample 7,500 problems without replacement, ensuring no overlap across iterations. Additional details are provided in Appendix A.1.

Implementation Details. To facilitate the generation of stepwise solutions for reward labeling, we first perform a **warm-up training** using synthetic solutions. Specifically, we leverage the MATH dataset (Hendrycks et al., 2021) and prompt *gpt-4o-2024-08-06* to systematically rewrite each solution in a step-by-step format, then organize these steps separated by "\n\n". The resulting model, denoted as $\mathbf{M}_0$, serves as the initialization for iterative self-training. In our experiments, each iteration operates under a fixed sampling budget, corresponding to an average of 8 samples per problem. The pre-sampling count n_p is set to 3, and thresholds τ_h and τ_l for difficulty estimation are set to 0.15 and 0.65, respectively. We utilize Skywork-PRM-7B (He et al., 2024b) as our PRM and perform three iterations in total. See more details in Appendix A.2.

Baselines. To ensure a comprehensive evaluation, we apply HS-STAR across a diverse set of open-source models, including DeepSeek-Math-7B (Shao et al., 2024a), Phi-3.5-Mini-Instruct (Abdin et al., 2024), Qwen2.5-3B, and Qwen2.5-7B (Qwen et al., 2025). We compare with the following baselines: (1) **Vanilla SFT**, using reference solutions from NuminaMath for training without any self-generated data; (2) **Stepwise Initialization** ($\mathbf{M}_0$), the base model trained on synthetic step-by-step solutions without any self-training; (3) **STAR-RFT**, using SFT as the training objective in STaRs; and (4) **STAR-DPO**, using DPO as the training objective in STaRs.

Evaluation. We evaluate our framework on seven mathematical reasoning benchmarks, including GSM8K (Cobbe et al., 2021), MATH500 (Yang et al., 2024), OlympiadBench (He et al., 2024a), Minerva-Math (Lewkowycz et al., 2022), College-Math (Tang et al., 2024), as well as competition-level benchmarks such as AMC23 (AI-MO, 2024b) and AIME24 (AI-MO, 2024a). We report **Pass@1** accuracy for all benchmarks, with the exception of AMC23 and AIME24. For these two, we follow standard protocol and report **Avg@32**, which is calculated from 32 generated samples per problem, using temperature as 0.6.

4.2 Main Results

Table 1 presents a comparative study of training methods across multiple mathematical reasoning benchmarks and backbone LLMs. We can draw the following conclusions:

HS-STAR achieves superior performance. Across all model backbones and benchmarks, HS-STAR consistently outperforms baseline methods. For example, it improves the overall accuracy by 2.2% on DeepSeek-Math-7B, 1.4% on Qwen2.5-3B, and 1.8% on Qwen2.5-7B compared to their respective best-performing baselines. These results demonstrate the significance of identifying and exploiting high-utility problems. Furthermore, on challenging datasets such as AIME24 and AMC23, HS-STAR also outperforms the most competitive counterparts, demonstrating the robustness of our boundary-focused sampling strategy.

DPO consistently outperforms RFT. Across most settings, STAR-DPO achieves higher accuracy than STAR-RFT. For instance, on Qwen2.5-7B and Qwen2.5-3B, STAR-DPO yields relative gains of 2.7% and 1.8%, respectively. We assume that this stems from DPO's ability to leverage both correct and incorrect responses, whereas RFT relies solely on correct trajectories and may underutilize informative failure cases.

Iterative self-training brings improvements. We observe that all STaR-based methods consistently outperform their initializations and vanilla SFT baselines, validating the effectiveness of the training paradigm. Among the backbones, the relatively modest improvement observed on Phi-3.5-Mini-Instruct is likely due to the extensive post-training it has already undergone. Moreover, we find that stepwise initialization not only enables format-consistent reasoning but also outperforms vanilla SFT, demonstrating its effectiveness as a lightweight and generalizable warm-up strategy.

Table 1: Main results across mathematical reasoning benchmarks. All STAR-based methods are trained iteratively for three self-training rounds. **Bold** values indicate the best performance, while underlined ones denote the second-best results. For AMC23 and AIME24, we report Avg@32, and Pass@1 is used for others.

Method	GSM8K	MATH 500	Olympiad Bench	Minerva Math	AMC23	College Math	AIME24	Avg.
DeepSeek-Math-7B	30.3	18.6	5.3	5.9	7.3	17.2	0.0	12.1
Vanilla SFT	54.4	28.4	9.9	7.0	10.9	28.4	0.6	19.9
Stepwise Init.	62.9	32.8	10.7	8.1	11.3	25.4	0.4	21.7
+STAR-RFT	66.0	30.2	8.6	11.8	11.7	26.8	0.5	22.2
+STAR-DPO	63.1	33.8	9.9	10.7	12.6	26.7	0.7	22.5
+HS-STAR (Ours)	**67.7**	**35.4**	**12.0**	**13.6**	**13.4**	**29.8**	**1.1**	**24.7**
Phi-3.5-Mini-Instruct	83.5	46.2	13.2	16.2	16.2	36.1	0.8	30.3
Vanilla SFT	81.7	47.8	14.7	11.4	15.8	32.0	0.5	29.1
Stepwise Init.	85.4	45.2	13.5	24.3	16.2	35.9	1.2	31.7
+STAR-RFT	84.9	45.2	15.6	23.9	16.6	36.1	1.1	31.9
+STAR-DPO	**86.5**	46.4	14.8	**24.6**	16.2	36.2	0.8	32.2
+HS-STAR (Ours)	86.1	**49.2**	**15.7**	24.3	**17.4**	**36.5**	**1.9**	**33.0**
Qwen2.5-3B	72.9	49.4	16.3	17.3	21.1	33.8	2.6	30.5
Vanilla SFT	62.9	58.6	**23.6**	13.2	25.5	31.0	**3.6**	31.2
Stepwise Init.	72.8	50.2	19.6	16.9	20.8	35.4	2.8	31.2
+STAR-RFT	80.1	51.4	19.6	19.1	28.2	39.7	1.6	34.2
+STAR-DPO	81.0	55.2	21.6	**24.6**	28.0	39.1	2.5	36.0
+HS-STAR (Ours)	**82.6**	**60.0**	22.7	24.3	**28.4**	**40.7**	3.0	**37.4**
Qwen2.5-7B	81.8	54.2	25.6	25.4	26.4	39.3	3.7	36.6
Vanilla SFT	84.2	66.8	25.9	17.3	37.0	36.8	6.9	39.3
Stepwise Init.	86.4	65.0	27.9	25.0	35.0	41.7	5.2	40.9
+STAR-RFT	86.7	66.8	27.2	31.4	45.2	38.7	4.8	43.0
+STAR-DPO	88.6	69.8	33.3	29.8	44.3	45.7	8.3	45.7
+HS-STAR (Ours)	**90.3**	**72.8**	**35.9**	**31.6**	**46.5**	**46.4**	**8.9**	**47.5**

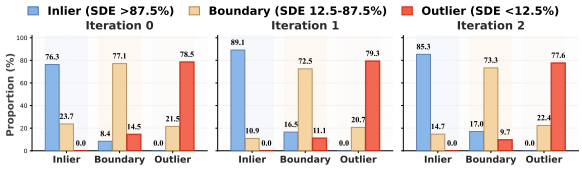


Figure 3: Estimation performance of our RDE on Qwen2.5-7B model. Sample categories identified by ours are presented along the horizontal axis, and for each category, the vertical dimension indicates the proportion of samples belonging to that category as estimated by SDE.

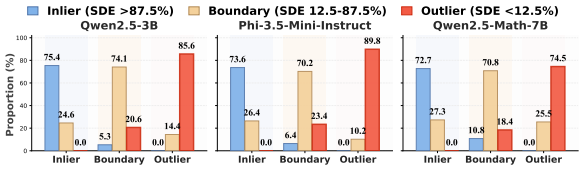


Figure 4: Estimation performance of our RDE on Qwen2.5-3B, Phi3.5-Mini-Instruct, and Qwen2.5-Math-3B. Notably, Qwen2.5-Math-7B is evaluated under the "Zero Training" setting.

4.3 "Zero Training" of HS-STAR

Settings. The recent emergence of DeepSeek-R1 (DeepSeek-AI et al., 2025) has sparked a trend of R1-Zero-like training (Chu et al., 2025; Yu et al., 2025), where reinforcement learning is applied directly to pre-trained models. Following this, we explore a similar "Zero Training" setup to further evaluate our approach. Specifically, we conduct HS-STAR on Qwen2.5-Math-7B (Yang et al., 2024) by skipping the initial warm-up SFT. We compare against various advanced LLM reasoning training methods over the same backbone, including Qwen2.5-Math-7B-Instruct (Yang et al., 2024), SimpleRL (Zeng et al., 2025a), PURE-VR (Cheng et al., 2025), DPO-VP (Tu et al., 2025), STAR-RFT (named Online RFT in Zeng et al. (2025b)), and STAR-DPO (referred to as Online DPO in Zhang et al. (2025a)). Detailed descriptions of these methods are provided in Appendix A.3.

Results. As illustrated in Table 2, we observe that HS-STAR also demonstrates strong performance in zero training setting, achieving a 6.4% improvement over the backbone model. Among self-training approaches, HS-STAR achieves the highest accuracy, surpassing the second-best method by 1.2%. Moreover, we find that HS-STAR can even achieve performance comparable to SimpleRL, which leverages GRPO (Shao et al., 2024b) for

Table 2: Comparison with other "Zero Training" models. All models are fine-tuned based on the Qwen2.5-Math-7B. We evaluate SimpleRL, PURE-VR, and DPO-VP using their publicly released checkpoints, while STAR-RFT and STAR-DPO are reproduced under the same experimental settings as ours.

Method	Training Strategy	MATH 500	Olympiad Bench	Minerva Math	AMC23	College Math	AIME24	Avg.
Qwen2.5-Math-7B	-	72.0	34.8	27.6	56.1	43.0	17.2	41.8
Qwen2.5-Math-7B-Instruct	-	**82.8**	40.3	35.7	59.5	46.9	11.4	46.1
SimpleRL	online RL	78.2	**42.5**	34.2	62.3	**49.1**	**23.9**	**48.4**
PURE-VR	online RL	79.0	40.6	**36.4**	63.1	47.3	15.6	47.0
DPO-VP	STaR	74.4	36.4	31.2	57.5	45.1	18.9	43.9
STAR-RFT	STaR	73.8	37.9	36.0	62.3	47.0	18.3	45.9
STAR-DPO	STaR	77.6	41.3	34.9	60.5	48.0	19.5	47.0
HS-STAR (Ours)	STaR	77.8	41.8	**36.4**	**64.3**	48.0	20.8	48.2

reinforcement learning. This suggests that the proposed framework can match the performance of online RL through a more flexible framework, while avoiding the complexity of hyperparameter tuning and the computational costs (Abdin et al., 2024; Tu et al., 2025; Liu et al., 2025; Fu et al., 2025; Wang et al., 2025b).

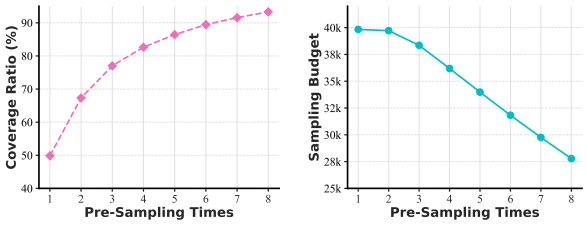
4.4 Analysis of Difficulty Estimation

Impact on Estimation Strategy of HS-STAR. Our HS-STAR can be seamlessly integrated with several alternative approaches to difficulty estimation. Specifically, we have developed three variants: HS-STAR-Acc, which solely utilizes accuracy-based estimation, HS-STAR-Reward, which solely utilizes reward-based estimation, and HS-STAR-SDE, which employs SDE (defined in Sec. 2.1) as an oracle measure of problem difficulty. Further details of these variants are provided in Appendix A.3. As summarized in Table 3, HS-STAR-SDE, which uses oracle difficulty and allocates more samples to boundary-level problems, leads to the best overall performance across all iterations, confirming the high utility of such problems. Among all variants that employ the same resource constraints, HS-STAR performs best, with accuracy only 0.4% lower than the HS-STAR-SDE oracle. In contrast, both ablation variants result in noticeable performance drops, yet still outperform the naive STAR-DPO without difficulty estimation and budget reallocation. These results suggest that RDE offers an effective solution for difficulty estimation by combining two complementary signals, without requiring extensive sampling.

Estimation Accuracy. To evaluate the performance of our difficulty estimation method, we employ the labels derived by the SDE as the ground truth. As illustrated in Fig. 3, our method achieve an estimation accuracy on the three types of samples, exceeding 70% across three iterations conducted on the Qwen2.5-7B model. Furthermore, as shown in Fig. 4, our method show considerable effectiveness across various models. Notably, it maintains high accuracy even when evaluated on the Qwen2.5-Math-7B model trained under "Zero Training" settings.

Table 3: Ablation study on difficulty estimation. We report the average performance across seven benchmarks.

Method	Iter. 1	Iter. 2	Iter. 3
HS-STAR-SDE (Oracle)	**45.7**	**47.0**	**47.9**
STAR-DPO	44.2	45.1	45.7
HS-STAR-Acc	44.7	46.2	46.8
HS-STAR-Reward	45.4	46.3	46.7
HS-STAR	45.6	46.6	47.5



(a) Boundary Samples Coverage in Re-Sampling Stage.

(b) Total Sampling Budget on Boundary Samples.

Figure 5: Analysis of the effects of Pre-Sampling times on Qwen2.5-7B. Subfig. 5a shows the trend of coverage of SDE estimated boundary samples as Pre-Sampling times vary. Subfig. 5b illustrates how the total sampling budget for SDE estimated boundary samples evolves as Pre-Sampling times vary.

4.5 Impact of Pre-Sampling Times

Since the number of pre-sampling directly influences both estimation accuracy and budget allocation, we examine its impact in detail. As shown in Fig. 5a, increasing the number of pre-sampling times improves the accuracy of difficulty estimation, leading to better coverage of Boundary sam-

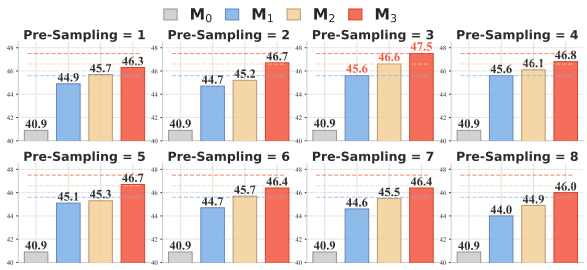


Figure 6: Performance under different Pre-Sampling Times on Qwen2.5-7B. For comparative analysis, we use the performance at a Pre-Sampling Time of 3 as the baseline, indicated by the dashed line.

Table 4: Impact of threshold ranges on precision, recall, and average performance. Our chosen setting strikes a balance, leading to the highest average performance.

Configuration	Precision	Recall	Avg.
Narrower Range	81.6	67.8	47.0
Broader Range	52.9	92.5	46.6
Standard	77.1	73.8	47.5

ples. However, this gain in estimation accuracy introduces a trade-off. As depicted in Fig. 5b, under a fixed total sampling budget, allocating more resources to pre-sampling reduces the budget available for exploiting high-utility Boundary samples, weakening overall reallocation effectiveness. This tradeoff is further confirmed in Fig. 6, which shows that performance peaks when pre-sampling is performed three times, balancing estimation accuracy and budget efficiency. Beyond this point, further increasing pre-sampling leads to performance degradation due to insufficient sampling of critical instances.

4.6 Sensitive Analysis of Thresholds

In our reward-guided difficulty estimation approach, the hyperparameters τ_l and τ_h are pivotal for defining and selecting boundary cases. These thresholds are empirically determined through pilot experiments, as detailed in Sec. 2. Importantly, this calibration constitutes a one-time overhead for a given process reward model. The resulting thresholds generalize robustly across all policy models in our main experiments, thereby obviating the need for model-specific fine-tuning.

Our selection process was primarily guided by the trade-off between precision and recall in identifying these boundary samples. Using the boundary cases identified by the SDE method as the ground truth, we define Precision as the fraction of samples classified as boundary cases by our method that are also identified as such by the SDE method, and Recall as the fraction of all boundary cases identified by the SDE method that our method successfully detects. These two metrics dictate the balance between the sampling budget allocated to boundary cases and the diversity of these cases captured during the Re-Sampling phase.

To further validate our choice and elucidate the impact of these thresholds, we conducted an ablation study that considered a narrower range by setting $\tau_l = 0.4$ and $\tau_h = 0.6$, as well as a broader range with $\tau_l = 0.2$ and $\tau_h = 0.8$. As the results in Table 4 indicate, a clear trade-off emerges. A narrower range yields higher precision but at the cost of significantly lower recall. Conversely, a broader range substantially increases recall, but this is achieved at the expense of a sharp decline in precision. These experimental results confirm that our chosen thresholds achieve an effective balance between precision and recall, and that this equilibrium is conducive to better overall performance.

Table 5: Ablation study of Re-Sampling strategies on Qwen2.5-7B. For these variants, the estimation of Inlier, Boundary, and Outlier samples is performed using RDE.

Re-Sampling Strategy	Iter. 1	Iter. 2	Iter. 3
w/o Re-Sampling	44.2	45.1	45.7
Inlier	42.0	43.0	44.2
Outlier	41.7	41.9	42.2
Inlier+Outlier	41.9	41.5	42.8
Inlier+Boundary	45.3	46.4	47.2
Boundary+Outlier	44.2	44.7	46.0
Boundary (Ours)	**45.6**	**46.6**	**47.5**

4.7 Comparison of Re-Sampling Strategies

In HS-STAR, the re-sampling budget is allocated based on difficulty levels estimated by our RDE strategy, with a focus on boundary-level problems. To better assess the utility of different difficulty levels, we compare re-sampling strategies constructed from all possible combinations of *Inlier*, *Outlier*, and *Boundary* samples, excluding the boundary-only configuration used in our main approach. The results are presented in Table 5, where re-sampling exclusively on *Boundary* samples consistently yields the best performance across all iterations, confirming the effectiveness of prioritizing such problems to maximize training utility. Strategies involving *Inlier+Boundary* also perform competitively, likely due to the predominance of boundary samples in the combined set. In contrast,

strategies based on *Inlier*, *Outlier*, or their combinations result in significantly lower performance. These findings highlight the importance of focusing on boundary-level queries during the re-sampling stage for effective self-improvement.

5 Related Work

5.1 Self-Taught Reasoners

Recent studies have shown that LLMs can progressively improve themselves by training on self-generated responses using SFT or DPO (Zelikman et al., 2022; Gulcehre et al., 2023; Huang et al., 2023; Yuan et al., 2024; Li et al., 2025; Wang et al., 2025c). In mathematical reasoning tasks, response selection is typically guided by answer correctness, enabling LLMs to act as self-taught reasoners without relying on human-annotated reasoning trajectories (Yuan et al., 2023; Singh et al., 2024; Hosseini et al., 2024; Pang et al., 2024; Wu et al., 2025; Zhang et al., 2025a).

Previous research has primarily explored two further directions. One line of work incorporates auxiliary reward model signals beyond answer correctness (Yang et al., 2024; Zeng et al., 2025b; Tu et al., 2025). Another focuses on enhancing the quality or accuracy of sampled responses, including designing MCTS strategy (Zhang et al., 2024; Tian et al., 2024; Chen et al., 2024a; Wang et al., 2024b) and integrating teacher guidance (Ding et al., 2025). However, these methods mainly aim to improve response quality, without accounting for the varying utility of problems. In contrast, our study reveals that boundary-level problems play a pivotal role in self-taught reasoning and introduces a hierarchical sampling strategy to efficiently exploit their utility.

5.2 Difficulty-Aware LLM Training

Difficulty-aware strategies have proven effective for improving the training of LLMs. For instance, in instruction tuning, prior work commonly adopts instruction-following difficulty (Li et al., 2024d,c) or uncertainty-based techniques (Liu et al., 2024; Zhang et al., 2025b) to select high-utility data. In mathematical reasoning, problem difficulty is typically estimated by pass rate. On this basis, DART-MATH (Tong et al., 2024) allocates more sampling budget to synthesize hard examples, while some recent studies advocate avoiding overly difficult questions (Tian et al., 2025b; Bae et al., 2025; Yu et al., 2025). Within STaR, a few studies have explored difficulty-aware sampling by allocating more resources to challenging problems (Ding et al., 2025; Xue et al., 2025). However, our analysis demonstrates that such difficult questions contribute significantly less compared to those near the model's capability boundary. Therefore, we propose HS-STAR that efficiently identifies and prioritizes boundary-level problems during self-training.

6 Conclusion

In this paper, we empirically demonstrate that the utility of self-training data is largely determined by the difficulty level of problems, with problems near the model's capability boundary being substantially more valuable than overly simple or excessively hard ones. Motivated by these findings, we propose HS-STAR, a hierarchical sampling framework that improves self-taught reasoning by explicitly estimating and exploiting problem utility. Concretely, HS-STAR first performs lightweight reward-guided difficulty estimation, then reallocates the sampling budget to prioritize high-utility boundary-level problems for preference optimization, thereby maximizing training effectiveness under a fixed sampling resource constraint. Experimental results confirm that our method significantly outperforms various baselines. We believe this work provides valuable insights for difficulty-aware optimization in LLM post-training.

Limitations

Despite the proposed framework HS-STAR effectively enhances the self-training for mathematical reasoning, it has two primary limitations.

- HS-STAR relies on difficulty estimation techniques such as reward-guided estimation to identify high-utility problems. Therefore, our framework is inherently tied to mathematical tasks, where problem difficulty is relatively well-defined. This limits the generalizability of HS-STAR to other domains where difficulty estimation is more ambiguous.
- Recent advances in rule-based RL have shown promising improvements in LLM reasoning. Although HS-STAR is developed for offline reinforced self-training, we believe that dynamically identifying high-utility problems during rollout could further improve the effectiveness of online RL, leaving this to our future work.

We believe that addressing these limitations could broaden the applicability of HS-STAR.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, and 110 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

AI-MO. 2024a. Aimo validation aime dataset. https://huggingface.co/datasets/AI-MO/aimo-validation-aime.

AI-MO. 2024b. Aimo validation amc dataset. https://huggingface.co/datasets/AI-MO/aimo-validation-amc.

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, and Donghyun Kwak. 2025. Online difficulty filtering for reasoning oriented reinforcement learning. *Preprint*, arXiv:2504.03380.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024a. Alphamath almost zero: Process supervision without process. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Qiguang Chen, Libo Qin, Jiaqi WANG, Jingxuan Zhou, and Wanxiang Che. 2024b. Unlocking the capabilities of thought: A reasoning boundary framework to quantify and optimize chain-of-thought. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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

Xiangxiang Chu, Hailang Huang, Xiao Zhang, Fei Wei, and Yong Wang. 2025. Gpg: A simple and strong reinforcement learning baseline for model reasoning. *arXiv preprint arXiv:2504.02546*.

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.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Yiwen Ding, Zhiheng Xi, Wei He, Lizhuoyuan Lizhuoyuan, Yitao Zhai, Shi Xiaowei, Xunliang Cai, Tao Gui, Qi Zhang, and Xuanjing Huang. 2025. Mitigating tail narrowing in LLM self-improvement via socratic-guided sampling. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 10627–10646, Albuquerque, New Mexico. Association for Computational Linguistics.

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

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. 2023. Reinforced self-training (rest) for language modeling. *Preprint*, arXiv:2308.08998.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024a. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, Bangkok, Thailand. Association for Computational Linguistics.

Jujie He, Tianwen Wei, Rui Yan, Jiacai Liu, Chaojie Wang, Yimeng Gan, Shiwen Tu, Chris Yuhao Liu, Liang Zeng, Xiaokun Wang, Boyang Wang, Yongcong Li, Fuxiang Zhang, Jiacheng Xu, Bo An, Yang Liu, and Yahui Zhou. 2024b. Skywork-o1 open series. https://huggingface.co/Skywork.

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 *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-STar: Training verifiers for self-taught reasoners. In *First Conference on Language Modeling*.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1051–1068, Singapore. Association for Computational Linguistics.

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

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay 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 *Advances in Neural Information Processing Systems*, volume 35, pages 3843–3857. Curran Associates, Inc.

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

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

Ming Li, Lichang Chen, Jiuhai Chen, Shwai He, Jiuxiang Gu, and Tianyi Zhou. 2024c. Selective reflection-tuning: Student-selected data recycling for LLM instruction-tuning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 16189–16211, Bangkok, Thailand. Association for Computational Linguistics.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024d. From quantity to quality: Boosting LLM performance with self-guided data selection for instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhiwei Li, Bao-Long Bi, Ling-Rui Mei, Junfeng Fang, Zhijiang Guo, Le Song, and Cheng-Lin Liu. 2025. From system 1 to system 2: A survey of reasoning large language models. *Preprint*, arXiv:2502.17419.

Liangxin Liu, Xuebo Liu, Derek F. Wong, Dongfang Li, Ziyi Wang, Baotian Hu, and Min Zhang. 2024. SelectIT: Selective instruction tuning for LLMs via uncertainty-aware self-reflection. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. *Preprint*, arXiv:2503.20783.

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

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

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 *Thirty-seventh Conference on Neural Information Processing Systems*.

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

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

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron T Parisi, Abhishek Kumar, Alexander A Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Fathy Elsayed, Hanie Sedghi, and 21 others. 2024. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research*. Expert Certification.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*.

Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. 2024. Mathscale: scaling instruction tuning for mathematical reasoning. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Shi-Yu Tian, Zhi Zhou, Kun-Yang Yu, Ming Yang, Lin-Han Jia, Lan-Zhe Guo, and Yu-Feng Li. 2025a. Vc search: Bridging the gap between well-defined and ill-defined problems in mathematical reasoning. *Preprint*, arXiv:2406.05055.

Xiaoyu Tian, Sitong Zhao, Haotian Wang, Shuaiting Chen, Yiping Peng, Yunjie Ji, Han Zhao, and Xiangang Li. 2025b. Deepdistill: Enhancing llm reasoning capabilities via large-scale difficulty-graded data training. *Preprint*, arXiv:2504.17565.

Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Lei Han, Haitao Mi, and Dong Yu. 2024. Toward self-improvement of llms via imagination, searching, and criticizing. In *Advances in Neural Information Processing Systems*, volume 37, pages 52723–52748. Curran Associates, Inc.

Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. 2024. Dart-math: Difficulty-aware rejection tuning for mathematical problem-solving. In *Advances in Neural Information Processing Systems*, volume 37, pages 7821–7846. Curran Associates, Inc.

Songjun Tu, Jiahao Lin, Xiangyu Tian, Qichao Zhang, Linjing Li, Yuqian Fu, Nan Xu, Wei He, Xiangyuan Lan, Dongmei Jiang, and Dongbin Zhao. 2025. Enhancing llm reasoning with iterative dpo: A comprehensive empirical investigation. *Preprint*, arXiv:2503.12854.

Haozhe Wang, Long Li, Chao Qu, Fengming Zhu, Weidi Xu, Wei Chu, and Fangzhen Lin. 2025a. To code or not to code? adaptive tool integration for math language models via expectation-maximization. *arXiv preprint arXiv:2502.00691*.

Haozhe Wang, Qixin Xu, Che Liu, Junhong Wu, Fangzhen Lin, and Wenhu Chen. 2025b. Emergent hierarchical reasoning in llms through reinforcement learning. *arXiv preprint arXiv:2509.03646*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024a. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

Xiyao Wang, Linfeng Song, Ye Tian, Dian Yu, Baolin Peng, Haitao Mi, Furong Huang, and Dong Yu. 2024b. Towards self-improvement of llms via mcts: Leveraging stepwise knowledge with curriculum preference learning. *Preprint*, arXiv:2410.06508.

Yifei Wang, Feng Xiong, Yong Wang, Linjing Li, Xiangxiang Chu, and Daniel Dajun Zeng. 2025c. Position bias mitigates position bias:mitigate position bias through inter-position knowledge distillation. *Preprint*, arXiv:2508.15709.

Ting Wu, Xuefeng Li, and Pengfei Liu. 2025. Progress or regress? self-improvement reversal in post-training. In *The Thirteenth International Conference on Learning Representations*.

Boyang Xue, Qi Zhu, Hongru Wang, Rui Wang, Sheng Wang, Hongling Xu, Fei Mi, Yasheng Wang, Lifeng Shang, Qun Liu, and Kam-Fai Wong. 2025. Dast: Difficulty-aware self-training on large language models. *Preprint*, arXiv:2503.09029.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *Preprint*, arXiv:2409.12122.

Minglai Yang, Ethan Huang, Liang Zhang, Mihai Surdeanu, William Wang, and Liangming Pan. 2025. How is llm reasoning distracted by irrelevant context? an analysis using a controlled benchmark. *Preprint*, arXiv:2505.18761.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *Preprint*, arXiv:2308.01825.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. STar: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*.

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

Weihao Zeng, Yuzhen Huang, Lulu Zhao, Yijun Wang, Zifei Shan, and Junxian He. 2025b. B-STar: Monitoring and balancing exploration and exploitation in self-taught reasoners. In *The Thirteenth International Conference on Learning Representations*.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. ReST-MCTS*: LLM self-training via process reward guided tree search. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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

Jia Zhang, Chen-Xi Zhang, Yao Liu, Yi-Xuan Jin, Xiao-Wen Yang, Bo Zheng, Yi Liu, and Lan-Zhe Guo. 2025b. D3: Diversity, difficulty, and dependability-aware data selection for sample-efficient llm instruction tuning. *Preprint*, arXiv:2503.11441.

Weiyao Zhu, Ou Wu, Fengguang Su, and Yingjun Deng. 2024. Exploring the learning difficulty of data: Theory and measure. *ACM Trans. Knowl. Discov. Data*, 18(4).

A Experimental Settings

This section offers comprehensive descriptions of the datasets, baselines, and implementations.

A.1 Dataset Details

NuminaMath-1.5 (Li et al., 2024a) is the second iteration of the widely used NuminaMath (Li et al., 2024b) dataset. This dataset provides a substantial collection of high-quality data suitable for post-training applications, comprising approximately 900,000 competition-level mathematics problems. Each problem is accompanied by a detailed solution presented in a Chain of Thought (CoT) format, which delineates the step-by-step reasoning process. The dataset encompasses a broad spectrum of mathematical content, drawing from diverse sources such as Chinese high school mathematics exercises and problems featured in prominent US and international mathematics olympiad competitions. The data collection primarily involved extracting content from online examination paper PDFs and mathematics discussion forums, thereby ensuring both the diversity and rigor of the included mathematical material.

A.2 Implementation Details

We conducted both data sampling and model evaluation using the vLLM framework (Kwon et al., 2023). During sampling, we set the temperature to 0.7. All models underwent full-parameter fine-tuning. Specifically, we used a learning rate of 5×10^{-7} for Qwen2.5-7B, while we trained Qwen2.5-Math-7B, Qwen2.5-3B, DeepSeek-Math-7B, and Phi-3.5-Instruct with a learning rate of 1×10^{-6}. Common hyperparameters included a maximum sequence length of 2048, a coefficient β of 0.1, and a batch size of 256. For models incorporating a warmup phase, τ_l and τ_h were set to 0.15 and 0.65, respectively. In the "Zero-Training" scenario, τ_l and τ_h were assigned values of 0.15 and 0.4, respectively.

A.3 Baseline Details

A.3.1 Baselines on Qwen2.5-Math-7B

Qwen2.5-Math-7B-Instruct. (Qwen et al., 2025) An instruct model in the Qwen2.5 series with strong mathematical reasoning capabilities.

SimpleRL. (Zeng et al., 2025a) A reinforcement learning framework that enables zero-RL training from a base model, utilizing simple rule-based rewards to improve reasoning accuracy.

Algorithm 1: HS-STAR

Input: Iterations T, initial policy model $\mathcal{M}_0$, dataset $\{\mathcal{D}_t\}_{t=1}^{T}$, sampling budgets n_p, n_t, functions $V(x,r)$, $S(x,r)$, thresholds τ_h, τ_l.
Output: Optimized model $\mathcal{M}_T$.

for $t = 0$ **to** $T-1$ **do**
  Initialize $\mathcal{D}_t^{\text{pairs}} \leftarrow \emptyset$, $\mathcal{D}_t^{\text{B}} \leftarrow \emptyset$
  foreach $x \in \mathcal{D}_t$ **do**
    $\mathcal{R}_{t,x}^{p} \leftarrow \{r_i \sim \mathcal{M}_t(x)\}_{i=1}^{n_p}$
    Calculate $\phi_{\text{a}}(\mathcal{R}_{t,x}^{p}) = \frac{1}{n_p}\sum_i V(x,r_i)$ and $\phi_{\text{r}}(\mathcal{R}_{t,x}^{p}) = \frac{1}{n_p}\sum_i S(x,r_i)$
    if x *is classified as Boundary* **then**
      Add x to $\mathcal{D}^{\text{B}}$
    end
  end
  if $|\mathcal{D}^{\text{B}}| > 0$ **then**
    $n_r \leftarrow \left\lfloor \frac{(n_t - n_p)\times|\mathcal{D}|}{|\mathcal{D}^{\text{B}}|} \right\rfloor$
    foreach $x \in \mathcal{D}^{\text{B}}$ **do**
      Generate responses: $\mathcal{R}_{t,x}^{r} \leftarrow \{r_i \sim M_t(x)\}_{i=1}^{n_r}$
    end
  end
  foreach $x \in \mathcal{D}^{\text{B}}$ **do**
    $\mathcal{R}_{t,x} \leftarrow \mathcal{R}_{t,x}^{p} \cup \mathcal{R}_{t,x}^{r}$
    Partition $\mathcal{R}_{t,x}$ into $\mathcal{R}_{t,x}^{\text{corr}}$ and $\mathcal{R}_{t,x}^{\text{incorr}}$
    Sort $\mathcal{R}_{t,x}^{\text{corr}}$ and $\mathcal{R}_{t,x}^{\text{incorr}}$ in descending order
    $k \leftarrow \min(|\mathcal{R}_{t,x}^{\text{corr}}|, |\mathcal{R}_{t,x}^{\text{incorr}}|)$
    Sample k pairs from top-k responses: $\mathcal{D}_t^{\text{pairs}} \leftarrow \mathcal{D}_t^{\text{pairs}} \cup \{(r_{(i)}^{\text{corr}}, r_{(i)}^{\text{incorr}})\}_{i=1}^{k}$
  end
  Update $\mathcal{M}_{t+1}$ using DPO loss on $\mathcal{D}_t^{\text{pairs}}$
end
return $\mathcal{M}_T$

PURE-VR. (Cheng et al., 2025) PURE is a reinforcement learning approach for LLM fine-tuning that replaces the standard sum-form credit assignment with a novel min-form, where the value function is defined as the minimum of future rewards.

DPO-VP. (Tu et al., 2025) DPO-VP enhances LLM reasoning via iterative preference learning with DPO. It iteratively refines the generator and reward model using simple verifiable rewards, achieving efficient performance comparable to RL.

STAR-RFT. STAR-RFT is an iterative self-training method. At each iteration, it filters and selects correct answers based on their correctness to use for further training, thereby effectively achieving self-improvement.

STAR-DPO. STAR-DPO is an iterative self-training method based on DPO. In each iteration, it partitions the generated responses based on their correctness and sorts the samples by reward to construct a preference dataset for DPO optimization.

Table 6: Prompt Templates for Stepwise Solutions Construction.

Category	Prompt Template
Reformat	Please reformat the provided solution for the given problem by dividing it into multiple detailed steps. These steps must explicitly present the final answer within `\boxed{}`. For each step, enrich the content with the minimal necessary details to enhance clarity. Ensure that any added information is precise and unambiguous to avoid potential misunderstandings. Return the response in explicit JSON format as follows: ["[STEP 1 CONTENT]", "[STEP 2 CONTENT]", //Continue for each step...]
Post-process	Please check and fix any LaTeX formatting errors in the following mathematical solution step. Return only the corrected step with proper LaTeX syntax.

Table 7: Detailed Results on HS-STAR Invariants.

Estimation Strategy	Iteration	GSM8K	MATH 500	Olympiad Bench	Minerva Math	AMC23	College Math	AIME24	Avg.
HS-STAR-SDE (Oracle)	1	88.8	71.8	34.4	28.3	44.8	45.2	7.3	45.8
	2	90.2	71.6	37.2	30.9	44.2	47.2	8.3	47.1
	3	90.6	73.8	34.5	32.7	46.2	46.7	10.9	47.9
STAR-DPO	1	87.4	69.8	30.8	25.4	43.7	45.4	6.7	44.2
	2	87.3	68.4	32.4	29.8	44.1	45.3	8.3	45.1
	3	88.6	69.8	33.3	29.8	44.3	45.7	8.3	45.7
HS-STAR-Acc	1	87.2	70.4	31.7	27.9	42.7	46.0	7.0	44.7
	2	88.6	71.8	32.3	32.0	45.1	45.7	8.2	46.2
	3	89.8	71.0	35.4	29.8	46.9	46.5	8.3	46.8
HS-STAR-Reward	1	87.6	70.4	33.5	28.3	45.1	45.3	7.6	45.4
	2	89.0	70.2	33.6	31.2	45.2	46.0	8.9	46.3
	3	89.3	73.8	35.4	28.3	44.5	46.8	9.0	46.7
HS-STAR (Ours)	1	88.0	69.8	33.3	29.8	45.2	46.0	7.3	45.6
	2	89.5	71.8	34.2	31.2	45.7	46.0	7.8	46.6
	3	90.3	72.8	35.9	31.6	46.5	46.4	8.9	47.5

A.3.2 HS-STAR Variants

STAR-DPO. This baseline configuration employs standard sampling techniques followed by iterative preference optimization.

HS-STAR-Acc. A variant of HS-STAR. In the Difficulty Estimation phase, the estimation is solely based on the accuracy of responses sampled during Pre-Sampling. Subsequently, Re-Sampling is performed on the identified boundary examples. Finally, the collected data from both phases is utilized for preference optimization.

HS-STAR-Reward A variant of HS-STAR. In the Difficulty Estimation phase, the estimation is solely based on the reward of responses sampled during Pre-Sampling. Subsequently, Re-Sampling is performed on the identified boundary examples. Finally, the collected data from both phases is utilized for preference optimization.

HS-STAR-SDE A variant of HS-STAR. In the Difficulty Estimation phase, the estimation is based on SDE method. Subsequently, Re-Sampling with the full sampling budget is conducted on the identified boundary examples. Finally, the collected data from both phases is utilized for preference optimization.

A.3.3 Re-Sampling Strategies

w/o Re-Sampling. This configuration serves as a standard baseline, employing conventional sampling techniques followed by iterative preference optimization without difficulty-based re-sampling.

Re-Sampling on Inlier. Following prior difficulty estimation in the pre-sampling phase, remaining sampling efforts are exclusively focused on Inlier samples for subsequent iterative preference optimization.

Re-Sampling on Outlier. Following prior diffi-

Table 8: Detailed Ablation Study on Re-sampling Strategies.

Re-Sampling Strategy	Iteration	GSM8K	MATH 500	Olympiad Bench	Minerva Math	AMC23	College Math	AIME24	Avg.
w/o Re-Sampling	1	87.4	69.8	30.8	25.4	43.7	45.4	6.7	44.2
	2	87.3	68.4	32.4	29.8	44.1	45.3	8.3	45.1
	3	88.6	69.8	33.3	29.8	44.3	45.7	8.3	45.7
Inlier	1	86.7	65.8	30.5	23.2	39.0	43.4	5.7	42.0
	2	87.3	67.4	30.8	24.6	40.3	44.4	6.1	43.0
	3	87.2	70.0	32.1	27.6	41.2	45.2	6.2	44.2
Outlier	1	85.7	66.2	29.8	25.4	36.1	42.7	6.2	41.7
	2	86.4	67.4	29.6	25.7	37.3	42.2	5.0	41.9
	3	86.1	67.2	30.4	25.0	37.7	43.2	5.7	42.2
Inlier + Boundary	1	87.7	69.8	33.8	27.6	44.9	45.3	8.1	45.3
	2	89.2	71.6	35.1	29.4	44.9	45.7	8.8	46.4
	3	89.5	72.0	36.1	33.1	45.0	46.5	8.5	47.2
Boundary + Outlier	1	87.1	70.6	32.0	24.6	42.5	45.2	7.2	44.2
	2	87.1	69.6	32.1	27.6	43.2	45.5	8.1	44.7
	3	88.4	71.2	33.6	30.5	44.6	45.7	8.2	46.0
Inlier + Outlier	1	86.8	65.8	30.2	24.6	38.5	42.8	4.9	41.9
	2	86.0	66.8	28.9	23.5	37.3	43.3	4.8	41.5
	3	86.1	67.8	30.4	25.4	40.4	43.7	6.1	42.8
Boundary	1	88.0	69.8	33.3	29.8	45.2	46.0	7.3	45.6
	2	89.5	71.8	34.2	31.2	45.7	46.0	7.8	46.6
	3	90.3	72.8	35.9	31.6	46.5	46.4	8.9	47.5

culty estimation in the pre-sampling phase, remaining sampling efforts are exclusively focused on Outlier samples for subsequent iterative preference optimization.

Re-Sampling on Inlier + Outlier. Following prior difficulty estimation in the pre-sampling phase, remaining sampling efforts are allocated to both Inlier and Outlier samples for subsequent iterative preference optimization.

Re-Sampling on Inlier + Boundary. Following prior difficulty estimation in the pre-sampling phase, remaining sampling efforts are allocated to both Inlier and Boundary samples for subsequent iterative preference optimization.

Re-Sampling on Outlier + Boundary. Following prior difficulty estimation in the pre-sampling phase, remaining sampling efforts are allocated to both Outlier and Boundary samples for subsequent iterative preference optimization.

Re-Sampling on Boundary (Ours). Following prior difficulty estimation in the pre-sampling phase, remaining sampling efforts are exclusively focused on Boundary samples for subsequent iterative preference optimization.

B Algorithm

The overall procedure of our algorithm is illustrated in Algorithm 1.

C Prompt Template

To construct the stepwise warmup dataset, we leveraged the MATH dataset (Hendrycks et al., 2021) and prompted GPT-4o-2024-08-06 to systematically rewrite each solution in a JSON format. Subsequently, these rewritten solutions were separated by the delimiter "\n\n". The prompt template used for this process is presented in Table 6. We initially employed a Reformat prompt to guide the model in restructuring the solutions in json format. In cases where the Reformat attempt failed, a Post-process prompt was utilized to further refine or reshape the output. Finally, the resulting data was filtered based on the provided answer.

D Additional Experimental Results

D.1 Iterative Results on Qwen2.5-7B

As illustrated in Fig. 7, HS-STAR consistently outperformed all baseline methods across all evaluated benchmarks. As the number of iterations increased, the performance of all methods gradually

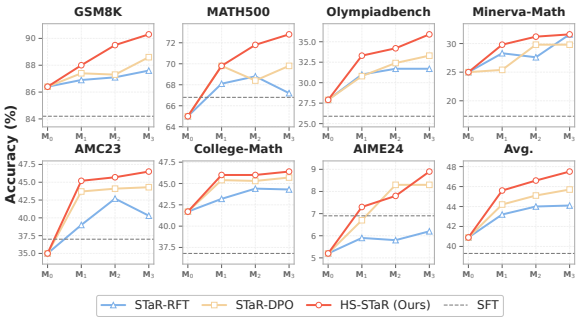


Figure 7: Comparison of the performance improvements of Qwen2.5-7B across three training iterations.

improved; notably, HS-STAR attained the highest M_3 accuracy on every benchmark. Moreover, the overall average accuracy highlights that HS-STAR delivers the most substantial improvement compared to other approaches.

D.2 Results on HS-STAR Invariants

As shown in Table 7, we additionally present the performance of various Difficulty Estimation ablation strategies across different evaluation datasets at each iterative round.

D.3 Results on Re-sampling Strategies

As shown in Table 8, we also provide the performance of various difficulty Re-Sampling ablation strategies across different evaluation datasets at each iteration round.